Shorewall 3.x Documentation

Tom Eastep

Copyright © 2001-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-11-16

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Note

The complete Shorewall Documentation is available for download in both Docbook XML and HTML formats.

If you are new to Shorewall, please read these two articles first.

- Introduction to Shorewall
- QuickStart Guides (HOWTOS)

The following article is also recommended reading for newcomers.

- Configuration File Basics
  - Comments in configuration files
  - Line Continuation
  - INCLUDE Directive
  - Port Numbers/Service Names
  - Port Ranges
  - Using Shell Variables
  - Using DNS Names

    - Complementing an IP address or Subnet
    - IP Address Ranges
    - Shorewall Configurations (making a test configuration)
    - Using MAC Addresses in Shorewall

The remainder of the Documentation supplements the QuickStart Guides. Please review the appropriate guide before trying to use this documentation directly.

1. 2.6 Kernel
2. Accounting
3. Actions
4. Aliased (virtual) Interfaces (e.g., eth0:0)
5. Bandwidth Control
6. Blacklisting
    - Static Blacklisting using /etc/shorewall/blacklist
    - Dynamic Blacklisting using /sbin/shorewall
7. Bridging
    - Bridge/Firewall (control traffic through the bridge)
    - Simple Bridge (don't need to control traffic through the bridge)
8. Commands (Description of all /sbin/shorewall commands)
9. Configuration File Reference Manual
    - params
    - zones
    - interfaces
    - hosts
    - policy
    - providers
    - rules
    - masq
    - proxyarp
    - nat
    - tunnels
    - tcrules
    - tcclasses
    - tcdevices
    - shorewall.conf
    - modules
    - tos
    - blacklist
    - rfc1918
    - routestopped
    - accounting

    - usersets and users
    - maclist
    - actions and action.template
    - netmap
    - ipsec
10. Corporate Network Example (Contributed by a Graeme Boyle)
11. DHCP
12. ECN Disabling by host or subnet
13. Error Messages
14. Extension Scripts (How to extend Shorewall without modifying Shorewall code through the use of files in /etc/shorewall -- /etc/shorewall/start, /etc/shorewall/stopped, etc.)
15. Fallback/Uninstall
16. FAQs
17. Features
18. Forwarding Traffic on the Same Interface
19. FTP and Shorewall
20. Getting help or answers to questions
21. Installation/Upgrade
22. IPP2P
23. IPSEC
24. IPSEC using Kernel 2.6 and Shorewall 2.1 or Later.
25. Ipsets
26. Kazaa Filtering
27. Kernel Configuration
28. Logging
29. Macros
30. MAC Verification
31. Multiple Internet Connections from a Single Firewall
32. Multiple Zones Through One Interface
33. My Shorewall Configuration (How I personally use Shorewall)
34. Netfilter Overview
35. Network Mapping
36. One-to-one NAT (Static NAT)
37. OpenVPN
38. Operating Shorewall
39. Packet Processing in a Shorewall-based Firewall
40. 'Ping' Management
41. Port Information
    - Which applications use which ports
    - Ports used by Trojans
42. Port Knocking

43. PPTP
44. Proxy ARP
45. Release Model
46. Requirements
47. Routing and Shorewall
48. Routing on One Interface
49. Samba
50. Shorewall Setup Guide
    - Introduction
    - Shorewall Concepts
    - Network Interfaces
    - Addressing, Subnets and Routing
        - IP Addresses
        - Subnets
        - Routing
        - Address Resolution Protocol (ARP)
        - RFC 1918
    - Setting up your Network
        - Routed
        - Non-routed
            - SNAT
            - DNAT
            - Proxy ARP
            - One-to-one NAT
        - Rules
        - Odds and Ends
    - DNS
    - Starting and Stopping the Firewall
51. SMB
52. Squid with Shorewall
53. Starting/stopping the Firewall
    - Description of all /sbin/shorewall commands
    - How to safely test a Shorewall configuration change
54. Static (one-to-one) NAT
55. Support
56. Traffic Accounting
57. Traffic Shaping/QOS
58. Troubleshooting (Things to try if it doesn't work)
59. UPnP
60. Upgrade Issues
61. VPN

- Basics
- IPSEC
- GRE and IPIP
- OpenVPN (My personal choice)
- PPTP
- 6to4
- IPSEC/PPTP passthrough from a system behind your firewall to a remote network
- Other VPN types

62. White List Creation

Appendix 1. GNU Free Documentation License

Version 1.2, November 2002

Table of Contents

PREAMBLE
APPLICABILITY AND DEFINITIONS
VERBATIM COPYING
COPYING IN QUANTITY
MODIFICATIONS
COMBINING DOCUMENTS
COLLECTIONS OF DOCUMENTS
AGGREGATION WITH INDEPENDENT WORKS
TRANSLATION
TERMINATION
FUTURE REVISIONS OF THIS LICENSE
ADDENDUM: How to use this License for your documents

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats

that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page.

If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant

Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no

> Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Introduction

Tom Eastep

Copyright © 2003-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-09-12

Table of Contents

Introduction
- Glossary
- What is Shorewall?

Shorewall Concepts
License

Introduction

The information in this document applies only to 3.x releases of Shorewall.

Glossary

- Netfilter - the packet filter facility built into the 2.4 and later Linux kernels.
- ipchains - the packet filter facility built into the 2.2 Linux kernels. Also the name of the utility program used to configure and control that facility. Netfilter can be used in ipchains compatibility mode.
- iptables - the utility program used to configure and control Netfilter. The term "iptables" is often used to refer to the combination of iptables+Netfilter (with Netfilter not in ipchains compatibility mode).

What is Shorewall?

The Shoreline Firewall, more commonly known as "Shorewall", is high-level tool for configuring Netfilter. You describe your firewall/gateway requirements using entries in a set of configuration

files. Shorewall reads those configuration files and with the help of the iptables utility, Shorewall configures Netfilter to match your requirements. Shorewall can be used on a dedicated firewall system, a multi-function gateway/router/server or on a standalone GNU/Linux system. Shorewall does not use Netfilter's ipchains compatibility mode and can thus take advantage of Netfilter's connection state tracking capabilities.

Shorewall is not a daemon. Once Shorewall has configured Netfilter, it's job is complete and there is no “Shorewall process” left running in your system. The /sbin/shorewall program can be used at any time to monitor the Netfilter firewall.

Shorewall is not the easiest to use of the available iptables configuration tools but I believe that it is the most flexible and powerful. So if you are looking for a simple point-and-click set-and-forget Linux firewall solution that requires a minimum of networking knowledge, I would encourage you to check out the following alternatives:

- http://www.m0n0.ch/wall/
- http://www.fs-security.com/

If you are looking for a Linux firewall solution that can handle complex and fast changing network environments then Shorewall is a logical choice.

Shorewall Concepts

The configuration files for Shorewall are contained in the directory `/etc/shorewall` -- for simple setups, you will only need to deal with a few of them.

Shorewall views the network where it is running as being composed of a set of *zones*. In the three-interface sample configuration for example, the following zone names are used:

```
#NAME                    DESCRIPTION
fw                       The firewall itself
net                      The Internet
loc                      Your Local Network
dmz                      Demilitarized Zone
```

Zones are defined in the `/etc/shorewall/zones` file.

Note that Shorewall recognizes the firewall system as its own zone. The name of the zone designating the firewall itself is stored in the shell variable $*FW* which may be used throughout the Shorewall configuration to refer to the firewall zone.

Rules about what traffic to allow and what traffic to deny are expressed in terms of zones.

- You express your default policy for connections from one zone to another zone in the `/etc/shorewall/policy` file. The basic choices for policy are:
  - ACCEPT - Accept the connection.
  - DROP - Ignore the connection request.
  - REJECT - Return an appropriate error to the connection request.

  Connection request logging may be specified as part of a policy and it is conventional to log DROP and REJECT policies.

- You define exceptions to these default policies in the `/etc/shorewall/rules` file.
- You only need concern yourself with connection requests. You don't need to define rules for how traffic that is part of an established connection is handled and in most cases you don't have to worry about how related connections are handled (ICMP error packets and related TCP connection requests such as used by FTP).

For each connection request entering the firewall, the request is first checked against the `/etc/shorewall/rules` file. If no rule in that file matches the connection request then the first policy in `/etc/shorewall/policy` that matches the request is applied. If there is a common action defined for the policy in /etc/shorewall/actions (or `/usr/share/shorewall/actions.std`) then that action is invoked before the policy is enforces. In the standard Shorewall distribution, the DROP policy has a common action called **Drop** and the REJECT policy has a common action called **Reject**. Common actions are used primarily to discard

The `/etc/shorewall/policy` file included with the three-interface sample has the following policies:

```
#SOURCE     DEST        POLICY      LOG LEVEL    LIMIT:BURST
loc         net         ACCEPT
net         all         DROP        info
all         all         REJECT      info
```

In the three-interface sample, the line below is included but commented out. If you want your firewall system to have full access to servers on the internet, uncomment that line.

```
#SOURCE     DEST        POLICY      LOG LEVEL    LIMIT:BURST
$FW         net         ACCEPT
```

The above policy will:

- Allow all connection requests from your local network to the internet
- Drop (ignore) all connection requests from the internet to your firewall or local network; these ignored connection requests will be logged using the *info* syslog priority (log level).

- Optionally accept all connection requests from the firewall to the internet (if you uncomment the additional policy)
- reject all other connection requests; these rejected connection requests will be logged using the *info* syslog priority (log level).

The simplest way to define a zone is to associate the zone with a network interface using the `/etc/shorewall/interfaces` file. In the three-interface sample, the three zones are defined using that file as follows:

```
#ZONE       INTERFACE      BROADCAST      OPTIONS
net         eth0           detect         dhcp,routefilter,norfc1918
loc         eth1           detect
dmz         eth2           detect
```

The above file defines the net zone as all hosts interfacing to the firewall through eth0, the loc zone as all hosts interfacing through eth1 and the dmz as all hosts interfacing through eth2.

To illustrate how rules provide exceptions to policies, suppose that you have the polcies listed above but you want to be able to connect to your firewall from the internet using Secure Shell (SSH). Recall that SSH connects uses TCP port 22.

```
#ACTION     SOURCE         DEST       PROTO      DEST
#                                                PORT(S)
ACCEPT      net            $FW        tcp        22
```

So although you have a policy of ignoring all connection attempts from the net zone (from the internet), the above exception to that policy allows you to connect to the SSH server running on your firewall.

Because Shorewall makes no assumptions about what traffic you want accepted, there are certain rules (exceptions) that need to be added to almost any configuration.

- The QuickStart guildes provide links to download pre-populated files for use in common setups and the Shorewall Setup Guide shows you examples for use with other more complex setups.
- To keep your firewall log from filling up with useless noise, Shorewall provides common actions that silently discard or reject such noise before it can be logged. As with everything in Shorewall, you can alter the behavior of these common actions (or do away with them entirely) as you see fit.

License

This program is free software; you can redistribute it and/or modify it under the terms of Version 2 of the GNU General Public License as published by the Free Software Foundation.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more detail.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA

Operating Shorewall

Tom Eastep

Copyright © 2004, 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-10-04

Table of Contents

Operational Components
Starting, Stopping and Clearing
Tracing Command Execution
Having Shorewall Start Automatically at Boot Time
Saving a Working Configuration for Error Recovery and Fast Startup
Additional Configuration Directories
Alternate Configuration Directories
Command Reference
Shorewall State Diagram

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Operational Components

There are a number of files that comprise the operational components of Shorewall.

- `/sbin/shorewall` — The program that you use to interact with Shorewall. Normally the root user's PATH includes `/sbin` and the program can be run from a shell prompt by simply typing **shorewall** followed by a command.

  ### Warning

  In some releases of KDE, the default configuration of the **konsole** program is brain dead with respect to the "Root Console". It executes the command "su" where it should execute "su -"; the latter will cause a login shell to be created which will in turn set PATH properly. You can correct this problem as follows:

  1. Click on "Settings" on the toolbar and select "Configure Konsole"
  2. Select the "Session" tab.
  3. Click on "Root Console"
  4. Change the Execute command from "su" to "su -"
  5. Click on "Save Session"
  6. Click on "Ok"

  To see a list of supported commands, use the **help** command:

  ```
  shorewall help
  ```

To get further information about a particular command, follow **help** by the command:

```
shorewall help start
```

- `/etc/shorewall` — The default directory where Shorewall looks for configuration files. See the sections entitled Additional Configuration Directories and Alternate Configuration Directories for information about how you can direct Shorewall to look in other directories.
- `/etc/init.d/shorewall` (`/etc/rc.d/firewall.rc` on Slackware) — The script run by **init** (the program responsible for startup and shutdown of your system) to start Shorewall at boot time and to stop Shorewall at shutdown.
- `/usr/share/shorewall/firewall` — The program responsible for configuring Netfilter based on your configuration files.
- `/usr/share/shorewall/functions` — A library of Bourne Shell functions used by both `/sbin/shorewall` and `/usr/share/shorewall/firewall`.

Starting, Stopping and Clearing

As explained in the Introduction, Shorewall is not something that runs all of the time in your system. Nevertheless, for integrating Shorewall into your initialization scripts it is useful to speak of *starting* Shorewall and *stopping* Shorewall.

- Shorewall is started using the **shorewall start** command. Once the start command completes successfully, Netfilter is configured as described in your Shorewall configuration files. If there is an error during **shorewall start**, then if you have a *saved configuration* then that configuration is restored. Otherwise, an implicit **shorewall stop** is executed.
- Shorewall is stopped using the **shorewall stop** command.

  ### Important

  The **shorewall stop** command does not remove all netfilter rules and open your firewall for all traffic to pass. It rather places your firewall in a safe state defined by the contents of your /etc/shorewall/routestopped file and the setting of ADMINISABSENTMINDED in /etc/shorewall/shorewall.conf.

- If you want to remove all Netfilter rules and open your firewall for all traffic to pass, use the **shorewall clear** command.
- If you change your configuration and want to install the changes, use the **shorewall restart** command.

For additional information, see the Shorewall State Diagram section.

Tracing Command Execution

If you include the word **trace** as the first parameter to an `/sbin/shorewall` command that transfers control to `/usr/share/shorewall/firewall`, execution of the latter program will be traced to STDERR.

Example 1. Tracing shorewall start

To trace the execution of **shorewall start** and write the trace to the file `/tmp/trace`, you would enter:

```
shorewall trace start 2> /tmp/trace
```

Having Shorewall Start Automatically at Boot Time

The .rpm, .deb and .tgz all try to configure your startup scripts so that Shorewall will start automatically at boot time. If you are using the **install.sh** script from the .tgz and it cannot determine how to configure automatic startup, a message to that effect will be displayed. You will need to consult your distribution's documentation to see how to integrate the `/etc/init.d/shorewall` script into the distribution's startup mechanism.

Caution

- Shorewall startup is disabled by default. Once you have configured your firewall, you can enable startup by editing

/etc/shorewall/shorewall.conf and setting STARTUP_ENABLED=Yes.. Note: Users of the .deb package must rather edit /etc/default/shorewall and set "startup=1".
- If you use dialup or some flavor of PPP where your IP address can change arbitrarily, you may want to start the firewall in your **/etc/ppp/ip-up.local** script. I recommend just placing "**/sbin/shorewall restart**" in that script.

Saving a Working Configuration for Error Recovery and Fast Startup

Once you have Shorewall working the way that you want it to, you can use **shorewall save** to *save* the commands necessary to recreate that configuration in a *restore script*.

In its simplest form, the save command is just:

```
shorewall save
```

That command creates the default restore script, /var/lib/shorewall/restore. The default may be changed using the RESTOREFILE option in /etc/shorewall/shorewall.conf. A different file name may also be specified in the **save** command:

```
shorewall save <filename>
```

Where <*filename*> is a simple file name (no slashes).

Once created, the default restore script serves several useful purposes:

- If you change your configuration and there is an error when you try to restart Shorewall, the restore script will be run to restore your firewall to working order.
- Bootup is faster. The -f option of the start command (e.g., **shorewall -f start**) causes Shorewall to look for the default restore script and if it exists, the script is run. This is much faster than starting Shorewall using the normal mechanism of reading the configuration files and running **iptables** dozens or even hundreds of times. By default, /etc/init.d/shorewall (/etc/rc.d/firewall.rc) uses the -f option when it is processing a request to start Shorewall.
- The **shorewall restore** command can be used at any time to quickly configure the firewall.

  ```
  shorewall restore [ <filename> ]
  ```

  If no <*filename*> is given, the default restore script is used. Otherwise, the script /var/lib/shorewall/<filename> is used.

The ability to have multiple restore scripts means that you can save different Shorewall firewall configurations and switch between them quickly using the **restore** command.

Restore scripts may be removed using the **shorewall forget** command:

```
shorewall forget [ <filename> ]
```

If no <*filename*> is given, the default restore script is removed. Otherwise, /var/lib/shorewall/<filename> is removed (of course, you can also use the Linux **rm** command from the shell prompt to remove these files).

Additional Configuration Directories

The CONFIG_PATH setting in /etc/shorewall/shorewall.conf determines where Shorewall looks for configuration files. The default setting is CONFIG_PATH=/etc/shorewall:/usr/share/shorewall which means that /etc/shorewall is searched first and if the file is not found then /usr/share/shorewall is searched. You can change the value of CONFIG_PATH to cause additional directories to be searched but CONFIG_PATH should *always* include both /etc/shorewall and /usr/share/shorewall.

When an alternate configuration directory is specified as described in the next section, that directory is searched *before* those directories listed in CONFIG_PATH.

Example - Search `/etc/shorewall`, `/etc/shorewall/actiondir` and `/usr/share/shorewall` in that order:

```
CONFIG_PATH=/etc/shorewall:/etc/shorewall/actiondir:/usr/share/shorewall
```

The above is the setting that I once used to allow me to place all of my user-defined 'action.' files in `/etc/shorewall/actiondir`.

Alternate Configuration Directories

As explained above, Shorewall normally looks for configuration files in the directories specified by the CONFIG_PATH option in `/etc/shorewall/shorewall.conf`. The **shorewall start**, **shorewall restart**, **shorewall check**, and **shorewall try** commands allow you to specify an additional directory for Shorewall to check before looking in the directories listed in CONFIG_PATH.

```
     shorewall {start|restart|check} <configuration-directory>
     shorewall try <configuration-directory> [ <timeout> ]
```

If a <*configuration-directory*> is specified, each time that Shorewall is going to read a file, it will first look in the <*configuration-directory*> . If the file is present in the <*configuration-directory*>, that file will be used; otherwise, the directories in the CONFIG_PATH will be searched. When changing the configuration of a production firewall, I recommend the following:

- If you haven't saved the current working configuration, do so using **shorewall save**.
- **mkdir /etc/test**
- **cd /etc/test**
- <copy any files that you need to change from /etc/shorewall to . and change them here>
- **shorewall check ./**
- <correct any errors found by check and check again>
- **shorewall try ./**

If the configuration starts but doesn't work, just “shorewall restart” to restore the old configuration. If the new configuration fails to start, the “try” command will automatically restore your configuration.

When the new configuration works then just:

- **cp -f * /etc/shorewall**
- **cd**
- **rm -rf /etc/test**
- **shorewall save**

Command Reference

add

shorewall add <interface>[:<host-list>] … <zone>

A <host-list> is a comma-separated list whose entries are:

- A host or network address
- The name of a bridge port
- The name of a bridge port followed by a colon (":") and a host or network address.

Adds an interface (and list of hosts if included) to a dynamic zone usually used with VPN's.

Example: **shorewall add ipsec0:192.0.2.24 vpn1**

adds the address 192.0.2.24 from interface ipsec0 to the zone vpn1.

allow

shorewall allow <address> ...

Re-enables receipt of packets from hosts previously blacklisted by a drop or reject command.

Shorewall allow, drop, rejct and save implement dynamic blacklisting.

check

shorewall [-q] check [<configuration-directory>]

Performs a cursory validation of the zones, interfaces, hosts, rules, policy, masq, blacklist, proxyarp, nat and provider files. Use this if you are unsure of any edits you have made to the shorewall configuration. See above for a recommended way to make changes.

clear

shorewall clear

Clear will remove all rules and chains installed by Shorewall. The firewall is then wide open and unprotected. Existing connections are untouched. Clear is often used to see if the firewall is causing connection problems.

delete

shorewall delete <interface>[:<host-list>] … <zone>

A <host-list> is a comma-separated list whose entries are:

- A host or network address
- The name of a bridge port
- The name of a bridge port followed by a colon (":") and a host or network address.

Deletes the specified interface (and host list if included) from the specified zone.

Example:

shorewall delete ipsec0:192.0.2.24 vpn1

deletes the address 192.0.2.24 from interface ipsec0 from zone vpn1

drop

shorewall drop <address> ...

Causes packets from the specified <*address*> to be ignored

dump

shorewall [-x] dump

Produce a verbose report about the firewall.

When -x is given, that option is also passed to iptables to display actual packet and byte counts.

forget

shorewall forget [<filename>]

Deletes `/var/lib/shorewall/<filename>`. If no <*filename*> is given then the file specified by RESTOREFILE in /etc/shorewall/shorewall.conf is removed.

help

shorewall help [<command> | host | address]

Display helpful information about the shorewall commands.

hits

hits

Produces several reports about the Shorewall packet log messages in the current log file specified by the LOGFILE option in /etc/shorewall/shorewall.conf.

ipcalc

shorewall ipcalc [<address> <mask> | <address>/<vlsm>]

Ipcalc displays the network address, broadcast address, network in CIDR notation and netmask corresponding to the input[s].

Example:

ipcalc 192.168.1.0/24

iprange

shorewall iprange <address1>-<address2>

Iprange decomposes the specified range of IP addresses into the equivalent list of network/host addresses.

logwatch

shorewall logwatch [<refresh interval>]

Monitors the log file specified by theLOGFILE option in /etc/shorewall/shorewall.conf and produces an audible alarm when new Shorewall messages are logged.

refresh

shorewall refresh: [-q] refresh

The rules involving the broadcast addresses of firewall interfaces, the black list, traffic control rules and ECN control rules are recreated to reflect any changes made to your configuration files. Existing connections are untouched If -q is specified, less detain is displayed making it easier to spot warnings.

reject

shorewall reject <address> ...

Causes packets from the specified <*address*>s to be rejected

reset

shorewall reset

All the packet and byte counters in the firewall are reset.

restart

shorewall [-q] restart <configuration-directory>

Restart is similar to **shorewall stop** followed by **shorewall start**. Existing connections are maintained. If -q is specified, less detail is displayed making it easier to spot warnings

restore

shorewall [-q] restore [<filename>]

Restore Shorewall to a state saved using the **shorewall save** command Existing connections are maintained. The <*filename*> names a restore file in `/var/lib/shorewall` created using **shorewall save**; if no <*filename*> is given then Shorewall will be restored from the file specified by the RESTOREFILE option in /etc/shorewall/shorewall.conf.

safe-restart

shorewall [-q] safe-restart [<filename>]

Only allowed if Shorewall is running. The current configuration is saved in `/var/lib/shorewall/safe-restart` (see the **save** command below). You will then be prompted asking if you want to accept the new configuration or not. If you answer "n" or if you fail to answer within 60 seconds (such as when your new configuration has disabled communication with your terminal), the configuration is restored from the saved configuration.

safe-start

shorewall [-q] safe-start [<filename>]

Shorewall is started normally. You will then be prompted asking if everything went all right. If you answer "n" or if you fail to answer within 60 seconds (such as when your new configuration has disabled communication with your terminal), a **shorewall clear** is performed for you.

save

shorewall save [<filename>]

The dynamic data is stored in /var/lib/shorewall/save. The state of the firewall is stored in `/var/lib/shorewall/<filename>` for use by the **shorewall restore** and **shorewall -f start** commands. If <*filename*> is not given then the state is saved in the file specified by the RESTOREFILE option in /etc/shorewall/shorewall.conf.

show

shorewall [-x] show [<chain> [<chain> ...] |classifiers|connections|log|nat|tc|tos]

shorewall [-x] show <chain> [<chain> ...] - produce a verbose report about the Netfilter chain(s). (**iptables -L chain -n -v**)

shorewall [-x] show nat - produce a verbose report about the nat table. (**iptables -t nat -L -n -v**)

shorewall [-x] show tos - produce a verbose report about the mangle table. (**iptables -t mangle -L -n -v**)

shorewall show log - display the last 20 packet log entries.

shorewall show capabilities - Displays your kernel/iptables capabilities

shorewall show connections - displays the IP connections currently being tracked by the firewall.

shorewall show classifiers - displays information about the traffic control/shaping classifiers.

shorewall show tc - displays information about the traffic control/shaping configuration.

shorewall show zones — Displays the composition of each zone.

When -x is given, that option is also passed to iptables to display actual packet and byte counts.

start

shorewall [-q] [-f] start [<configuration-directory>]

Start shorewall. Existing connections through shorewall managed interfaces are untouched. New connections will be allowed only if they are allowed by the firewall rules or policies. If -q is specified, less detail is displayed making it easier to spot warnings If -f is specified, the saved configuration specified by the RESTOREFILE option in /etc/shorewall/shorewall.conf will be restored if that saved configuration exists

stop

shorewall stop

Stops the firewall. All existing connections, except those listed in `/etc/shorewall/routestopped` or permitted by the ADMINISABSENTMINDED option in /etc/shorewall/shorewall.conf, are taken down. The only new traffic permitted through the firewall is from systems listed in `/etc/shorewall/routestopped` or by ADMINISABSENTMINDED.

status

shorewall status

Produce a short report about the firewall's status and state relative to the diagram below.

try

shorewall try <configuration-directory> [<timeout>]

Restart shorewall using the specified configuration. If an error occurs during the restart, then another shorewall restart is performed using the default configuration. If a timeout is specified then the restart is always performed after the timeout occurs and uses the default configuration.

When restarting using the default configuration, if the default restore script (as specified by the RESTOREFILE setting in /etc/shorewall/shorewall.conf) exists. then that script is used.

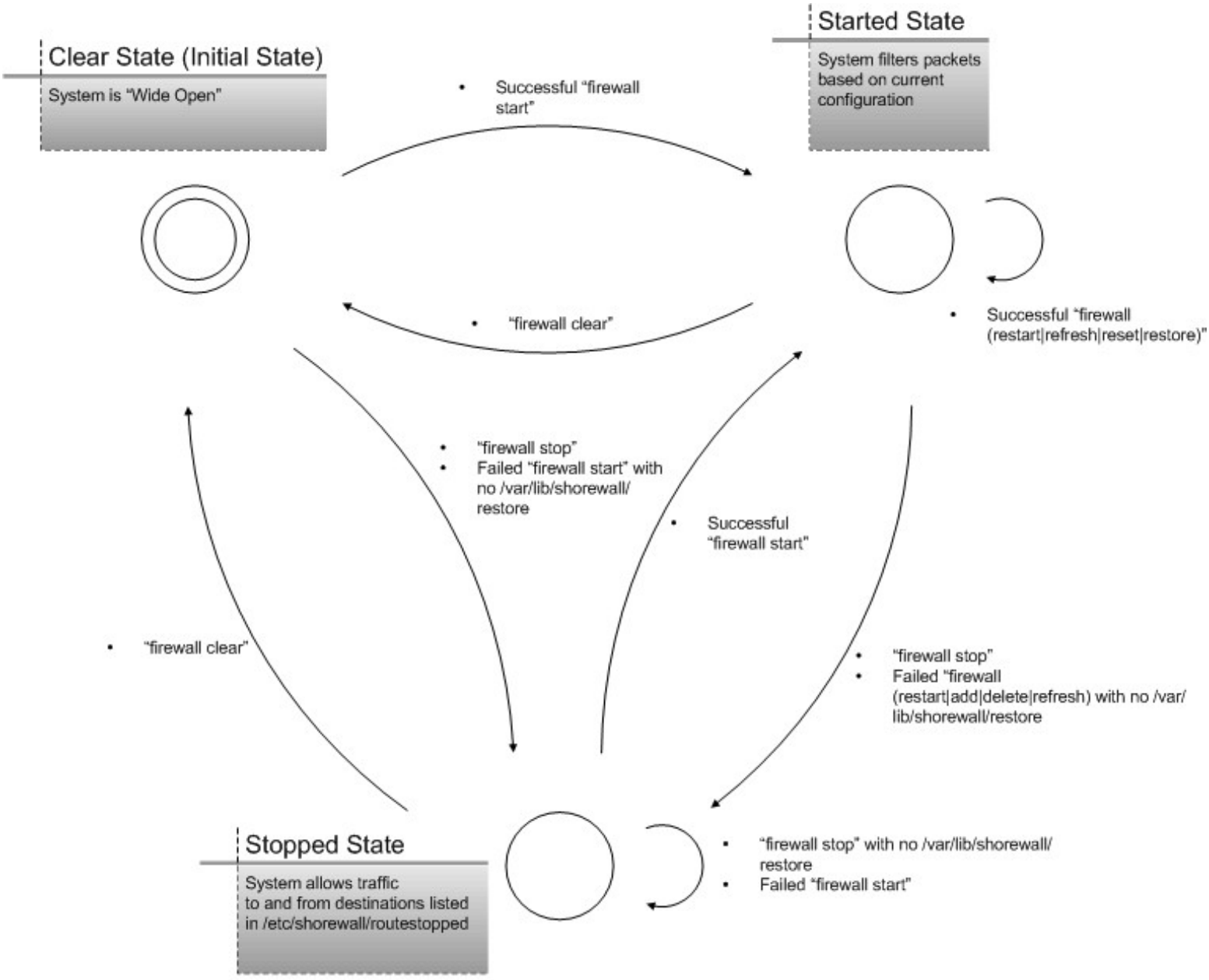
version

shorewall version

Show the current shorewall version

Shorewall State Diagram

The Shorewall State Diargram is depicted below.



Shorewall State Diagram

You will note that the commands that result in state transitions use the word “firewall” rather than “shorewall”. That is because the actual transitions are done by **/usr/share/shorewall/firewall**; **/sbin/shorewall** runs “firewall” according to the following table:

/sbin/shorewall Command	**Resulting /usr/share/shorewall/firewall Command**	**Effect if the Command Succeeds**
shorewall start	firewall start	The system filters packets based on your current Shorewall Configuration
shorewall stop	firewall stop	Only traffic to/from hosts listed in /etc/shorewall/hosts is passed to/from/through the firewall. If ADMINISABSENTMINDED=Yes in /etc/shorewall/shorewall.conf then in addition, all existing connections are retained and all connection requests from the firewall are accepted.
shorewall restart	firewall restart	Logically equivalent to “firewall stop;firewall start”
shorewall add	firewall add	Adds a host or subnet to a dynamic zone
shorewall delete	firewall delete	Deletes a host or subnet from a dynamic zone
shorewall refresh	firewall refresh	Reloads rules dealing with static blacklisting, traffic control and ECN.
shorewall reset	firewall reset	Resets traffic counters
shorewall clear	firewall clear	Removes all Shorewall rules, chains, addresses, routes and ARP entries.
shorewall try	firewall -c <new configuration> restart If unsuccessful then firewall start (standard configuration) If timeout then firewall restart (standard configuration)	

Shorewall 3.x Reference

Tom Eastep

Copyright © 2001-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-11-02

Abstract

This documentation is intended primarily for reference. Step-by-step instructions for configuring Shorewall in common setups may be found in the QuickStart Guides.

Table of Contents

Components
/etc/shorewall/params
/etc/shorewall/zones
/etc/shorewall/interfaces
/etc/shorewall/hosts Configuration
    Nested and Overlapping Zones
/etc/shorewall/policy Configuration
    Intra-Zone Traffic
    The CONTINUE policy
/etc/shorewall/rules
/etc/shorewall/masq
/etc/shorewall/proxyarp
/etc/shorewall/nat
/etc/shorewall/tunnels
/etc/shorewall/shorewall.conf
/etc/shorewall/modules Configuration
/etc/shorewall/tos Configuration
/etc/shorewall/blacklist
/usr/share/shorewall/rfc1918
/etc/shorewall/netmap
/etc/shorewall/routestopped
/etc/shorewall/maclist
/etc/shorewall/ecn
/etc/shorewall/accounting
1. Revision History

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than

Shorewall 3.0.0 then please see the documentation for that release.

Components

Shorewall consists of the following components:

params

a parameter file installed in `/etc/shorewall` that can be used to establish the values of shell variables for use in other files.

shorewall.conf

a parameter file installed in `/etc/shorewall` that is used to set several firewall parameters.

zones

a parameter file installed in `/etc/shorewall` that defines a network partitioning into “zones”

policy

a parameter file installed in `/etc/shorewall` that establishes overall firewall policy.

rules

a parameter file installed in `/etc/shorewall` and used to express firewall rules that are exceptions to the high-level policies established in /etc/shorewall/policy.

blacklist

a parameter file installed in `/etc/shorewall` and used to list blacklisted IP/subnet/MAC addresses.

ecn

a parameter file installed in `/etc/shorewall` and used to selectively disable Explicit Congestion Notification (ECN - RFC 3168).

functions

a set of shell functions used by both the firewall and shorewall shell programs. Installed in `/usr/share/shorewall`.

modules

a parameter file installed in `/etc/shorewall` and that specifies kernel modules and their parameters. Shorewall will automatically load the modules specified in this file.

tos

a parameter file installed in `/etc/shorewall` that is used to specify how the Type of Service (TOS) field in packets is to be set.

init.sh and init.debian.sh

a shell script installed in `/etc/init.d` to automatically start Shorewall during boot. The particular script installed depends on which distribution you are running.

interfaces

a parameter file installed in `/etc/shorewall` and used to describe the interfaces on the firewall system.

hosts

a parameter file installed in `/etc/shorewall` and used to describe individual hosts or subnetworks in zones.

maclist

a parameter file installed in `/etc/shorewall` and used to verify the MAC address (and possibly also the IP address(es)) of devices.

masq

This file also describes IP masquerading under Shorewall and is installed in `/etc/shorewall`.

firewall

a shell program that reads the configuration files in `/etc/shorewall` and configures your firewall. This file is installed in `/usr/share/shorewall`.

nat

a parameter file in `/etc/shorewall` used to define one-to-one NAT.

proxyarp

a parameter file in `/etc/shorewall` used to define Proxy Arp.

rfc1918

a parameter file in `/usr/share/shorewall` used to define the treatment of packets under the norfc1918 interface option.

routestopped

a parameter file in `/etc/shorewall` used to define those hosts that can access the firewall when Shorewall is stopped.

tcrules

a parameter file in `/etc/shorewall` used to define rules for classifying packets for Traffic Shaping/Control.

tcdevices

a parameter file in `/etc/shorewall` used to define the bandwidth for interfaces on which you want traffic shaping to be enabled.

tcclasses

a parameter file in `/etc/shorewall` used to define classes for traffic shaping.

tcstart

a set of shell functions used by Shorewall to setup traffic shaping.This file is installed in `/usr/share/shorewall`.

tunnels

a parameter file in `/etc/shorewall` used to define IPSec tunnels.

shorewall

a shell program (requiring a Bourne shell or derivative) used to control and monitor the firewall. This should be placed in `/sbin` or in `/usr/sbin` (the install.sh script and the rpm install this file in `/sbin`).

accounting

a parameter file in `/etc/shorewall` used to define traffic accounting rules.

version

a file created in `/usr/share/shorewall` that describes the version of Shorewall installed on your system.

actions and action.template

files in `/etc/shorewall` and `/usr/share/shorewall` respectively that allow you to define your own actions for rules in `/etc/shorewall/rules`.

actions.std and action.*

files in `/usr/share/shorewall` that define the actions included as a standard part of Shorewall.

providers

file in /etc/shorewall that is used to define multiple Internet Service Providers and load-balancing.

routes

file in /etc/shorewall that is used to interface to the experimental ROUTE target from Netfilter patch-o-matic-ng.

/etc/shorewall/params

You may use the file `/etc/shorewall/params` file to set shell variables that you can then use in some of the other configuration files.

It is suggested that variable names begin with an upper case letter to distinguish them from variables used internally within the Shorewall programs

Example 1. shell variables

```
NET_IF=eth0 NET_BCAST=130.252.100.255
NET_OPTIONS=blacklist,norfc1918
```

Example 2. /etc/shorewall/interfaces record

```
net $NET_IF $NET_BCAST $NET_OPTIONS
```

The result will be the same as if the record had been written

```
net eth0 130.252.100.255 blacklist,norfc1918
```

Variables may be used anywhere in the other configuration files.

/etc/shorewall/zones

This file is used to define the network zones. There is one entry in `/etc/shorewall/zones` for each zone; Columns in an entry are:

ZONE

short name for the zone. The name should be 5 characters or less in length and consist of lower-case letters or numbers. Short names must begin with a letter and the name assigned to the firewall is reserved for use by Shorewall itself. Note that the output produced by iptables is much easier to read if you select short names that are three characters or less in length. The name “all” may not be used as a zone name nor may the zone name assigned to the firewall itself via the FW variable in /etc/shorewall/shorewall.conf.

TYPE

ipsec - All traffic to/from this zone is encrypted.

ipv4 - By default, traffic to/from some of the hosts in this zone is not encrypted. Any encrypted hosts are designated using the **ipsec** option in /etc/shorewall/hosts.

firewall - Designates the firewall itself. You must have exactly one 'firewall' zone. No options are permitted with a 'firewall' zone.

OPTIONS, IN OPTIONS, OUT OPTIONS

Optional parameters that identify the security policy and security associations used in communication with hosts in this zone. A comma-separated list of the following:

proto[!]=ah|esp|ipcomp

mode[!]=transport|tunnel

reqid[!]=<*number*> — A number assigned to a security policy using the unique:<number> as the SPD level. See setkey(8).

tunnel-src[!]=<*address*>[/<*mask*>] — Tunnel Source; may only be included with mode=tunnel. Since tunnel source and destination are dependent on the direction of the traffic, this option and the following one should only be included in the IN OPTIONS and OUT OPTIONS columns.

tunnel-dst[!]=<*address*>[/<*mask*>] — Tunnel Destination; may only be included with mode=tunnel.

mss=<number> — Sets the MSS field in TCP syn packets forwarded to/from this zone. May be used to compensate for the lack of IPSEC pseudo-devices with their own MTU in the 2.6 Kernel IPSEC implementation. If specified in the IN OPTIONS, TCP SYN packets from the zone will have MSS altered; if specified in the OUT OPTIONS, TCP SYN packets to the zone will have MSS altered.

spi[!]=<*number*> — The security parameter index of the Security Association. Since a different SA is used for incoming and outgoing traffic, this option should only be listed in the IN OPTIONS and OUT OPTIONS columns.

strict — Must be specified when SPD rules are used (e.g., esp encapsulated with ah).

next — Separates rules when strict is used.

Warning

The order of entries in the `/etc/shorewall/zones` file is significant in some cases.

/etc/shorewall/interfaces

This file is used to tell the firewall which of your firewall's network interfaces are connected to which zone. There will be one entry in /etc/shorewall/interfaces for each of your interfaces. Columns in an entry are:

ZONE

A zone defined in the /etc/shorewall/zones file or “-”. If you specify “-”, you must use the /etc/shorewall/hosts file to define the zones accessed via this interface.

INTERFACE

the name of the interface (examples: eth0, ppp0, ipsec+). Each interface can be listed on only one record in this file.

Note

You do not need to include the loopback interface (lo) in this file.

BROADCAST

the broadcast address(es) for the sub-network(s) attached to the interface. This should be left empty for P-T-P interfaces (ppp*, ippp*); if you need to specify options for such an interface, enter “-” in this column. If you supply the special value “detect” in this column, the firewall will automatically determine the broadcast address. In order to use “detect”:

- the interface must be up before you start your firewall
- the interface must only be attached to a single sub-network (i.e., there must have a single broadcast address).

OPTIONS

a comma-separated list of options. Possible options include:

arp_filter

This option causes `/proc/sys/net/ipv4/conf/<interface>/arp_filter` to be set with the result that this interface will only answer ARP “who-has” requests from hosts that are routed out of that interface. Setting this option facilitates testing of your firewall where multiple firewall interfaces are connected to the same HUB/Switch (all interface connected to the single HUB/Switch should have this option specified). Note that using such a configuration in a production environment is strongly recommended against.

arp_ignore[=<*number*>]

respond to arp requests based on the value of <*number*>.

1 - reply only if the target IP address is local address configured on the incoming interface

2 - reply only if the target IP address is local address configured on the incoming interface and both with the sender's IP address are part from same subnet on this interface

3 - do not reply for local addresses configured with scope host, only resolutions for global and link addresses are replied

4-7 - reserved

8 - do not reply for all local addresses

If no <*number*> is given then the value 1 is assumed

Warning

DO NOT SPECIFY arp_ignore FOR ANY INTERFACE INVOLVED IN PROXY ARP.

routeback

This option causes Shorewall to set up handling for routing packets that arrive on this interface back out the same interface. If this option is specified, the ZONE column may not contain “-”.

tcpflags

This option causes Shorewall to make sanity checks on the header flags in TCP packets arriving on this interface. Checks include Null flags, SYN+FIN, SYN+RST and FIN+URG+PSH; these flag combinations are typically used for “silent” port scans. Packets failing these checks are logged according to the TCP_FLAGS_LOG_LEVEL option in /etc/shorewall/shorewall.conf and are disposed of according to the TCP_FLAGS_DISPOSITION option.

blacklist

This option causes incoming packets on this interface to be checked against the blacklist.

dhcp

The interface is assigned an IP address via DHCP or is used by a DHCP server running on the firewall. The firewall will be configured to allow DHCP traffic to and from the interface even when the firewall is stopped. You may also wish to use this option if you have a static IP but you are on a LAN segment that has a lot of Laptops that use DHCP and you select the **norfc1918** option (see below).

norfc1918

Packets arriving on this interface and that have a source or destination address that is reserved in RFC 1918 will be dropped after being optionally logged.

Beware that as IPv4 addresses become in increasingly short supply, ISPs are beginning to use RFC 1918 addresses within their own infrastructure. Also, many cable and DSL “modems” have an RFC 1918 address that can be used through a web browser for management and monitoring functions. If you want to specify **norfc1918** on your external interface but need to allow access to certain addresses from the above list, see FAQ 14.

routefilter

Invoke the Kernel's route filtering (anti-spoofing) facility on this interface. The kernel will reject any packets incoming on this interface that have a source address that would be routed outbound through another interface on the firewall.

Warning

If you specify this option for an interface then the interface must be up prior to starting the firewall.

proxyarp

This option causes Shorewall to set /proc/sys/net/ipv4/conf/<*interface*>/proxy_arp and is used when implementing Proxy ARP Sub-netting as described at http://www.tldp.org/HOWTO/mini/Proxy-ARP-Subnet/. Do **not** set this option if you are implementing Proxy ARP through entries in /etc/shorewall/proxyarp.

maclist

If this option is specified, all connection requests from this interface are subject to MAC Verification. May only be specified for ethernet interfaces.

detectnets

If this option is specified, the zone named in the ZONE column will contain only the hosts routed through the interface named in the INTERFACE column. **Do not set this option on your external (Internet) interface!** The interface must be in the UP state when Shorewall is [re]started.

nosmurfs

If this option is specified, incoming connection requests will be checked to ensure that they do not have a broadcast or multicast address as their source. Any such packets will be dropped after being optionally logged according to the setting of SMURF_LOG_LEVEL in /etc/shorewall/shorewall.conf.

logmartians

If this option is specified, the kernel's martian logging facility will be enabled on this interface (/proc/sys/net/ipv4/conf/<*interface*>/log_martians will be set to 1). See also the LOG_MARTIANS option in shorewall.conf.

sourceroute

If this option is not specified for an interface, then source-routed packets will not be accepted from that interface (sets `/proc/sys/net/ipv4/conf/<interface>` to 1).**Only set this option if you know what are you doing**.

upnp

Incoming requests from this interface may be remapped via UPNP (upnpd).

My recommendations concerning options:

- External Interface -- **tcpflags,blacklist,norfc1918,routefilter,nosmurfs,logmartians**
- Wireless Interface -- **maclist,routefilter,tcpflags,detectnets,nosmurfs**
- Use **dhcp** and **proxyarp** when needed.

Example 3. You have a conventional firewall setup in which eth0 connects to a Cable or DSL modem and eth1 connects to your local network and eth0 gets its IP address via DHCP. You want to check all packets entering from the internet against the black list. Your /etc/shorewall/interfaces file would be as follows:

```
#ZONE    INTERFACE    BROADCAST    OPTIONS
net      eth0         detect       dhcp,norfc1918,blacklist
```

Example 4. You have a standalone dialup GNU/Linux System. Your /etc/shorewall/interfaces file would be:

```
#ZONE    INTERFACE    BROADCAST    OPTIONS
net      ppp0
```

Example 5. You have local interface eth1 with two IP addresses - 192.168.1.1/24 and 192.168.12.1/24

```
#ZONE    INTERFACE    BROADCAST                    OPTIONS
loc      eth1         192.168.1.255,192.168.12.255
```

/etc/shorewall/hosts Configuration

For most applications, specifying zones entirely in terms of network interfaces is sufficient. There may be times though where you need to define a zone to be a more general collection of hosts. This is the purpose of the `/etc/shorewall/hosts` file.

Warning

The only time that you need entries in `/etc/shorewall/hosts` is where you have more than one zone connecting through a single interface.

IF YOU DON'T HAVE THIS SITUATION THEN DON'T TOUCH THIS FILE!!

Columns in this file are:

ZONE

A zone defined in the /etc/shorewall/zones file.

HOST(S)

The name of an interface defined in the /etc/shorewall/interfaces file followed by a colon (":") and a comma-separated list whose elements are either:

1. The IP address of a host
2. A subnetwork in the form *<subnet-address>/<mask width>*
3. A physical port name; only allowed when the interface names a bridge created by the **brctl addbr** command. This port must not be defined in `/etc/shorewall/interfaces` and may optionally followed by a colon (":") and a host or network IP. See the bridging documentation for details.

OPTIONS

A comma-separated list of option

routeback

This option causes Shorewall to set up handling for routing packets sent by this host group back back to the same group.

maclist

If specified, connection requests from the hosts specified in this entry are subject to MAC Verification. This option is only valid for ethernet interfaces.

tcpflags

This option causes Shorewall to make sanity checks on the header flags in TCP packets arriving from these hosts. Checks include Null flags, SYN+FIN, SYN+RST and FIN+URG+PSH; these flag combinations are typically used for “silent” port scans. Packets failing these checks are logged according to the TCP_FLAGS_LOG_LEVEL option in /etc/shorewall/shorewall.conf and are disposed of according to the TCP_FLAGS_DISPOSITION option.

blacklist (only makes sense for bridge ports)

This option causes incoming packets on this port to be checked against the blacklist.

norfc1918 -- only makes sense for bridge ports)

Packets arriving on this port and that have a source address that is reserved in RFC 1918 will be dropped after being optionally logged as specified in the section of RFC1918_LOG_LEVEL in shorewall.conf.

nosmurfs (only makes sense for bridge ports)

If this option is specified, incoming connection requests will be checked to ensure that they do not have a broadcast or multicast address as their source. Any such packets will be dropped after being optionally logged according to the setting of SMURF_LOG_LEVEL in /etc/shorewall/shorewall.conf.

ipsec

The hosts are accessed via a kernel 2.6 ipsec SA. Note that if the zone named in the ZONE column is specified as an IPSEC zone in the /etc/shorewall/zones file then you do NOT need to specify 'ipsec' here.

If you don't define any hosts for a zone, the hosts in the zone default to i0:0.0.0.0/0 , i1:0.0.0.0/0, ... where i0, i1, ... are the interfaces to the zone.

Note

You probably DON'T want to specify any hosts for your internet zone since the hosts that you specify will be the only ones that you will be able to access without adding additional rules.

Example 6. Your local interface is eth1 and you have two groups of local hosts that you want to make into separate zones:

```
192.168.1.0/25 192.168.1.128/25
```

Your /etc/shorewall/interfaces file might look like:

```
#ZONE    INTERFACE    BROADCAST    OPTIONS
net      eth0         detect       dhcp,norfc1918
-        eth1         192.168.1.127,192.168.1.255
```

The “-” in the ZONE column for eth1 tells Shorewall that eth1 interfaces to multiple zones.

```
#ZONE       HOST(S)                OPTIONS
loc1        eth1:192.168.1.0/25
loc2        eth1:192.168.1.128/25
```

Example 7. You have local interface eth1 with two IP addresses - 192.168.1.1/24 and 192.168.12.1/24

Your /etc/shorewall/interfaces file might look like:

```
#ZONE    INTERFACE    BROADCAST    OPTIONS
net      eth0         detect       dhcp,norfc1918
```

```
-        eth1         192.168.1.255,192.168.12.255
```

Your /etc/shorewall/hosts file might look like:

```
#ZONE       HOST(S)                OPTIONS
loc         eth1:192.168.1.0/24,192.168.12.0/24
```

Nested and Overlapping Zones

The `/etc/shorewall/interfaces` and `/etc/shorewall/hosts` file allow you to define nested or overlapping zones. Such overlapping/nested zones are allowed and Shorewall processes zones in the order that they appear in the `/etc/shorewall/zones` file. So if you have nested zones, you want the sub-zone to appear before the super-zone and in the case of overlapping zones, the rules that will apply to hosts that belong to both zones is determined by which zone appears first in `/etc/shorewall/zones`.

Hosts that belong to more than one zone may be managed by the rules of all of those zones. This is done through use of the special CONTINUE policy described below.

/etc/shorewall/policy Configuration

This file is used to describe the firewall policy regarding establishment of connections. Connection establishment is described in terms of *clients* who initiate connections and *servers* who receive those connection requests. Policies defined in `/etc/shorewall/policy` describe which zones are allowed to establish connections with other zones.

Policies established in `/etc/shorewall/policy` can be viewed as default policies. If no rule in /etc/shorewall/rules applies to a particular connection request then the policy from `/etc/shorewall/policy` is applied.

Five policies are defined:

ACCEPT

The connection is allowed.

DROP

The connection request is ignored.

REJECT

The connection request is rejected with an RST (TCP) or an ICMP destination-unreachable packet being returned to the client.

QUEUE

Send the connection request to a user-space process via the iptables QUEUE target (useful when you are using Snort-inline).

CONTINUE

The connection is neither ACCEPTed, DROPped nor REJECTed. CONTINUE may be used when one or both of the zones named in the entry are sub-zones of or intersect with another zone. For more information, see below.

NONE

Shorewall should not set up any infrastructure for handling traffic from the SOURCE zone to the DEST zone. When this policy is specified, the **LOG LEVEL** and **BURST:LIMIT** columns must be left blank.

For each policy specified in /etc/shorewall/policy, you can indicate that you want a message sent to your system log each time

that the policy is applied.

Entries in /etc/shorewall/policy have four columns as follows:

SOURCE

The name of a client zone (a zone defined in the /etc/shorewall/zones file , the name of the firewall zone or “all”).

DEST

The name of a destination zone (a zone defined in the /etc/shorewall/zones file , the name of the firewall zone or “all”). Shorewall automatically allows all traffic from the firewall to itself so the name of the firewall zone cannot appear in both the SOURCE and DEST columns.

POLICY

The default policy for connection requests from the SOURCE zone to the DESTINATION zone.

LOG LEVEL

Optional. If left empty, no log message is generated when the policy is applied. Otherwise, this column should contain an integer or name indicating a syslog level.

LIMIT:BURST - optional

If left empty, TCP connection requests from the **SOURCE** zone to the **DEST** zone will not be rate-limited. Otherwise, this column specifies the maximum rate at which TCP connection requests will be accepted followed by a colon (“:”) followed by the maximum burst size that will be tolerated. Example: **10/sec:40** specifies that the maximum rate of TCP connection requests allowed will be 10 per second and a burst of 40 connections will be tolerated. Connection requests in excess of these limits will be dropped. See the rules file documentation for an explanation of how rate limiting works.

In the SOURCE and DEST columns, you can enter “all” to indicate all zones.

The default `/etc/shorewall/policy` file is as follows.

```
#SOURCE    DEST      POLICY    LOG LEVEL    LIMIT:BURST
loc        net       ACCEPT
net        all       DROP      info
all        all       REJECT    info
```

This table may be interpreted as follows:

- All connection requests from the local network to hosts on the internet are accepted.
- All connection requests originating from the internet are ignored and logged at level KERNEL.INFO.
- All other connection requests are rejected and logged.

Warning

The firewall script processes the `/etc/shorewall/policy` file from top to bottom and **uses the first applicable policy that it finds**. For example, in the following policy file, the policy for (loc, loc) connections would be ACCEPT as specified in the first entry even though the third entry in the file specifies REJECT.

```
#SOURCE    DEST      POLICY    LOG LEVEL    LIMIT:BURST
loc        all       ACCEPT
net        all       DROP      info
loc        loc       REJECT    info
```

Intra-Zone Traffic

Shorewall allows a zone to be associated with more than one interface or with multiple networks that interface through a single interface. Shorewall will ACCEPT all traffic from a zone to itself provided that there is no explicit policy governing traffic from that zone to itself (an explicit policy does not specify “all” in either the SOURCE or DEST column).

The idea is this:

1. A zone should be homogeneous with respect to security requirements.
2. Traffic within a zone should not require rules or policies.
3. Shorewall will not restrict traffic within a zone.

Any time that you have multiple interfaces associated with a single zone, you should ask yourself if you really want traffic routed between those interfaces. Cases where you might not want that behavior are:

1. Multiple “net” interfaces to different ISPs. You don't want to route traffic from one ISP to the other through your firewall.
2. Multiple VPN clients. You don't necessarily want them to all be able to communicate between themselves using your gateway/router.

You can control the traffic from the firewall to itself. As with any zone, fw->fw traffic is enabled by default. It is not necessary to define the loopback interface (lo) in /etc/shorewall/interfaces in order to define fw->fw rules or a fw->fw policy.

The CONTINUE policy

Where zones are nested or overlapping, the CONTINUE policy allows hosts that are within multiple zones to be managed under the rules of all of these zones. Let's look at an example:

`/etc/shorewall/zones`:

```
#ZONE    TYPE        OPTION
$FW      firewall
sam      ipv4
net      ipv4
loc      ipv4
```

`/etc/shorewall/interfaces`:

```
#ZONE     INTERFACE     BROADCAST     OPTIONS
-         eth0          detect        dhcp,norfc1918
loc       eth1          detect
```

`/etc/shorewall/hosts`:

```
#ZONE     HOST(S)                     OPTIONS
net       eth0:0.0.0.0/0
sam       eth0:206.191.149.197
```

Note

Sam's home system is a member of both the **sam** zone and the **net** zone and as described above , that means that **sam** must be listed before **net** in `/etc/shorewall/zones`.

`/etc/shorewall/policy`:

```
#SOURCE      DEST        POLICY       LOG LEVEL
loc          net         ACCEPT
sam          all         CONTINUE
net          all         DROP         info
all          all         REJECT       info
```

The second entry above says that when Sam is the client, connection requests should first be process under rules where the source zone is **sam** and if there is no match then the connection request should be treated under rules where the source zone is **net**. It is important that this policy be listed BEFORE the next policy (**net** to **all**).

Partial `/etc/shorewall/rules`:

```
#ACTION   SOURCE    DEST            PROTO    DEST PORT(S)
...
DNAT      sam       loc:192.168.1.3 tcp      ssh
DNAT      net       loc:192.168.1.5 tcp      www
...
```

Given these two rules, Sam can connect to the firewall's internet interface with ssh and the connection request will be forwarded to 192.168.1.3. Like all hosts in the **net** zone, Sam can connect to the firewall's internet interface on TCP port 80 and the connection request will be forwarded to 192.168.1.5. The order of the rules is not significant.

Sometimes it is necessary to suppress port forwarding for a sub-zone. For example, suppose that all hosts can SSH to the firewall and be forwarded to 192.168.1.5 EXCEPT Sam. When Sam connects to the firewall's external IP, he should be connected to the firewall itself. Because of the way that Netfilter is constructed, this requires two rules as follows:

```
#ACTION   SOURCE    DEST            PROTO    DEST PORT(S)
...
DNAT      sam       $FW             tcp      ssh
DNAT      net       loc:192.168.1.3 tcp      ssh
...
```

The first rule allows Sam SSH access to the firewall. The second rule says that any clients from the net zone with the exception of those in the “sam” zone should have their connection port forwarded to 192.168.1.3. If you need to exclude more than one zone in this way, you can list the zones separated by commas (e.g., net!sam,joe,fred). This technique also may be used when the ACTION is REDIRECT.

/etc/shorewall/rules

The `/etc/shorewall/rules` file defines exceptions to the policies established in the `/etc/shorewall/policy` file. There is one entry in /etc/shorewall/rules for each of these rules. Entries in this file only govern the establishment of new connections — packets that are part of an existing connection or that establish a connection that is related to an existing connection are automatically accepted.

Rules for each pair of zones (source zone, destination zone) are evaluated in the order that they appear in the file — the first match determines the disposition of the connection request with a couple of caveats:

- LOG rules cause the connection request to be logged then processing continues with the next rule in the file.
- QUEUE rules cause the connection request to be passed to user-space -- the user-space application can later insert them back into the stream for further processing by following rules.
- CONTINUE rules may cause the connection request to be reprocessed using a different (source zone, destination zone)

pair.

The /etc/shorewall/rules file may now be into *sections*. Each section is introduced by a line that begins with the keyword SECTION which is followed by the section name. Sections are as listed below and must appear in the order shown.

ESTABLISHED

Rules in this section apply to packets in the ESTABLISHED state.

RELATED

Rules in this section apply to packets in the RELATED state.

NEW

Rules in this section apply to packets in the NEW and INVALID states.

Rules in the ESTABLISHED and RELATED sections are limited to the following ACTIONs:

ACCEPT, DROP, REJECT, QUEUE, LOG and User-defined actions.

Macros may be used in these sections provided that they expand to only these ACTIONs. At the end of the ESTABLISHED and RELATED sections, there is an implicit ACCEPT rule.

RESTRICTION: If you specify FASTACCEPT=Yes in /etc/shorewall.shorewall.conf then the ESTABLISHED and RELATED sections must be empty.

Caution

Unless you understand Netfilter well enough to be comfortable with the difference between ESTABLISHED, RELATED, INVALID and NEW connection tracking states, you should omit the ESTABLISHED and RELATED sections and place all of your rules in the NEW section.

Entries in the file have the following columns:

ACTION

ACCEPT, DROP, REJECT, CONTINUE

These have the same meaning here as in the policy file above.

ACCEPT+

Works like ACCEPT but also exempts the connection from matching DNAT and REDIRECT rules later in the file.

NONAT

Exempts matching connections from DNAT and REDIRECT rules later in the file.

DNAT

Causes the connection request to be forwarded to the system specified in the DEST column (port forwarding). “DNAT” stands for “**D**estination **N**etwork **A**ddress **T**ranslation”

DNAT-

The above ACTION (DNAT) generates two iptables rules:

1. a header-rewriting rule in the Netfilter “nat” table
2. an ACCEPT rule in the Netfilter “filter” table.

DNAT- works like DNAT but only generates the header-rewriting rule.

SAME

SAME is useful when more than one server IP address (an address range, for example) is given in the DEST column below. SAME works similar to DNAT with the exception that when multiple connections from an internet host match a SAME rule then all of the connections will be sent to the same internal server.

Note

Unlike when using DNAT rules, SAME rules may not alter the destination port number used for the connection.

SAME-

SAME generates two iptables rules:

1. a header-rewriting rule in the Netfilter “nat” table
2. an ACCEPT rule in the Netfilter “filter” table.

SAME- works like SAME but only generates the header-rewriting rule.

REDIRECT

Causes the connection request to be redirected to a port on the local (firewall) system.

REDIRECT-

The above ACTION (REDIRECT) generates two iptables rules:

1. a header-rewriting rule in the Netfilter “nat” table
2. an ACCEPT rule in the Netfilter “filter” table.

REDIRECT- works like REDIRECT but only generates the header-rewriting rule.

LOG

Log the packet -- requires a syslog level (see below).

QUEUE

Forward the packet to a user-space application. This facility is provided to allow interfacing to ftwall for Kazaa filtering.

Note

When the protocol specified in the PROTO column is TCP (“tcp”, “TCP” or “6”), Shorewall will only pass connection requests (SYN packets) to user space. This is for compatibility with ftwall.

<defined action>

An action defined in the `/etc/shorewall/actions` or `/usr/share/shorewall/actions.std` files.

<macro>

The name of a macro defined using a file with name macro.<name>. Macro files are usually placed in /etc/shorewall but may reside in any directory on the CONFIG_PATH.

The ACTION may optionally be followed by “:” and a syslog level (example: REJECT:info or ACCEPT:debug). This causes the packet to be logged at the specified level prior to being processed according to the specified ACTION. Note: if the ACTION is LOG then you MUST specify a syslog level. The log level may be optionally followed by a *log tag*. A log tag is a string of alphanumeric characters and is specified by following the log level with ":" and the log tag. Example:

```
ACCEPT:info:ftp net     dmz     tcp     21
```

The log tag is appended to the log prefix generated by the LOGPREFIX variable in `/etc/shorewall/conf`. If "ACCEPT:info" generates the log prefix "Shorewall:net2dmz:ACCEPT:" then "ACCEPT:info:ftp" will generate "Shorewall:net2dmz:ACCEPT:ftp " (note the trailing blank). The maximum length of a log prefix supported by iptables is 29 characters; if a larger prefix is generated, Shorewall will issue a warning message and will truncate the prefix to 29 characters.

Specifying a log level for a <*defined action*> will log all invocations of the action. For example:

```
AllowFTP:info          net        dmz
```

will log all net->dmz traffic that has not been handled by earlier rules. That's probably not what you want. If you want to log the FTP connections that are actually accepted, you need to log within the action itself. One way to do that would be to copy `/usr/share/shorewall/action.AllowFTP` to `/etc/shorewall` and modify the copy as follows:

```
#TARGET       SOURCE         DEST            PROTO   DEST    SOURCE          RATE
USER/
#                                                    PORT    PORT(S)         LIMIT
GROUP
ACCEPT:info   -              -               tcp     21
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

The use of DNAT or REDIRECT requires that you have NAT enabled in your kernel configuration.

SOURCE

Describes the source hosts to which the rule applies.. The contents of this field must begin with the name of a zone defined in /etc/shorewall/zones, $FW, “all” or "none". If the ACTION is DNAT or REDIRECT, sub-zones may be excluded from the rule by following the initial zone name with “!” and a comma-separated list of those sub-zones to be excluded. There is an example above.

If the source is "none" then the rule is ignored. This is most commonly used with Shell Variables where a shell variable is set to "none" if a rule is to be omitted.

If the source is not “all” then the source may be further restricted by adding a colon (“:”) followed by a comma-separated list of qualifiers. Qualifiers are may include:

interface name

refers to any connection requests arriving on the specified interface (example loc:eth4). The interface name may optionally be followed by a colon (“:”) and an IP address or subnet (examples: loc:eth4:192.168.4.22, net:eth0:192.0.2.0/24).

IP address

refers to a connection request from the host with the specified address (example net:155.186.235.151). If the ACTION is DNAT, this must not be a DNS name.

MAC Address

in Shorewall format.

subnet

refers to a connection request from any host in the specified subnet (example net:155.186.235.0/24). IP address ranges of the form <*first address*>-<*last address*> may be specified. This requires that your kernel and iptables have iprange

match support.

DEST

Describes the destination host(s) to which the rule applies. May take most of the forms described above for SOURCE plus the following two additional forms:

- An IP address followed by a colon and the port **number** that the server is listening on (service names from /etc/services are not allowed - example loc:192.168.1.3:80).
- A single port number (again, service names are not allowed) -- this form is only allowed if the ACTION is REDIRECT and refers to a server running on the firewall itself and listening on the specified port.

Restrictions:

- MAC addresses may not be specified.
- In DNAT rules, only IP addresses may be given -- DNS names are not permitted.
- You may not specify both an IP address and an interface name in the DEST column.

Like in the SOURCE column, a range of IP addresses may be specified in the DEST column as *<first address>-<last address>*. When the ACTION is DNAT or DNAT-, connections will be assigned to the addresses in the range in a round-robin fashion (load-balancing).

PROTO

Protocol. Must be a protocol name from /etc/protocols, a number, or “all”. Specifies the protocol of the connection request.

DEST PORT(S)

Port or port range (<low port>:<high port>) being connected to. May only be specified if the protocol is tcp, udp or icmp. For icmp, this column's contents are interpreted as an icmp type. If you don't want to specify DEST PORT(S) but need to include information in one of the columns to the right, enter “-” in this column. You may give a list of ports and/or port ranges separated by commas. Port numbers may be either integers or service names from /etc/services.

SOURCE PORTS(S)

May be used to restrict the rule to a particular client port or port range (a port range is specified as <low port number>:<high port number>). If you don't want to restrict client ports but want to specify something in the next column, enter “-” in this column. If you wish to specify a list of port number or ranges, separate the list elements with commas (with no embedded white space). Port numbers may be either integers or service names from /etc/services.

ORIGINAL DEST

This column may only be non-empty if the ACTION is DNAT or REDIRECT.

If DNAT or REDIRECT is the ACTION and the ORIGINAL DEST column is left empty, any connection request arriving at the firewall from the SOURCE that matches the rule will be forwarded or redirected. This works fine for connection requests arriving from the internet where the firewall has only a single external IP address. When the firewall has multiple external IP addresses or when the SOURCE is other than the internet, there will usually be a desire for the rule to only apply to those connection requests directed to particular IP addresses (see Example 2 below for another usage). Those IP addresses are specified in the ORIGINAL DEST column as a comma-separated list.

If this list begins with “!” then the rule will only apply if the original destination address matches none of the addresses listed.

RATE LIMIT

You may rate-limit ACCEPT, DNAT[-], REDIRECT[-] or LOG rules with an entry in this column. Entries have the form

```
<rate>/<interval>[:<burst>]
```

where <rate> is the number of connections per <interval> (“sec” or “min”) and <burst> is the largest burst permitted. If no burst value is given, a value of 5 is assumed.

There may be no whitespace embedded in the specification.

Example 8. Let's take

```
ACCEPT<2/sec:4> net dmz tcp 80
```

The first time this rule is reached, the packet will be accepted; in fact, since the burst is 4, the first four packets will be accepted. After this, it will be 500ms (1 second divided by the rate of 2) before a packet will be accepted from this rule, regardless of how many packets reach it. Also, every 500ms which passes without matching a packet, one of the bursts will be regained; if no packets hit the rule for 2 second, the burst will be fully recharged; back where we started.

Warning

When rate limiting is specified on a rule with “all” in the SOURCE or DEST fields below, the limit will apply to each pair of zones individually rather than as a single limit for all pairs of zones covered by the rule.

If you want to specify any following columns but no rate limit, place “-” in this column.

USER/GROUP

Output rules from the firewall itself may be restricted to a particular user or group.

The column may contain:

```
        [!][<user name or number>][:<group name or number>][+<program name>]
```

When this column is non-empty, the rule applies only if the program generating the output is running under the effective <user> and/or <group> specified (or is NOT running under that id if "!" is given).

Examples:

```
joe     #program must be run by joe
:kids   #program must be run by a member of the 'kids' group
!:kids  #program must not be run by a member of the 'kids' group
+upnpd  #program named upnpd (This feature was removed from Netfilter in kernel
version 2.6.14).
```

Example 9. You wish to forward all ssh connection requests from the internet to local system 192.168.1.3. You wish to limit the number of connections to 4/minute with a burst of 8:

```
#ACTION        SOURCE      DEST              PROTO      DEST PORT(S)
DNAT<4/min:8>  net         loc:192.168.1.3   tcp        ssh
```

Example 10. You want to redirect all local www connection requests EXCEPT those to your own http server (206.124.146.177) to a Squid transparent proxy running on the firewall and listening on port 3128. Squid will of course require access to remote web servers. This example shows yet another use for the ORIGINAL DEST column; here, connection requests that were NOT (notice the “!”) originally destined to 206.124.146.177 are redirected to local port 3128.

```
#ACTION  SOURCE  DEST   PROTO   DEST PORT(S)   SOURCE   ORIGINAL
#                                              PORT(S)  DEST
REDIRECT loc     3128   tcp     www            -        !206.124.146.177
ACCEPT   $FW     net    tcp     www
```

Example 11. You want to run a web server at 155.186.235.222 in your DMZ and have it accessible remotely and locally. the DMZ is managed by Proxy ARP or by classical sub-netting.

```
#ACTION        SOURCE      DEST                PROTO      DEST PORT(S)
ACCEPT         net         dmz:155.186.235.222 tcp        www
ACCEPT         loc         dmz:155.186.235.222 tcp        www
```

Example 12. You want to run wu-ftpd on 192.168.2.2 in your masqueraded DMZ. Your internet interface address is 155.186.235.151 and you want the FTP server to be accessible from the internet in addition to the local 192.168.1.0/24 and dmz 192.168.2.0/24 subnetworks.

Note

since the server is in the 192.168.2.0/24 subnetwork, we can assume that access to the server from that subnet will not involve the firewall (but see FAQ 2)

Note

unless you have more than one external IP address, you can leave the ORIGINAL DEST column blank in the first rule. You cannot leave it blank in the second rule though because then all ftp connections originating in the local subnet 192.168.1.0/24 would be sent to 192.168.2.2 regardless of the site that the user was trying to connect to. That is clearly not what you want.

```
#ACTION  SOURCE             DEST             PROTO  DEST PORT(S)   SOURCE   ORIGINAL
#                                                                   PORT(S)  DEST
DNAT     net                dmz:192.168.2.2  tcp    ftp
DNAT     loc:192.168.1.0/24 dmz:192.168.2.2  tcp    ftp            -
155.186.235.151
```

If you are running wu-ftpd, you should restrict the range of passive in your /etc/ftpaccess file. I only need a few simultaneous FTP sessions so I use port range 65500-65535. In /etc/ftpaccess, this entry is appropriate:

```
passive ports 0.0.0.0/0 65500 65534
```

If you are running pure-ftpd, you would include “-p 65500:65534” on the pure-ftpd runline.

The important point here is to ensure that the port range used for FTP passive connections is unique and will not overlap with any usage on the firewall system.

Example 13. You wish to allow unlimited DMZ access to the host with MAC address 02:00:08:E3:FA:55.

```
#ACTION  SOURCE                  DEST PROTO DEST PORT(S)
ACCEPT   loc:~02-00-08-E3-FA-55  dmz  all
```

Example 14. You wish to allow access to the SMTP server in your DMZ from all zones.

```
#ACTION  SOURCE   DEST PROTO DEST PORT(S)
```

```
ACCEPT   all      dmz  tcp   25
```

Note

When “all” is used as a source or destination, intra-zone traffic is not affected. In this example, if there were two DMZ interfaces then the above rule would NOT enable SMTP traffic between hosts on these interfaces.

Example 15. Your firewall's external interface has several IP addresses but you only want to accept SSH connections on address 206.124.146.176.

```
#ACTION  SOURCE   DEST                PROTO DEST PORT(S)
ACCEPT   net      fw:206.124.146.176 tcp   22
```

Example 16. (For advanced users). From the internet, you with to forward tcp port 25 directed to 192.0.2.178 and 192.0.2.179 to host 192.0.2.177 in your DMZ. You also want to allow access from the internet directly to tcp port 25 on 192.0.2.177.

```
#ACTION  SOURCE   DEST             PROTO  DEST PORT(S)   SOURCE   ORIGINAL
#                                                       PORT(S)  DEST
DNAT-    net      dmz:192.0.2.177  tcp    25             -        192.0.2.178
DNAT-    net      dmz:192.0.2.177  tcp    25             -        192.0.2.179
ACCEPT   net      dmz:192.0.2.177  tcp    25
```

Using “DNAT-” rather than “DNAT” avoids two extra copies of the third rule from being generated.

Example 17. You have 9 http servers behind a Shorewall firewall and you want connection requests to be distributed among your servers. The servers are 192.168.1.101-192.168.1.109.

```
#ACTION  SOURCE   DEST                            PROTO DEST PORT(S)
DNAT     net      loc:192.168.1.101-192.168.1.109 tcp   80
```

Example 18. You want to redirect all local www connection requests EXCEPT those from 192.168.1.4 and 192.168.1.199 to a Squid transparent proxy running on the firewall and listening on port 3128.

```
#ACTION  SOURCE  DEST   PROTO   DEST PORT(S)   SOURCE   ORIGINAL
#                                              PORT(S)  DEST
NONAT    loc:192.168.1.4,192.168.1.199 \
                 net    tcp     www
REDIRECT loc     3128   tcp     www            -
ACCEPT   $FW     net    tcp     www
```

The reason that NONAT is used in the above example rather than ACCEPT+ is that the example is assuming the usual ACCEPT loc->net policy. Since traffic from the local zone to the internet zone is accepted anyway, adding an additional ACCEPT rule is unnecessary and all that is required is to avoid the REDIRECT rule for HTTP connection requests from the two listed IP addresses.

Look here for information on other services.

/etc/shorewall/masq

The /etc/shorewall/masq file is used to define classical IP Masquerading and Source Network Address Translation (SNAT). There is one entry in the file for each subnet that you want to masquerade. In order to make use of this feature, you must have

NAT enabled.

Columns are:

INTERFACE

The interface that will masquerade the subnet; this is normally your internet interface. This interface name can be optionally qualified by adding “:” and a subnet or host IP. When this qualification is added, only packets addressed to that host or subnet will be masqueraded. The interface name can be qualified with ":" followed by a comma separated list of hosts and/or subnets. If this list begins with “!” (e.g., “eth0:!192.0.2.8/29,192.0.2.32/29”) then only packets addressed to destinations **not** listed will be masqueraded; otherwise (e.g., “eth0:192.0.2.8/29,192.0.2.32/29”), traffic will be masqueraded if it **does** match one of the listed addresses.

If you have set ADD_SNAT_ALIASES=Yes in /etc/shorewall/shorewall.conf, you can cause Shorewall to create an alias *label* of the form *interfacename:digit* (e.g., eth0:0) by placing that label in this column. See example 5 below. Alias labels created in this way allow the alias to be visible to the ipconfig utility. **THAT IS THE ONLY THING THAT THIS LABEL IS GOOD FOR AND IT MAY NOT APPEAR ANYWHERE ELSE IN YOUR SHOREWALL CONFIGURATION.**

Normally MASQUERADE/SNAT rules are evaluated after one-to-one NAT rules defined in the `/etc/shorewall/nat` file. If you preceed the interface name with a plus sign ("+") then the rule will be evaluated before one-to-one NAT.

Examples:

```
+eth0
+eth1:192.0.2.32/27
```

The effect of ADD_SNAT_ALIASES=Yes can be negated for an entry by following the interface name by ":" but no digit.

Examples:

```
eth0:
eth1::192.0.2.32/27
+eth3
```

SUBNET

The subnet that you want to have masqueraded through the INTERFACE. This may be expressed as a single IP address, a subnet or an interface name. In the latter instance, the interface must be configured and started before Shorewall is started as Shorewall will determine the subnet based on information obtained from the “ip” utility.

The subnet may be optionally followed by “!” and a comma-separated list of addresses and/or subnets that are to be excluded from masquerading.

ADDRESS

The source address to be used for outgoing packets. This column is optional and if left blank, the current primary IP address of the interface in the first column is used. If you have a static IP on that interface, listing it here makes processing of output packets a little less expensive for the firewall. If you specify an address in this column, it must be an IP address configured on the INTERFACE or you must have ADD_SNAT_ALIASES enabled in /etc/shorewall/shorewall.conf. You may include a range of IP addresses in this column to indicate that Netfilter should use the addresses in the range in round-robin fashion. Beginning with Shorewall version 1.4.7, you may include a list of ranges and/or addresses in this column; again, Netfilter will use all listed ranges/addresses in rounded-robin fashion.

You may also specify the source port range to be used (the PROTO column must specify tcp or udp) by following the address or address range if any with ":" and the port range (in the format <*low port*>-<*high port*>).

Examples:

```
#INTERFACE SUBNET          ADDRESS                PROTO
eth0       10.0.0.0/8      192.0.2.44:7000-8000   udp

#INTERFACE SUBNET          ADDRESS                PROTO
eth0       192.168.1.0/24  :4000-5000             tcp
```

Some internet application that establish multiple connections from a client assume that when SNAT is being used that all connections between the client and a particular client and a remote server will appear to the server to come from the same external IP address. You can ensure that this is the case by preceding the ADDRESS range by "SAME:".

Example:

```
#INTERFACE SUBNET          ADDRESS
eth0       10.0.0.0/8      SAME:192.0.2.44-192.168.2.50
```

If you want all connections from an internal system to use the same external IP address regardless of the remote server that they are connecting to then precede the ADDRESS range by "SAME:nodst:".

Example:

```
#INTERFACE SUBNET          ADDRESS
eth0       10.0.0.0/8      SAME:nodst:192.0.2.44-192.168.2.50
```

PROTO

If specified, must be a protocol number of a protocol name from /etc/protocols. Restricts the SNAT or Masquerade to that protocol.

PORT(S)

If the PROTO column specifies TCP (6) or UDP (17) then this column may be used to restrict to SNAT or Masquerade to traffic with a certain destination port or a set of destination ports. The column may contain:

- A port number or a port name from /etc/services.
- A comma-separated list of port numbers and/or port names. Your kernel must have Multiport match support. You can tell if your kernel has this support by issuing a **shorewall check** command and looking at the output under “Shorewall has detected the following iptables/netfilter capabilities:”.
- A range of port numbers of the form <*low port*>:<*high port*>

IPSEC

If you specify a value other than "-" in this column, you must be running kernel 2.6 and your kernel and iptables must include policy match support.

The value in this column is a comma-separated list of options from the following. Only packets that will be encrypted via an SA that matches these options will have their source address changed.

- Yes or yes • Match any SA. Normally used as the only option.
- reqid=<*number*> where <*number*> is specified using setkey(8) using the 'unique:<*number*>' option for the SPD level.
- spi=<*number*> where <*number*> is the SPI of the SA.
- proto=ah|esp|ipcomp
- mode=transport|tunnel
- tunnel-src=<*address*>[/<*mask*>] (only available with mode=tunnel)
- tunnel-dst=<*address*>[/<*mask*>] (only available with mode=tunnel)
- strict — Means that packets must match all rules.

- next — Separates rules; can only be used with strict.

Example 19. You have eth0 connected to a cable modem and eth1 connected to your local subnetwork 192.168.9.0/24. Your /etc/shorewall/masq file would look like:

```
#INTERFACE          SUBNET           ADDRESS
eth0                192.168.9.0/24
```

Example 20. You have a number of IPSEC tunnels through ipsec0 and you want to masquerade traffic from your 192.168.9.0/24 subnet to the remote subnet 10.1.0.0/16 only.

```
#INTERFACE          SUBNET           ADDRESS
ipsec0:10.1.0.0/16  192.168.9.0/24
```

Example 21. You have a DSL line connected on eth0 and a local network (192.168.10.0/24) connected to eth1. You want all local->net connections to use source address 206.124.146.176.

```
#INTERFACE          SUBNET           ADDRESS
eth0                192.168.10.0/24  206.124.146.176
```

Example 22. Same as example 3 except that you wish to exclude 192.168.10.44 and 192.168.10.45 from the SNAT rule.

```
#INTERFACE SUBNET                                      ADDRESS
eth0       192.168.10.0/24!192.168.10.44,192.168.10.45 206.124.146.176
```

Example 23. You have a second IP address (206.124.146.177) assigned to you and wish to use it for SNAT of the subnet 192.168.12.0/24. You want to give that address the name eth0:0. You must have ADD_SNAT_ALIASES=Yes in /etc/shorewall/shorewall.conf.

```
#INTERFACE           SUBNET           ADDRESS
eth0:0               192.168.12.0/24  206.124.146.177
```

Example 24. You want to use both 206.124.146.177 and 206.124.146.179 for SNAT of the subnet 192.168.12.0/24. Each address will be used on alternate outbound connections.

```
#INTERFACE            SUBNET           ADDRESS
eth0                  192.168.12.0/24 206.124.146.177,206.124.146.179
```

Example 25. You want all outgoing SMTP traffic entering the firewall on eth1 to be sent from eth0 with source IP address 206.124.146.177. You want all other outgoing traffic from eth1 to be sent from eth0 with source IP address 206.124.146.176.

```
#INTERFACE            SUBNET           ADDRESS            PROTO  PORT(S)
eth0                  eth1             206.124.146.177    tcp    25
eth0                  eth1             206.124.146.176
```

Note that the order of the entries in the above example is important.

/etc/shorewall/proxyarp

If you want to use proxy ARP on an entire sub-network, I suggest that you look at the Proxy ARP Subnet Mini HOWTO. If

you decide to use the technique described in that HOWTO, you can set the proxy_arp flag for an interface (`/proc/sys/net/ipv4/conf/`*<interface>*`/proxy_arp`) by including the **proxyarp** option in the interface's record in /etc/shorewall/interfaces. When using Proxy ARP sub-netting, you do **NOT** include any entries in /etc/shorewall/proxyarp.

The `/etc/shorewall/proxyarp` file is used to define Proxy ARP. The file is typically used for enabling Proxy ARP on a small set of systems since you need one entry in this file for each system using proxy ARP. Columns are:

ADDRESS

address of the system.

INTERFACE

the interface that connects to the system. If the interface is obvious from the subnetting, you may enter “-” in this column.

EXTERNAL

the external interface that you want to honor ARP requests for the ADDRESS specified in the first column.

HAVEROUTE

If you already have a route through INTERFACE to ADDRESS, this column should contain “Yes” or “yes”. If you want Shorewall to add the route, the column should contain “No” or “no”.

PERSISTENT

If you specify "No" or "no" in the HAVEROUTE column, Shorewall will automatically add a route to the host in the ADDRESS column through the interface in the INTERFACE column. If you enter “No” or “no” in the PERSISTENT column or if you leave the column empty, that route will be deleted if you issue a **shorewall stop** or **shorewall clear** command. If you place “Yes” or “yes” in the PERSISTENT column, then those commands will not cause the route to be deleted.

Note

After you have made a change to the `/etc/shorewall/proxyarp file`, you may need to flush the ARP cache of all routers on the LAN segment connected to the interface specified in the EXTERNAL column of the change/added entry(s). If you are having problems communicating between an individual host (A) on that segment and a system whose entry has changed, you may need to flush the ARP cache on host A as well.

ISPs typically have ARP configured with long TTL (hours!) so if your ISPs router has a stale cache entry (as seen using “tcpdump -nei <external interface> host <IP addr>”), it may take a long while to time out. I personally have had to contact my ISP and ask them to delete a stale entry in order to restore a system to working order after changing my proxy ARP settings.

Example 26. You have public IP addresses 155.182.235.0/28. You configure your firewall as follows:

```
eth0 - 155.186.235.1 (internet connection) eth1 -
      192.168.9.0/24 (masqueraded local systems) eth2 - 192.168.10.1
      (interface to your DMZ)
```

In your DMZ, you want to install a Web/FTP server with public address 155.186.235.4. On the Web server, you subnet just like the firewall's eth0 and you configure 155.186.235.1 as the default gateway. In your `/etc/shorewall/proxyarp` file, you will have:

```
#ADDRESS        INTERFACE        EXTERNAL          HAVEROUTE
155.186.235.4   eth2             eth0              NO
```

Tip

You may want to configure the servers in your DMZ with a subnet that is smaller than the subnet of your internet interface. See the Proxy ARP Subnet Mini HOWTO for details. In this case you will want to place “Yes” in the HAVEROUTE column.

/etc/shorewall/nat

The `/etc/shorewall/nat` file is used to define one-to-one NAT. There is one entry in the file for each one-to-one NAT relationship that you wish to define. In order to make use of this feature, you must have NAT enabled.

Important

If all you want to do is forward ports to servers behind your firewall, you do NOT want to use one-to-one NAT. Port forwarding can be accomplished with simple entries in the rules file. Also, in most cases Proxy ARP provides a superior solution to one-to-one NAT because the internal systems are accessed using the same IP address internally and externally.

Columns in an entry are:

EXTERNAL

External IP address

Caution

This should NOT be the primary IP address of the interface named in the next column.

INTERFACE

Interface that you want the EXTERNAL IP address to appear on. If you have set ADD_IP_ALIASES=Yes in /etc/shorewall/shorewall.conf, you can specify an alias label of the form *interfacename:digit* (e.g., eth0:0) and Shorewall will create the alias with that label. Alias labels created in this way allow the alias to be visible to the ipconfig utility. **THAT IS THE ONLY THING THAT THIS LABEL IS GOOD FOR AND IT MAY NOT APPEAR ANYWHERE ELSE IN YOUR SHOREWALL CONFIGURATION.**

The effect of ADD_IP_ALIASES=Yes can be negated for an entry by following the interface name by ":" but no digit.

Example:

```
eth0:
```

INTERNAL

Internal IP address.

ALL INTERFACES

If “Yes” or “yes”, NAT will be effective from all hosts. If “No” or “no” (or if left empty) then NAT will be effective only through the interface named in the INTERFACE column.

LOCAL

If Yes or yes, NAT will be effective from the firewall system. Note that with Shorewall 2.0.1 and earlier versions, this column was ignored if the ALL INTERFACES column did not contain "Yes" or "yes". This column's contents are independent of the value in ALL INTERFACES.

Look here for additional information and an example.

/etc/shorewall/tunnels

The /etc/shorewall/tunnels file allows you to define IPSec, GRE, IPIP, OpenVPN, PPTP, 6to4 and other tunnels with end-points on your firewall.

For an overview of Shorewall's VPN support, try this article.

Instructions for setting up IPSEC tunnels may be found here (if you are using kernel 2.6 with native IPSEC support, look here), instructions for IPIP and GRE tunnels are here, instructions for OpenVPN tunnels are here, instructions for PPTP tunnels are here, instructions for 6to4 tunnels are here, and instructions for integrating Shorewall with other types of tunnels are here.

/etc/shorewall/shorewall.conf

This file is used to set the following firewall parameters:

FASTACCEPT

> Normally, Shorewall accepting ESTABLISHED/RELATED packets until these packets reach the chain in which the original connection was accepted. So for packets going from the 'loc' zone to the 'net' zone, ESTABLISHED/RELATED packets are ACCEPTED in the 'loc2net' chain.
>
> If you set FASTACCEPT=Yes, then ESTABLISHED/RELEATED packets are accepted early in the INPUT, FORWARD and OUTPUT chains. If you set FASTACCEPT=Yes then you may not include rules in the ESTABLISHED or RELATED sections of /etc/shorewall/rules.

STARTUP_ENABLED

> When set to Yes or yes, Shorewall may be started. Used as guard against Shorewall being accidentally started before it has been configured.

MACLIST_TTL

> The performance of configurations with a large numbers of entries in /etc/shorewall/maclist can be improved by setting the MACLIST_TTL variable in /etc/shorewall/shorewall.conf.
>
> If your iptables and kernel support the "Recent Match" (see the output of "shorewall check" near the top), you can cache the results of a 'maclist' file lookup and thus reduce the overhead associated with MAC Verification.
>
> When a new connection arrives from a 'maclist' interface, the packet passes through then list of entries for that interface in /etc/shorewall/maclist. If there is a match then the source IP address is added to the 'Recent' set for that interface. Subsequent connection attempts from that IP address occurring within $MACLIST_TTL seconds will be accepted without having to scan all of the entries. After $MACLIST_TTL from the first accepted connection request from an IP address, the next connection request from that IP address will be checked against the entire list.
>
> If MACLIST_TTL is not specified or is specified as empty (e.g, MACLIST_TTL="" or is specified as zero then 'maclist' lookups will not be cached).

MACLIST_TABLE

> Normally, MAC verification occurs in the filter table (INPUT and FORWARD) chains. When forwarding a packet from an interface with MAC verification to a bridge interface, that doesn't work.

This problem can be worked around by setting MACLIST_TABLE=mangle which will cause Mac verification to occur out of the PREROUTING chain. Because REJECT isn't available in that environment, you may not specify MACLIST_DISPOSITION=REJECT with MACLIST_TABLE=mangle.

RFC1918_STRICT

Traditionally, the RETURN target in the 'rfc1918' file has caused norfc1918 processing to cease for a packet if the packet's source IP address matches the rule. Thus, if you have this entry in /etc/shorewall/rfc1918:

```
#SUBNETS                    TARGET
192.168.1.0/24              RETURN
```

then traffic from 192.168.1.4 to 10.0.3.9 will be accepted even though you also have:

```
#SUBNETS                    TARGET
10.0.0.0/8                  logdrop
```

Setting RFC1918_STRICT=Yes in shorewall.conf will cause such traffic to be logged and dropped since while the packet's source matches the RETURN rule, the packet's destination matches the 'logdrop' rule.

If not specified or specified as empty (e.g., RFC1918_STRICT="") then RFC1918_STRICT=No is assumed.

Warning

RFC1918_STRICT=Yes requires that your kernel and iptables support 'Connection Tracking' match.

LOGALLNEW

When set to a log level, this option causes Shorewall to generate a logging rule as the first rule in each builtin chain.

- The table name is used as the chain name in the log prefix.
- The chain name is used as the target in the log pref

Example: Using the default LOGFORMAT, the log prefix for logging from the nat table's PREROUTING chain is:

```
Shorewall:nat:PREROUTING
```

Important

There is no rate limiting on these logging rules so use LOGALLNEW at your own risk; it may cause high CPU and disk utilization and you may not be able to control your firewall after you enable this option.

Caution

DO NOT USE THIS OPTION IF THE RESULTING LOG MESSAGES WILL BE SENT TO ANOTHER SYSTEM.

DYNAMIC_ZONES

When set to Yes or yes, enables dynamic zones.

CONFIG_PATH

Specifies where configuration files other than `shorewall.conf` may be found. CONFIG_PATH is specifies as a list of directory names separated by colons (":"). When looking for a configuration file other than shorewall.conf:

- If the command is "try" or if "-c <configuration directory>" was specified in the command then the directory given in the command is searched first.
- Next, each directory in the CONFIG_PATH setting is searched in sequence.

If CONFIG_PATH is not given or if it is set to the empty value then the contents of `/usr/share/shorewall/configpath` are used. As released from shorewall.net, that file sets the CONFIG_PATH to **/etc/shorewall:/usr/share/shorewall** but your particular distribution may set it differently.

Note that the setting in `/usr/share/shorewall/configpath` is always used to locate `shorewall.conf`.

PKTTYPE

Normally Shorewall attempts to use the iptables packet type match extension to determine broadcast and multicast packets.

1. This can cause a message to appear during **shorewall start** (modprobe: cant locate module ipt_pkttype).
2. Some users have found problems with the packet match extension with the result that their firewall log is flooded with messages relating to broadcast packets.

If you are experiencing either of these problems, setting PKTTYPE=No will prevent Shorewall from trying to use the packet type match extension and to use IP address matching to determine which packets are broadcasts or multicasts.

RESTOREFILE

The simple name of a file in `/var/lib/shorewall` to be used as the default restore script in the shorewall save, shorewall restore, shorewall forget and shorewall -f start commands. See the Saved Configuration documentation for details.

BRIDGING

When set to Yes or yes, enables Shorewall Bridging support.

SMURF_LOG_LEVEL

Specifies the logging level for smurf packets (see the **nosmurfs** option in /etc/shorewall/interfaces). If set to the empty value (SMURF_LOG_LEVEL="") then smurfs are not logged.

MODULE_SUFFIX

The value of this variable determines the possible file extensions of kernel modules. The default value is "o gz ko and o.gz". See /etc/shorewall/modules for more details.

ADMINISABSENTMINDED

The value of this variable affects Shorewall's stopped state. When ADMINISABSENTMINDES=No, only traffic to/from those addresses listed in /etc/shorewall/routestopped is accepted when Shorewall is stopped. When ADMINISABSENTMINDED=Yes, in addition to traffic to/from addresses in `/etc/shorewall/routestopped`, connections that were active when Shorewall stopped continue to work and all new connections from the firewall system itself are allowed. If this variable is not set or is given the empty value then ADMINISABSENTMINDED=No is assumed.

SHOREWALL_SHELL

This parameter is used to specify the shell program to be used to interpret the firewall script (/usr/share/shorewall/firewall). If not specified or specified as a null value, /bin/sh is assumed.

IPTABLES

This parameter names the iptables executable to be used by Shorewall. If not specified or if specified as a null value, then the iptables executable located using the PATH option is used.

LOGFORMAT

The value of this variable generate the --log-prefix setting for Shorewall logging rules. It contains a “printf” formatting template which accepts three arguments (the chain name, logging rule number (optional) and the disposition). To use LOGFORMAT with fireparse, set it as:

```
LOGFORMAT="fp=%s:%d a=%s "
```

If the LOGFORMAT value contains the substring “%d” then the logging rule number is calculated and formatted in that position; if that substring is not included then the rule number is not included. If not supplied or supplied as empty (LOGFORMAT="") then “Shorewall:%s:%s:” is assumed.

Caution

/sbin/shorewall uses the leading part of the LOGFORMAT string (up to but not including the first “%”) to find log messages in the “show log”, “status” and “hits” commands. This part should not be omitted (the LOGFORMAT should not begin with “%”) and the leading part should be sufficiently unique for **/sbin/shorewall** to identify Shorewall messages.

CLEAR_TC

If this option is set to “No” then Shorewall won't clear the current traffic control rules during [re]start. This setting is intended for use by people that prefer to configure traffic shaping when the network interfaces come up rather than when the firewall is started. If that is what you want to do, set TC_ENABLED=Yes and CLEAR_TC=No and do not supply an `/etc/shorewall/tcstart` file. That way, your traffic shaping rules can still use the “fwmark” classifier based on packet marking defined in /etc/shorewall/tcrules. If not specified, CLEAR_TC=Yes is assumed.

MARK_IN_FORWARD_CHAIN

If your kernel has a FORWARD chain in the mangle table, you may set MARK_IN_FORWARD_CHAIN=Yes to cause the marking specified in the tcrules file to occur in that chain rather than in the PREROUTING chain. This permits you to mark inbound traffic based on its destination address when SNAT or Masquerading are in use. To determine if your kernel has a FORWARD chain in the mangle table, use the “**/sbin/shorewall show mangle**” command; if a FORWARD chain is displayed then your kernel will support this option. If this option is not specified or if it is given the empty value (e.g., MARK_IN_FORWARD_CHAIN="") then MARK_IN_FORWARD_CHAIN=No is assumed.

RFC1918_LOG_LEVEL

This parameter determines the level at which packets logged under the “norfc1918” mechanism are logged. The value must be a valid syslog level and if no level is given, then info is assumed.

TCP_FLAGS_DISPOSITION

Determines the disposition of TCP packets that fail the checks enabled by the tcpflags interface option and must have a value of ACCEPT (accept the packet), REJECT (send an RST response) or DROP (ignore the packet). If not set or if set to the empty value (e.g., TCP_FLAGS_DISPOSITION="") then TCP_FLAGS_DISPOSITION=DROP is assumed.

TCP_FLAGS_LOG_LEVEL

Determines the syslog level for logging packets that fail the checks enabled by the tcpflags interface option. The value must be a valid syslogd log level. If you don't want to log these packets, set to the empty value (e.g., TCP_FLAGS_LOG_LEVEL="").

MACLIST_DISPOSITION

Determines the disposition of connections requests that fail MAC Verification and must have the value ACCEPT (accept the connection request anyway), REJECT (reject the connection request) or DROP (ignore the connection request). If not set or if set to the empty value (e.g., MACLIST_DISPOSITION="") then MACLIST_DISPOSITION=REJECT is assumed.

MACLIST_LOG_LEVEL

Determines the syslog level for logging connection requests that fail MAC Verification. The value must be a valid syslogd log level. If you don't want to log these connection requests, set to the empty value (e.g., MACLIST_LOG_LEVEL="").

LOG_MARTIANS

If set to Yes or yes, sets `/proc/sys/net/ipv4/conf/all/log_martians` and `/proc/sys/net/ipv4/conf/default/log_martians` to 1. Default is which sets both of the above to zero. If you do not enable martian logging for all interfaces, you may still enable it for individual interfaces using the **logmartians** interface option in /etc/shorewall/interfaces.

DETECT_DNAT_ADDRS

If set to “Yes” or “yes”, Shorewall will detect the first IP address of the interface to the source zone and will include this address in DNAT rules as the original destination IP address. If set to “No” or “no”, Shorewall will not detect this address and any destination IP address will match the DNAT rule. If not specified or empty, “DETECT_DNAT_ADDRS=Yes” is assumed.

NAT_BEFORE_RULES

If set to “No” or “no”, port forwarding rules can override the contents of the /etc/shorewall/nat file. If set to “Yes” or “yes”, port forwarding rules cannot override one-to-one NAT. If not set or set to an empty value, “Yes” is assumed.

SUBSYSLOCK

This parameter should be set to the name of a file that the firewall should create if it starts successfully and remove when it stops. Creating and removing this file allows Shorewall to work with your distribution's initscripts. For RedHat, this should be set to /var/lock/subsys/shorewall. For Debian, the value is /var/state/shorewall and in LEAF it is /var/run/shorwall. Example: SUBSYSLOCK=/var/lock/subsys/shorewall.

MODULESDIR

This parameter specifies the directory where your kernel netfilter modules may be found. If you leave the variable empty, Shorewall will supply the value "/lib/modules/`uname -r`/kernel/net/ipv4/netfilter.

LOGRATE and LOGBURST

These parameters set the match rate and initial burst size for logged packets. Please see the iptables man page for a description of the behavior of these parameters (the iptables option --limit is set by LOGRATE and --limit-burst is set by LOGBURST). If both parameters are set empty, no rate-limiting will occur.

Example 27.

```
LOGRATE=10/minute
LOGBURST=5
```

For each logging rule, the first time the rule is reached, the packet will be logged; in fact, since the burst is 5, the first five packets will be logged. After this, it will be 6 seconds (1 minute divided by the rate of 10) before a message will be logged from the rule, regardless of how many packets reach it. Also, every 6 seconds which passes without matching a packet, one of the bursts will be regained; if no packets hit the rule for 30 seconds, the burst will be fully recharged; back where we started.

LOGFILE

This parameter tells the /sbin/shorewall program where to look for Shorewall messages when processing the “show log”, “monitor”, “status” and “hits” commands. If not assigned or if assigned an empty value, /var/log/messages is assumed.

IP_FORWARDING

This parameter determines whether Shorewall enables or disables IPV4 Packet Forwarding (/proc/sys/net/ipv4/ip_forward). Possible values are:

On or on

packet forwarding will be enabled.

Off or off

packet forwarding will be disabled.

Keep or keep

Shorewall will neither enable nor disable packet forwarding.

If this variable is not set or is given an empty value (IP_FORWARD="") then IP_FORWARD=On is assumed.

ADD_IP_ALIASES

This parameter determines whether Shorewall automatically adds the *external* address(es) in /etc/shorewall/nat. If the variable is set to “Yes” or “yes” then Shorewall automatically adds these aliases. If it is set to “No” or “no”, you must add these aliases yourself using your distribution's network configuration tools.

If this variable is not set or is given an empty value (ADD_IP_ALIASES="") then ADD_IP_ALIASES=Yes is assumed.

Warning

Addresses added by ADD_IP_ALIASES=Yes are deleted and re-added during **shorewall restart**. As a consequence, connections using those addresses may be severed.

ADD_SNAT_ALIASES

This parameter determines whether Shorewall automatically adds the SNAT *ADDRESS* in /etc/shorewall/masq. If the variable is set to “Yes” or “yes” then Shorewall automatically adds these addresses. If it is set to “No” or “no”, you must add these addresses yourself using your distribution's network configuration tools.

If this variable is not set or is given an empty value (ADD_SNAT_ALIASES="") then ADD_SNAT_ALIASES=No is assumed.

Warning

Addresses added by ADD_SNAT_ALIASES=Yes are deleted and re-added during **shorewall restart**. As a consequence, connections using those addresses may be severed.

RETAIN_ALIASES

During "shorewall start", IP addresses to be added as a consequence of ADD_IP_ALIASES=Yes and ADD_SNAT_ALIASES=Yes are quietly deleted when /etc/shorewall/nat and /etc/shorewall/masq are processed then are re-added later. This is done to help ensure that the addresses can be added with the specified labels but can have the undesirable side effect of causing routes to be quietly deleted. When RETAIN_ALIASES is set to Yes, existing addresses will not be deleted. Regardless of the setting of RETAIN_ALIASES, addresses added during "shorewall start" are still deleted at a subsequent "shorewall stop" or "shorewall restart".

LOGUNCLEAN

This parameter determines the logging level of mangled/invalid packets controlled by the “dropunclean" and "logunclean” interface options. If LOGUNCLEAN is empty (LOGUNCLEAN=) then packets selected by “dropclean” are dropped silently (“logunclean” packets are logged under the “info” log level). Otherwise, these packets are logged at the specified level (Example: LOGUNCLEAN=debug).

BLACKLIST_DISPOSITION

This parameter determines the disposition of packets from blacklisted hosts. It may have the value DROP if the packets are to be dropped or REJECT if the packets are to be replied with an ICMP port unreachable reply or a TCP RST (tcp only). If you do not assign a value or if you assign an empty value then DROP is assumed.

BLACKLIST_LOGLEVEL

This parameter determines if packets from blacklisted hosts are logged and it determines the syslog level that they are to be logged at. Its value is a syslog level (Example: BLACKLIST_LOGLEVEL=debug). If you do not assign a value or if you assign an empty value then packets from blacklisted hosts are not logged.

DELAYBLACKLISTLOAD

Users with a large static black list (`/etc/shorewall/blacklist`) may want to set the DELAYBLACKLISTLOAD option to Yes. When DELAYBLACKLISTLOAD=Yes, Shorewall will enable new connections before loading the blacklist rules. While this may allow connections from blacklisted hosts to slip by during construction of the blacklist, it can substantially reduce the time that all new connections are disabled during **shorewall [re]start**.

CLAMPMSS[=<value>]

This parameter enables the TCP Clamp MSS to PMTU feature of Netfilter and is usually required when your internet connection is through PPPoE or PPTP. If set to “Yes” or “yes”, the feature is enabled. If left blank or set to “No” or “no”, the feature is not enabled.

Note

This option requires CONFIG_IP_NF_TARGET_TCPMSS in your kernel.

You may also set CLAMPMSS to a numeric value (e.g., CLAMPMSS=1400). This will set the MSS field in TCP SYN packets going through the firewall to the value that you specify.

ROUTE_FILTER

If this parameter is given the value “Yes” or “yes” then route filtering (anti-spoofing) is enabled on all network interfaces which are brought up while Shorewall is in the started state. The default value is “no”.

/etc/shorewall/modules Configuration

The file `/etc/shorewall/modules` contains commands for loading the kernel modules required by Shorewall-defined firewall rules. Shorewall will source this file during start/restart provided that it exists and that the directory specified by the MODULESDIR parameter exists (see /etc/shorewall/shorewall.conf above).

The file that is released with Shorewall calls the Shorewall function “loadmodule” for the set of modules that I load.

The *loadmodule* function is called as follows:

```
loadmodule <modulename> [ <module parameters> ]
```

where

<modulename>

is the name of the modules without the trailing “.o” (example ip_conntrack).

<module parameters>

Optional parameters to the insmod utility.

The function determines if the module named by <*modulename*> is already loaded and if not then the function determines if the “.o” file corresponding to the module exists in the <*moduledirectory*>; if so, then the following command is executed:

```
insmod <moduledirectory>/<modulename>.o <module parameters>
```

If the file doesn't exist, the function determines of the “.o.gz” file corresponding to the module exists in the *moduledirectory*. If it does, the function assumes that the running configuration supports compressed modules and execute the following command:

```
insmod <moduledirectory>/<modulename>.o.gz <module parameters>
```

The value of the MODULE_SUFFIX option in determines which files the loadmodule function looks for if the named module doesn't exist. For each file <*extension*> listed in MODULE_SUFFIX (default "o gz ko o.gz"), the function will append a period (".") and the extension and if the resulting file exists then the following command will be executed:

```
insmod moduledirectory/<modulename>.<extension> <module parameters>
```

/etc/shorewall/tos Configuration

The `/etc/shorewall/tos` file allows you to set the Type of Service field in packet headers based on packet source, packet destination, protocol, source port and destination port. In order for this file to be processed by Shorewall, you must have mangle support enabled.

Entries in the file have the following columns:

SOURCE

The source zone. May be qualified by following the zone name with a colon (“:”) and either an IP address, an IP subnet, a MAC address in Shorewall Format or the name of an interface. This column may also contain the name of the firewall zone to indicate packets originating on the firewall itself or “all” to indicate any source.

DEST

The destination zone. May be qualified by following the zone name with a colon (“:”) and either an IP address or an IP subnet. Because packets are marked prior to routing, you may not specify the name of an interface. This column may also contain “all” to indicate any destination.

PROTOCOL

The name of a protocol in `/etc/protocols` or the protocol's number.

SOURCE PORT(S)

The source port or a port range. For all ports, place a hyphen (“-”) in this column.

DEST PORT(S)

The destination port or a port range. To indicate all ports, place a hyphen (“-”) in this column.

TOS

The type of service. Must be one of the following:

Minimize-Delay (16)
Maximize-Throughput (8)
Maximize-Reliability (4)

Minimize-Cost (2)

Normal-Service (0)

Here's a sample `/etc/shorewall/tos` file:

```
#SOURCE  DEST    PROTOCOL    SOURCE PORTS(S)   DEST PORTS(S)   TOS
all      all     tcp         -                 ssh             16
all      all     tcp         ssh               -               16
all      all     tcp         -                 ftp             16
all      all     tcp         ftp               -               16
all      all     tcp         -                 ftp-data        8
all      all     tcp         ftp-data          -               8
```

Warning

Users have reported that odd routing problems result from adding the ESP and AH protocols to the `/etc/shorewall/tos` file.

/etc/shorewall/blacklist

Each line in `/etc/shorewall/blacklist` contains an IP address, a MAC address in Shorewall Format or subnet address.

Example 28.

```
130.252.100.69
206.124.146.0/24
```

Packets **from** hosts listed in the blacklist file will be disposed of according to the value assigned to the BLACKLIST_DISPOSITION and BLACKLIST_LOGLEVEL variables in /etc/shorewall/shorewall.conf. Only packets arriving on interfaces that have the "blacklist" option in `/etc/shorewall/interfaces` are checked against the blacklist. The black list is designed to prevent listed hosts/subnets from accessing services on **your** network.

The blacklist file has three columns:

ADDRESS/SUBNET

As described above.

PROTOCOL

Optional. If specified, only packets specifying this protocol will be blocked.

PORTS

Optional; may only be given if PROTOCOL is tcp, udp or icmp. Expressed as a comma-separated list of destination port numbers or service names (from /etc/services). If present, only packets matching the specified protocol and one of the listed destination ports are blocked. When the PROTOCOL is icmp, the PORTS column contains a comma-separated list of ICMP type numbers or names (see "iptables -h icmp").

Shorewall also has a dynamic blacklist capability.

Important

The Shorewall blacklist file is **NOT** designed to police your users' web browsing -- to do that, I suggest that you install and configure Squid with SquidGuard.

/usr/share/shorewall/rfc1918

This file lists the subnets affected by the norfc1918 interface option. Columns in the file are:

SUBNET

The subnet using VLSM notation (e.g., 192.168.0.0/16).

TARGET

What to do with packets to/from the SUBNET:

RETURN

Process the packet normally thru the rules and policies. See also RFC1918_STRICT above.

DROP

Silently drop the packet.

logdrop

Log then drop the packet -- see the RFC1918_LOG_LEVEL parameter above.

If you want to modify this file, DO NOT MODIFY `/usr/share/shorewall/rfc1918`. Rather copy that file to `/etc/shorewall/rfc1918` and modify the copy.

/etc/shorewall/netmap

Network mapping is defined using the `/etc/shorewall/netmap` file. Columns in this file are:

TYPE

Must be DNAT or SNAT.

If DNAT, traffic entering INTERFACE and addressed to NET1 has it's destination address rewritten to the corresponding address in NET2.

If SNAT, traffic leaving INTERFACE with a source address in NET1 has it's source address rewritten to the corresponding address in NET2.

NET1

Must be expressed in CIDR format (e.g., 192.168.1.0/24).

INTERFACE

A firewall interface. This interface must have been defined in `/etc/shorewall/interfaces`.

NET2

A second network expressed in CIDR format.

For more information, see the Network Mapping documentation.

/etc/shorewall/routestopped

This file defines the hosts that are accessible from the firewall when the firewall is stopped. Entries in this file are also active while Shorewall is being stopped or [re]started.

Columns in the file are:

INTERFACE

The firewall interface through which the host(s) communicate with the firewall.

HOST(S) - (Optional)

A comma-separated list of IP/Subnet addresses. If not supplied or supplied as “-” then 0.0.0.0/0 is assumed.

Example 29. When your firewall is stopped, you want firewall accessibility from local hosts 192.168.1.0/24 and from your DMZ. Your DMZ interfaces through eth1 and your local hosts through eth2.

```
#INTERFACE          HOST(S)
eth2                192.168.1.0/24
eth1                -
```

Note

Shorewall allows connections defined by the contents of `/etc/shorewall/routestopped` during the potentially lengthy processing of the commands **shorewall start**, **shorewall restart**, **shorewall try** and **shorewall stop**.

/etc/shorewall/maclist

This file is described in the MAC Validation Documentation.

/etc/shorewall/ecn

This file is described in the ECN Control Documentation.

/etc/shorewall/accounting

This file is described in the Traffic Accounting Documentation.

1. Revision History

Revision History		
Revision 1.25	2005-08-28	TE
Changes for 3.0.		
Revision 1.24	2005-03-15	TE
Update for use of /etc/shorewall/routestopped during [re]start.		
Revision 1.23	2005-03-10	TE
Changes for Shorewall 2.2.2.		
Revision 1.20	2004-10-25	TE
Changes for Shorewall 2.2 Beta 1.		

Revision 1.19	2004-09-12	TE
Changes for Shorewall 2.1.		
Revision 1.18	2004-08-22	TE
Add /etc/shorewall/ipsec documentation.		
Revision 1.17	2004-04-05	TE
Update for Shorewall 2.0.2		
Revision 1.16	2004-03-17	TE
Clarified LOGBURST and LOGLIMIT.		
Revision 1.15	2004-02-16	TE
Move the rfc1918 file to /usr/share/shorewall.		
Revision 1.14	2004-02-13	TE
Add a note about the order of rules.		
Revision 1.13	2004-02-03	TE
Update for Shorewall 2.0.		
Revision 1.12	2004-01-21	TE
Add masquerade destination list.		
Revision 1.12	2004-01-18	TE
Correct typo.		
Revision 1.11	2004-01-05	TE
Standards Compliance		
Revision 1.10	2004-01-05	TE
Improved formatting of DNAT- and REDIRECT- for clarity		
Revision 1.9	2003-12-25	MN
Initial Docbook Conversion Complete		

Shorewall QuickStart Guides (HOWTOs)

Tom Eastep

Copyright © 2001-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-01-07

Table of Contents

The Guides
- If you already have a router.
- If you want the firewall system to handle a **single public IP address**
- If you want the firewall system to handle more than one public IP address
- Guides that Others have Written

With thanks to Richard who reminded me once again that we must all first walk before we can run.

The French Translations of the single-IP guides are courtesy of Patrice Vetsel. Updated for Shorewall 2.0 by Fabien Demassieux.

The French Translation of the Shorewall Setup Guide is courtesy of Fabien Demassieux.

The Guides

These guides provide step-by-step instructions for configuring Shorewall in common firewall setups.

If you already have a router.

If you already have a router on your premises and you simply want to add a firewall between the router and your local systems then you want a bridge configuration.

If you want the firewall system to handle a single public IP address

These guides are designed to get your first firewall up and running quickly in the three most common Shorewall configurations. If you want to learn more about Shorewall than is explained in these simple guides then the Shorewall Setup Guide is for you.

- Standalone Linux System (Version Française)
- Two-interface Linux System acting as a firewall/router for a small local network (Version Française)
- Three-interface Linux System acting as a firewall/router for a small local network and a DMZ.. (Version Française)

If you want the firewall system to handle more than one public IP address

The Shorewall Setup Guide outlines the steps necessary to set up a firewall where there are multiple public IP addresses involved or if you want to learn more about Shorewall than is explained in the single-address guides above (Version Française)

Guides that Others have Written

Andrew Allen has provided this guide for installing Shorewall on standalone webhosting servers.

Shorewall and Bridged Firewalls

Tom Eastep

Copyright © 2004, 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-10-21

Table of Contents

Background
Requirements
Application
Configuring the Bridge
Configuring Shorewall
Combination Router/Bridge
Limitations
Other Links

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Background

Systems where Shorewall runs normally function as *routers*. In the context of the Open System Interconnect (OSI) reference model, a router operates at layer 3, Shorewall may also be deployed on a GNU Linux System that acts as a *bridge*. Bridges are layer-2 devices in the OSI model (think of a bridge as an ethernet switch).

Some differences between routers and bridges are:

1. Routers determine packet destination based on the destination IP address while bridges route traffic based on the destination MAC address in the ethernet frame.
2. As a consequence of the first difference, routers can be connected to more than one IP network while a bridge may be part of only a single network.
3. In most configurations, routers don't forward broadcast packets while a bridges do.

Note

Section 4 of RFC 1812 describes the conditions under which a router may or must forward broadcasts.

Requirements

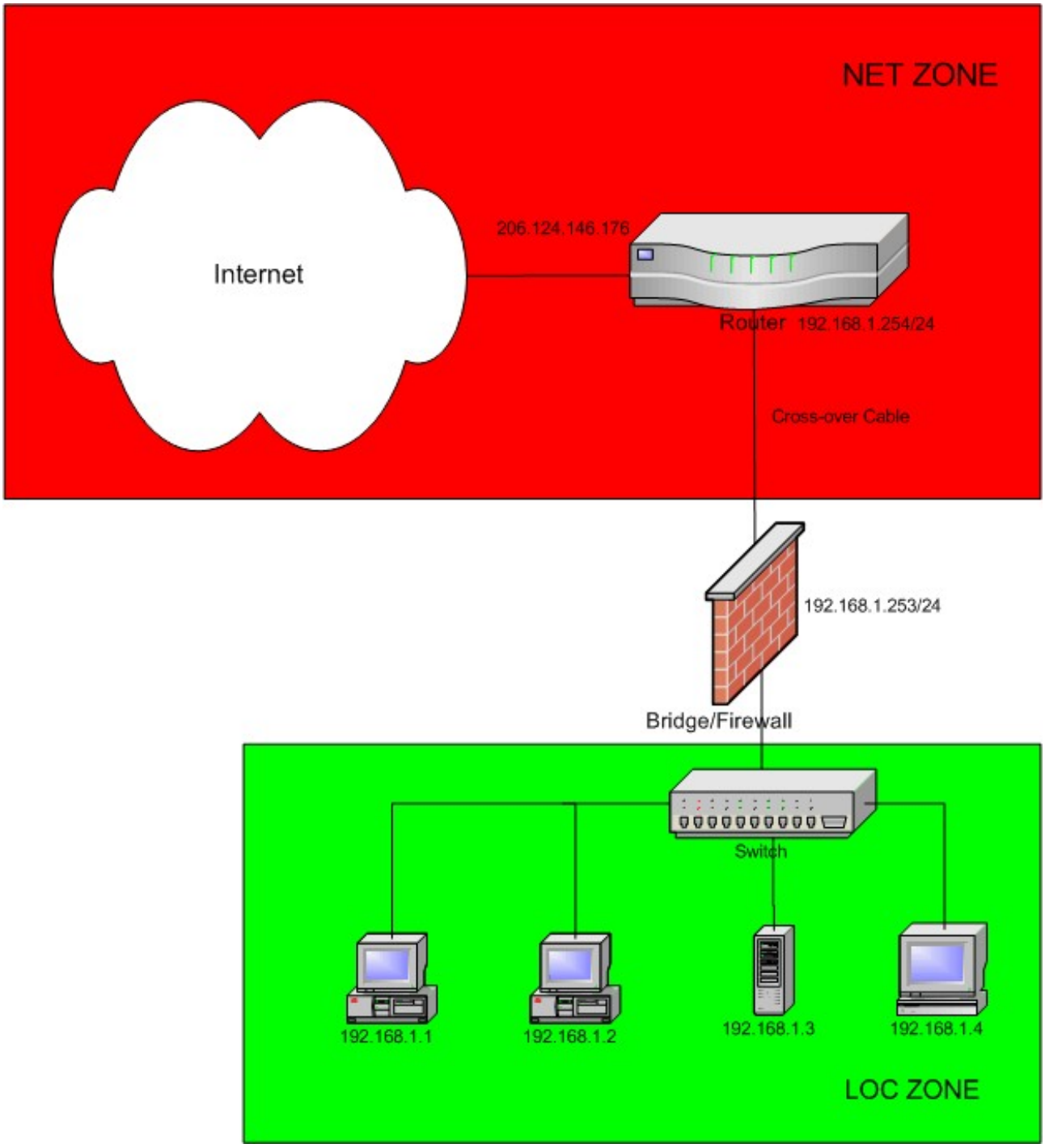
Note that if you need a bridge but do not need to restrict the traffic through the bridge then any version of Shorewall will work. See the Simple Bridge documentation for details.

In order to use Shorewall as a bridging firewall:

- Your kernel must contain bridge support (CONFIG_BRIDGE=m or CONFIG_BRIDGE=y).
- Your kernel must contain Netfilter physdev match support (CONFIG_IP_NF_MATCH_PHYSDEV=m or CONFIG_IP_NF_MATCH_PHYSDEV=y). Physdev match is standard in the 2.6 kernel series but must be patched into the 2.4 kernels (see http://bridge.sf.net). Bering and Bering uCLibc users must find and install ipt_physdev.o for their distribution and add “ipt_physdev” to /etc/modules.
- Your iptables must contain physdev match support. iptables 1.2.9 and later contain this support.
- You must have the bridge utilities (bridge-utils) package installed.

Application

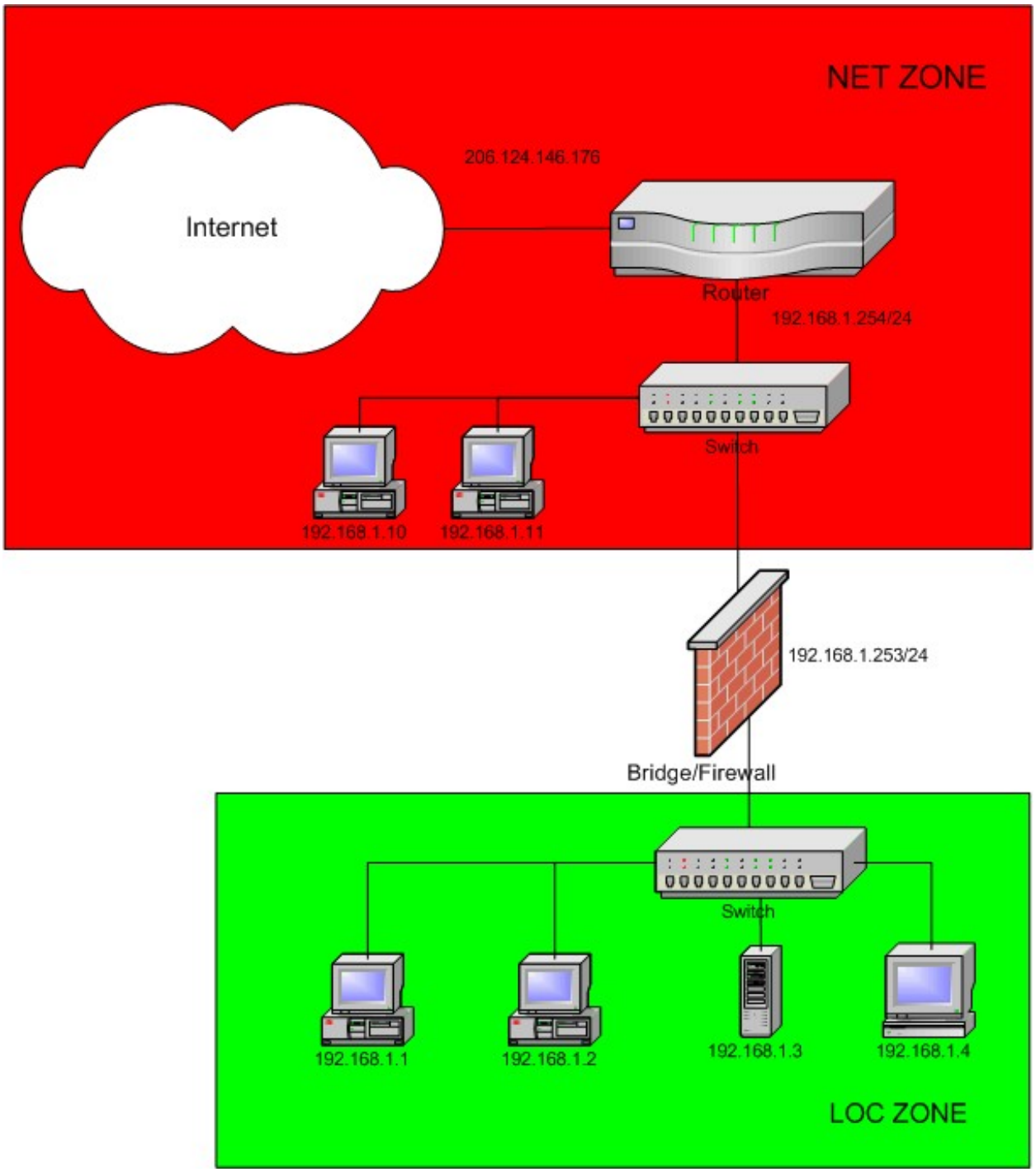
The following diagram shows a typical application of a bridge/firewall. There is already an existing router in place whose internal interface supports a network and you want to insert a firewall between the router and the systems in the local network. In the example shown, the network uses RFC 1918 addresses but that is not a requirement; the bridge would work exactly the same if public IP addresses were used (remember that the bridge doesn't deal with IP addresses).



There are a several key differences in this setup and a normal Shorewall configuration:

- The Shorewall system (the Bridge/Firewall) has only a single IP address even though it has two ethernet interfaces! The IP address is configured on the bridge itself rather than on either of the network cards.
- The systems connected to the LAN are configured with the router's IP address (192.168.1.254 in the above diagram) as their default gateway.
- **traceroute** doesn't detect the Bridge/Firewall as an intermediate router.
- If the router runs a DHCP server, the hosts connected to the LAN can use that server without having **dhcrelay** running on the Bridge/Firewall.

There are other possibilities here -- there could be a hub or switch between the router and the Bridge/Firewall and there could be other systems connected to that switch. All of the systems on the local side of the **router** would still be configured with IP addresses in 192.168.1.0/24 as shown below.



Configuring the Bridge

Configuring the bridge itself is quite simple and uses the **brctl** utility from the bridge-utils package. Bridge configuration information may be found at http://bridge.sf.net.

Unfortunately, many Linux distributions don't have good bridge configuration tools and the network configuration GUIs don't detect

the presence of bridge devices. Here is an excerpt from a Debian `/etc/network/interfaces` file for a two-port bridge with a static IP address:

```
auto br0
iface br0 inet static
        address 192.168.1.253
        netmask 255.255.255.0
        network 192.168.1.0
        broadcast 192.168.1.255
        pre-up /sbin/ip link set eth0 up
        pre-up /sbin/ip link set eth1 up
        pre-up /usr/sbin/brctl addbr br0
        pre-up /usr/sbin/brctl addif br0 eth0
        pre-up /usr/sbin/brctl addif br0 eth1
```

While it is not a requirement to give the bridge an IP address, doing so allows the bridge/firewall to access other systems and allows the bridge/firewall to be managed remotely. The bridge must also have an IP address for REJECT rules and policies to work correctly — otherwise REJECT behaves the same as DROP. It is also a requirement for bridges to have an IP address if they are part of a bridge/router.

Important

Get your bridge configuration working first, including bridge startup at boot, before you configure and start Shorewall.

The bridge may have its IP address assigned via DHCP. Here's an example of an /etc/sysconfig/network/ifcfg-br0 file from a SuSE™ system:

```
BOOTPROTO='dhcp'
REMOTE_IPADDR=''
STARTMODE='onboot'
UNIQUE='3hqH.MjuOqWfSZ+C'
WIRELESS='no'
MTU=''
```

Here's an /etc/sysconfig/network-scripts/ifcfg-br0 file for a Mandrake™ system:

```
DEVICE=br0
BOOTPROTO=dhcp
ONBOOT=yes
```

On both the SuSE and Mandrake systems, a separate script is required to configure the bridge itself.

Here are scripts that I used on a Suse™ 9.1 system.

`/etc/sysconfig/network/ifcfg-br0`

```
BOOTPROTO='dhcp'
REMOTE_IPADDR=''
STARTMODE='onboot'
UNIQUE='3hqH.MjuOqWfSZ+C'
WIRELESS='no'
MTU=''
```

`/etc/init.d/bridge`

```
#!/bin/sh
```

```
################################################################################
#   Script to create a bridge
#
#     (c) 2004 - Tom Eastep (teastep@shorewall.net)
#
#   Modify the following variables to match your configuration
#
#### BEGIN INIT INFO
# Provides:       bridge
# Required-Start: coldplug
# Required-Stop:
# Default-Start:  2 3 5
# Default-Stop:   0 1 6
# Description:    starts and stops a bridge
### END INIT INFO
#
# chkconfig: 2345 05 89
# description: GRE/IP Tunnel
#
################################################################################


PATH=$PATH:/sbin:/usr/sbin:/usr/local/sbin

INTERFACES="eth1 eth0"
BRIDGE="br0"
MODULES="tulip"

do_stop() {
    echo "Stopping Bridge $BRIDGE"
    brctl delbr $BRIDGE
    for interface in $INTERFACES; do
        ip link set $interface down
    done
}

do_start() {

      echo "Starting Bridge $BRIDGE"
      for module in $MODULES; do
          modprobe $module
      done

      sleep 5

      for interface in $INTERFACES; do
          ip link set $interface up
      done

      brctl addbr $BRIDGE

      for interface in $INTERFACES; do
          brctl addif $BRIDGE $interface
      done
}

case "$1" in
  start)
      do_start
    ;;
  stop)
      do_stop
    ;;
```

```
  restart)
      do_stop
      sleep 1
      do_start
    ;;
  *)
    echo "Usage: $0 {start|stop|restart}"
    exit 1
esac
exit 0
```

Axel Westerhold has contributed this example of configuring a bridge with a static IP address on a Fedora System (Core 1 and Core 2 Test 1). Note that these files also configure the bridge itself so there is no need for a separate bridge config script.

`/etc/sysconfig/network-scripts/ifcfg-br0:`

```
DEVICE=br0
TYPE=Bridge
IPADDR=192.168.50.14
NETMASK=255.255.255.0
ONBOOT=yes
```

`/etc/sysconfig/network-scripts/ifcfg-eth0:`

```
DEVICE=eth0
TYPE=ETHER
BRIDGE=br0
ONBOOT=yes
```

`/etc/sysconfig/network-scripts/ifcfg-eth1:`

```
DEVICE=eth1
TYPE=ETHER
BRIDGE=br0
ONBOOT=yes
```

Florin Grad at Mandrake™ provides this script for configuring a bridge:

```
#!/bin/sh
# chkconfig: 2345 05 89
# description: Layer 2 Bridge
#

[ -f /etc/sysconfig/bridge ] && . /etc/sysconfig/bridge

PATH=$PATH:/sbin:/usr/sbin:/usr/local/sbin

do_stop() {
    echo "Stopping Bridge"
    for i in $INTERFACES $BRIDGE_INTERFACE ; do
        ip link set $i down
    done
    brctl delbr $BRIDGE_INTERFACE
}

do_start() {

   echo "Starting Bridge"
   for i in $INTERFACES ; do
```

```
        ip link set $i up
   done
   brctl addbr br0
   for i in $INTERFACES ; do
        ip link set $i up
        brctl addif br0 $i
   done
   ifup $BRIDGE_INTERFACE
}

case "$1" in
  start)
      do_start
    ;;
  stop)
      do_stop
    ;;
  restart)
      do_stop
      sleep 1
      do_start
    ;;
  *)
    echo "Usage: $0 {start|stop|restart}"
    exit 1
esac
exit 0
```

The `/etc/sysconfig/bridge file`:

```
BRIDGE_INTERFACE=br0          #The name of your Bridge
INTERFACES="eth0 eth1"        #The physical interfaces to be bridged
```

Andrzej Szelachowski contributed the following.

```
Here is how I configured bridge in Slackware:

1) I had to compile bridge-utils (It's not in the standard distribution)
2) I've created rc.bridge in /etc/rc.d:

#########################
#! /bin/sh

ifconfig eth0 0.0.0.0
ifconfig eth1 0.0.0.0
#ifconfig lo 127.0.0.1 #this line should be uncommented if you don't use rc.inet1

brctl addbr most

brctl addif most eth0
brctl addif most eth1

ifconfig most 192.168.1.31 netmask 255.255.255.0 up
#route add default gw 192.168.1.1 metric 1 #this line should be uncommented if
                                           #you don't use rc.inet1
#########################

3) I made rc.brige executable and added the following line to /etc/rc.d/rc.local

/etc/rc.d/rc.bridge
```

Joshua Schmidlkofer writes:

```
Bridge Setup for Gentoo

#install bridge-utils
emerge bridge-utils

## create a link for net.br0
cd /etc/init.d
ln -s net.eth0 net.br0

# Remove net.eth*, add net.br0 and bridge.
rc-update del net.eth0
rc-update del net.eth1
rc-update add net.br0 default
rc-update add bridge boot



/etc/conf.d/bridge:

  #bridge contains the name of each bridge you want created.
  bridge="br0"

  # bridge_<bridge>_devices contains the devices to use at bridge startup.
  bridge_br0_devices="eth0 eth1"

/etc/conf.d/net

   iface_br0="10.0.0.1     broadcast 10.0.0.255 netmask 255.255.255.0"
   #for dhcp:
   #iface_br0="dhcp"
   #comment this out if you use dhcp.
   gateway="eth0/10.0.0.1"
```

Users who successfully configure bridges on other distributions, with static or dynamic IP addresses, are encouraged to send me their configuration so I can post it here.

Configuring Shorewall

Bridging in Shorewall is enabled using the BRIDGING option in `/etc/shorewall/shorewall.conf`:

```
BRIDGING=Yes
```

In the scenario pictured above, there would probably be two zones defined -- one for the internet and one for the local LAN so in `/etc/shorewall/zones`:

```
#ZONE   TYPE            OPTIONS
fw      firewall
net     ipv4
loc     ipv4
#LAST LINE - ADD YOUR ENTRIES ABOVE THIS ONE - DO NOT REMOVE
```

A conventional two-zone policy file is appropriate here — `/etc/shorewall/policy`:

```
#SOURCE     DEST        POLICY        LOG       LIMIT:BURST
loc         net         ACCEPT
net         all         DROP          info
all         all         REJECT        info
```

```
#LAST LINE - ADD YOUR ENTRIES ABOVE THIS ONE - DO NOT REMOVE
```

Only the bridge device itself is configured with an IP address so only that device is defined to Shorewall in `/etc/shorewall/interfaces`:

```
#ZONE    INTERFACE      BROADCAST       OPTIONS
-       br0             192.168.1.255
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

The zones are defined using the `/etc/shorewall/hosts` file. Assuming that the router is connected to `eth0` and the switch to `eth1`:

```
#ZONE           HOST(S)                         OPTIONS
net             br0:eth0
loc             br0:eth1
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS LINE -- DO NOT REMOVE
```

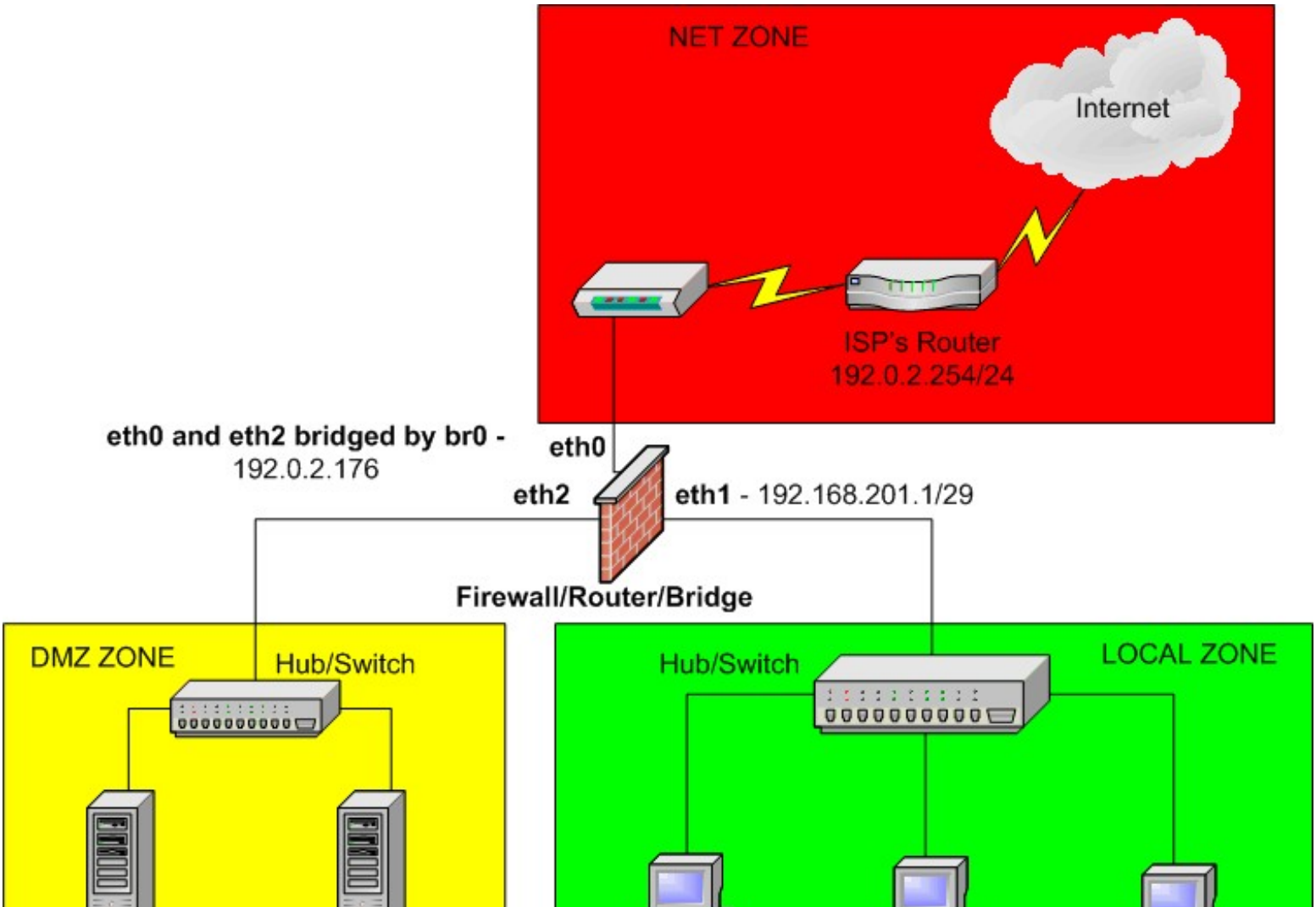
When Shorewall is stopped, you want to allow only local traffic through the bridge — `/etc/shorewall/routestopped`:

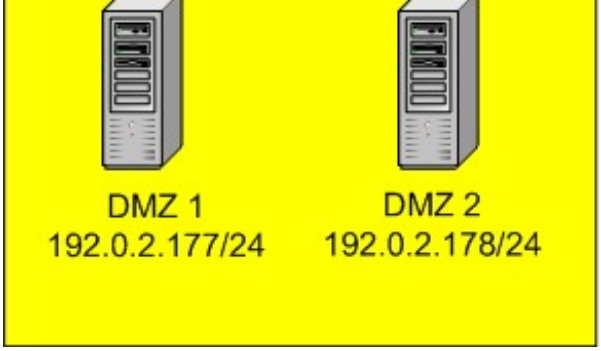
```
#INTERFACE      HOST(S)         OPTIONS
br0             192.168.1.0/24  routeback
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

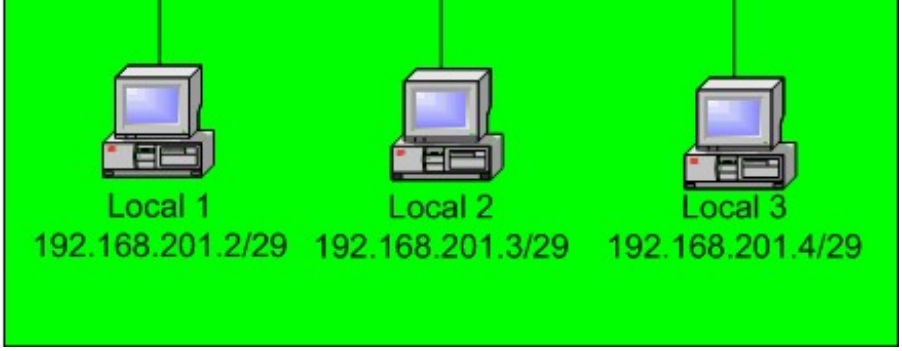
The `/etc/shorewall/rules` file from the two-interface sample is a good place to start for defining a set of firewall rules.

Combination Router/Bridge

A system running Shorewall doesn't have to be exclusively a bridge or a router -- it can act as both. Here's an example:







This is basically the same setup as shown in the Shorewall Setup Guide with the exception that the DMZ is bridged rather than using Proxy ARP. Changes in the configuration shown in the Setup Guide are as follows:

1. The `/etc/shorewall/proxyarp` file is empty in this configuration.
2. The `/etc/shorewall/interfaces` file is as follows:

```
#ZONE    INTERFACE       BROADCAST      OPTIONS
-        br0             detect         routefilter
loc      eth1            detect
```

3. The `/etc/shorewall/hosts` file would have:

```
#ZONE    HOSTS                          OPTIONS
net      br0:eth0
dmz      br0:eth2
```

4. The DMZ systems need a route to the 192.168.201.0/24 network via 192.0.2.176 to enable them to communicate with the local network.

Limitations

Bridging doesn't work with some wireless cards — see http://bridge.sf.net.

Other Links

- Here is an article in Spanish detailing bridging a public and local network using Shorewall. This is another router/bridge configuration.

Shorewall and a Simple Bridge

Tom Eastep

Copyright © 2004, 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-10-12

Table of Contents

Background
Application

Background

Systems where Shorewall runs normally function as *routers*. In the context of the Open System Interconnect (OSI) reference model, a router operates at layer 3. Shorewall may also be deployed on a GNU Linux System that acts as a *bridge*. Bridges are layer-2 devices in the OSI model (think of a bridge as an ethernet switch).

Some differences between routers and bridges are:

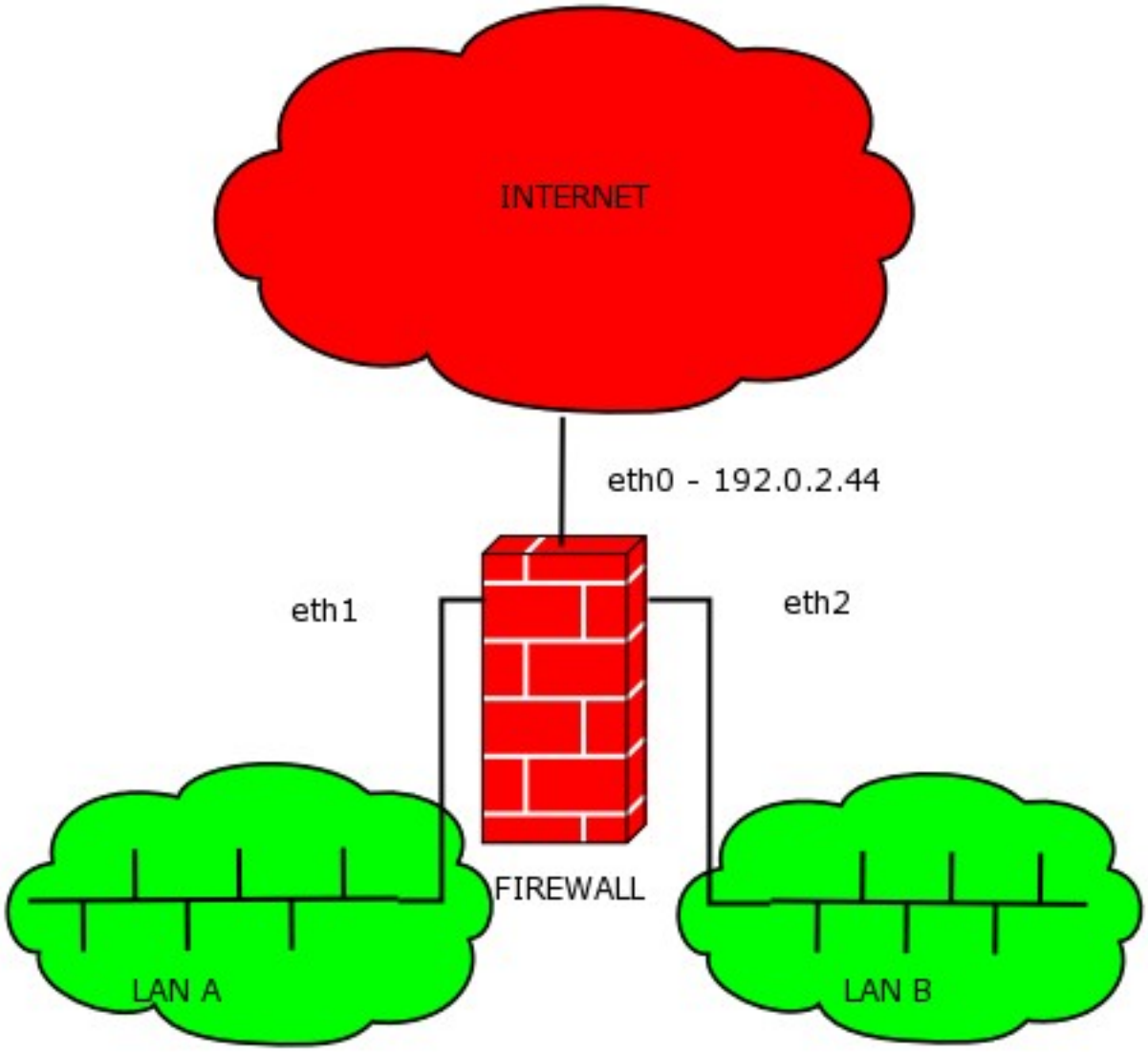
1. Routers determine packet destination based on the destination IP address while bridges route traffic based on the destination MAC address in the ethernet frame.
2. As a consequence of the first difference, routers can be connected to more than one IP network while a bridge may be part of only a single network.
3. A router cannot forward broadcast packets while a bridge can.

Application

There are cases where you want to create a bridge to join two or more LAN segments and you don't need to restrict the traffic between those segments. This is the environment that is described in this article.

If you do need to restrict traffic through the bridge, please refer to the Shorewall Bridge/Firewall documentation. Also please refer to that documentation for information about how to create a bridge.

The following diagram shows a firewall for two bridged LAN segments.



Firewall bridges LAN A and LAN B using bridge 'br0' with IP address 10.0.1.254/24

This is fundimentally the Two-interface Firewall described in the Two-interface Quickstart Guide. The bridge-specific changes are restricted to the `/etc/shorewall/interfaces` file.

`/etc/shorewall/interfaces`:

```
#ZONE          INTERFACE       BROADCAST       OPTIONS
net            eth0            detect          ...
loc            br0             10.0.1.255      routeback,...
```

So the key points here are:

- The **loc** interface is `br0`.
- Neither `eth1` nor `eth2` have IP addresses and neither are mentioned in the Shorewall configuration.
- The **routeback** option is specified for `br0`.
- The default gateway for hosts in the local segments will be 10.0.1.254 — the IP address of the bridge itself.

Basic Two-Interface Firewall

Tom Eastep

Copyright © 2002-, 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-11-10

Table of Contents

Introduction
- System Requirements
- Conventions

PPTP/ADSL
Shorewall Concepts
Network Interfaces
IP Addresses
IP Masquerading (SNAT)
Port Forwarding (DNAT)
Domain Name Server (DNS)
Other Connections
Some Things to Keep in Mind
Starting and Stopping Your Firewall
Additional Recommended Reading
Adding a Wireless Segment to your Two-Interface Firewall

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Introduction

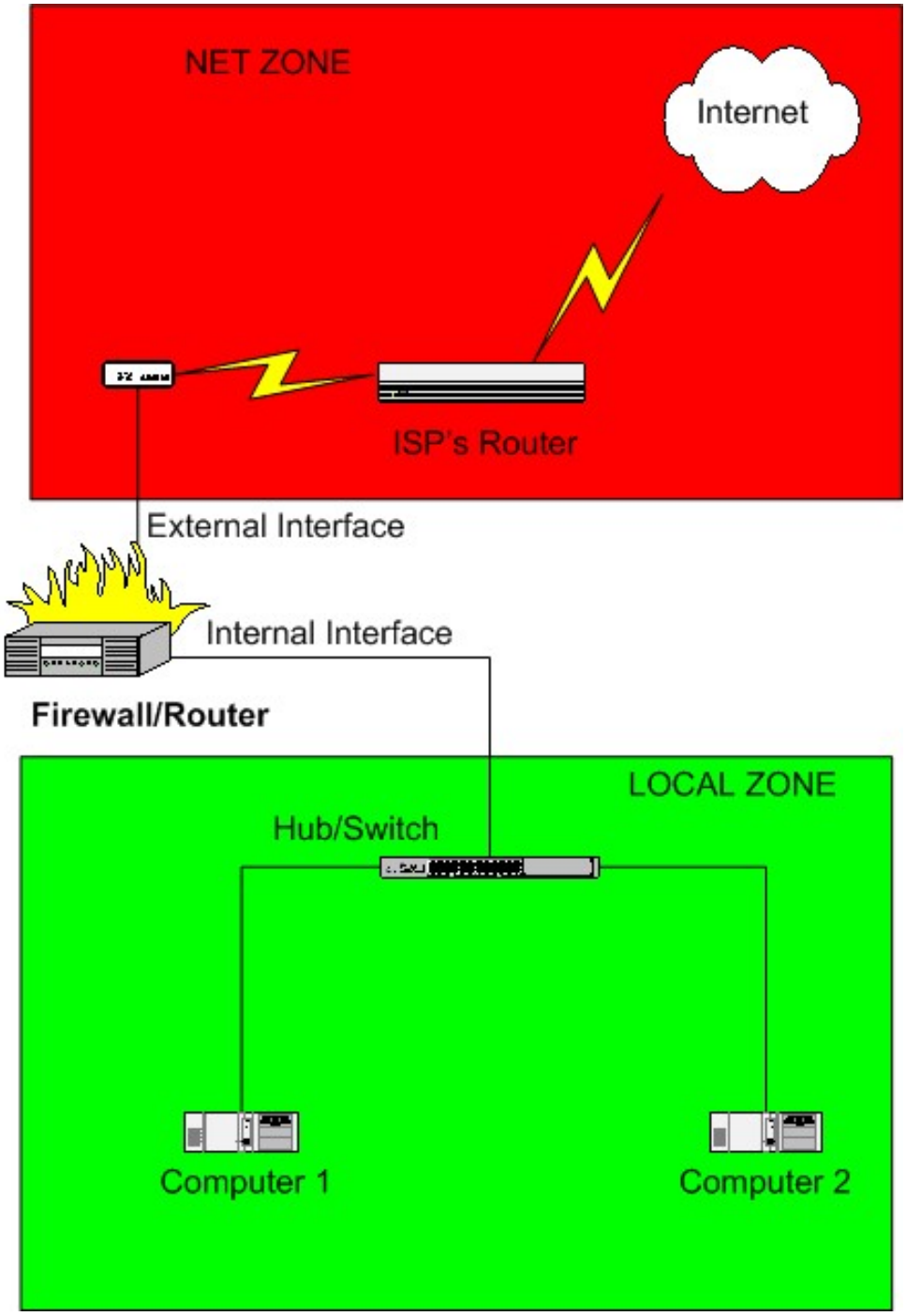
Setting up a Linux system as a firewall for a small network is a fairly straight-forward task if you understand the basics and follow the documentation.

This guide doesn't attempt to acquaint you with all of the features of Shorewall. It rather focuses on what is required to configure Shorewall in its most common configuration:

- Linux system used as a firewall/router for a small local network.
- **Single public IP address.** If you have more than one public IP address, this is not the guide you want -- see the Shorewall Setup Guide instead.
- Internet connection through cable modem, DSL, ISDN, Frame Relay, dial-up ...

Here is a schematic of a typical installation:

Figure 1. Common two interface firewall configuration



Caution

If you edit your configuration files on a Windows™ system, you must save them as Unix™ files if your editor supports that option or you must run them through **dos2unix** before trying to use them. Similarly, if you copy a configuration file from your Windows™ hard drive to a floppy disk, you must run **dos2unix** against the copy before using it with Shorewall.

- Windows™ Version of **dos2unix**
- Linux Version of **dos2unix**

System Requirements

Shorewall requires that you have the **iproute/iproute2** package installed (on RedHat™, the package is called **iproute**). You can tell if this package is installed by the presence of an **ip** program on your firewall system. As `root`, you can use the **which** command to check for this program:

```
[root@gateway root]# which ip
/sbin/ip
[root@gateway root]#
```

I recommend that you first read through the guide to familiarize yourself with what's involved then go back through it again making your configuration changes.

Conventions

Points at which configuration changes are recommended are flagged with ➡.

Configuration notes that are unique to LEAF/Bering are marked with .

PPTP/ADSL

➡

If you have an ADSL Modem and you use PPTP to communicate with a server in that modem, you must make the changes recommended here in addition to those detailed below. ADSL with PPTP is most commonly found in Europe, notably in Austria.

Shorewall Concepts

The configuration files for Shorewall are contained in the directory `/etc/shorewall` -- for simple setups, you will only need to deal with a few of these as described in this guide.

Warning

Note to Debian Users

If you install using the .deb, you will find that your `/etc/shorewall` directory is empty. This is intentional. The released configuration file skeletons may be found on your system in the directory `/usr/share/doc/shorewall/default-config`. Simply copy the files you need from that directory to `/etc/shorewall` and modify the copies.

Note that you must copy `/usr/share/doc/shorewall/default-config/shorewall.conf` and /usr/share/doc/shorewall/default-config/modules to `/etc/shorewall` even if you do not modify those files.

➡

Important

After you have installed Shorewall, locate the two-interfaces samples:

1. If you installed using an RPM, the samples will be in the Samples/two-interfaces/ subdirectory of the Shorewall documentation directory. If you don't know where the Shorewall documentation directory is,

you can find the samples using this command:

```
~# rpm -ql shorewall | fgrep two-interfaces
/usr/share/doc/packages/shorewall/Samples/two-interfaces
/usr/share/doc/packages/shorewall/Samples/two-interfaces/interfaces
/usr/share/doc/packages/shorewall/Samples/two-interfaces/masq
/usr/share/doc/packages/shorewall/Samples/two-interfaces/policy
/usr/share/doc/packages/shorewall/Samples/two-interfaces/routestopped
/usr/share/doc/packages/shorewall/Samples/two-interfaces/rules
/usr/share/doc/packages/shorewall/Samples/two-interfaces/zones
~#
```

2. If you installed using the tarball, the samples are in the Samples/two-interfaces directory in the tarball.
3. If you installed using the .deb, the samples are in /usr/share/doc/shorewall/examples/two-interfaces.

As each file is introduced, I suggest that you look through the actual file on your system -- each file contains detailed configuration instructions and default entries.

Shorewall views the network where it is running as being composed of a set of zones. In the two-interface sample configuration, the following zone names are used:

```
#ZONE   TYPE     OPTIONS                 IN                      OUT
#                                        OPTIONS                 OPTIONS
fw      firewall
net     ipv4
loc     ipv4
```

Zones are defined in the `/etc/shorewall/zones` file.

Note that Shorewall recognizes the firewall system as its own zone - when the /etc/shorewall/zones file is processed, the name of the firewall zone is stored in the shell variable $FW which may be used to refer to the firewall zone throughout the Shorewall configuration.

Rules about what traffic to allow and what traffic to deny are expressed in terms of zones.

- You express your default policy for connections from one zone to another zone in the `/etc/shorewall/policy` file.
- You define exceptions to those default policies in the `/etc/shorewall/rules` file.

For each connection request entering the firewall, the request is first checked against the `/etc/shorewall/rules` file. If no rule in that file matches the connection request then the first policy in `/etc/shorewall/policy` that matches the request is applied. If there is a comon action defined for the policy in `/etc/shorewall/actions` or `/usr/share/shorewall/actions.std` then that action is peformed before the action is applied.

The `/etc/shorewall/policy` file included with the two-interface sample has the following policies:

```
#SOURCE    DEST        POLICY      LOG LEVEL    LIMIT:BURST
loc        net         ACCEPT
net        all         DROP        info
all        all         REJECT      info
```

In the two-interface sample, the line below is included but commented out. If you want your firewall system to have full access to servers on the internet, uncomment that line.

```
#SOURCE     DEST        POLICY      LOG LEVEL    LIMIT:BURST
$FW         net         ACCEPT
```

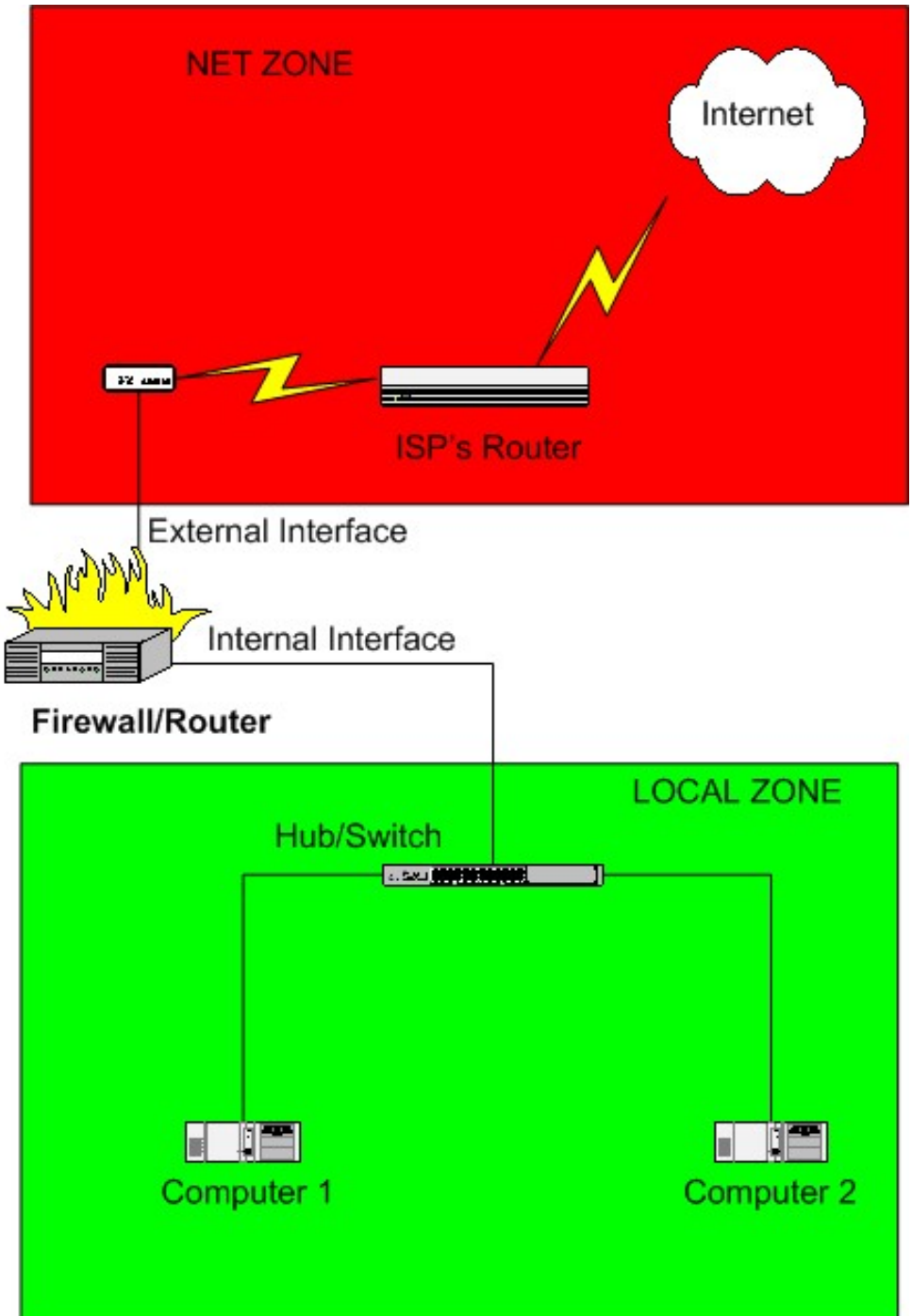
The above policy will:

- Allow all connection requests from your local network to the internet
- Drop (ignore) all connection requests from the internet to your firewall or local network
- Optionally accept all connection requests from the firewall to the internet (if you uncomment the additional policy)
- reject all other connection requests.

It is important to note that Shorewall policies (and rules) refer to **connections** and not packet flow. With the policies defined in the `/etc/shorewall/policy` file shown above, connections are allowed from the *loc* zone to the *net* zone even though connections are not allowed from the *loc* zone to the firewall itself.

At this point, edit your `/etc/shorewall/policy` and make any changes that you wish.

Network Interfaces



The firewall has two network interfaces. Where Internet connectivity is through a cable or DSL “Modem”, the *External Interface* will be the ethernet adapter that is connected to that “Modem” (e.g., `eth0`) unless you connect via *Point-to-Point Protocol* over Ethernet (PPPoE) or *Point-to-Point Tunneling Protocol* (PPTP) in which case the External Interface will be a `ppp` interface (e.g., `ppp0`). If you connect via a regular modem, your External Interface will also be `ppp0`. If you connect via ISDN, your external interface will be `ippp0`.

If your external interface is `ppp0` or `ippp0` then you will want to set `CLAMPMSS=yes` in `/etc/shorewall/shorewall.conf`.

Your *Internal Interface* will be an ethernet adapter (`eth1` or `eth0`) and will be connected to a hub or switch. Your other computers will be connected to the same hub/switch (note: If you have only a single internal system, you can connect the firewall directly to the computer using a cross-over cable).

Warning

Do not connect the internal and external interface to the same hub or switch except for testing.You can test using this kind of configuration if you specify the arp_filter option in `/etc/shorewall/interfaces` for all interfaces connected to the common hub/switch. **Using such a setup with a production firewall is strongly recommended against**.

The Shorewall two-interface sample configuration assumes that the external interface is `eth0` and the internal interface is `eth1`. If your configuration is different, you will have to modify the sample `/etc/shorewall/interfaces` file accordingly. While you are there, you may wish to review the list of options that are specified for the interfaces. Some hints:

Tip

If your external interface is `ppp0` or `ippp0`, you can replace the `detect` in the second column with a “-” (minus the quotes).

Tip

If your external interface is `ppp0` or `ippp0` or if you have a static IP address, you can remove `dhcp` from the option list.

Tip

If your internal interface is a bridge create using the **brctl** utility then **you must add the `routeback` option to the option list.**

IP Addresses

Before going further, we should say a few words about Internet Protocol (IP) addresses. Normally, your ISP will assign you a single Public IP address. This address may be assigned via the Dynamic Host Configuration Protocol (DHCP) or as part of establishing your connection when you dial in (standard modem) or establish your PPP connection. In rare cases, your ISP may assign you a static IP address; that means that you configure your firewall's external interface to use that address permanently. However your external address is assigned, it will be shared by all of your systems when you access the

Internet. You will have to assign your own addresses in your internal network (the Internal Interface on your firewall plus your other computers). **RFC 1918** reserves several *Private* IP address ranges for this purpose:

```
10.0.0.0     - 10.255.255.255
172.16.0.0   - 172.31.255.255
192.168.0.0 - 192.168.255.255
```

Before starting Shorewall, **you should look at the IP address of your external interface and if it is one of the above ranges, you should remove the 'norfc1918' option from the external interface's entry in `/etc/shorewall/interfaces`.**

You will want to assign your addresses from the same sub-network (subnet). For our purposes, we can consider a subnet to consists of a range of addresses `x.y.z.0 - x.y.z.255`. Such a subnet will have a Subnet Mask of `255.255.255.0`. The address `x.y.z.0` is reserved as the *Subnet Address* and `x.y.z.255` is reserved as the *Subnet Broadcast Address*. In Shorewall, a subnet is described using Classless InterDomain Routing (CIDR) notation with consists of the subnet address followed by `/24`. The “24” refers to the number of consecutive leading “1” bits from the left of the subnet mask.

Range:	`10.10.10.0 - 10.10.10.255`
Subnet Address:	`10.10.10.0`
Broadcast Address:	`10.10.10.255`
CIDR Notation:	`10.10.10.0/24`

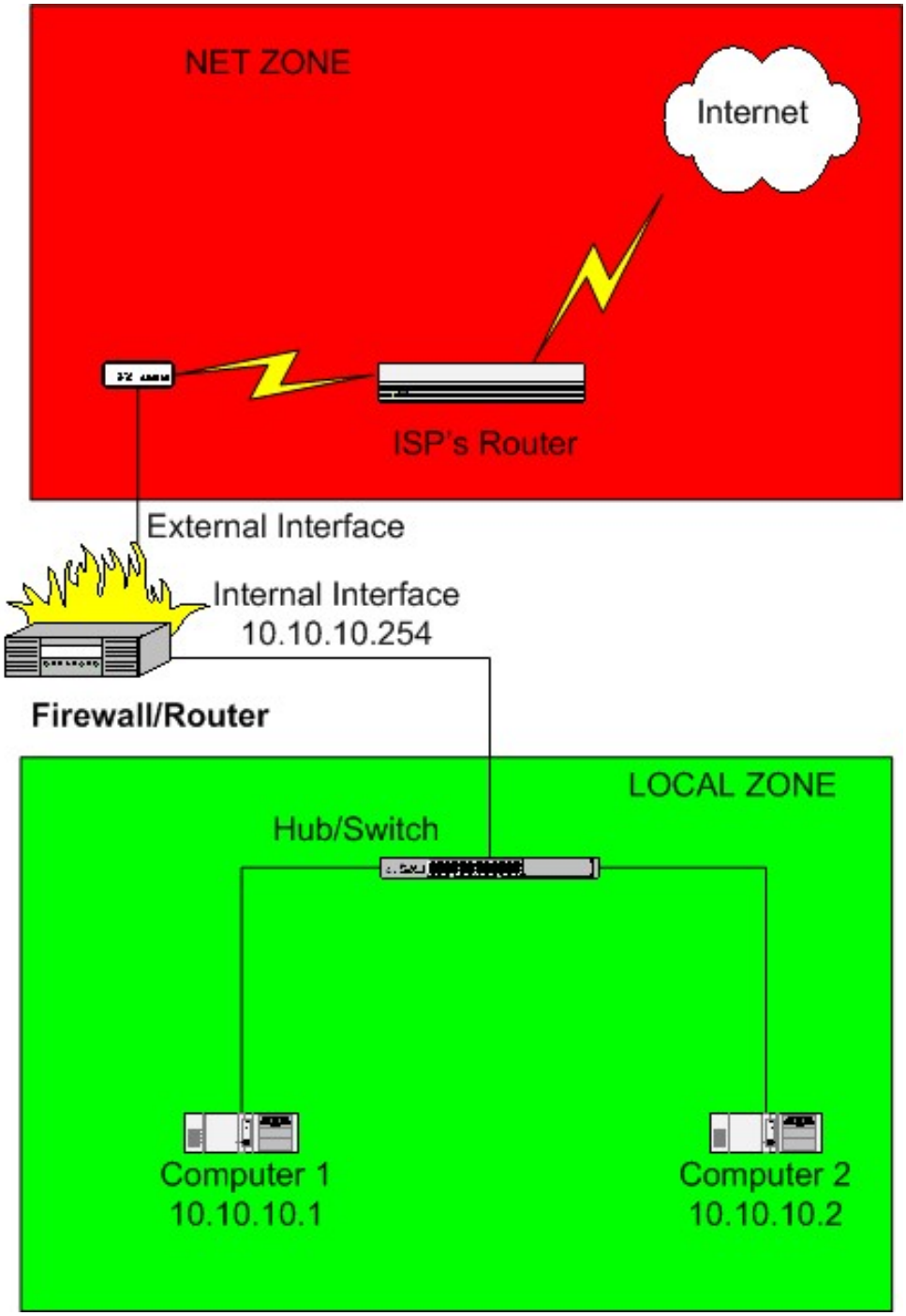
It is conventional to assign the internal interface either the first usable address in the subnet (`10.10.10.1` in the above example) or the last usable address (`10.10.10.254`).

One of the purposes of subnetting is to allow all computers in the subnet to understand which other computers can be communicated with directly. To communicate with systems outside of the subnetwork, systems send packets through a gateway (router).

Your local computers (computer 1 and computer 2 in the above diagram) should be configured with their default gateway to be the IP address of the firewall's internal interface.

The foregoing short discussion barely scratches the surface regarding subnetting and routing. If you are interested in learning more about IP addressing and routing, I highly recommend “IP Fundamentals: What Everyone Needs to Know about Addressing & Routing”, Thomas A. Maufer, Prentice-Hall, 1999, ISBN 0-13-975483-0 (link).

The remainder of this quide will assume that you have configured your network as shown here:



The default gateway for computer's 1 & 2 would be `10.10.10.254`.

Warning

Your ISP might assign your external interface an **RFC 1918** address. If that address is in the `10.10.10.0/24` subnet then **you will need to select a DIFFERENT RFC 1918 subnet for your local network.**

IP Masquerading (SNAT)

The addresses reserved by RFC 1918 are sometimes referred to as non-routable because the Internet backbone routers don't forward packets which have an RFC-1918 destination address. When one of your local systems (let's assume computer 1) sends a connection request to an internet host, the firewall must perform *Network Address Translation* (NAT). The firewall rewrites the source address in the packet to be the address of the firewall's external interface; in other words, the firewall makes it look as if the firewall itself is initiating the connection. This is necessary so that the destination host will be able to route return packets back to the firewall (remember that packets whose destination address is reserved by RFC 1918 can't be

routed across the internet so the remote host can't address its response to computer 1). When the firewall receives a return packet, it rewrites the destination address back to `10.10.10.1` and forwards the packet on to computer 1.

On Linux systems, the above process is often referred to as *IP Masquerading* but you will also see the term *Source Network Address Translation* (SNAT) used. Shorewall follows the convention used with Netfilter:

- *Masquerade* describes the case where you let your firewall system automatically detect the external interface address.
- *SNAT* refers to the case when you explicitly specify the source address that you want outbound packets from your local network to use.

In Shorewall, both *Masquerading* and *SNAT* are configured with entries in the `/etc/shorewall/masq` file. You will normally use Masquerading if your external IP is dynamic and SNAT if the IP is static.

If your external firewall interface is `eth0`, you do not need to modify the file provided with the sample. Otherwise, edit `/etc/shorewall/masq` and change the first column to the name of your external interface and the second column to the name of your internal interface.

If your external IP is static, you can enter it in the third column in the `/etc/shorewall/masq` entry if you like although your firewall will work fine if you leave that column empty. Entering your static IP in column 3 makes processing outgoing packets a little more efficient.

If you are using the Debian package, please check your `shorewall.conf` file to ensure that the following is set correctly; if it is not, change it appropriately:

- `IP_FORWARDING=On`

Port Forwarding (DNAT)

One of your goals may be to run one or more servers on your local computers. Because these computers have RFC-1918 addresses, it is not possible for clients on the internet to connect directly to them. It is rather necessary for those clients to address their connection requests to the firewall who rewrites the destination address to the address of your server and forwards the packet to that server. When your server responds, the firewall automatically performs SNAT to rewrite the source address in the response.

The above process is called *Port Forwarding* or *Destination Network Address Translation* (DNAT). You configure port forwarding using DNAT rules in the `/etc/shorewall/rules` file.

The general form of a simple port forwarding rule in `/etc/shorewall/rules` is:

```
#ACTION    SOURCE     DEST                                          PROTO       DEST
PORT(S)
DNAT       net        loc:<server local ip address>[:<server port>] <protocol> <port>
```

Shorewall has macros for many popular applications. Look at /usr/share/shorewall/macro.* to see what is available in your release. Macros simplify creating DNAT rules by supplying the protocol and port(s) as shown in the following examples.

Example 1. Web Server

You run a Web Server on computer 2 and you want to forward incoming TCP port 80 to that system:

```
#ACTION    SOURCE     DEST              PROTO      DEST PORT(S)
Web/DNAT   net        loc:192.168.1.5
```

Example 2. FTP Server

You run an FTP Server on computer 1 so you want to forward incoming TCP port 21 to that system:

```
#ACTION     SOURCE     DEST              PROTO      DEST PORT(S)
FTP/DNAT    net        loc:10.10.10.1
```

For FTP, you will also need to have FTP connection tracking and NAT support in your kernel. For vendor-supplied kernels, this means that the `ip_conntrack_ftp` and `ip_nat_ftp` modules must be loaded. Shorewall will automatically load these modules if they are available and located in the standard place under `/lib/modules/<kernel version>/kernel/net/ipv4/netfilter`.

A couple of important points to keep in mind:

- You must test the above rule from a client outside of your local network (i.e., don't test from a browser running on computers 1 or 2 or on the firewall). If you want to be able to access your web server and/or FTP server from inside your firewall using the IP address of your external interface, see Shorewall FAQ #2.
- Many ISPs block incoming connection requests to port 80. If you have problems connecting to your web server, try the following rule and try connecting to port 5000.

  ```
  #ACTION     SOURCE     DEST                PROTO      DEST PORT(S)
  DNAT        net        loc:10.10.10.2:80   tcp        5000
  ```

At this point, modify `/etc/shorewall/rules` to add any DNAT rules that you require.

Important

When testing DNAT rules like those shown above, you must test from a client OUTSIDE YOUR FIREWALL (in the 'net' zone). You cannot test these rules from inside the firewall!

For DNAT troubleshooting tips, see FAQs 1a and 1b.

Domain Name Server (DNS)

Normally, when you connect to your ISP, as part of getting an IP address your firewall's *Domain Name Service* (DNS) resolver will be automatically configured (e.g., the `/etc/resolv.conf` file will be written). Alternatively, your ISP may have given you the IP address of a pair of DNS name servers for you to manually configure as your primary and secondary name servers. Regardless of how DNS gets configured on your firewall, it is your responsibility to configure the resolver in your internal systems. You can take one of two approaches:

- You can configure your internal systems to use your ISP's name servers. If your ISP gave you the addresses of their servers or if those addresses are available on their web site, you can configure your internal systems to use those addresses. If that information isn't available, look in /etc/resolv.conf on your firewall system -- the name servers are given in "nameserver" records in that file.
- You can configure a *Caching Name Server* on your firewall. Red Hat™ has an RPM for a caching name server (the

RPM also requires the **bind**RPM) and for Bering users, there is **dnscache.lrp**. If you take this approach, you configure your internal systems to use the firewall itself as their primary (and only) name server. You use the internal IP address of the firewall (`10.10.10.254` in the example above) for the name server address. To allow your local systems to talk to your caching name server, you must open port 53 (both UDP and TCP) from the local network to the firewall; you do that by adding the following rules in `/etc/shorewall/rules`.

```
#ACTION     SOURCE     DEST                PROTO      DEST PORT(S)
DNS/ACCEPT  loc        $FW
```

Other Connections

The two-interface sample includes the following rules:

```
#ACTION      SOURCE     DEST                PROTO      DEST PORT(S)
DNS/ACCEPT   $FW        net
```

This rule allows DNS access from your firewall and may be removed if you uncommented the line in `/etc/shorewall/policy` allowing all connections from the firewall to the internet.

In the rule shown above, “DNS/ACCEPT” is an example of a *macro invocation*. Shorewall includes a number of macros (see `/usr/share/shorewall/macro.*`) and you can add your own.

You don't have to use defined macros when coding a rule in `/etc/shorewall/rules`; Shorewall will start slightly faster if you code your rules directly rather than using macros. The the rule shown above could also have been coded as follows:

```
#ACTION     SOURCE     DEST                PROTO      DEST PORT(S)
ACCEPT      $FW        net                 udp        53
ACCEPT      $FW        net                 tcp        53
```

In cases where Shorewall doesn't include a defined macro to meet your needs, you can either define the macro yourself or you can simply code the appropriate rules directly.

The sample also includes:

```
#ACTION       SOURCE     DEST                PROTO      DEST PORT(S)
SSH/ACCEPT    loc        $FW
```

That rule allows you to run an SSH server on your firewall and connect to that server from your local systems.

If you wish to enable other connections from your firewall to other systems, the general format using a macro is:

```
#ACTION          SOURCE     DEST                PROTO      DEST PORT(S)
<macro>/ACCEPT   $FW        <destination zone>
```

The general format when not using defined actions is:

```
#ACTION     SOURCE     DEST                PROTO      DEST PORT(S)
ACCEPT      $FW        <destination zone> <protocol> <port>
```

Example 3. Web Server on Firewall

You want to run a Web Server on your firewall system:

```
#ACTION      SOURCE     DEST                PROTO      DEST PORT(S)
Web/ACCEPT   net        $FW
Web/ACCEPT   loc
```

$FWThose two rules would of course be in addition to the rules listed above under "You can configure a Caching Name Server on your firewall".

If you don't know what port and protocol a particular application uses, look here.

Important

I don't recommend enabling **telnet** to/from the internet because it uses clear text (even for login!). If you want shell access to your firewall from the internet, use SSH:

```
#ACTION       SOURCE     DEST                PROTO      DEST PORT(S)
SSH/ACCEPT    net        $FW
```

Bering users will want to add the following two rules to be compatible with Jacques's Shorewall configuration.

```
#ACTION     SOURCE     DEST     PROTO      DEST PORT(S)
ACCEPT      loc        $FW      udp        53          #Allow DNS Cache to work
ACCEPT      loc        $FW      tcp        80          #Allow Weblet to work
```

Now edit your `/etc/shorewall/rules` file to add or delete other connections as required.

Some Things to Keep in Mind

- **You cannot test your firewall from the inside**. Just because you send requests to your firewall external IP address does not mean that the request will be associated with the external interface or the "net" zone. Any traffic that you generate from the local network will be associated with your local interface and will be treated as loc->fw traffic.
- **IP addresses are properties of systems, not of interfaces**. It is a mistake to believe that your firewall is able to forward packets just because you can ping the IP address of all of the firewall's interfaces from the local network. The only conclusion you can draw from such pinging success is that the link between the local system and the firewall works and that you probably have the local system's default gateway set correctly.
- **All IP addresses configured on firewall interfaces are in the $FW (fw) zone**. If 192.168.1.254 is the IP address of your internal interface then you can write "**$FW:192.168.1.254**" in a rule but you may not write "**loc:192.168.1.254**". Similarly, it is nonsensical to add 192.168.1.254 to the **loc** zone using an entry in `/etc/shorewall/hosts`.
- **Reply packets do NOT automatically follow the reverse path of the one taken by the original request**. All packets are routed according to the routing table of the host at each step of the way. This issue commonly comes up when people install a Shorewall firewall parallel to an existing gateway and try to use DNAT through Shorewall without changing the default gateway of the system receiving the forwarded requests. Requests come in through the Shorewall firewall where the destination IP address gets rewritten but replies go out unmodified through the old gateway.
- **Shorewall itself has no notion of inside or outside**. These concepts are embodied in how Shorewall is configured.

Starting and Stopping Your Firewall

The installation procedure configures your system to start Shorewall at system boot but startup is disabled so that your system won't try to start Shorewall before configuration is complete. Once you have completed configuration of your firewall, you must edit /etc/shorewall/shorewall.conf and set STARTUP_ENABLED=Yes.

Important

Users of the .deb package must edit `/etc/default/shorewall` and set `startup=1`.

The firewall is started using the "**shorewall start**" command and stopped using "**shorewall stop**". When the firewall is stopped, routing is enabled on those hosts that have an entry in `/etc/shorewall/routestopped`. A running firewall may be restarted using the "**shorewall restart**" command. If you want to totally remove any trace of Shorewall from your Netfilter configuration, use "**shorewall clear**".

The two-interface sample assumes that you want to enable routing to/from `eth1` (the local network) when Shorewall is stopped. If your local network isn't connected to `eth1` or if you wish to enable access to/from other hosts, change `/etc/shorewall/routestopped` accordingly.

Warning

If you are connected to your firewall from the internet, do not issue a "**shorewall stop**" command unless you have added an entry for the IP address that you are connected from to `/etc/shorewall/routestopped`. Also, I don't recommend using "**shorewall restart**"; it is better to create an alternate configuration and test it using the "**shorewall try**" command.

Additional Recommended Reading

I highly recommend that you review the Common Configuration File Features page -- it contains helpful tips about Shorewall features than make administering your firewall easier.

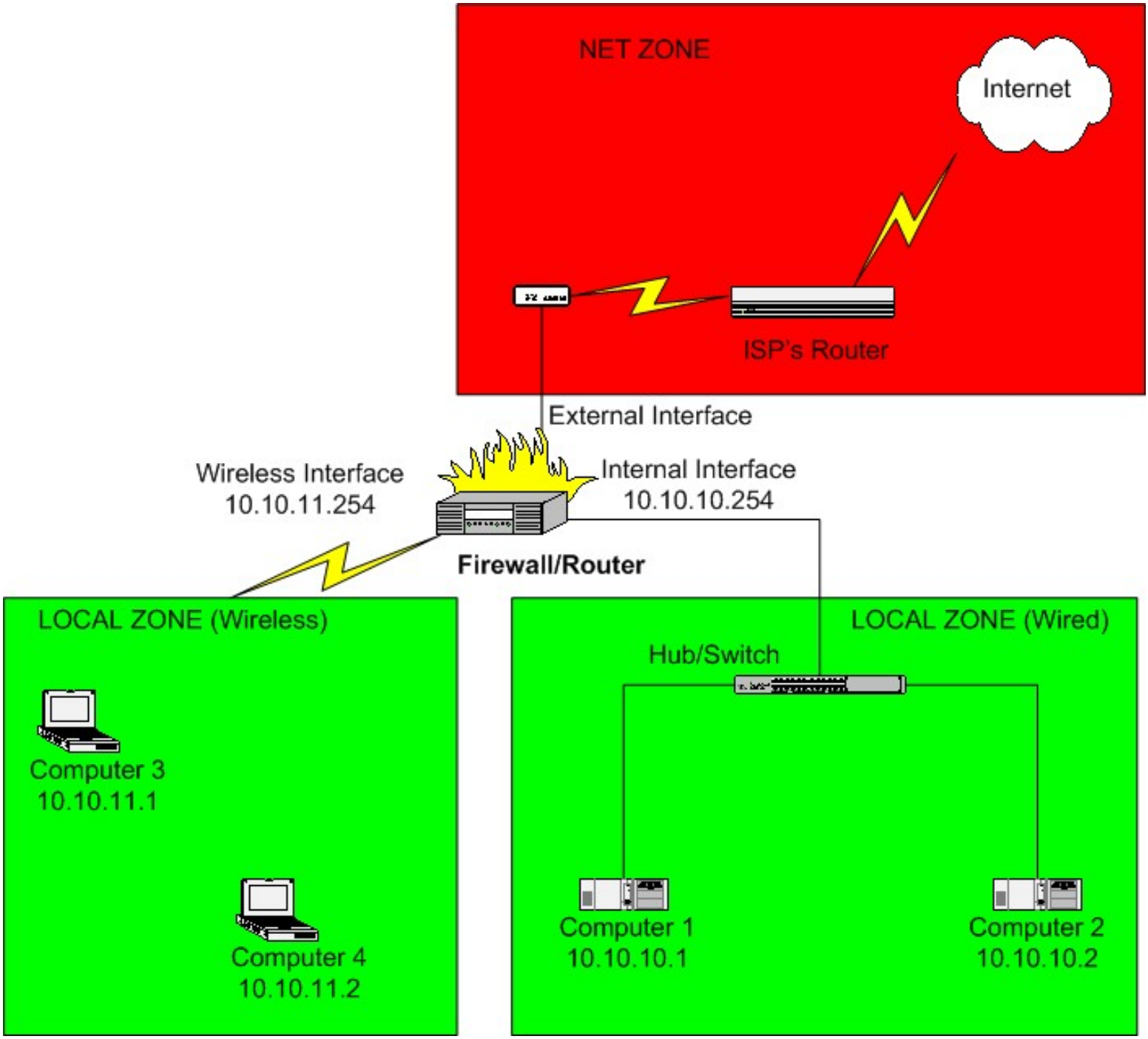
Adding a Wireless Segment to your Two-Interface Firewall

Once you have the two-interface setup working, the next logical step is to add a Wireless Network. The first step involves adding an additional network card to your firewall, either a Wireless card or an ethernet card that is connected to a Wireless Access Point.

Caution

When you add a network card, it won't necessarily be detected as the next highest ethernet interface. For example, if you have two ethernet cards in your system (`eth0` and `eth1`) and you add a third card that uses the same driver as one of the other two, that third card won't necessarily be detected as `eth2`; it could rather be detected as `eth0` or `eth1`! You can either live with that or you can shuffle the cards around in the slots until the new card is detected as `eth2`.

Your new network will look similar to what is shown in the following figure.



The first thing to note is that the computers in your wireless network will be in a different subnet from those on your wired local LAN. In the above example, we have chosen to use the network 10.10.11.0/24. Computers 3 and 4 would be configured with a default gateway IP address of 10.10.11.254.

Second, we have chosen to include the wireless network as part of the local zone. Since Shorewall allows intra-zone traffic by default, traffic may flow freely between the local wired network and the wireless network.

There are only two changes that need to be made to the Shorewall configuration:

- An entry needs to be added to `/etc/shorewall/interfaces` for the wireless network interface. If the wireless interface is `wlan0`, the entry might look like:

```
#ZONE      INTERFACE        BROADCAST           OPTIONS
```

```
loc        wlan0            detect              maclist
```

As shown in the above entry, I recommend using the maclist option for the wireless segment. By adding entries for computers 3 and 4 in `/etc/shorewall/maclist`, you help ensure that your neighbors aren't getting a free ride on your internet connection. Start by omitting that option; when you have everything working, then add the option and configure your `/etc/shorewall/maclist` file.

- You need to add an entry to the `/etc/shorewall/masq` file to masquerade traffic from the wireless network to the internet. If your internet interface is `eth0` and your wireless interface is `wlan0`, the entry would be:

```
#INTERFACE            SUBNET              ADDRESS
eth0                  wlan0
```

One other thing to note. To get Microsoft™ networking working between the wireless and wired networks, you will need either a WINS server or a PDC. I personally use Samba configured as a WINS server running on my firewall. Running a WINS server on your firewall requires the rules listed in the Shorewall/Samba documentation.

Shorewall Setup Guide

Tom Eastep

Copyright © 2001-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-11-01

Table of Contents

Introduction
Shorewall Concepts
Network Interfaces
Addressing, Subnets and Routing
- IP Addresses
- Subnets
- Routing
- Address Resolution Protocol (ARP)
- RFC 1918

Setting Up Your Network
- Routed
- Non-routed
  - SNAT
  - DNAT
  - Proxy ARP
  - One-to-one NAT
- Rules
- Odds and Ends

DNS
Some Things to Keep in Mind
Starting and Stopping the Firewall

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Introduction

This guide is intended for users who are setting up Shorewall in an environment where a set of public IP addresses must be managed or who want to know more about Shorewall than is contained in the single-address guides. Because the range of possible applications is so broad, the Guide will give you general guidelines and will point you to other resources as necessary.

Caution

If you run LEAF Bering, your Shorewall configuration is NOT what I release -- I suggest that you consider installing a stock Shorewall lrp from the shorewall.net site before you proceed. Shorewall requires that the iproute/iproute2

package be installed (on RedHat, the package is called iproute). You can tell if this package is installed by the presence of an **ip** program on your firewall system. As root, you can use the “which” command to check for this program:

```
[root@gateway root]# which ip
/sbin/ip
[root@gateway root]#
```

I recommend that you first read through the guide to familiarize yourself with what's involved then go back through it again making your configuration changes. Points at which configuration changes are recommended are flagged with ➡.

Caution

If you edit your configuration files on a Windows system, you must save them as Unix files if your editor supports that option or you must run them through dos2unix before trying to use them with Shorewall. Similarly, if you copy a configuration file from your Windows hard drive to a floppy disk, you must run dos2unix against the copy before using it with Shorewall.

- Windows Version of dos2unix
- Linux Version of dos2unix

Shorewall Concepts

➡

The configuration files for Shorewall are contained in the directory `/etc/shorewall` -- for most setups, you will only need to deal with a few of these as described in this guide. Skeleton files are created during the Shorewall Installation Process.

Warning

Note to Debian Users

If you install using the .deb, you will find that your `/etc/shorewall` directory is empty. This is intentional. The released configuration file skeletons may be found on your system in the directory `/usr/share/doc/shorewall/default-config`. Simply copy the files you need from that directory to `/etc/shorewall` and modify the copies.

Note that you must copy `/usr/share/doc/shorewall/default-config/shorewall.conf` and /usr/share/doc/shorewall/default-config/modules to `/etc/shorewall` even if you do not modify those files.

As each file is introduced, I suggest that you look through the actual file on your system -- each file contains detailed configuration instructions and some contain default entries.

Shorewall views the network where it is running as being composed of a set of zones. In this guide, we will use the following zones:

fw

The firewall system itself.

net

The public Internet.

loc

A private local network using private IP addresses.

dmz

A Demilitarized Zone holding publicly-accessible servers.

Zones are defined in the file `/etc/shorewall/zones`.

Important

The `/etc/shorewall/zones` file included in the release is empty. You can create the standard set of zones described above by copying and pasting the following into the file:

```
#ZONE   TYPE            OPTIONS
fw      firewall
net     ipv4
loc     ipv4
dmz     ipv4
```

Note that Shorewall recognizes the firewall system as its own zone - The above example follows the usual convention of naming the Firewall zone **fw**. The name specified for the firewall zone (**fw** in the above example) is stored in the shell variable *$FW* when the /etc/shorewall/zones file is processed. With the exception of the name assigned to the firewall zone, Shorewall attaches absolutely no meaning to zone names. Zones are entirely what YOU make of them. That means that you should not expect Shorewall to do something special “because this is the internet zone” or “because that is the DMZ”.

Edit the /etc/shorewall/zones file and make any changes necessary.

Rules about what traffic to allow and what traffic to deny are expressed in terms of zones.

- You express your default policy for connections from one zone to another zone in the `/etc/shorewall/policy` file.
- You define exceptions to those default policies in the `/etc/shorewall/rules`.

Shorewall is built on top of the Netfilter kernel facility. Netfilter implements a connection tracking function that allows what is often referred to as stateful inspection of packets. This stateful property allows firewall rules to be defined in terms of connections rather than in terms of packets. With Shorewall, you:

1. Identify the source (client) zone.
2. Identify destination (server) zone.
3. If the POLICY from the client's zone to the server's zone is what you want for this client/server pair, you need do nothing further.
4. If the POLICY is not what you want, then you must add a rule. That rule is expressed in terms of the client's zone and the server's zone.

Just because connections of a particular type are allowed from zone A to the firewall and are also allowed from the firewall to zone B **DOES NOT mean that these connections are allowed from zone A to zone B** (in other words, policies and rules involving the firewall zone are not transitibe). It rather means that you can have a proxy running on the firewall that accepts a connection from zone A and then establishes its own separate connection from the firewall to zone B.

For each connection request entering the firewall, the request is first checked against the `/etc/shorewall/rules` file. If no rule in that file matches the connection request then the first policy in `/etc/shorewall/policy` that matches the request is applied after the request is passed to the appropriate common action (if any).

Prior to Shorewall 2.2.0, the default `/etc/shorewall/policy` file had the following policies:

```
#SOURCE ZONE     DESTINATION ZONE    POLICY     LOG     LIMIT:BURST
#                                               LEVEL
loc              net                 ACCEPT
net              all                 DROP       info
all              all                 REJECT     info
```

Important

The currently released policy file is empty. You can copy and paste the above entries to create a starting point from which to customize your policies.

The above policies will:

1. allow all connection requests from your local network to the internet
2. drop (ignore) all connection requests from the internet to your firewall or local network and log a message at the info level (here is a description of log levels).
3. reject all other connection requests and log a message at the info level. When a request is rejected, the firewall will return an RST (if the protocol is TCP) or an ICMP port-unreachable packet for other protocols.

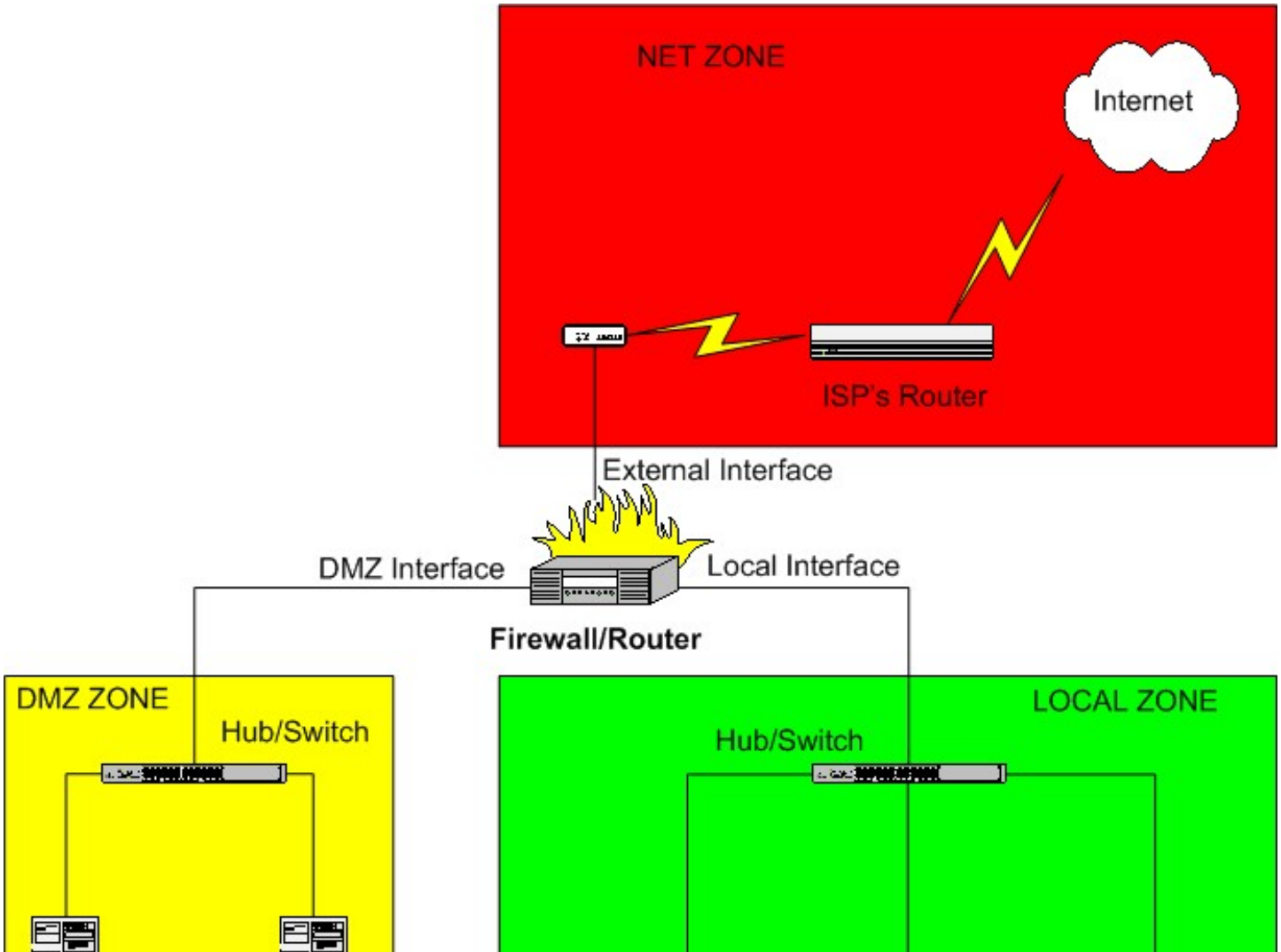
At this point, edit your `/etc/shorewall/policy` and make any changes that you wish.

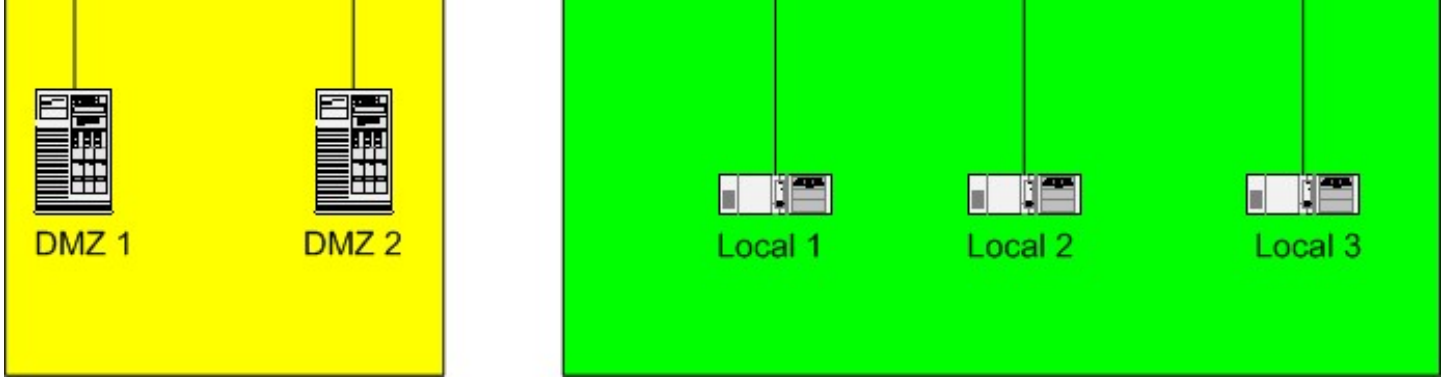
Network Interfaces

For the remainder of this guide, we'll refer to the following diagram. While it may not look like your own network, it can be used to illustrate the important aspects of Shorewall configuration.

In this diagram:

- The DMZ Zone consists of systems DMZ 1 and DMZ 2. A DMZ is used to isolate your internet-accessible servers from your local systems so that if one of those servers is compromised, you still have the firewall between the compromised system and your local systems.
- The Local Zone consists of systems Local 1, Local 2 and Local 3.
- All systems from the ISP outward comprise the Internet Zone.





The simplest way to define zones is to associate the zone name (previously defined in /etc/shorewall/zones) with a network interface. This is done in the /etc/shorewall/interfaces file. The firewall illustrated above has three network interfaces. Where Internet connectivity is through a cable or DSL "Modem", the *External Interface* will be the Ethernet adapter that is connected to that "Modem" (e.g., `eth0`) unless you connect via Point-to-Point Protocol over Ethernet (PPPoE) or Point-to-Point Tunneling Protocol (PPTP) in which case the External Interface will be a ppp interface (e.g., `ppp0`). If you connect via a regular modem, your External Interface will also be `ppp0`. If you connect using ISDN, you external interface will be `ippp0`.

If your external interface is `ppp0` or `ippp0` then you will want to set CLAMPMSS=yes in `/etc/shorewall/shorewall.conf`.

Your *Local Interface* will be an Ethernet adapter (`eth0`, `eth1` or `eth2`) and will be connected to a hub or switch. Your local computers will be connected to the same switch (note: If you have only a single local system, you can connect the firewall directly to the computer using a cross-over cable).

Your *DMZ Interface* will also be an Ethernet adapter (`eth0`, `eth1` or `eth2`) and will be connected to a hub or switch. Your DMZ computers will be connected to the same switch (note: If you have only a single DMZ system, you can connect the firewall directly to the computer using a cross-over cable).

Caution

Do not connect the internal and external interface to the same hub or switch except for testing AND you are running Shorewall version 1.4.7 or later. When using these recent versions, you can test using this kind of configuration if you specify the **arp_filter** option in `/etc/shorewall/interfaces` for all interfaces connected to the common hub/switch. Using such a setup with a production firewall is strongly recommended against.

For the remainder of this Guide, we will assume that:

- The External Interface is `eth0`.
- The Local Interface `eth1`.
- The DMZ Interface `eth2`.

The Shorewall default configuration does not define the contents of any zone. To define the above configuration using the /etc/shorewall/interfaces file, that file would might contain:

```
#ZONE    INTERFACE      BROADCAST     OPTIONS
net      eth0           detect        norfc1918
loc      eth1           detect
dmz      eth2           detect
```

Note that the **$FW** zone has no entry in the /etc/shorewall/interfaces file.

Edit the `/etc/shorewall/interfaces` file and define the network interfaces on your firewall and associate each interface with a zone. If you have a zone that is interfaced through more than one interface, simply include one entry for each interface and repeat the zone name as many times as necessary.

Example 1. Multiple Interfaces to a Zone

```
#ZONE    INTERFACE      BROADCAST     OPTIONS
net      eth0           detect        norfc1918
loc      eth1           detect
loc      eth2           detect
```

You may define more complicated zones using the `/etc/shorewall/hosts` file but in most cases, that isn't necessary. See Shorewall_and_Aliased_Interfaces.html and Multiple_Zones.html for examples.

Addressing, Subnets and Routing

Normally, your ISP will assign you a set of Public IP addresses. You will configure your firewall's external interface to use one of those addresses permanently and you will then have to decide how you are going to use the rest of your addresses. Before we tackle that question though, some background is in order.

If you are thoroughly familiar with IP addressing and routing, you may go to the next section.

The following discussion barely scratches the surface of addressing and routing. If you are interested in learning more about this subject, I highly recommend "*IP Fundamentals: What Everyone Needs to Know about Addressing & Routing*", Thomas A. Maufer, Prentice-Hall, 1999, ISBN 0-13-975483-0.

IP Addresses

IP version 4 (IPv4) addresses are 32-bit numbers. The notation w.x.y.z refers to an address where the high-order byte has value "w", the next byte has value "x", etc. If we take the address 192.0.2.14 and express it in hexadecimal, we get:

```
C0.00.02.0E
```

or looking at it as a 32-bit integer

```
C000020E
```

Subnets

You will still hear the terms "Class A network", "Class B network" and "Class C network". In the early days of IP, networks only came in three sizes (there were also Class D networks but they were used differently):

Class A - netmask 255.0.0.0, size = 2 ** 24
Class B - netmask 255.255.0.0, size = 2 ** 16
Class C - netmask 255.255.255.0, size = 256

The class of a network was uniquely determined by the value of the high order byte of its address so you could look at an IP address and immediately determine the associated netmask. The netmask is a number that when logically ANDed with an address isolates the network number; the remainder of the address is the host number. For example, in the Class C address 192.0.2.14, the network number is hex C00002 and the host number is hex 0E.

As the internet grew, it became clear that such a gross partitioning of the 32-bit address space was going to be very limiting (early on, large corporations and universities were assigned their own class A network!). After some false starts, the current technique of subnetting these networks into smaller subnetworks evolved; that technique is referred to as *Classless InterDomain Routing* (CIDR). Today, any system that you are likely to work with will understand CIDR and Class-based networking is largely a thing of the past.

A *subnetwork* (often referred to as a *subnet*) is a contiguous set of IP addresses such that:

1. The number of addresses in the set is a power of 2; and
2. The first address in the set is a multiple of the set size.
3. The first address in the subnet is reserved and is referred to as the *subnet address*.
4. The last address in the subnet is reserved as the subnet's *broadcast address*.

As you can see by this definition, in each subnet of size n there are (n - 2) usable addresses (addresses that can be assigned to hosts). The first and last address in the subnet are used for the subnet address and subnet broadcast address respectively. Consequently, small subnetworks are more wasteful of IP addresses than are large ones.

Since n is a power of two, we can easily calculate the *Base-2 Logarithm* (log2) of n. For the more common subnet sizes, the size and its base-2 logarithm are given in the following table:

Table 1. Natural Logarithms

n	**log2 n**	**(32 - log2 n)**
8	3	29
16	4	28
32	5	27
64	6	26
128	7	25
256	8	24
512	9	23
1024	10	22
2048	11	21
4096	12	20
8192	13	19
16384	14	18
32768	15	17
65536	16	16

You will notice that the above table also contains a column for (32 - log2 **n)**. That number is the *Variable Length Subnet Mask* (VLSM) for a network of size n. From the above table, we can derive the following one which is a little easier to use.

Table 2. VLSM

Subnet Size	**VLSM**	**Subnet Mask**
8	/29	255.255.255.248
16	/28	255.255.255.240
32	/27	255.255.255.224
64	/26	255.255.255.192
128	/25	255.255.255.128
256	/24	255.255.255.0
512	/23	255.255.254.0
1024	/22	255.255.252.0
2048	/21	255.255.248.0
4096	/20	255.255.240.0
8192	/19	255.255.224.0

16384	/18	255.255.192.0
32768	/17	255.255.128.0
65536	/16	255.255.0.0
2 ** 24	/8	255.0.0.0

Notice that the VLSM is written with a slash ("/") -- you will often hear a subnet of size 64 referred to as a "slash 26" subnet and one of size 8 referred to as a "slash 29".

The subnet's mask (also referred to as its *netmask*) is simply a 32-bit number with the first "VLSM" bits set to one and the remaining bits set to zero. For example, for a subnet of size 64, the subnet mask has 26 leading one bits:

```
11111111111111111111111111000000 = FFFFFFC0 = FF.FF.FF.C0 = 255.255.255.192
```

The subnet mask has the property that if you logically AND the subnet mask with an address in the subnet, the result is the subnet address. Just as important, if you logically AND the subnet mask with an address outside the subnet, the result is NOT the subnet address. As we will see below, this property of subnet masks is very useful in routing.

For a subnetwork whose address is **a.b.c.d** and whose Variable Length Subnet Mask is **/v**, we denote the subnetwork as "**a.b.c.d/v**" using *CIDR Notation*. Example:

Table 3. Subnet

Subnet:	10.10.10.0 - 10.10.10.127
Subnet Size:	128
Subnet Address:	10.10.10.0
Broadcast Address:	10.10.10.127
CIDR Notation:	10.10.10.0/25

There are two degenerate subnets that need mentioning; namely, the subnet with one member and the subnet with 2 ** 32 members.

Table 4. /32 and /0

Subnet Size	VLSM Length	Subnet Mask	CIDR Notation
1	32	255.255.255.255	a.b.c.d/32
32	0	0.0.0.0	0.0.0.0/0

So any address **a.b.c.d** may also be written **a.b.c.d/32** and the set of all possible IP addresses is written **0.0.0.0/0**.

A Shorewall user has contributed a useful graphical summary of the above information.

Later in this guide, you will see the notation **a.b.c.d/v** used to describe the ip configuration of a network interface (the "ip" utility also uses this syntax). This simply means that the interface is configured with ip address **a.b.c.d** and with the netmask that corresponds to VLSM /**v**.

Example 2. 192.0.2.65/29

The interface is configured with IP address 192.0.2.65 and netmask 255.255.255.248.

Beginning with Shorewall 1.4.6, /sbin/shorewall supports an ipcalc command that automatically calculates information about a [sub]network.

Example 3. Using the ipcalc command

```
shorewall ipcalc 10.10.10.0/25
   CIDR=10.10.10.0/25
   NETMASK=255.255.255.128
   NETWORK=10.10.10.0
   BROADCAST=10.10.10.127
```

Example 4. Using the ipcalc command

```
shorewall ipcalc 10.10.10.0 255.255.255.128
   CIDR=10.10.10.0/25
   NETMASK=255.255.255.128
   NETWORK=10.10.10.0
   BROADCAST=10.10.10.127
```

Routing

One of the purposes of subnetting is that it forms the basis for routing. Here's the routing table on my firewall (compressed for PDF):

```
[root@gateway root]# netstat -nr
Kernel IP routing table
Destination     Gateway         Genmask         Flgs MSS Win irtt Iface
192.168.9.1     0.0.0.0         255.255.255.255 UH   40  0      0 texas
206.124.146.177 0.0.0.0         255.255.255.255 UH   40  0      0 eth1
206.124.146.180 0.0.0.0         255.255.255.255 UH   40  0      0 eth3
192.168.3.0     0.0.0.0         255.255.255.0   U    40  0      0 eth3
192.168.2.0     0.0.0.0         255.255.255.0   U    40  0      0 eth1
192.168.1.0     0.0.0.0         255.255.255.0   U    40  0      0 eth2
206.124.146.0   0.0.0.0         255.255.255.0   U    40  0      0 eth0
192.168.9.0     192.0.2.223     255.255.255.0   UG   40  0      0 texas
127.0.0.0       0.0.0.0         255.0.0.0       U    40  0      0 lo
0.0.0.0         206.124.146.254 0.0.0.0         UG   40  0      0 eth0
[root@gateway root]#
```

The device *texas* is a GRE tunnel to a peer site in the Dallas, Texas area.

The first three routes are *host routes* since they indicate how to get to a single host. In the “netstat” output this can be seen by the “Genmask” (Subnet Mask) of 255.255.255.255 and the “H” in the Flags column. The remainder are *“net” routes* since they tell the kernel how to route packets to a subnetwork. The last route is the *default route* and the gateway mentioned in that route is called the *default gateway*.

When the kernel is trying to send a packet to IP address **A**, it starts at the top of the routing table and:

- **A** is logically ANDed with the “Genmask” value in the table entry.
- The result is compared with the “Destination” value in the table entry.
- If the result and the “Destination” value are the same, then:
  - If the “Gateway” column is non-zero, the packet is sent to the gateway over the interface named in the “Iface” column.
  - Otherwise, the packet is sent directly to **A** over the interface named in the “iface” column.
- Otherwise, the above steps are repeated on the next entry in the table.

Since the default route matches any IP address (**A** LAND 0.0.0.0 = 0.0.0.0), packets that don't match any of the other routing table entries are sent to the default gateway which is usually a router at your ISP. Lets take an example. Suppose that we want to route a packet to 192.168.1.5. That address clearly doesn't match any of the host routes in the table but if we logically and that address with 255.255.255.0, the result is 192.168.1.0 which matches this routing table entry:

```
192.168.1.0   0.0.0.0       255.255.255.0   U     40  0         0 eth2
```

So to route a packet to 192.168.1.5, the packet is sent directly over eth2.

One more thing needs to be emphasized -- all outgoing packet are sent using the routing table and reply packets are not a special case. There seems to be a common mis-conception whereby people think that request packets are like salmon and contain a genetic code that is magically transferred to reply packets so that the replies follow the reverse route taken by the request. That isn't the case; the replies may take a totally different route back to the client than was taken by the requests -- they are totally independent.

Address Resolution Protocol (ARP)

When sending packets over Ethernet, IP addresses aren't used. Rather Ethernet addressing is based on *Media Access Control* (MAC) addresses. Each Ethernet device has it's own unique MAC address which is burned into a PROM on the device during manufacture. You can obtain the MAC of an Ethernet device using the “ip” utility:

```
[root@gateway root]# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc htb qlen 100
link/ether 02:00:08:e3:fa:55 brd ff:ff:ff:ff:ff:ff
inet 206.124.146.176/24 brd 206.124.146.255 scope global eth0
inet 206.124.146.178/24 brd 206.124.146.255 scope global secondary eth0
inet 206.124.146.179/24 brd 206.124.146.255 scope global secondary eth0
[root@gateway root]#
```

As you can see from the above output, the MAC is 6 bytes (48 bits) wide. A card's MAC is usually also printed on a label attached to the card itself. Because IP uses IP addresses and Ethernet uses MAC addresses, a mechanism is required to translate an IP address into a MAC address; that is the purpose of the *Address Resolution Protocol* (ARP). Here is ARP in action:

```
[root@gateway root]# tcpdump -nei eth2 arp
tcpdump: listening on eth2
09:56:49.766757 2:0:8:e3:4c:48 0:6:25:aa:8a:f0 arp 42:
                arp who-has 192.168.1.19 tell 192.168.1.254
09:56:49.769372 0:6:25:aa:8a:f0 2:0:8:e3:4c:48 arp 60:
                arp reply 192.168.1.19 is-at 0:6:25:aa:8a:f0

2 packets received by filter
0 packets dropped by kernel
[root@gateway root]#
```

In this exchange, 192.168.1.254 (MAC 2:0:8:e3:4c:48) wants to know the MAC of the device with IP address 192.168.1.19. The system having that IP address is responding that the MAC address of the device with IP address 192.168.1.19 is 0:6:25:aa:8a:f0.

In order to avoid having to exchange ARP information each time that an IP packet is to be sent, systems maintain an *ARP cache* of IP<->MAC correspondences. You can see the ARP cache on your system (including your Windows system) using the “arp” command:

```
[root@gateway root]# arp -na
? (206.124.146.177) at 00:A0:C9:15:39:78 [ether] on eth1
? (192.168.1.3) at 00:A0:CC:63:66:89 [ether] on eth2
? (192.168.1.5) at 00:A0:CC:DB:31:C4 [ether] on eth2
? (206.124.146.254) at 00:03:6C:8A:18:38 [ether] on eth0
? (192.168.1.19) at 00:06:25:AA:8A:F0 [ether] on eth2
```

The leading question marks are a result of my having specified the “n” option (Windows “arp” doesn't allow that option) which causes the “arp” program to forego IP->DNS name translation. Had I not given that option, the question marks would have been replaced with the FQDN corresponding to each IP address. Notice that the last entry in the table records the information we saw using tcpdump above.

RFC 1918

IP addresses are allocated by the Internet Assigned Number Authority (IANA) who delegates allocations on a geographic basis to Regional Internet Registries (RIRs). For example, allocation for the Americas and for sub-Sahara Africa is delegated to the American Registry for Internet Numbers (ARIN). These RIRs may in turn delegate to national registries. Most of us don't deal with these registrars but rather get our IP addresses from our ISP. It's a fact of life that most of us can't afford as many Public IP addresses as we have devices to assign them to so we end up making use of Private IP addresses. RFC 1918 reserves several IP address ranges for this purpose:

```
10.0.0.0    - 10.255.255.255
172.16.0.0  - 172.31.255.255
192.168.0.0 - 192.168.255.255
```

The addresses reserved by RFC 1918 are sometimes referred to as *non-routable* because the Internet backbone routers don't forward packets which have an RFC-1918 destination address. This is understandable given that anyone can select any of these addresses for their private use but the term non-routable is somewhat unfortunate because it leads people to the erroneous conclusion that traffic destined for one of these addresses can't be sent through a router. This is definitely not true; private routers (including your Shorewall-based firewall) can forward RFC 1918 addresed traffic just fine.

When selecting addresses from these ranges, there's a couple of things to keep in mind:

- As the IPv4 address space becomes depleted, more and more organizations (including ISPs) are beginning to use RFC 1918 addresses in their infrastructure.
- You don't want to use addresses that are being used by your ISP or by another organization with whom you want to establish a VPN relationship.

So it's a good idea to check with your ISP to see if they are using (or are planning to use) private addresses before you decide the addresses that you are going to use.

Warning

In this document, external “real” IP addresses are of the form 192.0.2.x. 192.0.2.0/24 is reserved by RFC 3330 for use as public IP addresses in printed examples and test networks. These "real" addresses are not to be confused with addresses in 192.168.0.0/16; as described above, those addresses are reserved by RFC 1918 for private use.

Setting Up Your Network

The choice of how to set up your network depends primarily on how many Public IP addresses you have vs. how many addressable entities you have in your network. Regardless of how many addresses you have, your ISP will handle that set of addresses in one of two ways:

- **Routed** - Traffic to any of your addresses will be routed through a single gateway address. This will generally only be done if your ISP has assigned you a complete subnet (/29 or larger). In this case, you will assign the gateway address as the IP address of your firewall/router's external interface.
- **Non-routed** - Your ISP will send traffic to each of your addresses directly.

In the subsections that follow, we'll look at each of these separately.

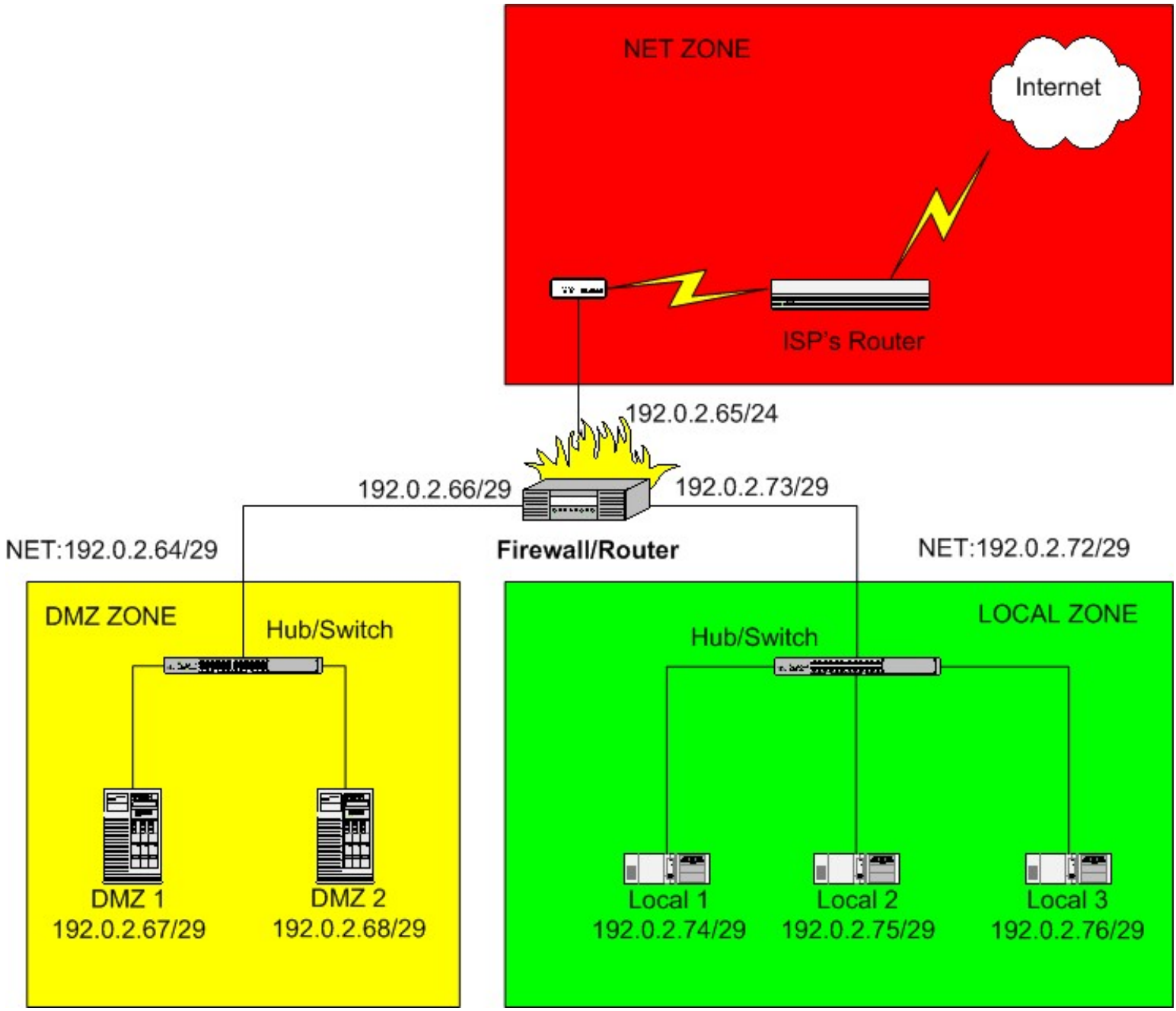
Before we begin, there is one thing for you to check:

If you are using the Debian package, please check your shorewall.conf file to ensure that the following are set correctly; if they are not, change them appropriately:

- NAT_ENABLED=Yes (Shorewall versions earlier than 1.4.6)
- IP_FORWARDING=On

Routed

Let's assume that your ISP has assigned you the subnet 192.0.2.64/28 routed through 192.0.2.65. That means that you have IP addresses 192.0.2.64 - 192.0.2.79 and that your firewall's external IP address is 192.0.2.65. Your ISP has also told you that you should use a netmask of 255.255.255.0 (so your /28 is part of a larger /24). With this many IP addresses, you are able to subnet your /28 into two /29's and set up your network as shown in the following diagram.



Here, the DMZ comprises the subnet 192.0.2.64/29 and the Local network is 192.0.2.72/29. The default gateway for hosts in the DMZ would be configured to 192.0.2.66 and the default gateway for hosts in the local network would be 192.0.2.73.

Notice that this arrangement is rather wasteful of public IP addresses since it is using 192.0.2.64 and 192.0.2.72 for subnet addresses, 192.0.2.71 and 192.0.2.79 for subnet broadcast addresses and 192.0.2.66 and 168.0.2.73 for internal addresses on the firewall/router. Nevertheless, it shows how subnetting can work and if we were dealing with a /24 rather than a /28 network, the use of 6 IP addresses out of 256 would be justified because of the simplicity of the setup.

The astute reader may have noticed that the Firewall/Router's external interface is actually part of the DMZ subnet (192.0.2.64/29). What if DMZ 1 (192.0.2.67) tries to communicate with 192.0.2.65? The routing table on DMZ 1 will look like this:

```
Kernel IP routing table
Destination     Gateway     Genmask         Flags MSS Window irtt Iface
```

```
192.0.2.64      0.0.0.0     255.255.255.248 U     40  0         0 eth0
0.0.0.0         192.0.2.66  0.0.0.0         UG    40  0         0 eth0
```

This means that DMZ 1 will send an ARP “who-has 192.0.2.65” request and no device on the DMZ Ethernet segment has that IP address. Oddly enough, the firewall will respond to the request with the MAC address of its DMZ Interface!! DMZ 1 can then send Ethernet frames addressed to that MAC address and the frames will be received (correctly) by the firewall/router.

It is this rather unexpected ARP behavior on the part of the Linux Kernel that prompts the warning earlier in this guide regarding the connecting of multiple firewall/router interfaces to the same hub or switch. When an ARP request for one of the firewall/router's IP addresses is sent by another system connected to the hub/switch, all of the firewall's interfaces that connect to the hub/switch can respond! It is then a race as to which “here-is” response reaches the sender first.

Non-routed

If you have the above situation but it is non-routed, you can configure your network exactly as described above with one additional twist; simply specify the “proxyarp” option on all three firewall interfaces in the /etc/shorewall/interfaces file.

Most of us don't have the luxury of having enough public IP addresses to set up our networks as shown in the preceding example (even if the setup is routed).

For the remainder of this section, assume that your ISP has assigned you IP addresses 192.0.2.176-180 and has told you to use netmask 255.255.255.0 and default gateway 192.0.2.254.

Clearly, that set of addresses doesn't comprise a subnetwork and there aren't enough addresses for all of the network interfaces. There are four different techniques that can be used to work around this problem.

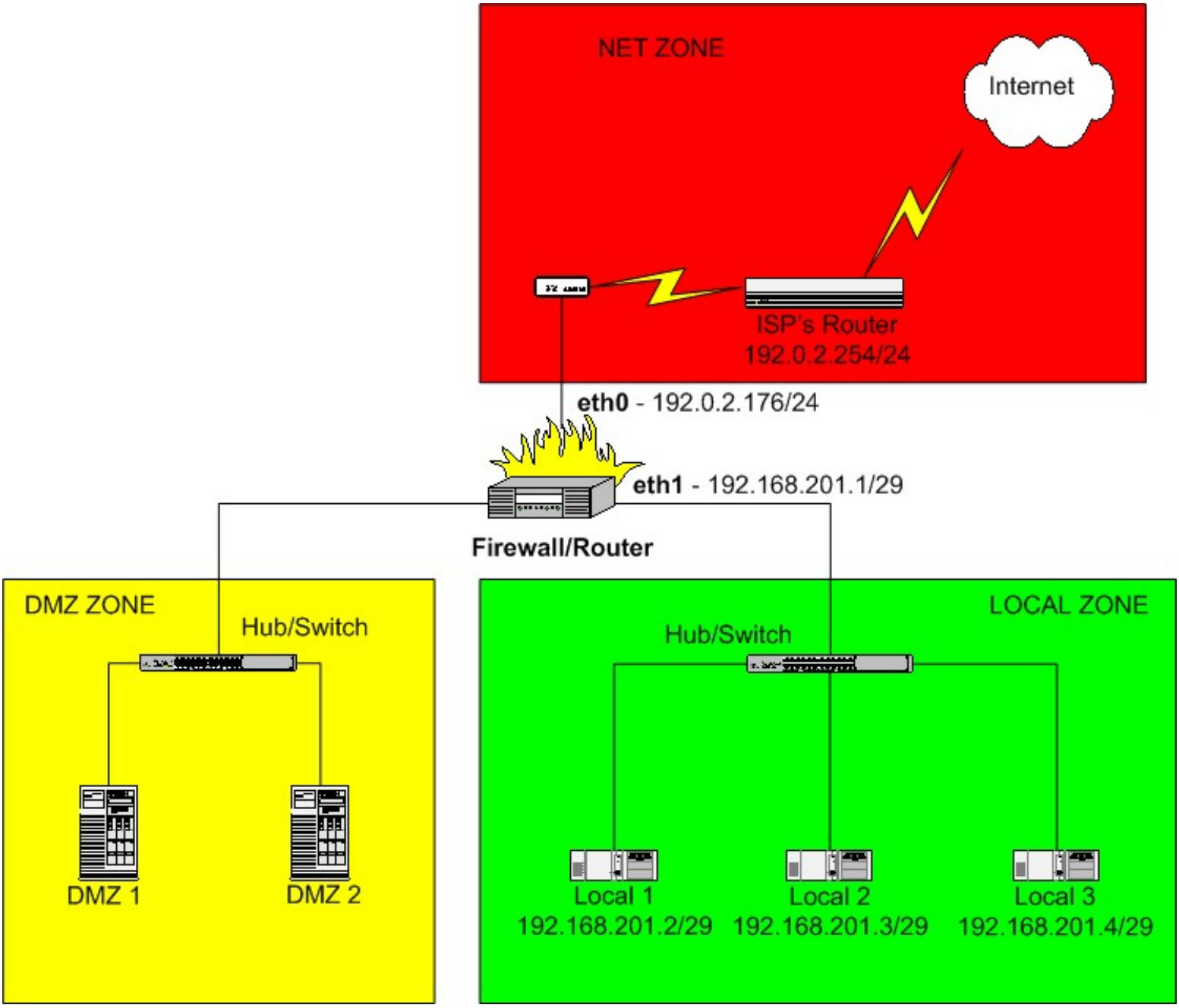
- *Source Network Address Translation* (SNAT).
- *Destination Network Address Translation* (DNAT) also known as *Port Forwarding*.
- *Proxy ARP*.
- *Network Address Translation* (NAT) also referred to as *One-to-one NA*T.

Often a combination of these techniques is used. Each of these will be discussed in the sections that follow.

SNAT

With SNAT, an internal LAN segment is configured using RFC 1918 addresses. When a host **A** on this internal segment initiates a connection to host **B** on the internet, the firewall/router rewrites the IP header in the request to use one of your public IP addresses as the source address. When **B** responds and the response is received by the firewall, the firewall changes the destination address back to the RFC 1918 address of **A** and forwards the response back to **A.**

Let's suppose that you decide to use SNAT on your local zone and use public address 192.0.2.176 as both your firewall's external IP address and the source IP address of internet requests sent from that zone.



The local zone has been subnetted as 192.168.201.0/29 (netmask 255.255.255.248).

The systems in the local zone would be configured with a default gateway of 192.168.201.1 (the IP address of the firewall's local interface).

SNAT is configured in Shorewall using the `/etc/shorewall/masq` file.

```
#INTERFACE      SUBNET              ADDRESS
eth0            192.168.201.0/29    192.0.2.176
```

This example used the normal technique of assigning the same public IP address for the firewall external interface and for SNAT. If you wanted to use a different IP address, you would either have to use your distributions network configuration tools to add that IP address to the external interface or you could set ADD_SNAT_ALIASES=Yes in /etc/shorewall/shorewall.conf and Shorewall will add the address for you.

DNAT

When SNAT is used, it is impossible for hosts on the internet to initiate a connection to one of the internal systems since those systems do not have a public IP address. DNAT provides a way to allow selected connections from the internet.

Suppose that your daughter wants to run a web server on her system “Local 3”. You could allow connections to the internet to her server by adding the following entry in `/etc/shorewall/rules`:

```
#ACTION   SOURCE   DEST                PROTO   DEST     SOURCE     ORIGINAL
#                                              PORT(S)  PORT(S)    DEST
DNAT      net      loc:192.168.201.4   tcp     www
```

If one of your daughter's friends at address **A** wants to access your daughter's server, she can connect to http://192.0.2.176 (the firewall's external IP address) and the firewall will rewrite the destination IP address to 192.168.201.4 (your daughter's system) and forward the request. When your daughter's server responds, the firewall will rewrite the source address back to 192.0.2.176 and send the response back to **A**.

This example used the firewall's external IP address for DNAT. You can use another of your public IP addresses (place it in the ORIGINAL DEST column in the rule above) but Shorewall will not add that address to the firewall's external interface for you.

Important

When testing DNAT rules like those shown above, you must test from a client OUTSIDE YOUR FIREWALL (in the 'net' zone). You cannot test these rules from inside the firewall!

For DNAT troubleshooting tips, see FAQs 1a and 1b.

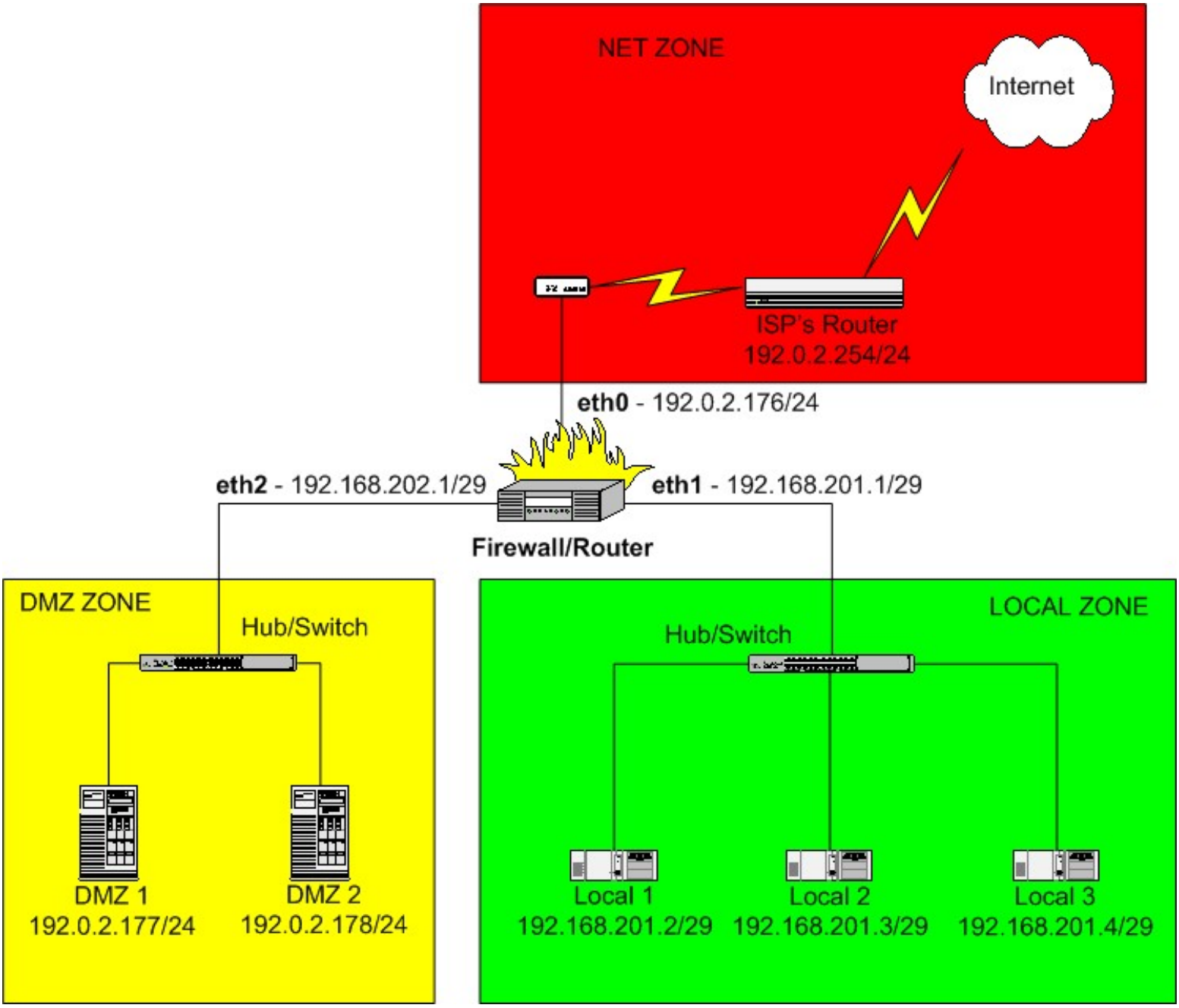
Proxy ARP

The idea behind Proxy ARP is that:

- A host **H** behind your firewall is assigned one of your public IP addresses (**A**), and is assigned the same netmask (**M**) as the firewall's external interface.
- The firewall responds to ARP “who has” requests for **A** from machines outside of the firewall.
- When **H** issues an ARP “who has” request for a machine with an address in the network defined by **M** where the target machine is outside of the firewall, the firewall will respond to **H** (with the MAC of the firewall interface that **H** is connected to).

For a more complete description of how Proxy ARP works, please see the Shorewall Proxy Documentation.

Let us suppose that we decide to use Proxy ARP on the DMZ in our example network.



Here, we've assigned the IP addresses 192.0.2.177 to system DMZ 1 and 192.0.2.178 to DMZ 2. Notice that we've just assigned an arbitrary RFC 1918 IP address and subnet mask to the DMZ interface on the firewall. That address and netmask isn't relevant - just be sure it doesn't overlap another subnet that you've defined.

The Shorewall configuration of Proxy ARP is done using the `/etc/shorewall/proxyarp` file.

```
#ADDRESS      INTERFACE     EXTERNAL      HAVE ROUTE          PERSISTANT
192.0.2.177   eth2          eth0          No
192.0.2.178   eth2          eth0          No
```

Because the HAVE ROUTE column contains No, Shorewall will add host routes thru eth2 to 192.0.2.177 and 192.0.2.178. The ethernet interfaces on DMZ 1 and DMZ 2 should be configured to have the IP addresses shown but should have the same default gateway as the firewall itself -- namely 192.0.2.254. In other words, they should be configured just like they would be if they were parallel to the firewall rather than behind it.

Caution

Do not add the Proxy ARP'ed address(es) (192.0.2.177 and 192.0.2.178 in the above example) to the external

interface (eth0 in this example) of the firewall.

A word of warning is in order here. ISPs typically configure their routers with a long ARP cache timeout. If you move a system from parallel to your firewall to behind your firewall with Proxy ARP, it will probably be HOURS before that system can communicate with the internet. There are a couple of things that you can try:

1. (Courtesy of Bradey Honsinger) A reading of Stevens' TCP/IP Illustrated, Vol 1 reveals that a

   > “gratuitous” ARP packet should cause the ISP's router to refresh their ARP cache (section 4.7). A gratuitous ARP is simply a host requesting the MAC address for its own IP; in addition to ensuring that the IP address isn't a duplicate,...
   >
   > “if the host sending the gratuitous ARP has just changed its hardware address..., this packet causes any other host...that has an entry in its cache for the old hardware address to update its ARP cache entry accordingly.”

   Which is, of course, exactly what you want to do when you switch a host from being exposed to the Internet to behind Shorewall using proxy ARP (or one-to-one NAT for that matter). Happily enough, recent versions of Redhat's iputils package include “arping”, whose “-U” flag does just that:

   ```
   arping -U -I <net if> <newly proxied IP>

   arping -U -I eth0 66.58.99.83 # for example
   ```

   Stevens goes on to mention that not all systems respond correctly to gratuitous ARPs, but googling for “arping -U” seems to support the idea that it works most of the time.

2. You can call your ISP and ask them to purge the stale ARP cache entry but many either can't or won't purge individual entries.

You can determine if your ISP's gateway ARP cache is stale using ping and tcpdump. Suppose that we suspect that the gateway router has a stale ARP cache entry for 192.0.2.177. On the firewall, run tcpdump as follows:

```
tcpdump -nei eth0 icmp
```

Now from 192.0.2.177, ping the ISP's gateway (which we will assume is 192.0.2.254):

```
ping 192.0.2.254
```

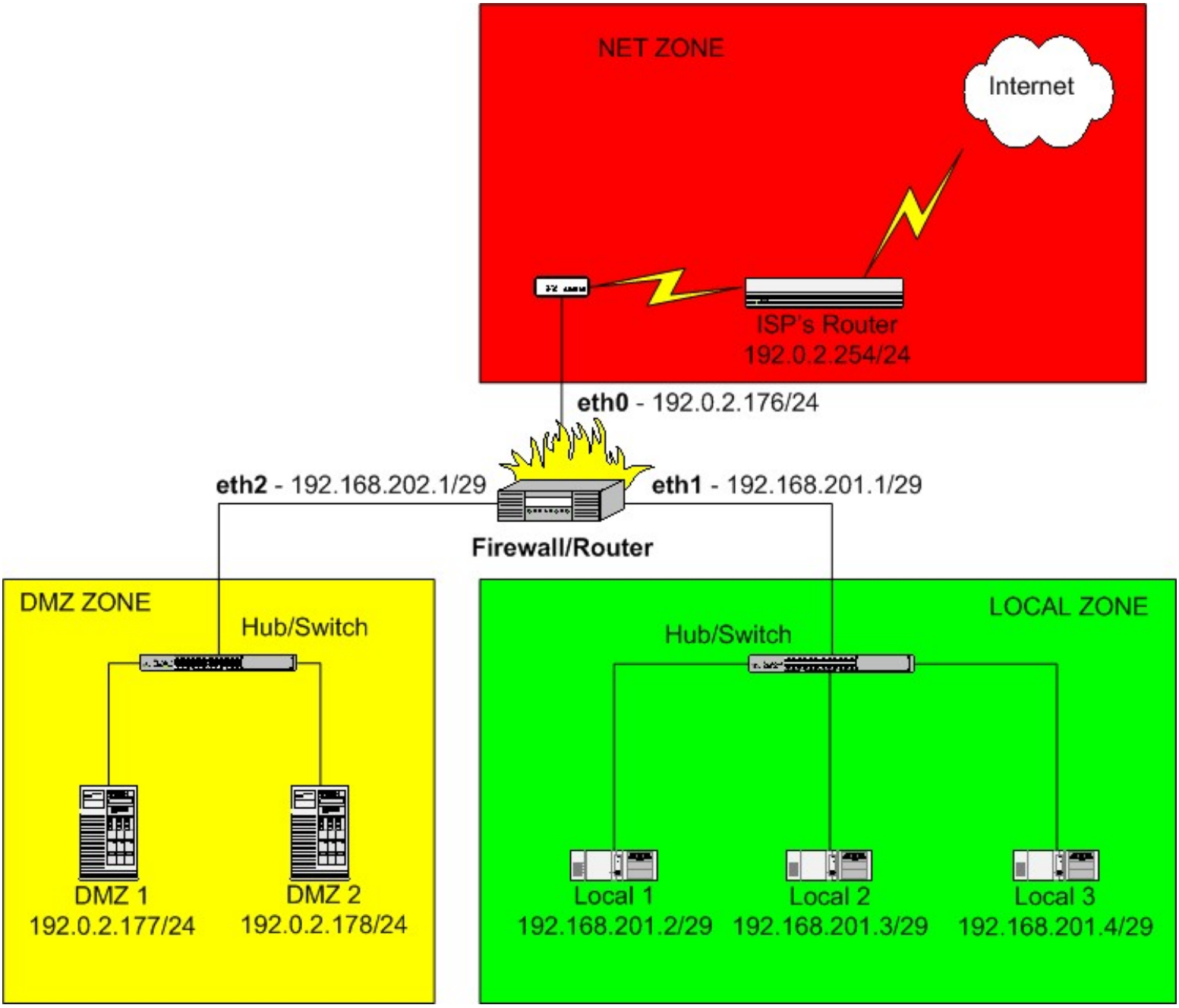
We can now observe the tcpdump output:

```
13:35:12.159321 0:4:e2:20:20:33 0:0:77:95:dd:19 ip 98:
                192.0.2.177 > 192.0.2.254: icmp: echo request (DF)
13:35:12.207615 0:0:77:95:dd:19 0:c0:a8:50:b2:57 ip 98:
                192.0.2.254 > 192.0.2.177 : icmp: echo reply
```

Notice that the source MAC address in the echo request is different from the destination MAC address in the echo reply!! In this case 0:4:e2:20:20:33 was the MAC of the firewall's eth0 NIC while 0:c0:a8:50:b2:57 was the MAC address of DMZ 1. In other words, the gateway's ARP cache still associates 192.0.2.177 with the NIC in DMZ 1 rather than with the firewall's eth0.

One-to-one NAT

With one-to-one NAT, you assign local systems RFC 1918 addresses then establish a one-to-one mapping between those addresses and public IP addresses. For outgoing connections SNAT (Source Network Address Translation) occurs and on incoming connections DNAT (Destination Network Address Translation) occurs. Let's go back to our earlier example involving your daughter's web server running on system Local 3.



Recall that in this setup, the local network is using SNAT and is sharing the firewall external IP (192.0.2.176) for outbound connections. This is done with the following entry in `/etc/shorewall/masq`:

```
#INTERFACE      SUBNET                ADDRESS
eth0            192.168.201.0/29      192.0.2.176
```

Suppose now that you have decided to give your daughter her own IP address (192.0.2.179) for both inbound and outbound connections. You would do that by adding an entry in `/etc/shorewall/nat`.

```
#EXTERNAL    INTERFACE    INTERNAL        ALL INTERFACES    LOCAL
192.0.2.179  eth0         192.168.201.4   No                No
```

With this entry in place, you daughter has her own IP address and the other two local systems share the firewall's IP address.

Once the relationship between 192.0.2.179 and 192.168.201.4 is established by the nat file entry above, it is no longer appropriate to use a DNAT rule for you daughter's web server -- you would rather just use an ACCEPT rule:

```
#ACTION   SOURCE   DEST                PROTO   DEST     SOURCE     ORIGINAL
#                                             PORT(S)  PORT(S)    DEST
ACCEPT    net      loc:192.168.201.4   tcp     www
```

A word of warning is in order here. ISPs typically configure their routers with a long ARP cache timeout. If you move a system from parallel to your firewall to behind your firewall with one-to-one NAT, it will probably be HOURS before that system can communicate with the internet. There are a couple of things that you can try:

1. (Courtesy of Bradey Honsinger) A reading of Stevens' TCP/IP Illustrated, Vol 1 reveals that a

   > “gratuitous” ARP packet should cause the ISP's router to refresh their ARP cache (section 4.7). A gratuitous ARP is simply a host requesting the MAC address for its own IP; in addition to ensuring that the IP address isn't a duplicate,...
   >
   > “if the host sending the gratuitous ARP has just changed its hardware address..., this packet causes any other host...that has an entry in its cache for the old hardware address to update its ARP cache entry accordingly.”

   Which is, of course, exactly what you want to do when you switch a host from being exposed to the Internet to behind Shorewall using one-to-one NAT. Happily enough, recent versions of Redhat's iputils package include “arping”, whose “-U” flag does just that:

   ```
   arping -U -I <net if> <newly proxied IP>

   arping -U -I eth0 66.58.99.83 # for example
   ```

   Stevens goes on to mention that not all systems respond correctly to gratuitous ARPs, but googling for “arping -U” seems to support the idea that it works most of the time.

2. You can call your ISP and ask them to purge the stale ARP cache entry but many either can't or won't purge individual entries.

You can determine if your ISP's gateway ARP cache is stale using ping and tcpdump. Suppose that we suspect that the gateway router has a stale ARP cache entry for 192.0.2.177. On the firewall, run tcpdump as follows:

```
tcpdump -nei eth0 icmp
```

Now from 192.0.2.177, ping the ISP's gateway (which we will assume is 192.0.2.254):

```
ping 192.0.2.254
```

We can now observe the tcpdump output:

```
13:35:12.159321 0:4:e2:20:20:33 0:0:77:95:dd:19 ip 98:
                192.0.2.177 > 192.0.2.254: icmp: echo request (DF)
13:35:12.207615 0:0:77:95:dd:19 0:c0:a8:50:b2:57 ip 98:
                192.0.2.254 > 192.0.2.177 : icmp: echo reply
```

Notice that the source MAC address in the echo request is different from the destination MAC address in the echo reply!! In this case 0:4:e2:20:20:33 was the MAC of the firewall's eth0 NIC while 0:c0:a8:50:b2:57 was the MAC address of DMZ 1. In other words, the gateway's ARP cache still associates 192.0.2.177 with the NIC in DMZ 1 rather than with the firewall's eth0.

Rules

Note

Shorewall has a macro facility that includes macros for many standard applications. This section does not use those macros but rather defines the rules directly.

With the default policies described earlier in this document, your local systems (Local 1-3) can access any server on the internet and the DMZ can't access any other host (including the firewall). With the exception of DNAT rules which cause address translation and allow the translated connection request to pass through the firewall, the way to allow connection requests through your firewall is to use ACCEPT rules.

Note

Since the SOURCE PORT(S) and ORIG. DEST. Columns aren't used in this section, they won't be shown

You probably want to allow ping between your zones:

```
#ACTION   SOURCE   DEST                PROTO   DEST
#                                              PORT(S)
ACCEPT    net      dmz                 icmp    echo-request
ACCEPT    net      loc                 icmp    echo-request
ACCEPT    dmz      loc                 icmp    echo-request
ACCEPT    loc      dmz                 icmp    echo-request
```

Let's suppose that you run mail and pop3 servers on DMZ 2 and a Web Server on DMZ 1. The rules that you would need are:

```
#ACTION   SOURCE           DEST                PROTO   DEST     COMMENTS
#                                                      PORT(S)
ACCEPT    net              dmz:192.0.2.178     tcp     smtp     #Mail from
                                                                  #Internet
ACCEPT    net              dmz:192.0.2.178     tcp     pop3     #Pop3 from
                                                                  #Internet
ACCEPT    loc              dmz:192.0.2.178     tcp     smtp     #Mail from local
                                                                  #Network
ACCEPT    loc              dmz:192.0.2.178     tcp     pop3     #Pop3 from local
                                                                  #Network
ACCEPT    $FW              dmz:192.0.2.178     tcp     smtp     #Mail from the
                                                                  #Firewall
ACCEPT    dmz:192.0.2.178  net                 tcp     smtp     #Mail to the
                                                                  #Internet
ACCEPT    net              dmz:192.0.2.177     tcp     http     #WWW from
                                                                  #Internet
ACCEPT    net              dmz:192.0.2.177     tcp     https    #Secure WWW
                                                                  #from Internet
ACCEPT    loc              dmz:192.0.2.177     tcp     https    #Secure WWW
                                                                  #from local
                                                                  #Network
```

If you run a public DNS server on 192.0.2.177, you would need to add the following rules:

```
#ACTION   SOURCE           DEST                PROTO   DEST     COMMENTS
#                                                      PORT(S)
ACCEPT    net              dmz:192.0.2.177     udp     domain   #UDP DNS from
                                                                  #Internet
ACCEPT    net              dmz:192.0.2.177     tcp     domain   #TCP DNS from
                                                                  #Internet
ACCEPT    loc              dmz:192.0.2.177     udp     domain   #UDP DNS from
                                                                  #Local Network
```

```
ACCEPT   loc              dmz:192.0.2.177    tcp    domain  #TCP DNS from
                                                           #Local Network
ACCEPT   $FW              dmz:192.0.2.177    udp    domain  #UDP DNS from
                                                           #the Firewall
ACCEPT   $FW              dmz:192.0.2.177    tcp    domain  #TCP DNS from
                                                           #the Firewall
ACCEPT   dmz:192.0.2.177 net                 udp    domain  #UDP DNS to
                                                           #the Internet
ACCEPT   dmz:192.0.2.177 net                 tcp    domain  #TCPP DNS to
                                                           #the Internet
```

You probably want some way to communicate with your firewall and DMZ systems from the local network -- I recommend SSH which through its scp utility can also do publishing and software update distribution.

```
#ACTION  SOURCE           DEST                PROTO  DEST    COMMENTS
#                                             PORT(S)
ACCEPT   loc              dmz                 tcp    ssh     #SSH to the DMZ
ACCEPT   net              $FW                 tcp    ssh     #SSH to the
                                                             #Firewall
```

Odds and Ends

The above discussion reflects my personal preference for using Proxy ARP for my servers in my DMZ and SNAT/NAT for my local systems. I prefer to use NAT only in cases where a system that is part of an RFC 1918 subnet needs to have it's own public IP.

If you haven't already, it would be a good idea to browse through `/etc/shorewall/shorewall.conf` just to see if there is anything there that might be of interest. You might also want to look at the other configuration files that you haven't touched yet just to get a feel for the other things that Shorewall can do.

In case you haven't been keeping score, here's the final set of configuration files for our sample network. Only those that were modified from the original installation are shown.

`/etc/shorewall/interfaces` (The “options” will be very site-specific).

```
#ZONE    INTERFACE      BROADCAST     OPTIONS
net      eth0           detect        rfc1918,routefilter
loc      eth1           detect
dmz      eth2           detect
```

The setup described here requires that your network interfaces be brought up before Shorewall can start. This opens a short window during which you have no firewall protection. If you replace “detect” with the actual broadcast addresses in the entries above, you can bring up Shorewall before you bring up your network interfaces.

```
#ZONE    INTERFACE      BROADCAST     OPTIONS
net      eth0           192.0.2.255   rfc1918
loc      eth1           192.168.201.7
dmz      eth2           192.168.202.7
```

`/etc/shorewall/masq` - Local Subnet

```
#INTERFACE     SUBNET                ADDRESS
eth0           192.168.201.0/29      192.0.2.176
```

`/etc/shorewall/proxyarp` - DMZ

```
#ADDRESS     EXTERNAL     INTERFACE     HAVE ROUTE
192.0.2.177  eth2         eth0          No
192.0.2.178  eth2         eth0          No
```

/etc/shorewall/nat- Daughter's System

```
#EXTERNAL   INTERFACE   INTERNAL       ALL INTERFACES   LOCAL
192.0.2.179 eth0        192.168.201.4  No               No
```

/etc/shorewall/rules

```
#ACTION  SOURCE          DEST                PROTO  DEST    COMMENTS
#                                                   PORT(S)
ACCEPT   net             dmz                 icmp   echo-request
ACCEPT   net             loc                 icmp   echo-request
ACCEPT   dmz             loc                 icmp   echo-request
ACCEPT   loc             dmz                 icmp   echo-request
ACCEPT   net             loc:192.168.201.4   tcp    www     #Daughter's
                                                            #Server
ACCEPT   net             dmz:192.0.2.178     tcp    smtp    #Mail from
                                                            #Internet
ACCEPT   net             dmz:192.0.2.178     tcp    pop3    #Pop3 from
                                                            #Internet
ACCEPT   loc             dmz:192.0.2.178     tcp    smtp    #Mail from local
                                                            #Network
ACCEPT   loc             dmz:192.0.2.178     tcp    pop3    #Pop3 from local
                                                            #Network
ACCEPT   $FW             dmz:192.0.2.178     tcp    smtp    #Mail from the
                                                            #Firewall
ACCEPT   dmz:192.0.2.178 net                 tcp    smtp    #Mail to the
                                                            #Internet
ACCEPT   net             dmz:192.0.2.177     tcp    http    #WWW from
                                                            #Internet
ACCEPT   net             dmz:192.0.2.177     tcp    https   #Secure WWW
                                                            #from Internet
ACCEPT   loc             dmz:192.0.2.177     tcp    https   #Secure WWW
                                                            #from local
                                                            #Network
ACCEPT   net             dmz:192.0.2.177     udp    domain  #UDP DNS from
                                                            #Internet
ACCEPT   net             dmz:192.0.2.177     tcp    domain  #TCP DNS from
                                                            #Internet
ACCEPT   loc             dmz:192.0.2.177     udp    domain  #UDP DNS from
                                                            #Local Network
ACCEPT   loc             dmz:192.0.2.177     tcp    domain  #TCP DNS from
                                                            #Local Network
ACCEPT   $FW             dmz:192.0.2.177     udp    domain  #UDP DNS from
                                                            #the Firewall
ACCEPT   $FW             dmz:192.0.2.177     tcp    domain  #TCP DNS from
                                                            #the Firewall
ACCEPT   dmz:192.0.2.177 net                 udp    domain  #UDP DNS to
                                                            #the Internet
ACCEPT   dmz:192.0.2.177 net                 tcp    domain  #TCPP DNS to
                                                            #the Internet
ACCEPT   loc             dmz                 tcp    ssh     #SSH to the DMZ
ACCEPT   net             $FW                 tcp    ssh     #SSH to the
                                                            #Firewall
```

DNS

Given the collection of RFC 1918 and public addresses in this setup, it only makes sense to have separate internal and external DNS servers. You can combine the two into a single BIND 9 server using Views. If you are not interested in Bind 9 views, you can go to the next section.

Suppose that your domain is foobar.net and you want the two DMZ systems named www.foobar.net and mail.foobar.net and you want the three local systems named "winken.foobar.net, blinken.foobar.net and nod.foobar.net. You want your firewall to be known as firewall.foobar.net externally and it's interface to the local network to be know as gateway.foobar.net and its interface to the dmz as dmz.foobar.net. Let's have the DNS server on 192.0.2.177 which will also be known by the name ns1.foobar.net.

The `/etc/named.conf` file would look like this:

```
options {
        directory "/var/named";
        listen-on { 127.0.0.1 ; 192.0.2.177; };
        transfer-format many-answers;
        max-transfer-time-in 60;

allow-transfer {
        // Servers allowed to request zone tranfers
       <secondary NS IP>; };
};

logging {
        channel xfer-log {
                file "/var/log/named/bind-xfer.log";
                print-category yes;
                print-severity yes;
                print-time yes;
                severity info;
};

        category xfer-in { xfer-log; };
        category xfer-out { xfer-log; };
        category notify { xfer-log; };
};

#
# This is the view presented to our internal systems
#

view "internal" {
        #
        # These are the clients that see this view
        #
        match-clients { 192.168.201.0/29;
                        192.168.202.0/29;
                        127.0.0.0/8;
                        192.0.2.176/32;
                        192.0.2.178/32;
                        192.0.2.179/32;
                        192.0.2.180/32; };
        #
        # If this server can't complete the request, it should use
        # outside servers to do so
        #
        recursion yes;
```

```
        zone "." in {
                type hint;
                file "int/root.cache";
        };

        zone "foobar.net" in {
                type master;
                notify no;
                allow-update { none; };
                file "int/db.foobar";
        };

        zone "0.0.127.in-addr.arpa" in {
                type master;
                notify no;
                allow-update { none; };
                file "int/db.127.0.0";
        };

        zone "201.168.192.in-addr.arpa" in {
                type master;
                notify no;
                allow-update { none; };
                file "int/db.192.168.201";
        };

        zone "202.168.192.in-addr.arpa" in {
                type master;
                notify no;
                allow-update { none; };
                file "int/db.192.168.202";
        };

        zone "176.2.0.192.in-addr.arpa" in {
                type master;
                notify no;
                allow-update { none; };
                file "db.192.0.2.176";
        };

        zone "177.2.0.192.in-addr.arpa" in {
                type master;
                notify no;
                allow-update { none; };
                file "db.192.0.2.177";
        };

        zone "178.2.0.192.in-addr.arpa" in {
                type master;
                notify no;
                allow-update { none; };
                file "db.192.0.2.178";
        };

        zone "179.2.0.192.in-addr.arpa" in {
                type master;
                notify no;
                allow-update { none; };
                file "db.206.124.146.179";
        };

};
```

```
#
# This is the view that we present to the outside world
#
view "external" {
        match-clients { any; };
        #
        # If we can't answer the query, we tell the client so
        #
        recursion no;

        zone "foobar.net" in {
                type master;
                notify yes;
                allow-update {none; };
                file "ext/db.foobar";
        };

        zone "176.2.0.192.in-addr.arpa" in {
                type master;
                notify yes;
                allow-update { none; };
                file "db.192.0.2.176";
        };

        zone "177.2.0.192.in-addr.arpa" in {
                type master;
                notify yes;
                allow-update { none; };
                file "db.192.0.2.177";
        };

        zone "178.2.0.192.in-addr.arpa" in {
                type master;
                notify yes;
                allow-update { none; };
                file "db.192.0.2.178";
        };

        zone "179.2.0.192.in-addr.arpa" in {
                type master;
                notify yes;
                allow-update { none; };
                file "db.192.0.2.179";
        };
};
```

Here are the files in `/var/named` (those not shown are usually included in your bind disbribution).

`db.192.0.2.176` - This is the reverse zone for the firewall's external interface

```
; ##############################################################
; Start of Authority (Inverse Address Arpa) for 192.0.2.176/32
; Filename: db.192.0.2.176
; ##############################################################
@ 604800 IN SOA ns1.foobar.net. netadmin.foobar.net. (
                        2001102303 ; serial
                        10800 ; refresh (3 hour)
                        3600 ; retry (1 hour)
                        604800 ; expire (7 days)
                        86400 ) ; minimum (1 day)
```

```
;
; ##############################################################
; Specify Name Servers for all Reverse Lookups (IN-ADDR.ARPA)
; ##############################################################
@        604800  IN NS   ns1.foobar.net.
@        604800  IN NS   <name of secondary ns>.
;
; ##############################################################
; Iverse Address Arpa Records (PTR's)
; ##############################################################
176.2.0.192.in-addr.arpa. 86400 IN PTR firewall.foobar.net.
```

`db.192.0.2.177` - Reverse zone www server

```
; ##############################################################
; Start of Authority (Inverse Address Arpa) for 192.0.2.177/32
; Filename: db.192.0.2.177
; ##############################################################
@ 604800 IN SOA ns1.foobar.net. netadmin.foobar.net. (
                       2001102303 ; serial
                       10800 ; refresh (3 hour)
                       3600 ; retry (1 hour)
                       604800 ; expire (7 days)
                       86400 ) ; minimum (1 day)
;
; ##############################################################
; Specify Name Servers for all Reverse Lookups (IN-ADDR.ARPA)
; ##############################################################
@        604800  IN NS   ns1.foobar.net.
@        604800  IN NS   <name of secondary ns>.
;
; ##############################################################
; Iverse Address Arpa Records (PTR's)
; ##############################################################
177.2.0.192.in-addr.arpa. 86400 IN PTR www.foobar.net.
```

`db.192.0.2.178` - Reverse zone for the mail server

```
; ##############################################################
; Start of Authority (Inverse Address Arpa) for 192.0.2.178/32
; Filename: db.192.0.2.178
; ##############################################################
@ 604800 IN SOA ns1.foobar.net. netadmin.foobar.net. (
                      2001102303 ; serial
                      10800 ; refresh (3 hour)
                      3600 ; retry (1 hour)
                      604800 ; expire (7 days)
                      86400 ) ; minimum (1 day)
;
; ##############################################################
; Specify Name Servers for all Reverse Lookups (IN-ADDR.ARPA)
; ##############################################################
@        604800  IN NS   ns1.foobar.net.
@        604800  IN NS   <name of secondary ns>.
;
; ##############################################################
; Iverse Address Arpa Records (PTR's)
; ##############################################################
178.2.0.192.in-addr.arpa. 86400 IN PTR mail.foobar.net.
```

`db.192.0.2.179` - Reverse zone for Daughter's public web server

```
; ##############################################################
; Start of Authority (Inverse Address Arpa) for 192.0.2.179/32
; Filename: db.192.0.2.179
; ##############################################################
@ 604800 IN SOA ns1.foobar.net. netadmin.foobar.net. (
                     2001102303 ; serial
                     10800 ; refresh (3 hour)
                     3600 ; retry (1 hour)
                     604800 ; expire (7 days)
                     86400 ) ; minimum (1 day)
;
; ##############################################################
; Specify Name Servers for all Reverse Lookups (IN-ADDR.ARPA)
; ##############################################################
@        604800  IN NS   ns1.foobar.net.
@        604800  IN NS   <name of secondary ns>.
;
; ##############################################################
; Iverse Address Arpa Records (PTR's)
; ##############################################################
179.2.0.192.in-addr.arpa. 86400 IN PTR nod.foobar.net.
```

`int/db.127.0.0` - Reverse zone for localhost

```
; ##############################################################
; Start of Authority (Inverse Address Arpa) for 127.0.0.0/8
; Filename: db.127.0.0
; ##############################################################
@ 604800 IN SOA ns1.foobar.net. netadmin.foobar.net. (
                            2001092901 ; serial
                            10800 ; refresh (3 hour)
                            3600 ; retry (1 hour)
                            604800 ; expire (7 days)
                            86400 ) ; minimum (1 day)
; ##############################################################
; Specify Name Servers for all Reverse Lookups (IN-ADDR.ARPA)
; ##############################################################
@     604800          IN NS   ns1.foobar.net.

; ##############################################################
; Iverse Address Arpa Records (PTR's)
; ##############################################################
1       86400           IN PTR  localhost.foobar.net.
```

`int/db.192.168.201` - Reverse zone for the local network. This is only shown to internal clients.

```
; ##############################################################
; Start of Authority (Inverse Address Arpa) for 192.168.201.0/29
; Filename: db.192.168.201
; ##############################################################
@ 604800 IN SOA ns1.foobar.net netadmin.foobar.net. (
                               2002032501 ; serial
                               10800 ; refresh (3 hour)
                               3600 ; retry (1 hour)
                               604800 ; expire (7 days)
                               86400 ) ; minimum (1 day)
```

```
; ##############################################################
; Specify Name Servers for all Reverse Lookups (IN-ADDR.ARPA)
; ##############################################################
@       604800          IN NS   ns1.foobar.net.
; ##############################################################
; Iverse Address Arpa Records (PTR's)
; ##############################################################
1       86400           IN PTR  gateway.foobar.net.
2       86400           IN PTR  winken.foobar.net.
3       86400           IN PTR  blinken.foobar.net.
4       86400           IN PTR  nod.foobar.net.
```

`int/db.192.168.202` - Reverse zone for the firewall's DMZ Interface

```
; ##############################################################
; Start of Authority (Inverse Address Arpa) for 192.168.202.0/29
; Filename: db.192.168.202
; ##############################################################
@ 604800 IN SOA ns1.foobar.net netadmin.foobar.net. (
                                2002032501 ; serial
                                10800 ; refresh (3 hour)
                                3600 ; retry (1 hour)
                                604800 ; expire (7 days)
                                86400 ) ; minimum (1 day)

; ##############################################################
; Specify Name Servers for all Reverse Lookups (IN-ADDR.ARPA)
; ##############################################################
@             604800  IN NS   ns1.foobar.net.

; ##############################################################
; Iverse Address Arpa Records (PTR's)
; ##############################################################
1               86400 IN PTR  dmz.foobar.net.
```

`int/db.foobar` - Forward zone for internal clients.

```
;##############################################################
; Start of Authority for foobar.net.
; Filename: db.foobar
;##############################################################
@ 604800 IN SOA ns1.foobar.net. netadmin.foobar.net. (
                        2002071501 ; serial
                        10800 ; refresh (3 hour)
                        3600 ; retry (1 hour)
                        604800 ; expire (7 days)
                        86400 ); minimum (1 day)
;##############################################################
; foobar.net Nameserver Records (NS)
;##############################################################
@                 604800  IN NS   ns1.foobar.net.

;##############################################################
; Foobar.net Office Records (ADDRESS)
;##############################################################
localhost 86400   IN A    127.0.0.1

firewall       86400   IN A    192.0.2.176
www            86400   IN A    192.0.2.177
```

```
ns1             86400   IN A    192.0.2.177
mail            86400   IN A    192.0.2.178

gateway         86400   IN A    192.168.201.1
winken          86400   IN A    192.168.201.2
blinken         86400   IN A    192.168.201.3
nod             86400   IN A    192.168.201.4

dmz             86400   IN A    192.168.202.1
```

`ext/db.foobar` - Forward zone for external clients.

```
;##############################################################
; Start of Authority for foobar.net.
; Filename: db.foobar
;##############################################################
@ 86400 IN SOA ns1.foobar.net. netadmin.foobar.net. (
                        2002052901 ; serial
                        10800 ; refresh (3 hour)
                        3600 ; retry (1 hour)
                        604800 ; expire (7 days)
                        86400 ); minimum (1 day)
;#############################################################
; Foobar.net Nameserver Records (NS)
;#############################################################
@                86400   IN NS   ns1.foobar.net.
@                86400   IN NS   <secondary NS>.
;#############################################################
; Foobar.net        Foobar Wa Office Records (ADDRESS)
;#############################################################
localhost         86400   IN A    127.0.0.1
;
; The firewall itself
;
firewall          86400   IN A    192.0.2.176
;
; The DMZ
;
ns1               86400   IN A    192.0.2.177
www               86400   IN A    192.0.2.177
mail              86400   IN A    192.0.2.178
;
; The Local Network
;
nod             86400   IN A    192.0.2.179

;#############################################################
; Current Aliases for foobar.net (CNAME)
;#############################################################

;#############################################################
; foobar.net MX Records (MAIL EXCHANGER)
;#############################################################
foobar.net.      86400   IN A    192.0.2.177
                 86400   IN MX 0 mail.foobar.net.
                 86400   IN MX 1 <backup MX>.
```

Some Things to Keep in Mind

- **You cannot test your firewall from the inside**. Just because you send requests to your firewall external IP address does not mean that the request will be associated with the external interface or the “net” zone. Any traffic that you generate from the local network will be associated with your local interface and will be treated as loc->$FW traffic.
- **IP addresses are properties of systems, not of interfaces**. It is a mistake to believe that your firewall is able to forward packets just because you can ping the IP address of all of the firewall's interfaces from the local network. The only conclusion you can draw from such pinging success is that the link between the local system and the firewall works and that you probably have the local system's default gateway set correctly.
- **All IP addresses configured on firewall interfaces are in the $FW (fw) zone**. If 192.168.1.254 is the IP address of your internal interface then you can write “**$FW:192.168.1.254**” in a rule but you may not write “**loc:192.168.1.254**”. Similarly, it is nonsensical to add 192.168.1.254 to the **loc** zone using an entry in `/etc/shorewall/hosts`.
- **Reply packets do NOT automatically follow the reverse path of the one taken by the original request**. All packets are routed according to the routing table of the host at each step of the way. This issue commonly comes up when people install a Shorewall firewall parallel to an existing gateway and try to use DNAT through Shorewall without changing the default gateway of the system receiving the forwarded requests. Requests come in through the Shorewall firewall where the destination IP address gets rewritten but replies go out unmodified through the old gateway.
- **Shorewall itself has no notion of inside or outside**. These concepts are embodied in how Shorewall is configured.

Starting and Stopping the Firewall

The Installation procedure configures your system to start Shorewall at system boot.

The firewall is started using the “shorewall start” command and stopped using “shorewall stop”. When the firewall is stopped, routing is enabled on those hosts that have an entry in `/etc/shorewall/routestopped`. A running firewall may be restarted using the “shorewall restart” command. If you want to totally remove any trace of Shorewall from your Netfilter configuration, use “shorewall clear”.

Edit the `/etc/shorewall/routestopped` file and configure those systems that you want to be able to access the firewall when it is stopped.

Caution

If you are connected to your firewall from the internet, do not issue a “shorewall stop” command unless you have added an entry for the IP address that you are connected from to `/etc/shorewall/routestopped`. Also, I don't recommend using “shorewall restart”; it is better to create an *an alternate configuration* and test it using the “**shorewall try**” command.

Shorewall Installation and Upgrade

Tom Eastep

Copyright © 2001-, 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-08-23

Table of Contents

Install using RPM
Install using tarball
Install the .lrp
Install the .deb
General Notes about Upgrading Shorewall
Upgrade using RPM
Upgrade using tarball
Upgrade the .lrp
Configuring Shorewall
Uninstall/Fallback

Caution

This article applies to Shorewall 3.0 and later. If you are installing or upgradeing to a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Important

Before attempting installation, I strongly urge you to read and print a copy of the Shorewall QuickStart Guide for the configuration that most closely matches your own.

Important

Before upgrading, be sure to review the Upgrade Issues.

Install using RPM

To install Shorewall using the RPM:

1. **Be sure that you have the correct RPM package!**

   The standard RPM package from shorewall.net and the mirrors is known to work with **Suse™**, **Power PPC™**, **Trustix™** and **TurboLinux™**. There is also an RPM package provided by Simon Matter that is taylored for **RedHat/Fedora™** and another package from Jack Coates that is customized for **Mandrake™**. All of these are available from the download page.

   If you try to install the wrong package, it probably won't work.
2. Install the RPM

```
rpm -ivh <shorewall rpm>
```

Caution

Some users are in the habit of using the **rpm -U** command for installing packages as well as for updating them. If you use that command when installing the Shorewall RPM then you will have to manually enable Shorewall startup at boot time by running **chkconfig**, **insserv** or whatever utility you use to manipulate you init symbolic links.

Note

Some SuSE users have encountered a problem whereby rpm reports a conflict with kernel <= 2.2 even though a 2.4 kernel is installed. If this happens, simply use the --nodeps option to rpm.

```
rpm -ivh --nodeps <shorewall rpm>
```

Note

Shorewall is dependent on the iproute package. Unfortunately, some distributions call this package iproute2 which will cause the installation of Shorewall to fail with the diagnostic:

```
error: failed dependencies:iproute is needed by shorewall-2.2.x-1
```

This problem should not occur if you are using the correct RPM package (see 1., above) but may be worked around by using the --nodeps option of rpm.

```
rpm -ivh --nodeps <shorewall rpm>
```

3. Edit the configuration files to match your configuration.

Warning

YOU CAN **NOT** SIMPLY INSTALL THE RPM AND ISSUE A “shorewall start” COMMAND. SOME CONFIGURATION IS REQUIRED BEFORE THE FIREWALL WILL START. IF YOU ISSUE A “start” COMMAND AND THE FIREWALL FAILS TO START, YOUR SYSTEM WILL NO LONGER ACCEPT ANY NETWORK TRAFFIC. IF THIS HAPPENS, ISSUE A “shorewall clear” COMMAND TO RESTORE NETWORK CONNECTIVITY.

4. Enable startup by removing `/etc/shorewall/startup_disabled` (If you are running Shorewall 2.1.3 or later, edit /`etc/shorewall/shorewall.conf` and set STARTUP_ENABLED to Yes).
5. Start the firewall by typing

```
shorewall start
```

Install using tarball

To install Shorewall using the tarball and install script:

1. unpack the tarball (tar -zxf shorewall-x.y.z.tgz).
2. cd to the shorewall directory (the version is encoded in the directory name as in “shorewall-3.0.0”).
3. Type:

```
./install.sh
```

4. Edit the configuration files to match your configuration.
5. Enable Startup by editing `/etc/shorewall/shorewall.conf` and set STARTUP_ENABLED=Yes.
6. Start the firewall by typing

```
shorewall start
```

7. If the install script was unable to configure Shorewall to be started automatically at boot, see these instructions.

Install the .lrp

To install my version of Shorewall on a fresh Bering disk, simply replace the “shorwall.lrp” file on the image with the file that you downloaded. For example, if you download `shorewall-lrp-2.2.0.tgz` then you will rename the file to `shorwall.lrp` and replace the file by that name on the Bering disk with the new file. Then proceed to configure Shorewall as described in the Bering (or Bering uClibc) documentation.

Install the .deb

Important

Once you have installed the .deb package and before you attempt to configure Shorewall, please heed the advice of Lorenzo Martignoni, the Shorewall Debian Maintainer:

> “For more information about Shorewall usage on Debian system please look at /usr/share/doc/shorewall/README.Debian provided by [the] shorewall Debian package.”

The easiest way to install Shorewall on Debian, is to use apt-get:

apt-get install shorewall

To ensure that you are installing the latest version of Shorewall, please modify your `/etc/apt/sources.list` file as described here.

Once you have completed configuring Shorewall, you can enable startup at boot time by setting startup=1 in `/etc/default/shorewall`.

General Notes about Upgrading Shorewall

Most problems associated with upgrades come from two causes:

- The user didn't read and follow the migration considerations in the release notes (these are also reproduced in the Shorewall Upgrade Issues).
- The user mis-handled the `/etc/shorewall/shorewall.conf` file during upgrade. Shorewall is designed to allow the default behavior of the product to evolve over time. To make this possible, the design assumes that **you will not replace your current shorewall.conf file during upgrades**. It is recommended that after you first install Shorewall that you modify `/etc/shorewall/shorewall.conf` so as to prevent your package manager from overwriting it during subsequent upgrades (since the addition of STARTUP_ENABLED, such modification is assured since you must manually change the setting of that option). If you feel absolutely compelled to have the latest comments and options in your shorewall.conf then you must proceed carefully. You should determine which new options have been added and you must reset their value (e.g. OPTION=""); otherwise, you will get different behavior from what you expect.

Upgrade using RPM

If you already have the Shorewall RPM installed and are upgrading to a new version:

1. **Be sure that you have the correct RPM package!**

   The standard RPM package from shorewall.net and the mirrors is known to work with Suse, Power PPC, Trustix and TurboLinux. There is also an RPM package provided by Simon Matter that is taylored for RedHat/Fedora and another package from Jack Coates that is customized for Mandrake. If you try to upgrade using the wrong package, it probably won't work.
2. Upgrade the RPM

   ```
   rpm -Uvh <shorewall rpm file>
   ```

   ### Note

Some SuSE users have encountered a problem whereby rpm reports a conflict with kernel <= 2.2 even though a 2.4 kernel is installed. If this happens, simply use the --nodeps option to rpm.

```
rpm -Uvh --nodeps <shorewall rpm>
```

Note

Beginning with Shorewall 1.4.0, Shorewall is dependent on the iproute package. Unfortunately, some distributions call this package iproute2 which will cause the upgrade of Shorewall to fail with the diagnostic:

```
error: failed dependencies:iproute is needed by shorewall-1.4.0-1
```

This may be worked around by using the --nodeps option of rpm.

```
rpm -Uvh --nodeps <shorewall rpm>
```

3. See if there are any incompatibilities between your configuration and the new Shorewall version and correct as necessary.

```
shorewall check
```

4. Restart the firewall.

```
shorewall restart
```

Upgrade using tarball

If you already have Shorewall installed and are upgrading to a new version using the tarball:

1. unpack the tarball.

```
tar -zxf shorewall-x.y.z.tgz
```

2. cd to the shorewall directory (the version is encoded in the directory name as in “shorewall-3.0.1”).
3. Type:

```
./install.sh
```

4. See if there are any incompatibilities between your configuration and the new Shorewall version and correct as necessary.

```
shorewall check
```

5. Start the firewall by typing

```
shorewall start
```

6. If the install script was unable to configure Shorewall to be started automatically at boot, see these instructions.

Upgrade the .lrp

The following was contributed by Charles Steinkuehler on the Leaf mailing list:

> It's *VERY* simple...just put in a new CD and reboot! :-) Actually, I'm only slightly kidding...that's exactly how I upgrade my prodution firewalls. The partial backup feature I added to Dachstein allows configuration data to be stored seperately from the rest of the package.
>
> Once the config data is seperated from the rest of the package, it's an easy matter to upgrade the pacakge while keeping your current configuration (in my case, just inserting a new CD and re-booting).
>
> Users who aren't running with multiple package paths and using partial backups can still upgrade a package, it just takes a bit of extra work. The general idea is to use a partial backup to save your configuration, replace the package, and restore your old configuration files. Step-by-step instructions for one way to do this (assuming a conventional single-floppy LEAF system) would be:
>
> - Make a backup copy of your firewall disk ('NEW'). This is the disk you will add the upgraded package(s) to.
> - Format a floppy to use as a temporary location for your configuration file(s) ('XFER'). This disk should have the same format as your firewall disk (and could simply be another backup copy of your current firewall).
> - Make sure you have a working copy of your existing firewall ('OLD') in a safe place, that you *DO NOT* use durring this process. That way, if anything goes wrong you can simply reboot off the OLD disk to get back to a working configuration.
> - Remove your current firewall configuration disk and replace it with the XFER disk.
> - Use the lrcfg backup menu to make a partial backup of the package(s) you want to upgrade, being sure to backup the files to the XFER disk. From the backup menu:
>
> ```
> t e <enter> p <enter>
> b <package1> <enter>
> b <package2> <enter>
> ...
> ```
>
> - Download and copy the package(s) you want to upgrade onto the NEW disk.
> - Reboot your firewall using the NEW disk...at this point your upgraded packages will have their default configuration.
> - Mount the XFER disk (mount -t msdos /dev/fd0u1680 /mnt)
> - CD to the root directory (cd /)
> - Manually extract configuration data for each package you upgraded:
>
> ```
> tar -xzvf /mnt/package1.lrp
> tar -xzvf /mnt/package2.lrp
> ...
> ```

- Unmount (umount /mnt) and remove the XFER disk
- Using lrcfg, do *FULL* backups of your upgraded packages.
- Reboot, verifying the firewall works as expected. Some configuration files may need to be 'tweaked' to work properly with the upgraded package binaries.

Important

The new package file <package>.local can be used to fine-tune which files are included (and excluded) from the partial backup (see the Dachstein-CD README for details). If this file doesn't exist, the backup scripts assume anything from the <package>.list file that resides in /etc or /var/lib/lrpkg is part of the configuration data and is used to create the partial backup. If shorewall puts anything in /etc that isn't a user modified configuration file, a proper shorwall.local file should be created prior to making the partial backup [**Editor's note**: Shorewall places only user-modifiable files in /etc].

Note

It's obviously possible to do the above 'in-place', without using multiple disks, and even without making a partial backup (ie: copy current config files to /tmp, manually extract new package on top of current running firewall, then copy or merge config data from /tmp and backup...or similar), but anyone capable of that level of command line gymnastics is probably doing it already, without needing detailed instructions! :-)

For information on other LEAF/Bering upgrade tools, check out this article by Alex Rhomberg.

Configuring Shorewall

You will need to edit some or all of the configuration files to match your setup. In most cases, the Shorewall QuickStart Guides contain all of the information you need.

Uninstall/Fallback

See “Fallback and Uninstall”.

Upgrade Issues

Tom Eastep

Copyright © 2002, 2003, 2004 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005/11/18

Table of Contents

Important
Version >= 3.0.0
Version >= 2.4.0
Version >= 2.2.0
Version >= 2.0.2 RC1
Version >= 2.0.2 Beta 1
Version >= 2.0.1
VERSION >= 2.0.0-Beta1
Version >= 1.4.9
Version >= 1.4.8
Version >= 1.4.6
Version >= 1.4.4
Version 1.4.4
Version >= 1.4.2
Version >= 1.4.1
Version 1.4.1
Version >= 1.4.0
Version 1.4.0

Important

It is important that you read all of the sections on this page where the version number mentioned in the section title is later than what you are currently running.

In the descriptions that follows, the term *group* refers to a particular network or subnetwork (which may be `0.0.0.0/0` or it may be a host address) accessed through a particular interface.

Examples:

```
eth0:0.0.0.0/0
eth2:192.168.1.0/24
eth3:192.0.2.123
```

You can use the **shorewall check** command to see the groups associated with each of your zones.

Version >= 3.0.0

1. The "monitor" command has been eliminated.
2. The "DISPLAY" and "COMMENTS" columns in the /etc/shorewall/zones file have been removed and have been replaced by the former columns of the /etc/shorewall/ipsec file. The latter file has been removed.

   Additionally the FW option in shorewall.conf has been deprecated and is no longer set to 'fw' by default. New users are expected to define the firewall zone in /etc/shorewall/zones.

   Adhering to the principle of least astonishment, the old `/etc/shorewall/ipsec` file will continue to be supported. A new IPSECFILE variable in /etc/shorewall/shorewall.conf determines the name of the file that Shorewall looks in for IPSEC information. If that variable is not set or is set to the empty value then IPSECFILE=ipsec is assumed. So if you simply upgrade and don't do something idiotic like replace your current shorewall.conf file with the new one, your old configuration will continue to work. A dummy 'ipsec' file is included in the release so that your package manager (e.g., rpm) won't remove your existing file.

   The shorewall.conf file included in this release sets IPSECFILE=zones so that new users are expected to use the new zone file format.
3. The DROPINVALID option has been removed from shorewall.conf. The behavior will be as if DROPINVALID=No had been specified. If you wish to drop invalid state packets, use the dropInvalid built-in action.
4. The 'nobogons' interface and hosts option as well as the BOGON_LOG_LEVEL option have been eliminated.
5. Most of the standard actions have been replaced by parameterized macros (see below). So for example, the action.AllowSMTP and action.DropSMTP have been removed an a parameterized macro macro.SMTP has been added to replace them.

   In order that current users don't have to immediately update their rules and user-defined actions, Shorewall can substitute an invocation of the a new macro for an existing invocation of one of the old actions. So if your rules file calls AllowSMTP, Shorewall will replace that call with SMTP/ACCEPT. Because this substitution is expensive, it is conditional based on the setting of MAPOLDACTIONS in shorewall.conf. If this option is set to YES or if it is not set (such as if you are using your old shorewall.conf file) then Shorewall will perform the substitution. Once you have converted to use the new macros, you can set MAPOLDACTIONS=No and invocations of those actions will go much quicker during 'shorewall [re]start'.
6. The STATEDIR variable in /etc/shorewall/shorewall.conf has been removed. STATEDIR is now fixed at /var/lib/shorewall. If you have previously set STATEDIR to another directory, please copy the files from that directory to /var/lib/shorewall/ before [re]starting Shorewall after the upgrade to this version.
7. The "shorewall status" command now just gives the status of Shorewall (started or not-started). The previous status command has been renamed "dump". The command also shows the state relative to the state diagram at http://shorewall.net/starting_and_stopping_shorewall.htm. In addition to the state, the time and date at which that state was entered is shown.

   Note that at least one "shorewall [re]start" must be issued after upgrading to this release before "shorewall status" will show anything but "Unknown" for the state.
8. The "shorewall forget" command now removes the dynamic blacklist save file (/var/lib/shorewall/save).
9. In previous versions of Shorewall, the rules generated by entries in `/etc/shorewall/tunnels` preceded those rules generated by entries in `/etc/shorewall/rules`. Beginning with this release, the rules generated by entries in the tunnels file will appear *AFTER* the rules generated by the rules file. This may cause you problems if you have REJECT, DENY or CONTINUE rules in your rules file that would cause the tunnel transport packets to not reach the rules that ACCEPT them. See http://www.shorewall.net/VPNBasics.html for information on the rules generated by entries in the tunnels file.
10. The NEWNOTSYN and LOGNEWNOTSYN options in shorewall.conf have been removed as have the 'newnotsyn' options in `/etc/shorewall/interfaces` and `/etc/shorewall/hosts`.

    TCP new-not-syn packets may be blocked using the 'dropNonSyn' or 'rejNonSyn' built-in actions.

    Example: Reject all new-not-syn packets from the net and log them at the 'info' level.

```
#ACTION          SOURCE           DEST           PROTO
SECTION NEW
```

```
rejNonSyn:info   net              all            tcp
```

Note that the rule is added at the front of the NEW section of the rules file.

11. A new TC_SCRIPT option replaces TC_ENABLED in shorewall.conf. If the option is not set then the internal shaper (tc4shorewall by Arne Bernin) is used. Otherwise, the script named in the variable is used.

    Users who currently use an `/etc/shorewall/tcstart` file and wish to continue to do so should set TC_SCRIPT=/etc/shorewall/tcstart in shorewall.conf.

Version >= 2.4.0

1. Shorewall now enforces the restriction that mark values used in `/etc/shorewall/tcrules` are less than 256. If you are using mark values >= 256, you must change your configuration before you upgrade.
2. The value "ipp2p" is no longer accepted in the PROTO column of the `/etc/shorewall/rules` file. This support has never worked as intended and cannot be made to work in a consistent way. A "Howto" article on filtering P2P with Shorewall and ipp2p will be forthcoming.
3. LEAF/Bering packages for 2.4.0 and later releases are not available from shorewall.net. See the LEAF site for those packages.

Version >= 2.2.0

1. Shorewall configuration files except shorewall.conf are now empty (they contain only comments). If you wish to retain the defaults in any of the following files, you should copy these files before upgrading them then restore them after the upgrade:

   /etc/shorewall/zones
   /etc/shorewall/policy
   /etc/shorewall/tos
2. The following builtin actions have been removed and have been replaced by the new action logging implementation described in the new features below.

   logNotSyn
   rLogNotSyn
   dLogNotSyn
3. If shorewall.conf is upgraded to the latest version, it needs to be modified to set STARTUP_ENABLED=Yes.
4. The Leaf/Bering version of Shorewall was previously named:

   shorwall-<version>.lrp

   Beginning with 2.1, that file will now be named:

   shorewall-lrp-<version>.tgz

   Simply rename that file to 'shorwall.lrp' when installing it on your LEAF/Bering system.
5. The ORIGINAL DEST column of the /etc/shorewall/rules file may no longer contain a second (SNAT) address. You must use an entry in /etc/shorewall/masq instead.

   Example from Shorewall FAQ #1:

   Prior to Shorewall 2.1:

   /etc/shorewall/interfaces

```
#ZONE      INTERFACE       BROADCAST      OPTIONS
loc        eth1            detect         routeback
```

/etc/shorewall/rules

```
#ACTION   SOURCE    DEST              PROTO    DEST PORT    SOURCE   ORIGINAL
#                                                            PORT     DEST
DNAT      loc       loc:192.168.1.12 tcp      80           -
130.252.100.69:192.168.1.254
```

Shorewall 2.1 and Later:

/etc/shorewall/interfaces

```
#ZONE      INTERFACE       BROADCAST      OPTIONS
loc        eth1            detect         routeback
```

/etc/shorewall/masq:

```
#INTERFACE     SUBNETS       ADDRESS         PROTO    PORT(S)
eth1           eth1          192.168.1.254   tcp      80
```

/etc/shorewall/rules:

```
#ACTION   SOURCE    DEST              PROTO    DEST PORT    SOURCE   ORIGINAL
#                                                            PORT     DEST
DNAT      loc       loc:192.168.1.12 tcp      80           -        130.252.100.69
```

6. The 'logunclean' and 'dropunclean' options that were deprecated in Shorewall 2.0 have now been removed completely.
7. The default port for 'openvpn' tunnels (/etc/shorewall/tunnels) has been changed to 1194 to match a similar change in the OpenVPN product. The IANA has registered port 1194 for use by OpenVPN.
8. A new IPTABLES variable has been added to shorewall.conf. This variable names the iptables executable that Shorewall will use. The variable is set to "/sbin/iptables". If you use the new shorewall.conf, you may need to change this setting to maintain compatibility with your current setup (if you use your existing shorewall.conf that does not set IPTABLES then you should experience no change in behavior).

Version >= 2.0.2 RC1

- If you are upgrading from Shorewall 1.4.x and you have commands in your `/etc/shorewall/common` file that are not directly related to the **common** chain then you will want to move those commands to `/etc/shorewall/initdone`.

Version >= 2.0.2 Beta 1

- Extension Scripts - In order for extension scripts to work properly with the new iptables-save/restore integration introduced in Shorewall 2.0.2 Beta 1, some change may be required to your extension scripts.

  If your extension scripts are executing commands other than **iptables** then those commands must also be written to the restore file (a temporary file in `/var/lib/shorewall` that is renamed `/var/lib/shorewall/restore-base` at the completion of the `/sbin/shorewall` command). The following functions should be of help:
  1. save_command() -- saves the passed command to the restore file.

     Example:

```
save_command echo Operation Complete
```

That command would simply write "echo Operation Complete" to the restore file.

2. run_and_save_command() -- saves the passed command to the restore file then executes it. The return value is the exit status of the command. Example:

```
run_and_save_command "echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all"
```

Note that as in this example, when the command involves file redirection then the entire command must be enclosed in quotes. This applies to all of the functions described here.

3. ensure_and_save_command() -- runs the passed command. If the command fails, the firewall is restored to it's prior saved state and the operation is terminated. If the command succeeds, the command is written to the restore file

- Dynamic Zone support. - If you don't need to use the **shorewall add** and **shorewall delete** commands, you should set DYNAMIC_ZONES=No in `/etc/shorewall/shorewall.conf`.

Version >= 2.0.1

- The function of 'norfc1918' is now split between that option and a new 'nobogons' option. The rfc1918 file released with Shorewall now contains entries for only those three address ranges reserved by RFC 1918. A 'nobogons' interface option has been added which handles bogon source addresses (those which are reserved by the IANA, those reserved for DHCP auto-configuration and the class C test-net reserved for testing and documentation examples). This will allow users to perform RFC 1918 filtering without having to deal with out of date data from IANA. Those who are willing to update their `/usr/share/shorewall/bogons` file regularly can specify the 'nobogons' option in addition to 'norfc1918'. The level at which bogon packets are logged is specified in the new BOGON_LOG_LEVEL variable in shorewall.conf. If that option is not specified or is specified as empty (e.g, BOGON_LOG_LEVEL="") then bogon packets whose TARGET is 'logdrop' in `/usr/share/shorewall/bogons` are logged at the 'info' level.

Warning

If you have a file named `rfc1918` in your `/etc/shorewall` directory, you must either remove that file or you must copy `/usr/share/shorewall/rfc1918` to `/etc/shorewall` and modify that file appropriately. Do not leave the old `rfc1918` file in `/etc/shorewall`.

VERSION >= 2.0.0-Beta1

- The 'dropunclean' and 'logunclean' interface options are no longer supported. If either option is specified in `/etc/shorewall/interfaces`, a threatening message will be generated.
- The NAT_BEFORE_RULES option has been removed from `shorewall.conf`. The behavior of Shorewall 2.0 is as if NAT_BEFORE_RULES=No had been specified. In other words, DNAT rules now always take precedence over one-to-one NAT specifications.
- The default value for the ALL INTERFACES column in `/etc/shorewall/nat` has changed. In Shorewall 1.*, if the column was left empty, a value of "Yes" was assumed. This has been changed so that a value of "No" is now assumed.
- The following files don't exist in Shorewall 2.0:

  `/etc/shorewall/common.def`
  `/etc/shorewall/common`
  `/etc/shorewall/icmpdef`
  `/etc/shorewall/action.template` (moved to `/usr/share/shorewall/action.template`)

  The `/etc/shorewall/action` file now allows an action to be designated as the "common" action for a particular policy type by following the action name with ":" and the policy (DROP, REJECT or ACCEPT).

  The file /usr/share/shorewall/actions.std has been added to define those actions that are released as part of Shorewall 2.0 In that file are two actions as follows:

Drop:DROP
Reject:REJECT

The “Drop” action is the common action for DROP policies while the “Reject” action is the default action for REJECT policies. These actions will be performed on packets prior to applying the DROP or REJECT policy respectively. In the first release, the difference between "Reject" and "Drop" is that "Reject" REJECTs SMB traffic while "Drop" silently drops such traffic.

As described above, Shorewall allows a common action for ACCEPT policies but does not specify such an action in the default configuration.

For more information see the User-defined Action Page.

- The `/etc/shorewall` directory no longer contains `users` file or a `usersets` file. Similar functionality is now available using user-defined actions.

  Now, action files created by copying `/usr/share/shorewall/action.template` may now specify a USER and or GROUP name/id in the final column just like in the rules file (see below). It is thus possible to create actions that control traffic from a list of users and/or groups.
- The last column in `/etc/shorewall/rules` is now labeled USER/GROUP and may contain:

  [!]<*user number*>[:]
  [!]<*user name*>[:]
  [!]:<*group number*>
  [!]:<*group name*>
  [!]<*user number*>:<*group number*>
  [!]<*user name*>:<*group number*>
  [!]<*user inumber*>:<*group name*>
  [!]<*user name*>:<*group name*>
- If your kernel has IPV6 support (recent SuSe™ for example), and you don't use IPV6 then you will probably want to set DISABLE_IPV6=Yes in /etc/shorewall/shorewall.conf. You must have ipv6tables installed.

Version >= 1.4.9

- The default value of NEWNOTSYN set in /etc/shorewall/shorewall.conf has been changed from 'No' to 'Yes'. I find that NEWNOTSYN=No tends to result in lots of "stuck" connections because any network timeout during TCP session tear down results in retries being dropped (Netfilter has removed the connection from the conntrack table but the end-points haven't completed shutting down the connection). I therefore have chosen NEWNOTSYN=Yes as the default value and I advise caution in using NEWNOTSYN=Yes.

Version >= 1.4.8

- The meaning of `ROUTE_FILTER=Yes` has changed. Previously this setting was documented as causing route filtering to occur on all network interfaces; this didn't work. Beginning with this release, `ROUTE_FILTER=Yes` causes route filtering to occur on all interfaces brought up while Shorewall is running. This means that it may be appropriate to set `ROUTE_FILTER=Yes` and use the routefilter option in `/etc/shorewall/interfaces` entries.

Version >= 1.4.6

- The `NAT_ENABLED`, `MANGLE_ENABLED` and `MULTIPORT` options have been removed from `shorewall.conf`. These capabilities are now automatically detected by Shorewall.
- An undocumented feature previously allowed entries in the host file as follows:

```
zone    eth1:192.168.1.0/24,eth2:192.168.2.0/24
```

This capability was never documented and has been removed in 1.4.6 to allow entries of the following format:

```
zone    eth1:192.168.1.0/24,192.168.2.0/24
```

Version >= 1.4.4

If you are upgrading from 1.4.3 and have set the `LOGMARKER` variable in `/etc/shorewall/shorewall.conf`, then you must set the new `LOGFORMAT` variable appropriately and remove your setting of `LOGMARKER`.

Version 1.4.4

If you have zone names that are 5 characters long, you may experience problems starting Shorewall because the `--log-prefix` in a logging rule is too long. Upgrade to Version 1.4.4a to fix this problem.

Version >= 1.4.2

There are some cases where you may want to handle traffic from a particular group to itself. While I personally think that such a setups are ridiculous, there are two cases covered in this documentation where it can occur:

- In FAQ #2
- When running Squid as a transparent proxy in your local zone.

If you have either of these cases, you will want to review the current documentation and change your configuration accordingly.

Version >= 1.4.1

- Beginning with Version 1.4.1, traffic between groups in the same zone is accepted by default. Previously, traffic from a zone to itself was treated just like any other traffic; any matching rules were applied followed by enforcement of the appropriate policy. With 1.4.1 and later versions, unless you have explicit rules for traffic from Z to Z or you have an explicit Z to Z policy (where "Z" is some zone) then traffic between the groups in zone Z will be accepted. If you do have one or more explicit rules for Z to Z or if you have an explicit Z to Z policy then the behavior is as it was in prior versions.
    1. If you have a Z Z ACCEPT policy for a zone to allow traffic between two interfaces to the same zone, that policy can be removed and traffic between the interfaces will traverse fewer rules than previously.
    2. If you have a Z Z DROP or Z Z REJECT policy or you have Z->Z rules then your configuration should not require any change.
    3. If you are currently relying on a implicit policy (one that has "all" in either the SOURCE or DESTINATION column) to prevent traffic between two interfaces to a zone Z and you have no rules for Z->Z then you should add an explicit DROP or REJECT policy for Z to Z.
- Sometimes, you want two separate zones on one interface but you don't want Shorewall to set up any infrastructure to handle traffic between them.

Example 1. The `zones`, `interfaces` and, `hosts` file contents

```
/etc/shorewall/zones
z1      Zone1   The first Zone
z2      Zone2   The second Zone

/etc/shorewall/interfaces
z2      eth1    192.168.1.255
```

```
/etc/shorewall/hosts
z1      eth1:192.168.1.3
```

Here, zone z1 is nested in zone z2 and the firewall is not going to be involved in any traffic between these two zones. Beginning with Shorewall 1.4.1, you can prevent Shorewall from setting up any infrastructure to handle traffic between z1 and z2 by using the new NONE policy:

Example 2. The contents of `policy`

```
/etc/shorewall/policy
z1      z2      NONE
z2      z1      NONE
```

Note that NONE policies are generally used in pairs unless there is asymmetric routing where only the traffic on one direction flows through the firewall and you are using a NONE policy in the other direction.

Version 1.4.1

- In Version 1.4.1, Shorewall will never create rules to deal with traffic from a given group back to itself. The `multi` interface option is no longer available so if you want to route traffic between two subnetworks on the same interface then I recommend that you upgrade to Version 1.4.2 and use the `routeback` interface or host option.

Version >= 1.4.0

Important

Shorewall >=1.4.0 requires the **iproute** package ('`ip`' utility).

Note

Unfortunately, some distributions call this package **iproute2** which will cause the upgrade of Shorewall to fail with the diagnostic:

```
error: failed dependencies:iproute is needed by shorewall-1.4.0-1
```

This may be worked around by using the `--nodeps` option of **rpm** (**rpm -Uvh --nodeps `your_shorewall_rpm.rpm`**).

If you are upgrading from a version < 1.4.0, then:

- The `noping` and `forwardping` interface options are no longer supported nor is the `FORWARDPING` option in `shorewall.conf`. ICMP echo-request (ping) packets are treated just like any other connection request and are subject to rules and policies.
- Interface names of the form `<device>:<integer>` in `/etc/shorewall/interfaces` now generate a Shorewall error at startup (they always have produced warnings in iptables).
- The `MERGE_HOSTS` variable has been removed from `shorewall.conf`. Shorewall 1.4 behaves like 1.3 did when `MERGE_HOSTS=Yes`; that is zone contents are determined by *BOTH* the interfaces and hosts files when there are entries for the zone in both files.
- The `routestopped` option in the interfaces and hosts file has been eliminated; use entries in the `routestopped` file instead.
- The Shorewall 1.2 syntax for `DNAT` and `REDIRECT` rules is no longer accepted; you must convert to using the new syntax.

- The `ALLOWRELATED` variable in `shorewall.conf` is no longer supported. Shorewall 1.4 behavior is the same as 1.3 with `ALLOWRELATED=Yes`.
- Late-arriving DNS replies are now dropped by default; there is no need for your own `/etc/shorewall/common` file simply to avoid logging these packets.
- The `firewall`, `functions` and `version` files have been moved to `/usr/share/shorewall`.
- The `icmp.def` file has been removed. If you include it from `/etc/shorewall/icmpdef`, you will need to modify that file.
- If you followed the advice in FAQ #2 and call `find_interface_address` in `/etc/shorewall/params`, that code should be moved to `/etc/shorewall/init`.

Version 1.4.0

- The `multi` interface option is no longer supported. Shorewall will generate rules for sending packets back out the same interface that they arrived on in two cases:
  - There is an *explicit* policy for the source zone to or from the destination zone. An explicit policy names both zones and does not use the `all` reserved word.
  - There are one or more rules for traffic for the source zone to or from the destination zone including rules that use the `all` reserved word. Exception: if the source zone and destination zone are the same then the rule must be explicit - it must name the zone in both the `SOURCE` and `DESTINATION` columns.

Traffic Shaping/Control

Tom Eastep

Arne Bernin

Copyright © 2001-2004 Thomas M. Eastep

Copyright © 2005 Arne Bernin & Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-10-15

Table of Contents

Introduction
Linux traffic shaping and control
Linux Kernel Configuration
Enable TC support in Shorewall
Using builtin traffic shaping/control
- /etc/shorewall/tcrules
- /etc/shorewall/tcdevices
- /etc/shorewall/tcclasses
- ppp devices
- Real life examples
  - Configuration to replace Wondershaper
  - A simple setup

Using your own tc script
- Replacing builtin tcstart file
- Traffic control outside Shorewall

Introduction

Starting with Version 2.5.5, Shorewall has builtin support for traffic shaping and control. Before this version, the support was quite limited. You were able to use your own tcstart script (and you still are), but besides the tcrules file it was not possible to define classes or queueing discplines inside the Shorewall config files.

The support for traffic shaping and control still does not cover all options available (and especially all algorithms that can be used to queue traffic) in the Linux kernel but it should fit most needs. If you are using your own script for traffic control and you still want to use it in the future, you will find information on how to do this, later in this document. But for this to work, you will also need to enable traffic shaping in the kernel and Shorewall as covered by the next sections.

Linux traffic shaping and control

This section gives a brief introduction of how controlling traffic with the linux kernel works. Although this might be enough for

configuring it in the Shorewall configuration files, it still might be a good idea to take a deeper look into the Linux Advanced Routing and Shaping HOWTO. At the time of writing this, the current version is 1.0.0.

Since kernel 2.2 Linux has extensive support for controlling traffic. You can define different algorithms that are used to queue the traffic before it leaves an interface. The standard one is called pfifo and is (as the name suggests) of the type First In First out. This means, that it does not shape anything, if you have a connection that eats up all your bandwidth, this qeueing algorithm will not stop it from doing so.

For Shorewall traffic shaping we use two algorithms, one is called HTB (Hierarchical Token Bucket) and SFQ (Stochastic Fairness Queuing). SFQ is easy to explain: it just tries to track your connections (tcp or udp streams) and balances the traffic between them. This normally works well. HTB allows you to define a set of classes, and you can put the traffic you want into these classes. You can define minimum and maximum bandwitdh settings for those classes and order them hierachically (the less priorized classes only get bandwitdth if the more important have what they need). Shorewall builtin traffic shaping allows you to define these classes (and their bandwidth limits), and it uses SFQ inside these classes to make sure, that different data streams are handled equally.

You can only shape outgoing traffic. The reason for this is simple, the packets were already received by your network card before you can decide what to do with them. So the only choice would be to drop them which normally makes no sense (since you received the packet already, it went through the possible bottleneck (the incoming connection). The next possible bottleneck might come if the packet leaves on another interface, so this will be the place where queuing might occur. So, defining queues for incoming packages is not very useful, you just want to have it forwarded to the outgoing interface as fast as possible.

There is one exception, though. Limiting incoming traffic to a value a bit slower than your actual line speed will avoid queueing on the other end of that connection. This is mostly useful if you don't have access to traffic control on the other side and if this other side has a faster network connection than you do (the line speed between the systems is the bottleneck, e.g. a DSL connection to you providers router, the router itself is normally connected to a much faster backbone). So, if you drop packages that are coming in too fast, the underlaying protocol might recognize this and slow down the connection. TCP has a builtin mechanism for this, UDP has not (but the protocol over UDP might recognize it , if there is any).

The reason why queing is bad in these cases is, that you might have packets which need to be priorized over others, e.g. VoIP or ssh. For this type of connections it is important that packets arrive in a certain amount of time. For others like http downloads, it does not really matter if it takes a few seconds more.

If you have a large queue on the other side and the router there does not care about QoS or the QoS bits are not set properly, your important packets will go into the same queue as your less timecritical download packets which will result in a large delay.

You shape and control outgoing traffic by assigning the traffic to *classes*. Each class is associated with exactly one network interface and has a number of attributes:

1. PRIORITY - Used to give preference to one class over another when selecting a packet to send. The priority is a numeric value with 1 being the highest priority, 2 being the next highest, and so on.
2. RATE - The minimum bandwidth this class should get, when the traffic load rises. Classes with a higher priority (lower PRIORITY value) are served even if there are others that have a guaranteed bandwith but have a lower priority (higher PRIORITY value).
3. CEIL - The maximum bandwidth the class is allowed to use when the link is idle.
4. MARK - Netfilter has a facility for *marking* packets. Packet marks have a numberic value which is limited in Shorewall to the values 1-255. You assign packet marks to different types of traffic using entries in the `/etc/shorewall/tcrules` file.

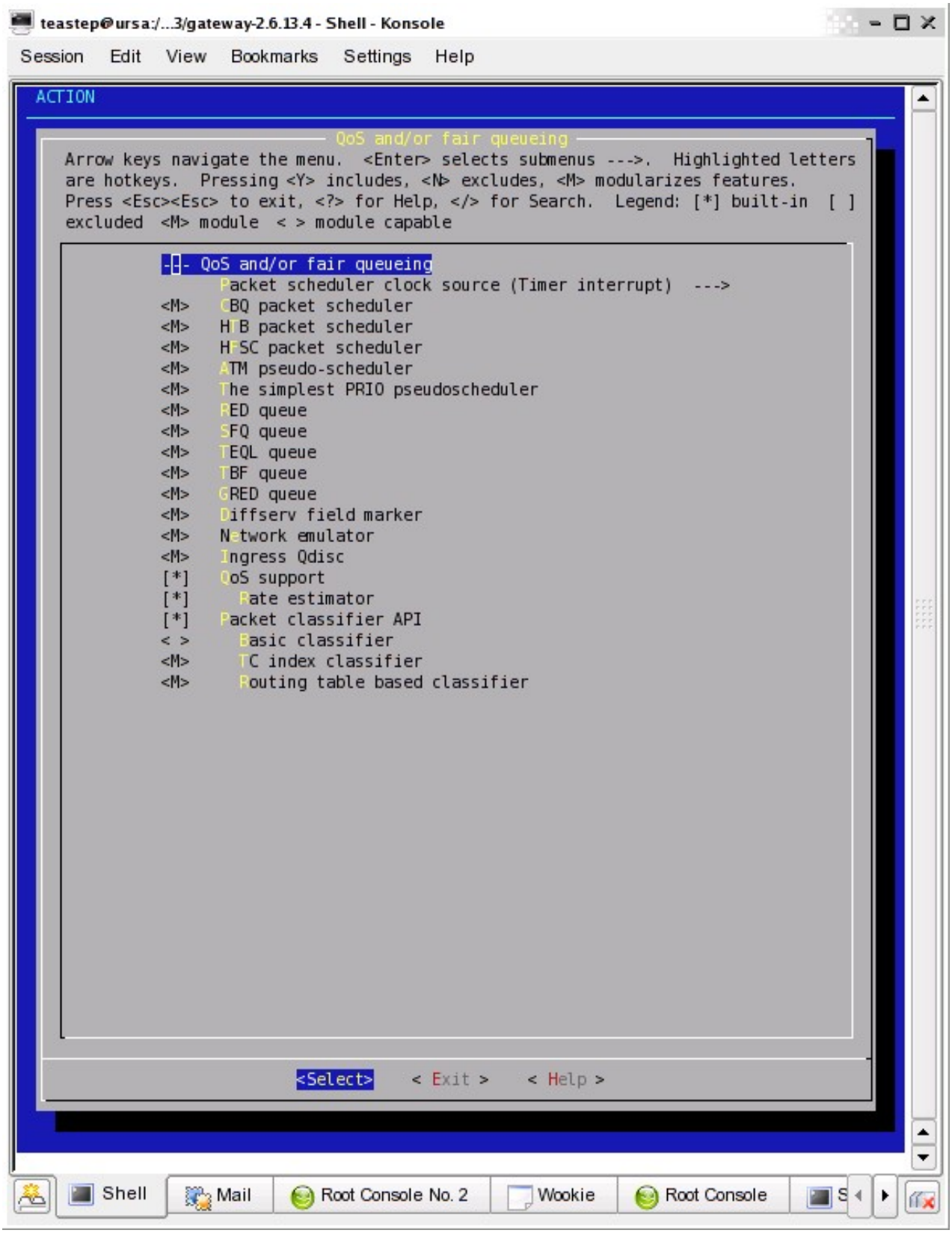
One class for each interface must be designated as the *default class*. This is the class to which unmarked traffic (packets to which you have not assigned a mark value in `/etc/shorewall/tcrules`) is assigned.

Linux Kernel Configuration

You will need at least kernel 2.4.18 for this to work, please take a look at the following screenshot for what settings you need to

enable. For builtin support, you need the HTB scheduler, the PRIO pseudoscheduler and SFQ queue. The other scheduler or queue algorithms are not needed.

This screen shot show how I've configured QoS in my Kernel:



Enable TC support in Shorewall

You need this support whether you use the builtin support or whether you provide your own tcstart script.

To enable the builtin traffic shaping and control in Shorewall, you have to do the following:

- Set **TC_ENABLED** to "Internal" in /etc/shorewall/shorewall.conf. Setting TC_ENABLED=Yes causes Shorewall to look for an external tcstart file (See a later section for details).
- Setting **CLEAR_TC** parameter in /etc/shorewall/shorewall.conf to Yes will clear the traffic shaping configuration during Shorewall [re]start and Shorewall stop. This is normally what you want when using the builtin support (and also if you use your own tcstart script)
- The other steps that follow depend on whether you use your own script or the builtin solution. They will be explained in the following sections.

Using builtin traffic shaping/control

For defining bandwidths (for either devices or classes) please use kbit or kbps(for Kilobytes per second) and make sure there is **NO** space between the number and the unit (it is 100kbit **not** 100 kbit). Using mbit, mbps or a raw number (which means bytes) could be used, but note that only integer numbers are supported (0.5 is **not valid**).

To properly configure the settings for your devices you might need to find out, the real up- and downstream rates you have. This is especially the case, if you are using a DSL connection or one of another type that do not have a guaranteed bandwidth.Don't trust the values your provider tells you for this, especially measuring the real download speed is important! There are several online tools that help you find out, search for "dsl speed test" on google (For Germany you can use arcor speed check). Be sure to choose a test located near you.

/etc/shorewall/tcrules

The fwmark classifier provides a convenient way to classify packets for traffic shaping. The “/etc/shorewall/tcrules” file is used for specifying these marks in a tabular fashion.

Normally, packet marking occurs in the PREROUTING chain before any address rewriting takes place. This makes it impossible to mark inbound packets based on their destination address when SNAT or Masquerading are being used. You can cause packet marking to occur in the FORWARD chain by using the MARK_IN_FORWARD_CHAIN option in shorewall.conf.

Columns in the file are as follows:

- MARK - Specifies the mark value is to be assigned in case of a match. This is an integer in the range 1-255. This value may be optionally followed by “:” and either “F” or “P” to designate that the marking will occur in the FORWARD or PREROUTING chains respectively. If this additional specification is omitted, the chain used to mark packets will be determined by the setting of the MARK_IN_FORWARD_CHAIN option in shorewall.conf.
- SOURCE - The source of the packet. If the packet originates on the firewall, place “fw” in this column. Otherwise, this is a comma-separated list of interface names, IP addresses, MAC addresses in Shorewall Format and/or Subnets.

  Examples

  ```
  eth0     192.168.2.4,192.168.1.0/24
  ```
- DEST -- Destination of the packet. Comma-separated list of IP addresses and/or subnets.
- PROTO - Protocol - Must be the name of a protocol from /etc/protocol, a number or “all”
- PORT(S) - Destination Ports. A comma-separated list of Port names (from /etc/services), port numbers or port ranges (e.g., 21:22); if the protocol is “icmp”, this column is interpreted as the destination icmp type(s).
- CLIENT PORT(S) - (Optional) Port(s) used by the client. If omitted, any source port is acceptable. Specified as a comma-separate list of port names, port numbers or port ranges.
- USER/GROUP (Added in Shorewall version 1.4.10) - (Optional) This column may only be non-empty if the SOURCE is the firewall itself. When this column is non-empty, the rule applies only if the program generating the output is running under the effective user and/or group. It may contain :

  [!][<user name or number>]:[<group name or number>][+<program name>]

The colon is optionnal when specifying only a user.

Examples:

```
joe     #program must be run by joe
:kids   #program must be run by a member of the 'kids' group
!:kids  #program must not be run by a member of the 'kids' group
+upnpd  #program named upnpd (This feature was removed from Netfilter in kernel
version 2.6.14).
```

Example 1.

All packets arriving on eth1 should be marked with 1. All packets arriving on eth2 and eth3 should be marked with 2. All packets originating on the firewall itself should be marked with 3.

```
#MARK   SOURCE    DESTINATION    PROTOCOL   USER/GROUP
1       eth1      0.0.0.0/0      all
2       eth2      0.0.0.0/0      all
2       eth3      0.0.0.0/0      all
3       fw        0.0.0.0/0      all
```

Example 2.

All GRE (protocol 47) packets not originating on the firewall and destined for 155.186.235.151 should be marked with 12.

```
#MARK   SOURCE    DESTINATION     PROTOCOL   USER/GROUP
12      0.0.0.0/0 155.182.235.151 47
```

Example 3.

All SSH packets originating in 192.168.1.0/24 and destined for 155.186.235.151 should be marked with 22.

```
#MARK   SOURCE         DESTINATION     PROTOCOL   USER/GROUP
22      192.168.1.0/24 155.182.235.151 tcp        22
```

/etc/shorewall/tcdevices

This file allows you to define the incoming and outgoing bandwidth for the devices you want traffic shaping to be enabled. That means, if you want to use traffic shaping for a device, you have to define it here.

Columns in the file are as follows:

- INTERFACE - Name of interface. Each interface may be listed only once in this file. You may NOT specify the name of an alias (e.g., eth0:0) here; see FAQ #18. You man NOT specify wildcards here, e.g. if you have multiple ppp interfaces, you need to put them all in here!
- IN-BANDWIDTH - The incoming Bandwidth of that interface. Please note that you are not able to do traffic shaping on incoming traffic, as the traffic is already received before you could do so. This Column allows you to define the maximum traffic allowed for this interface in total, if the rate is exceeded, the packets are dropped. You want this mainly if you have a DSL or Cable Connection to avoid queuing at your providers side. If you don't want any traffic to be dropped set this to a value faster than your interface maximum rate.
- OUT-BANDWIDTH - Specifiy the outgoing bandwidth of that interface. This is the maximum speed your connection can handle. It is also the speed you can refer as "full" if you define the tc classes. Outgoing traffic above this rate will be dropped.

Example 4.

Suppose you are using PPP over Ethernet (DSL) and ppp0 is the interface for this. The device has an outgoing bandwidth of 500kbit and an incoming bandwidth of 6000kbit

```
#INTERFACE    IN-BANDWITH      OUT-BANDWIDTH
ppp0           6000kbit         500kbit
```

/etc/shorewall/tcclasses

This file allows you to define the actual classes that are used to split the outgoing traffic.

- INTERFACE - Name of interface. Each interface may be listed only once in this file. You may NOT specify the name of an alias (e.g., eth0:0) here; see FAQ #18. You man NOT specify wildcards here, e.g. if you have multiple ppp interfaces, you need to put them all in here!
- MARK - The mark value which is an integer in the range 1-255. You define these marks in the tcrules file, marking the traffic you want to go into the queueing classes defined in here. You can use the same marks for different Interfaces.
- RATE - The minimum bandwidth this class should get, when the traffic load rises. Please note that first the classes which equal or a lesser priority value are served even if there are others that have a guaranteed bandwith but a lower priority.
- CEIL - The maximum bandwidth this class is allowed to use when the link is idle. Useful if you have traffic which can get full speed when more important services (e.g. interactive like ssh) are not used. You can use the value "full" in here for setting the maximum bandwidth to the defined output bandwidth of that interface.
- PRIORITY - you have to define a priority for the class. packets in a class with a higher priority (=lesser value) are handled before less priorized onces. You can just define the mark value here also, if you are increasing the mark values with lesser priority.
- OPTIONS - A comma-separated list of options including the following:
  - default - this is the default class for that interface where all traffic should go, that is not classified otherwise.

    ### Note

    defining default for exactly **one** class per interface is mandatory!

  - tos-<tosname> - this lets you define a filter for the given <tosname> which lets you define a value of the Type Of Service bits in the ip package which causes the package to go in this class. Please note, that this filter overrides all mark settings, so if you define a tos filter for a class all traffic having that mark will go in it regardless of the mark on the package. You can use the following for this option: tos-minimize-delay (16) tos-maximize-throughput (8) tos-maximize-reliability (4) tos-minimize-cost (2) tos-normal-service (0)

    ### Note

    Each of this options is only valid for **one** class per interface.

  - tcp-ack - if defined causes an tc filter to be created that puts all tcp ack packets on that interface that have an size of <=64 Bytes to go in this class. This is useful for speeding up downloads. Please note that the size of the ack packages is limited to 64 bytes as some applications (p2p for example) use to make every package an ack package which would cause them all into here. We want only packages WITHOUT payload to match, so the size limit. Bigger packets just take their normal way into the classes.

    ### Note

    This option is only valid for **class** per interface.

ppp devices

If you use ppp/pppoe/pppoa) to connect to your internet provider and you use traffic shaping you need to restart shorewall traffic shaping. The reason for this is, that if the ppp connection gets restartet (and it usally does this at least daily), all “tc” filters/qdiscs related to that interface are deleted.

The easiest way to achieve this, is just to restart shorewall once the link is up. To achieve this add a small executable script to“/etc/ppp/ip-up.d”.

```
#! /bin/sh

/sbin/shorewall restart
```

Real life examples

Configuration to replace Wondershaper

You are able to fully replace the wondershaper script by using the buitin traffic control.You can find example configuration files at "http://www1.shorewall.net/pub/shorewall/Samples/tc4shorewall/. Please note that they are just examples and need to be adjusted to work for you. In this examples it is assumed that your interface for you internet connection is ppp0 (for DSL) , if you use another connection type, you have to change it. You also need to change the settings in the tcdevices.wondershaper file to reflect your line speed. The relevant lines of the config files follow here. Please note that this is just an 1:1 replacement doing exactly what wondershaper should do. You are free to change it...

tcdevices file

```
#INTERFACE    IN-BANDWITH      OUT-BANDWIDTH
ppp0            5000kbit        500kbit
```

tcclasses file

```
#INTERFACE      MARK    RATE            CEIL        PRIORITY    OPTIONS
ppp0            1       full            full            1       tcp-ack,tos-minimize-
delay
ppp0            2       9*full/10       9*full/10       2       default
ppp0            3       8*full/10       8*full/10       2
```

tcrules file

```
#MARK           SOURCE          DEST            PROTO   PORT(S) CLIENT   USER
#                                                               PORT(S)
1:P             0.0.0.0/0       0.0.0.0/0       icmp    echo-request
1:P             0.0.0.0/0       0.0.0.0/0       icmp    echo-reply
# mark traffic which should have a lower priority with a 3:
# mldonkey
3               0.0.0.0/0       0.0.0.0/0       udp     -        4666
```

Wondershaper allows you to define a set of hosts and/or ports you want to classify as low priority. To achieve this , you have to add these hosts to tcrules and set the mark to 3 (true if you use the example configuration files).

Setting hosts to low priority

lets assume the following settings from your old wondershaper script (don't assume these example values are really useful, they are only used for demonstrating ;-):

```
# low priority OUTGOING traffic - you can leave this blank if you want
```

```
# low priority source netmasks
NOPRIOHOSTSRC="192.168.1.128/25 192.168.3.28"

# low priority destination netmasks
NOPRIOHOSTDST=60.0.0.0/24

# low priority source ports
NOPRIOPORTSRC="6662 6663"

# low priority destination ports
NOPRIOPORTDST="6662 6663"
```

This would result in the following additional settings to the tcrules file:

```
3               192.168.1.128/25 0.0.0.0/0      all
3               192.168.3.28     0.0.0.0/0      all
3               0.0.0.0/0        60.0.0.0/24    all
3               0.0.0.0/0        0.0.0.0/0      udp     6662,6663
3               0.0.0.0/0        0.0.0.0/0      udp     -         6662,6663
3               0.0.0.0/0        0.0.0.0/0      tcp     6662,6663
3               0.0.0.0/0        0.0.0.0/0      tcp     -         6662,6663
```

A simple setup

This is a simple setup for people sharing an internet connection and using different computers for this. It just basically shapes between 2 hosts which have the ip addresses 192.168.2.23 and 192.168.2.42

tcdevices file

```
#INTERFACE    IN-BANDWITH      OUT-BANDWIDTH
ppp0            6000kbit        700kbit
```

We have 6mbit down and 700kbit upstream.

tcclasses file

```
#INTERFACE      MARK    RATE            CEIL            PRIORITY    OPTIONS
ppp0            1       10kbit          50kbit          1           tcp-ack
ppp0            2       300kbit         full            2
ppp0            3       300kbit         full            2
ppp0            4       90kbit          200kbit         3           default
```

We add a class for tcp ack packets with highest priority, so that downloads are fast. The following 2 classes share most of the bandwidth between the 2 hosts, if the connection is idle, they may use full speed. As the hosts should be treated equally they have the same priority. The last class is for the remaining traffic.

tcrules file

```
#MARK            SOURCE          DEST            PROTO   PORT(S)      CLIENT   USER
#                                                                     PORT(S)
1:P              0.0.0.0/0       0.0.0.0/0       icmp    echo-request
1:P              0.0.0.0/0       0.0.0.0/0       icmp    echo-reply
2:P              192.168.2.23    0.0.0.0/0       all
3:P              192.168.2.42    0.0.0.0/0       all
```

We mark icmp ping and replies so they will go into the fast interactive class and set a mark for each host.

Using your own tc script

Replacing builtin tcstart file

If you prefer your own tcstart file, just install it in /etc/shorewall/tcstart.

In your tcstart script, when you want to run the “tc” utility, use the run_tc function supplied by Shorewall if you want tc errors to stop the firewall.

1. Set TC_ENABLED=Yes and CLEAR_TC=Yes
2. Supply an /etc/shorewall/tcstart script to configure your traffic shaping rules.
3. Optionally supply an /etc/shorewall/tcclear script to stop traffic shaping. That is usually unnecessary.
4. If your tcstart script uses the “fwmark” classifier, you can mark packets using entries in /etc/shorewall/tcrules.

Traffic control outside Shorewall

To start traffic shaping when you bring up your network interfaces, you will have to arrange for your traffic shaping configuration script to be run at that time. How you do that is distribution dependent and will not be covered here. You then should:

1. Set TC_ENABLED=No and CLEAR_TC=No
2. If your script uses the “fwmark” classifier, you can mark packets using entries in /etc/shorewall/tcrules.

Shorewall FAQs

Shorewall Community

Tom Eastep

Copyright © 2001-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled " GNU Free Documentation License ".

2005-10-25

Table of Contents

Installing Shorewall
- Where do I find Step by Step Installation and Configuration Instructions?
- (FAQ 37) I just installed Shorewall on Debian and the /etc/shorewall directory is empty!!!
- (FAQ 44) I can't install/upgrade the RPM — I keep getting the message "error: failed dependencies:iproute is needed..."
- (FAQ 50) When I install/upgrade I get multiple instance of the message "warning: user teastep does not exist - using root"

Port Forwarding (Port Redirection)
- (FAQ 1) I want to forward UDP port 7777 to my personal PC with IP address 192.168.1.5. I've looked everywhere and can't find how to do it.
  - (FAQ 1a) Ok -- I followed those instructions but it doesn't work
  - (FAQ 1b) I'm still having problems with port forwarding
  - (FAQ 1c) From the internet, I want to connect to port 1022 on my firewall and have the firewall forward the connection to port 22 on local system 192.168.1.3. How do I do that?
  - (FAQ 1d) I have a web server in my DMZ and I use port forwarding to make that server accessible from the Internet. That works fine but when my local users try to connect to the server using the Firewall's external IP address, it doesn't work.
  - (FAQ 1e) In order to discourage brute force attacks I would like to redirect all connections on a non-standard port (4104) to port 22 on the router/firewall. I notice that setting up a REDIRECT rule causes the firewall to open both ports 4104 and 22 to connections from the net. Is it possible to only redirect 4104 to the localhost port 22 and have connection attempts to port 22 from the net dropped?
- (FAQ 30) I'm confused about when to use DNAT rules and when to use ACCEPT rules.
- (FAQ 38) Where can I find more information about DNAT?
- (FAQ 48) How do I Set up Transparent Proxy with Shorewall?

DNS and Port Forwarding/NAT
- (FAQ 2) I port forward www requests to www.mydomain.com (IP 130.151.100.69) to system 192.168.1.5 in my local network. External clients can browse http://www.mydomain.com but internal clients can't.
  - (FAQ 2a) I have a zone "Z" with an RFC1918 subnet and I use one-to-one NAT to assign non-RFC1918 addresses to hosts in Z. Hosts in Z cannot communicate with each other using their external (non-RFC1918 addresses) so they can't access each other using their DNS names.
  - (FAQ 2b) I have a web server in my DMZ and I use port forwarding to make that server accessible from the Internet as www.mydomain.com. That works fine but when my local users try to connect to www.mydomain.com, it doesn't work.

Netmeeting/MSN
- (FAQ 3) I want to use Netmeeting or MSN Instant Messenger with Shorewall. What do I do?

Open Ports
- (FAQ 4) I just used an online port scanner to check my firewall and it shows some ports as "closed" rather than "blocked".

- Why?
  - (FAQ 4a) I just ran an nmap UDP scan of my firewall and it showed 100s of ports as open!!!!
  - (FAQ 4b) I have a port that I can't close no matter how I change my rules.
  - (FAQ 4c) How do I use Shorewall with PortSentry?
- (FAQ 51) How do I "Open a Port" with Shorewall

Connection Problems
- (FAQ 5) I've installed Shorewall and now I can't ping through the firewall
- (FAQ 15) My local systems can't see out to the net
- (FAQ 29) FTP Doesn't Work
- (FAQ 33) From clients behind the firewall, connections to some sites fail. Connections to the same sites from the firewall itself work fine. What's wrong.
- (FAQ 35) I have two Ethernet interfaces to my local network which I have bridged. When Shorewall is started, I'm unable to pass traffic through the bridge. I have defined the bridge interface (br0) as the local interface in /etc/shorewall/interfaces; the bridged Ethernet interfaces are not defined to Shorewall. How do I tell Shorewall to allow traffic through the bridge?

Logging
- (FAQ 6) Where are the log messages written and how do I change the destination?
  - (FAQ 6a) Are there any log parsers that work with Shorewall?
  - (FAQ 6b) DROP messages on port 10619 are flooding the logs with their connect requests. Can i exclude these error messages for this port temporarily from logging in Shorewall?
  - (FAQ 6d) Why is the MAC address in Shorewall log messages so long? I thought MAC addresses were only 6 bytes in length.
- (FAQ 16) Shorewall is writing log messages all over my console making it unusable!
- (FAQ 17) Why are these packets being Dropped/Rejected?/How do I decode Shorewall log messages?
- (FAQ 21) I see these strange log entries occasionally; what are they?

Routing
- (FAQ 32) My firewall has two connections to the internet from two different ISPs. How do I set this up in Shorewall?
- (FAQ 49) When I start Shorewall, my routing table gets blown away. Why does Shorewall do that?

Starting and Stopping
- (FAQ 7) When I stop Shorewall using “shorewall stop”, I can't connect to anything. Why doesn't that command work?
- (FAQ 8) When I try to start Shorewall on RedHat, I get messages about insmod failing -- what's wrong?
  - (FAQ 8a) When I try to start Shorewall on RedHat I get a message referring me to FAQ #8
- (FAQ 9) Why can't Shorewall detect my interfaces properly at startup?
- (FAQ 22) I have some iptables commands that I want to run when Shorewall starts. Which file do I put them in?
- (FAQ 34) How can I speed up start (restart)?
- (FAQ 43) I just installed the Shorewall RPM and Shorewall doesn't start at boot time.
- (FAQ 45) Why does "shorewall start fail" when trying to set up SNAT/Masquerading?

About Shorewall
- (FAQ 10) What Distributions does it work with?
- (FAQ 11) What Features does it have?
- (FAQ 12) Is there a GUI?
- (FAQ 13) Why do you call it “Shorewall”?
- (FAQ 23) Why do you use such ugly fonts on your web site?
- (FAQ 25) How to I tell which version of Shorewall I am running?
- (FAQ 31) Does Shorewall provide protection against....
- (FAQ 36) Does Shorewall Work with the 2.6 Linux Kernel?

RFC 1918
- (FAQ 14) I'm connected via a cable modem and it has an internal web server that allows me to configure/monitor it but as expected if I enable rfc1918 blocking for my eth0 interface (the internet one), it also blocks the cable modems web server.
  - (FAQ 14a) Even though it assigns public IP addresses, my ISP's DHCP server has an RFC 1918 address. If I enable RFC 1918 filtering on my external interface, my DHCP client cannot renew its lease.
  - (FAQ 14b) I connect to the internet with PPPoE. When I try to access the built-in web server in my DSL Modem, I get connection Refused.

Alias IP Addresses/Virtual Interfaces
- (FAQ 18) Is there any way to use aliased ip addresses with Shorewall, and maintain separate rulesets for different IPs?

Miscellaneous
(FAQ 19) I have added entries to /etc/shorewall/tcrules but they don't seem to do anything. Why?
(FAQ 20) I have just set up a server. Do I have to change Shorewall to allow access to my server from the internet?
(FAQ 24) How can I allow conections to let's say the ssh port only from specific IP Addresses on the internet?
(FAQ 26) When I try to use any of the SYN options in nmap on or behind the firewall, I get “operation not permitted”. How can I use nmap with Shorewall?"
(FAQ 27) I'm compiling a new kernel for my firewall. What should I look out for?
(FAQ 27a) I just built (or downloaded or otherwise acquired) and installed a new kernel and now Shorewall won't start. I know that my kernel options are correct.
(FAQ 28) How do I use Shorewall as a Bridging Firewall?
(FAQ 39) How do I block connections to a particular domain name?
(FAQ 42) How can I tell which features my kernel and iptables support?

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Installing Shorewall

Where do I find Step by Step Installation and Configuration Instructions?

Answer: Check out the QuickStart Guides.

(FAQ 37) I just installed Shorewall on Debian and the /etc/shorewall directory is empty!!!

Important

Once you have installed the .deb package and before you attempt to configure Shorewall, please heed the advice of Lorenzo Martignoni, the Shorewall Debian Maintainer:

“For more information about Shorewall usage on Debian system please look at /usr/share/doc/shorewall/README.Debian provided by [the] shorewall Debian package.”

If you install using the .deb, you will find that your `/etc/shorewall` directory is empty. This is intentional. The released configuration file skeletons may be found on your system in the directory `/usr/share/doc/shorewall/default-config`. Simply copy the files you need from that directory to `/etc/shorewall` and modify the copies.

Note that you must copy `/usr/share/doc/shorewall/default-config/shorewall.conf` and `/usr/share/doc/shorewall/default-config/modules` to `/etc/shorewall` even if you do not modify those files.

(FAQ 44) I can't install/upgrade the RPM — I keep getting the message "error: failed dependencies:iproute is needed..."

Answer: Read the Installation Instructions!

(FAQ 50) When I install/upgrade I get multiple instance of the message "warning: user teastep does not exist - using root"

Answer: You may safely ignore this warning message. It was caused by a minor packaging error that has since been corrected. It makes no difference to Shorewall's operation.

Port Forwarding (Port Redirection)

(FAQ 1) I want to forward UDP port 7777 to my personal PC with IP address 192.168.1.5. I've looked everywhere and can't find how to do it.

Answer: The first example in the rules file documentation shows how to do port forwarding under Shorewall. The format of a port-forwarding rule to a local system is as follows:

```
#ACTION    SOURCE      DEST                                      PROTO        DEST PORT
DNAT       net         loc:<local IP address>[:<local port>]  <protocol>   <port #>
```

So to forward UDP port 7777 to internal system 192.168.1.5, the rule is:

```
#ACTION    SOURCE   DEST              PROTO    DEST PORT
DNAT       net      loc:192.168.1.5   udp      7777
```

If you want to forward requests directed to a particular address (*<external IP>*) on your firewall to an internal system:

```
#ACTION SOURCE DEST                                    PROTO       DEST PORT  SOURCE
ORIGINAL
#                                                                              PORT
DEST.
DNAT    net    loc:<local IP address>[:<local port>]  <protocol>  <port #>   -
<external IP>
```

Finally, if you need to forward a range of ports, in the DEST PORT column specify the range as *<low-port>:<high-port>*.

(FAQ 1a) Ok -- I followed those instructions but it doesn't work

Answer: That is usually the result of one of four things:

- You are trying to test from inside your firewall (no, that won't work -- see the section called "(FAQ 2) I port forward www requests to www.mydomain.com (IP 130.151.100.69) to system 192.168.1.5 in my local network. External clients can browse http://www.mydomain.com but internal clients can't.").
- You have a more basic problem with your local system (the one that you are trying to forward to) such as an incorrect default gateway (it should be set to the IP address of your firewall's internal interface).
- Your ISP is blocking that particular port inbound.
- You are running Mandrake Linux prior to 10.0 final and have configured Internet Connection Sharing. In that case, the name of your local zone is 'masq' rather than 'loc' (change all instances of 'loc' to 'masq' in your rules). You may want to consider re-installing Shorewall in a configuration which matches the Shorewall documentation. See the two-interface QuickStart Guide for details.

(FAQ 1b) I'm still having problems with port forwarding

Answer: To further diagnose this problem:

- As root, type " **iptables -t nat -Z** ". This clears the NetFilter counters in the nat table.
- Try to connect to the redirected port from an external host.
- As root type " **shorewall show nat** "
- Locate the appropriate DNAT rule. It will be in a chain called *<source zone>*_dnat ("net_dnat" in the above examples).
- Is the packet count in the first column non-zero? If so, the connection request is reaching the firewall and is being redirected to the server. In this case, the problem is usually a missing or incorrect default gateway setting on the local system (the system you are trying to forward to -- its default gateway should be the IP address of the firewall's interface to that system).
- If the packet count is zero:

- the connection request is not reaching your server (possibly it is being blocked by your ISP); or
- you are trying to connect to a secondary IP address on your firewall and your rule is only redirecting the primary IP address (You need to specify the secondary IP address in the “ORIG. DEST.” column in your DNAT rule); or
- your DNAT rule doesn't match the connection request in some other way. In that case, you may have to use a packet sniffer such as tcpdump or ethereal to further diagnose the problem.

(FAQ 1c) From the internet, I want to connect to port 1022 on my firewall and have the firewall forward the connection to port 22 on local system 192.168.1.3. How do I do that?

In /`etc/shorewall/rules`:

```
#ACTION     SOURCE    DEST                  PROTO     DEST PORT
DNAT        net       loc:192.168.1.3:22    tcp       1022
```

(FAQ 1d) I have a web server in my DMZ and I use port forwarding to make that server accessible from the Internet. That works fine but when my local users try to connect to the server using the Firewall's external IP address, it doesn't work.

Answer: Let's assume the following:

- External IP address is 206.124.146.176 on `eth0`.
- Server's IP address is 192.168.2.4

You can enable access to the server from your local network using the firewall's external IP address by adding this rule:

```
#ACTION     SOURCE    DEST                  PROTO     DEST PORT(S)     SOURCE       ORIGINAL
#                                                                      PORT         DEST
DNAT        loc       dmz:192.168.2.4       tcp       80               -
206.124.146.176
```

If your external IP address is dynamic, then you must do the following:

In `/etc/shorewall/init`:

```
ETH0_IP=`find_interface_address eth0`
```

For users of Shorewall 2.1.0 and later:

```
ETH0_IP=`find_first_interface_address eth0`
```

and make your DNAT rule:

```
#ACTION     SOURCE         DEST                 PROTO     DEST PORT    SOURCE     ORIGINAL
#                                                                       PORT       DEST.
DNAT        loc            dmz:192.168.2.4      tcp       80           -          $ETH0_IP
```

(FAQ 1e) In order to discourage brute force attacks I would like to redirect all connections on a non-standard port (4104) to port 22 on the router/firewall. I notice that setting up a REDIRECT rule causes the firewall to open both ports 4104 and 22 to connections from the net. Is it possible to only redirect 4104 to the localhost port 22 and have connection attempts to port 22 from the net dropped?

Answer courtesy of Ryan: Assume that the IP address of your local firewall interface is 192.168.1.1. If you add the following rule then from the net, you will have 4104 listening, from your LAN, port 22.

```
#ACTION SOURCE  DEST                     PROTO   DEST PORT(S)
```

```
DNAT    net     fw:192.168.1.1:22       tcp     4104
```

(FAQ 30) I'm confused about when to use DNAT rules and when to use ACCEPT rules.

It would be a good idea to review the QuickStart Guide appropriate for your setup; the guides cover this topic in a tutorial fashion. DNAT rules should be used for connections that need to go the opposite direction from SNAT/MASQUERADE. So if you masquerade or use SNAT from your local network to the internet then you will need to use DNAT rules to allow connections from the internet to your local network. In all other cases, you use ACCEPT unless you need to hijack connections as they go through your firewall and handle them on the firewall box itself; in that case, you use a REDIRECT rule.

(FAQ 38) Where can I find more information about DNAT?

Ian Allen has written a Paper about DNAT and Linux.

(FAQ 48) How do I Set up Transparent Proxy with Shorewall?

Answer: See Shorewall_Squid_Usage.html.

DNS and Port Forwarding/NAT

(FAQ 2) I port forward www requests to www.mydomain.com (IP 130.151.100.69) to system 192.168.1.5 in my local network. External clients can browse http://www.mydomain.com but internal clients can't.

Answer: I have two objections to this setup.

- Having an internet-accessible server in your local network is like raising foxes in the corner of your hen house. If the server is compromised, there's nothing between that server and your other internal systems. For the cost of another NIC and a cross-over cable, you can put your server in a DMZ such that it is isolated from your local systems - assuming that the Server can be located near the Firewall, of course :-)
- The accessibility problem is best solved using Bind Version 9 “views” (or using a separate DNS server for local clients) such that www.mydomain.com resolves to 130.141.100.69 externally and 192.168.1.5 internally. That's what I do here at shorewall.net for my local systems that use one-to-one NAT.

Assuming that your external interface is eth0 and your internal interface is eth1 and that eth1 has IP address 192.168.1.254 with subnet 192.168.1.0/24, then:

Warning

All traffic redirected through use of this hack will look to the server as if it came from the firewall (192.168.1.254) rather than from the original client!

- In `/etc/shorewall/interfaces`:

```
#ZONE    INTERFACE    BROADCAST    OPTIONS
loc      eth1         detect       routeback
```

- In `/etc/shorewall/masq`:

```
#INTERFACE              SUBNET          ADDRESS         PROTO   PORT(S)
eth1:192.168.1.5        eth1            192.168.1.254   tcp     www
```

- In `/etc/shorewall/rules`:

```
#ACTION    SOURCE       DEST                PROTO    DEST PORT   SOURCE    ORIGINAL
```

```
#                                                                    PORT      DEST.
DNAT        loc           loc:192.168.1.5    tcp      www          -
130.151.100.69
```

That rule only works of course if you have a static external IP address. If you have a dynamic IP address then include this in `/etc/shorewall/init`:

```
ETH0_IP=`find_first_interface_address eth0`
```

and make your DNAT rule:

```
#ACTION     SOURCE         DEST                PROTO     DEST PORT    SOURCE     ORIGINAL
#                                                                      PORT       DEST.
DNAT        loc            loc:192.168.1.5     tcp       www          -          $ETH0_IP
```

Using this technique, you will want to configure your DHCP/PPPoE client to automatically restart Shorewall each time that you get a new IP address.

(FAQ 2a) I have a zone "Z" with an RFC1918 subnet and I use one-to-one NAT to assign non-RFC1918 addresses to hosts in Z. Hosts in Z cannot communicate with each other using their external (non-RFC1918 addresses) so they can't access each other using their DNS names.

Note

If the ALL INTERFACES column in /etc/shorewall/nat is empty or contains "Yes", you will also see log messages like the following when trying to access a host in Z from another host in Z using the destination hosts's public address:

```
Oct 4 10:26:40 netgw kernel:
          Shorewall:FORWARD:REJECT:IN=eth1 OUT=eth1 SRC=192.168.118.200
          DST=192.168.118.210 LEN=48 TOS=0x00 PREC=0x00 TTL=127 ID=1342 DF
          PROTO=TCP SPT=1494 DPT=1491 WINDOW=17472 RES=0x00 ACK SYN URGP=0
```

Answer: This is another problem that is best solved using Bind Version 9 "views". It allows both external and internal clients to access a NATed host using the host's DNS name.

Another good way to approach this problem is to switch from one-to-one NAT to Proxy ARP. That way, the hosts in Z have non-RFC1918 addresses and can be accessed externally and internally using the same address.

If you don't like those solutions and prefer to stupidly route all Z->Z traffic through your firewall then:

1. Set the routeback option on the interface to Z.
2. Set the ALL INTERFACES column in the nat file to "Yes".

Example 1. Example:

Zone: dmz Interface: eth2 Subnet: 192.168.2.0/24 Address: 192.168.2.254

In `/etc/shorewall/interfaces`:

```
#ZONE     INTERFACE     BROADCAST        OPTIONS
dmz       eth2          192.168.2.255    routeback
```

In `/etc/shorewall/nat`, be sure that you have "Yes" in the ALL INTERFACES column.

In /etc/shorewall/masq:

```
#INTERFACE     SUBNETS      ADDRESS
eth2           eth2         192.168.2.254
```

Like the idiotic hack in FAQ 2 above, this will make all dmz->dmz traffic appear to originate on the firewall.

(FAQ 2b) I have a web server in my DMZ and I use port forwarding to make that server accessible from the Internet as www.mydomain.com. That works fine but when my local users try to connect to www.mydomain.com, it doesn't work.

Answer: Let's assume the following:

- External IP address is 206.124.146.176 on `eth0` (www.mydomain.com).
- Server's IP address is 192.168.2.4

You can enable access to the server from your local network using the firewall's external IP address by adding this rule:

```
#ACTION     SOURCE    DEST                 PROTO     DEST PORT(S)     SOURCE       ORIGINAL
#                                                                     PORT         DEST
DNAT        loc       dmz:192.168.2.4      tcp       80               -
206.124.146.176
```

If your external IP address is dynamic, then you must do the following:

In `/etc/shorewall/init`:

```
ETH0_IP=`find_first_interface_address eth0`
```

and make your DNAT rule:

```
#ACTION     SOURCE         DEST                 PROTO     DEST PORT    SOURCE     ORIGINAL
#                                                                       PORT       DEST.
DNAT        loc            dmz:192.168.2.4      tcp       80           -          $ETH0_IP
```

Netmeeting/MSN

(FAQ 3) I want to use Netmeeting or MSN Instant Messenger with Shorewall. What do I do?

Answer: There is an H.323 connection tracking/NAT module that helps with Netmeeting. Note however that one of the Netfilter developers recently posted the following:

```
> I know PoM -ng is going to address this issue, but till it is ready, and
> all the extras are ported to it, is there any way to use the h.323
> contrack module kernel patch with a 2.6 kernel?
> Running 2.6.1 - no 2.4 kernel stuff on the system, so downgrade is not
> an option... The module is not ported yet to 2.6, sorry.
> Do I have any options besides a gatekeeper app (does not work in my
> network) or a proxy (would prefer to avoid them)?

I suggest everyone to setup a proxy (gatekeeper) instead: the module is
really dumb and does not deserve to exist at all. It was an excellent tool
to debug/develop the newnat interface.
```

Look here for a solution for MSN IM but be aware that there are significant security risks involved with this solution. Also check

the Netfilter mailing list archives at http://www.netfilter.org.

Open Ports

(FAQ 4) I just used an online port scanner to check my firewall and it shows some ports as “closed” rather than “blocked”. Why?

Answer: The default Shorewall setup invokes the **Drop** action prior to enforcing a DROP policy and the default policy to all zone from the internet is DROP. The Drop action is defined in `/usr/share/shorewall/action.Drop` which in turn invokes the **RejectAuth** action (defined in `/usr/share/shorewall/action.RejectAuth`). This is necessary to prevent outgoing connection problems to services that use the “Auth” mechanism for identifying requesting users. That is the only service which the default setup rejects.

If you are seeing closed TCP ports other than 113 (auth) then either you have added rules to REJECT those ports or a router outside of your firewall is responding to connection requests on those ports.

(FAQ 4a) I just ran an nmap UDP scan of my firewall and it showed 100s of ports as open!!!!

Answer: Take a deep breath and read the nmap man page section about UDP scans. If nmap gets **nothing** back from your firewall then it reports the port as open. If you want to see which UDP ports are really open, temporarily change your net->all policy to REJECT, restart Shorewall and do the nmap UDP scan again.

(FAQ 4b) I have a port that I can't close no matter how I change my rules.

I had a rule that allowed telnet from my local network to my firewall; I removed that rule and restarted Shorewall but my telnet session still works!!!

Answer: Rules only govern the establishment of new connections. Once a connection is established through the firewall it will be usable until disconnected (tcp) or until it times out (other protocols). If you stop telnet and try to establish a new session your firerwall will block that attempt.

(FAQ 4c) How do I use Shorewall with PortSentry?

Here's a writeup on a nice integration of Shorewall and PortSentry.

(FAQ 51) How do I "Open a Port" with Shorewall

Answer: It depends…

If the application serving the port is running on the same system as Shorewall then add this rule:

```
#ACTION        SOURCE          DEST        PROTO          DEST PORT(S)
ACCEPT         net             $FW         <protocol>     <port number>
```

Where <protocol> is either *tcp* or *udp* and <port number> is the port that you wish to "open".

If the application serving the port is running on one of the systems in your local network then please see FAQ 1.

Connection Problems

(FAQ 5) I've installed Shorewall and now I can't ping through the firewall

Answer: For a complete description of Shorewall “ping” management, see this page.

(FAQ 15) My local systems can't see out to the net

Answer: Every time I read “systems can't see out to the net”, I wonder where the poster bought computers with eyes and what those computers will “see” when things are working properly. That aside, the most common causes of this problem are:

1. The default gateway on each local system isn't set to the IP address of the local firewall interface.
2. The entry for the local network in the /etc/shorewall/masq file is wrong or missing.
3. The DNS settings on the local systems are wrong or the user is running a DNS server on the firewall and hasn't enabled UDP and TCP port 53 from the firewall to the internet.

(FAQ 29) FTP Doesn't Work

See the Shorewall and FTP page.

(FAQ 33) From clients behind the firewall, connections to some sites fail. Connections to the same sites from the firewall itself work fine. What's wrong.

Answer: Most likely, you need to set CLAMPMSS=Yes in /etc/shorewall/shorewall.conf.

(FAQ 35) I have two Ethernet interfaces to my local network which I have bridged. When Shorewall is started, I'm unable to pass traffic through the bridge. I have defined the bridge interface (br0) as the local interface in /etc/shorewall/interfaces; the bridged Ethernet interfaces are not defined to Shorewall. How do I tell Shorewall to allow traffic through the bridge?

Answer: Add the *routeback* option to `br0` in /etc/shorewall/interfaces.

For more information on this type of configuration, see the Shorewall Simple Bridge documentation.

Logging

(FAQ 6) Where are the log messages written and how do I change the destination?

Answer: NetFilter uses the kernel's equivalent of syslog (see “man syslog”) to log messages. It always uses the LOG_KERN (kern) facility (see “man openlog”) and you get to choose the log level (again, see “man syslog”) in your policies and rules. The destination for messages logged by syslog is controlled by `/etc/syslog.conf` (see “man syslog.conf”). When you have changed /etc/syslog.conf, be sure to restart syslogd (on a RedHat system, “service syslog restart”).

By default, older versions of Shorewall ratelimited log messages through settings in `/etc/shorewall/shorewall.conf` -- If you want to log all messages, set:

```
LOGLIMIT=""
LOGBURST=""
```

It is also possible to set up Shorewall to log all of its messages to a separate file.

(FAQ 6a) Are there any log parsers that work with Shorewall?

Answer: Here are several links that may be helpful:

http://www.shorewall.net/pub/shorewall/parsefw/
http://www.fireparse.com

http://cert.uni-stuttgart.de/projects/fwlogwatch
http://www.logwatch.org
http://gege.org/iptables
http://home.regit.org/ulogd-php.html

I personally use Logwatch. It emails me a report each day from my various systems with each report summarizing the logged activity on the corresponding system.

(FAQ 6b) DROP messages on port 10619 are flooding the logs with their connect requests. Can i exclude these error messages for this port temporarily from logging in Shorewall?

Temporarily add the following rule:

```
DROP net fw udp 10619
```

(FAQ 6d) Why is the MAC address in Shorewall log messages so long? I thought MAC addresses were only 6 bytes in length.

What is labeled as the MAC address in a Shorewall log message is actually the Ethernet frame header. It contains:

- the destination MAC address (6 bytes)
- the source MAC address (6 bytes)
- the ethernet frame type (2 bytes)

Example 2. Example

```
MAC=00:04:4c:dc:e2:28:00:b0:8e:cf:3c:4c:08:00
```

- Destination MAC address = 00:04:4c:dc:e2:28
- Source MAC address = 00:b0:8e:cf:3c:4c
- Ethernet Frame Type = 08:00 (IP Version 4)

(FAQ 16) Shorewall is writing log messages all over my console making it unusable!

Answer:

- Find where klogd is being started (it will be from one of the files in /etc/init.d -- sysklogd, klogd, ...). Modify that file or the appropriate configuration file so that klogd is started with “-c <*n*> ” where <*n*> is a log level of 5 or less; or
- See the “dmesg” man page (“man dmesg”). You must add a suitable “dmesg” command to your startup scripts or place it in /etc/shorewall/start.

Tip

Under RedHat and Mandrake, the max log level that is sent to the console is specified in /etc/sysconfig/init in the LOGLEVEL variable. Set “LOGLEVEL=5” to suppress info (log level 6) messages on the console.

Tip

Under Debian, you can set KLOGD=“-c 5” in `/etc/init.d/klogd` to suppress info (log level 6) messages on the console.

Tip

Under SuSE, add “-c 5” to KLOGD_PARAMS in /etc/sysconfig/syslog to suppress info (log level 6) messages on the

console.

(FAQ 17) Why are these packets being Dropped/Rejected?/How do I decode Shorewall log messages?

Answer: Logging of dropped/rejected packets occurs out of a number of chains (as indicated in the log message) in Shorewall:

man1918 or logdrop

The destination address is listed in `/usr/share/shorewall/rfc1918` with a **logdrop** target -- see `/usr/share/shorewall/rfc1918` .

rfc1918 or logdrop

The source or destination address is listed in `/usr/share/shorewall/rfc1918` with a **logdrop** target -- see `/usr/share/shorewall/rfc1918` .

all2<zone>, <zone>2all or all2all

You have a policy that specifies a log level and this packet is being logged under that policy. If you intend to ACCEPT this traffic then you need a rule to that effect.

<zone1>2<zone2>

Either you have a policy for **<zone1>** to **<zone2>** that specifies a log level and this packet is being logged under that policy or this packet matches a rule that includes a log level.

@<source>2<dest>

You have a policy for traffic from <**source**> to <**dest**> that specifies TCP connection rate limiting (value in the LIMIT:BURST column). The logged packet exceeds that limit and was dropped. Note that these log messages themselves are severely rate-limited so that a syn-flood won't generate a secondary DOS because of excessive log message. These log messages were added in Shorewall 2.2.0 Beta 7.

<interface>_mac

The packet is being logged under the **maclist** interface option.

logpkt

The packet is being logged under the **logunclean** interface option.

badpkt

The packet is being logged under the **dropunclean** interface option as specified in the **LOGUNCLEAN** setting in `/etc/shorewall/shorewall.conf` .

blacklst

The packet is being logged because the source IP is blacklisted in the `/etc/shorewall/blacklist` file.

INPUT or FORWARD

The packet has a source IP address that isn't in any of your defined zones (“shorewall check” and look at the printed zone definitions) or the chain is FORWARD and the destination IP isn't in any of your defined zones. If the chain is FORWARD and the IN and OUT interfaces are the same, then you probably need the **routeback** option on that interface in `/etc/shorewall/interfaces` or you need the **routeback** option in the relevant entry in `/etc/shorewall/hosts` .

OUTPUT

The packet has a destination IP address that isn't in any of your defined zones("shorewall check" and look at the printed zone definitions).

logflags

The packet is being logged because it failed the checks implemented by the **tcpflags** interface option.

Example 3. Here is an example:

```
Jun 27 15:37:56 gateway kernel:
        Shorewall:all2all:REJECT:IN=eth2
          OUT=eth1
          SRC=192.168.2.2
          DST=192.168.1.3 LEN=67 TOS=0x00 PREC=0x00 TTL=63 ID=5805 DF PROTO=UDP
        SPT=1803 DPT=53 LEN=47
```

Let's look at the important parts of this message:

all2all:REJECT

This packet was REJECTed out of the **all2all** chain -- the packet was rejected under the “all”->“all” REJECT policy (all2all above).

IN=eth2

the packet entered the firewall via eth2. If you see “IN=” with no interface name, the packet originated on the firewall itself.

OUT=eth1

if accepted, the packet would be sent on eth1. If you see “OUT=” with no interface name, the packet would be processed by the firewall itself.

SRC=192.168.2.2

the packet was sent by 192.168.2.2

DST=192.168.1.3

the packet is destined for 192.168.1.3

PROTO=UDP

UDP Protocol

DPT=53

The destination port is 53 (DNS)

For additional information about the log message, see http://logi.cc/linux/netfilter-log-format.php3.

In this case, 192.168.2.2 was in the “dmz” zone and 192.168.1.3 is in the “loc” zone. I was missing the rule:

```
ACCEPT dmz loc udp 53
```

(FAQ 21) I see these strange log entries occasionally; what are they?

```
Nov 25 18:58:52 linux kernel:
      Shorewall:net2all:DROP:IN=eth1 OUT=
      MAC=00:60:1d:f0:a6:f9:00:60:1d:f6:35:50:08:00 SRC=206.124.146.179
      DST=192.0.2.3 LEN=56 TOS=0x00 PREC=0x00 TTL=110 ID=18558 PROTO=ICMP
      TYPE=3 CODE=3 [SRC=192.0.2.3 DST=172.16.1.10 LEN=128 TOS=0x00 PREC=0x00
      TTL=47 ID=0 DF PROTO=UDP SPT=53 DPT=2857 LEN=108 ]
```

192.0.2.3 is external on my firewall... 172.16.0.0/24 is my internal LAN

Answer: While most people associate the Internet Control Message Protocol (ICMP) with “ping”, ICMP is a key piece of the internet. ICMP is used to report problems back to the sender of a packet; this is what is happening here. Unfortunately, where NAT is involved (including SNAT, DNAT and Masquerade), there are a lot of broken implementations. That is what you are seeing with

these messages. When Netfilter displays these messages, the part before the "[" describes the ICMP packet and the part between the "[" and "]" describes the packet for which the ICMP is a response.

Here is my interpretation of what is happening -- to confirm this analysis, one would have to have packet sniffers placed a both ends of the connection.

Host 172.16.1.10 behind NAT gateway 206.124.146.179 sent a UDP DNS query to 192.0.2.3 and your DNS server tried to send a response (the response information is in the brackets -- note source port 53 which marks this as a DNS reply). When the response was returned to to 206.124.146.179, it rewrote the destination IP TO 172.16.1.10 and forwarded the packet to 172.16.1.10 who no longer had a connection on UDP port 2857. This causes a port unreachable (type 3, code 3) to be generated back to 192.0.2.3. As this packet is sent back through 206.124.146.179, that box correctly changes the source address in the packet to 206.124.146.179 but doesn't reset the DST IP in the original DNS response similarly. When the ICMP reaches your firewall (192.0.2.3), your firewall has no record of having sent a DNS reply to 172.16.1.10 so this ICMP doesn't appear to be related to anything that was sent. The final result is that the packet gets logged and dropped in the all2all chain. I have also seen cases where the source IP in the ICMP itself isn't set back to the external IP of the remote NAT gateway; that causes your firewall to log and drop the packet out of the rfc1918 chain because the source IP is reserved by RFC 1918.

Routing

(FAQ 32) My firewall has two connections to the internet from two different ISPs. How do I set this up in Shorewall?

Answer: See this article on Shorewall and Routing.

(FAQ 49) When I start Shorewall, my routing table gets blown away. Why does Shorewall do that?

Answer: This is usually the consequence of a one-to-one nat configuration blunder:

1. Specifying the primary IP address for an interface in the EXTERNAL column of `/etc/shorewall/nat` even though the documentation (and the comments in the file) warn you not to do that.
2. Specifying ADD_IP_ALIASES=Yes and RETAIN_ALIASES=No in /etc/shorewall/shorewall.conf.

This combination causes Shorewall to delete the primary IP address from the network interface specified in the INTERFACE column which usually causes all routes out of that interface to be deleted. The solution is to **not specify the primary IP address of an interface in the EXTERNAL column**.

Starting and Stopping

(FAQ 7) When I stop Shorewall using “shorewall stop”, I can't connect to anything. Why doesn't that command work?

The “ **stop** ” command is intended to place your firewall into a safe state whereby only those hosts listed in `/etc/shorewall/routestopped`' are activated. If you want to totally open up your firewall, you must use the “ **shorewall clear** ” command.

(FAQ 8) When I try to start Shorewall on RedHat, I get messages about insmod failing -- what's wrong?

Answer: The output you will see looks something like this:

```
/lib/modules/2.4.17/kernel/net/ipv4/netfilter/ip_tables.o: init_module: Device or
resource busy
Hint: insmod errors can be caused by incorrect module parameters, including invalid
```

```
IO or IRQ parameters
/lib/modules/2.4.17/kernel/net/ipv4/netfilter/ip_tables.o: insmod
/lib/modules/2.4.17/kernel/net/ipv4/netfilter/ip_tables.o failed
/lib/modules/2.4.17/kernel/net/ipv4/netfilter/ip_tables.o: insmod ip_tables failed
iptables v1.2.3: can't initialize iptables table `nat': iptables who? (do you need to
insmod?)
Perhaps iptables or your kernel needs to be upgraded.
```

This problem is usually corrected through the following sequence of commands

```
service ipchains stop
chkconfig --delete ipchains
rmmod ipchains
```

Also, be sure to check the errata for problems concerning the version of iptables (v1.2.3) shipped with RH7.2.

(FAQ 8a) When I try to start Shorewall on RedHat I get a message referring me to FAQ #8

Answer: This is usually cured by the sequence of commands shown above in the section called "(FAQ 8) When I try to start Shorewall on RedHat, I get messages about insmod failing -- what's wrong?".

(FAQ 9) Why can't Shorewall detect my interfaces properly at startup?

I just installed Shorewall and when I issue the start command, I see the following:

```
Processing /etc/shorewall/params ...
Processing /etc/shorewall/shorewall.conf ...
Starting Shorewall...
Loading Modules...
Initializing...
Determining Zones...
   Zones: net loc
Validating interfaces file...
Validating hosts file...
Determining Hosts in Zones...
    Net Zone: eth0:0.0.0.0/0

        Local Zone: eth1:0.0.0.0/0
Deleting user chains...
Creating input Chains...
...
```

Why can't Shorewall detect my interfaces properly?

Answer: The above output is perfectly normal. The Net zone is defined as all hosts that are connected through eth0 and the local zone is defined as all hosts connected through `eth1`. If you are running Shorewall 1.4.10 or later, you can consider setting the **detectnets** interface option on your local interface (`eth1` in the above example). That will cause Shorewall to restrict the local zone to only those networks routed through that interface.

(FAQ 22) I have some iptables commands that I want to run when Shorewall starts. Which file do I put them in?

You can place these commands in one of the Shorewall Extension Scripts. Be sure that you look at the contents of the chain(s) that you will be modifying with your commands to be sure that the commands will do what they are intended. Many iptables commands published in HOWTOs and other instructional material use the -A command which adds the rules to the end of the chain. Most chains that Shorewall constructs end with an unconditional DROP, ACCEPT or REJECT rule and any rules that you add after that

will be ignored. Check “man iptables” and look at the -I (--insert) command.

(FAQ 34) How can I speed up start (restart)?

Using a light-weight shell such as **ash** can dramatically decrease the time required to **start** or **restart** Shorewall. See the SHOREWALL_SHELL variable in `shorewall.conf` .

Use a fast terminal emulator -- in particular the KDE konsole scrolls much faster than the Gnome terminal. Also use the '-q' option if you are restarting remotely or from a slow terminal (or redirect the output to a file as in **shorewall restart > /dev/null**).

Shorewall also supports a fast start capability. To use this capability:

1. With Shorewall in the started state, run **shorewall save**. This creates the script `/var/lib/shorewall/restore`.
2. Use the **-f** option to the start command (e.g., **shorewall -f start**). This causes Shorewall to look for the `/var/lib/shorewall/restore` script and if that script exists, it is run. Running `/var/lib/shorewall/restore` takes much less time than a full **shorewall start**.
3. The `/etc/init.d/shorewall` script that is run at boot time uses the **-f** option.
4. The `/var/lib/shorewall/restore` script can be run any time to restore the firewall. The script may be run directly or it may be run indirectly using the **shorewall restore** command.

If you change your Shorewall configuration, you must execute a **shorewall start** (without **-f**) or **shorewall restart** prior to doing another **shorewall save**. The **shorewall save** command saves the currently running configuration and not the one reflected in your updated configuration files.

Likewise, if you change your Shorewall configuration then once you are satisfied that it is working properly, you must do another **shorewall save**. Otherwise at the next reboot, you will revert to the old configuration stored in `/var/lib/shorewall/restore`.

(FAQ 43) I just installed the Shorewall RPM and Shorewall doesn't start at boot time.

Answer: When you install using the "rpm -U" command, Shorewall doesn't run your distribution's tool for configuring Shorewall startup. You will need to run that tool (insserv, chkconfig, run-level editor, …) to configure Shorewall to start in the run-levels that you run your firewall system at.

(FAQ 45) Why does "shorewall start fail" when trying to set up SNAT/Masquerading?

shorewall start produces the following output:

```
…
Processing /etc/shorewall/policy...
   Policy ACCEPT for fw to net using chain fw2net
   Policy ACCEPT for loc0 to net using chain loc02net
   Policy ACCEPT for loc1 to net using chain loc12net
   Policy ACCEPT for wlan to net using chain wlan2net
Masqueraded Networks and Hosts:
iptables: Invalid argument
   ERROR: Command "/sbin/iptables -t nat -A …" Failed
```

Answer: 99.999% of the time, this error is caused by a mismatch between your iptables and kernel.

a. Your iptables must be compiled against a kernel source tree that is Netfilter-compatible with the kernel that you are running.
b. If you rebuild iptables using the defaults and install it, it will be installed in /usr/local/sbin/iptables. As shown above, you have the IPTABLES variable in shorewall.conf set to "/sbin/iptables".

About Shorewall

(FAQ 10) What Distributions does it work with?

Shorewall works with any GNU/Linux distribution that includes the proper prerequisites.

(FAQ 11) What Features does it have?

Answer: See the Shorewall Feature List.

(FAQ 12) Is there a GUI?

Answer: Yes. Shorewall support is included in Webmin 1.060 and later versions. See http://www.webmin.com

(FAQ 13) Why do you call it “Shorewall”?

Answer: Shorewall is a concatenation of “ *Shore*line” (the city where I live) and “Fire*wall* ”. The full name of the product is actually “Shoreline Firewall” but “Shorewall” is much more commonly used.

(FAQ 23) Why do you use such ugly fonts on your web site?

The Shorewall web site is almost font neutral (it doesn't explicitly specify fonts except on a few pages) so the fonts you see are largely the default fonts configured in your browser. If you don't like them then reconfigure your browser.

(FAQ 25) How to I tell which version of Shorewall I am running?

At the shell prompt, type:

```
/sbin/shorewall version
```

(FAQ 31) Does Shorewall provide protection against....

IP Spoofing: Sending packets over the WAN interface using an internal LAP IP address as the source address?

> Answer: Yes.

Tear Drop: Sending packets that contain overlapping fragments?

> Answer: This is the responsibility of the IP stack, not the Netfilter-based firewall since fragment reassembly occurs before the stateful packet filter ever touches each packet.

Smurf and Fraggle: Sending packets that use the WAN or LAN broadcast address as the source address?

> Answer: Shorewall can be configured to do that using the blacklisting facility. Shorewall versions 2.0.0 and later filter these packets under the *nosmurfs* interface option in /etc/shorewall/interfaces.

Land Attack: Sending packets that use the same address as the source and destination address?

> Answer: Yes, if the routefilter interface option is selected.

DOS: - SYN Dos - ICMP Dos - Per-host Dos protection

> Answer: Shorewall has facilities for limiting SYN and ICMP packets. Netfilter as included in standard Linux kernels doesn't support per-remote-host limiting except by explicit rule that specifies the host IP address; that form of limiting is supported by Shorewall.

(FAQ 36) Does Shorewall Work with the 2.6 Linux Kernel?

Shorewall works with the 2.6 Kernels with a couple of caveats:

- Netfilter/iptables doesn't fully support IPSEC in the 2.6 Kernels -- kernel and iptables patches are available and the details may be found at the Shorewall IPSEC-2.6 page.
- The 2.6 Kernels do not provide support for the logunclean and dropunclean options in `/etc/shorewall/interfaces`. Note that support for those options was also removed from Shorewall in version 2.0.0.

RFC 1918

(FAQ 14) I'm connected via a cable modem and it has an internal web server that allows me to configure/monitor it but as expected if I enable rfc1918 blocking for my eth0 interface (the internet one), it also blocks the cable modems web server.

Is there any way it can add a rule before the rfc1918 blocking that will let all traffic to and from the 192.168.100.1 address of the modem in/out but still block all other rfc1918 addresses?

Answer: If you are running a version of Shorewall earlier than 1.3.1, create /etc/shorewall/start and in it, place the following:

```
run_iptables -I rfc1918 -s 192.168.100.1 -j ACCEPT
```

If you are running version 1.3.1 or later, add the following to /etc/shorewall/rfc1918 (Note: If you are running Shorewall 2.0.0 or later, you may need to first copy `/usr/share/shorewall/rfc1918` to `/etc/shorewall/rfc1918`):

Be sure that you add the entry ABOVE the entry for 192.168.0.0/16.

```
#SUBNET         TARGET
192.168.100.1   RETURN
```

Note

If you add a second IP address to your external firewall interface to correspond to the modem address, you must also make an entry in /etc/shorewall/rfc1918 for that address. For example, if you configure the address 192.168.100.2 on your firewall, then you would add two entries to /etc/shorewall/rfc1918:

```
#SUBNET         TARGET
192.168.100.1   RETURN
192.168.100.2   RETURN
```

(FAQ 14a) Even though it assigns public IP addresses, my ISP's DHCP server has an RFC 1918 address. If I enable RFC 1918 filtering on my external interface, my DHCP client cannot renew its lease.

The solution is the same as the section called "(FAQ 14) I'm connected via a cable modem and it has an internal web server that allows me to configure/monitor it but as expected if I enable rfc1918 blocking for my eth0 interface (the internet one), it also blocks the cable modems web server." above. Simply substitute the IP address of your ISPs DHCP server.

(FAQ 14b) I connect to the internet with PPPoE. When I try to access the built-in web server in my DSL Modem, I get connection Refused.

I see the following in my log:

```
Mar  1 18:20:07 Mail kernel: Shorewall:OUTPUT:REJECT:IN= OUT=eth0 SRC=192.168.1.2
DST=192.168.1.1 LEN=60
TOS=0x00 PREC=0x00 TTL=64 ID=26774 DF PROTO=TCP SPT=32797 DPT=80 WINDOW=5840 RES=0x00
SYN URGP=0
```

Answer: The fact that the message is being logged from the OUTPUT chain means that the destination IP address is not in any defined zone (see FAQ 17). You need to:

1. Add a zone for the modem in `/etc/shorewall/zones`:

```
#ZONE    TYPE          OPTIONS
modem    ipv4
```

2. Define the zone to be associated with `eth0` (or whatever interface connects to your modem) in `/etc/shorewall/interfaces`:

```
#ZONE      INTERFACE   BROADCAST    OPTIONS
modem      eth0        detect
```

3. Allow web traffic to the modem in `/etc/shorewall/rules`:

```
#ACTION    SOURCE      DEST      PROTO     DEST PORT(S)
ACCEPT     fw          modem     tcp       80
ACCEPT     loc         modem     tcp       80
```

Note that many of these ADSL/Cable Modems have no default gateway or their default gateway is at a fixed IP address that is different from the IP address you have assigned to your external interface. In either case, you may have problems browsing the modem from your local network even if you have the correct routes established on your firewall. This is usually solved by masquerading traffic from your local network to the modem.

`/etc/shorewall/masq`:

```
#INTERFACE          SUBNET          ADDRESS
eth0                eth1                        # eth1 = interface to local network
```

For an example of this when the ADSL/Cable modem is bridged, see my configuration. In that case, I masquerade using the IP address of my local interface!

Alias IP Addresses/Virtual Interfaces

(FAQ 18) Is there any way to use aliased ip addresses with Shorewall, and maintain separate rulesets for different IPs?

Answer: Yes. See Shorewall and Aliased Interfaces.

Miscellaneous

(FAQ 19) I have added entries to /etc/shorewall/tcrules but they don't seem to do anything. Why?

You probably haven't set TC_ENABLED=Yes in /etc/shorewall/shorewall.conf so the contents of the tcrules file are simply being ignored.

(FAQ 20) I have just set up a server. Do I have to change Shorewall to allow access to my server from the internet?

Yes. Consult the QuickStart guide that you used during your initial setup for information about how to set up rules for your server.

(FAQ 24) How can I allow conections to let's say the ssh port only from specific IP

Addresses on the internet?

In the SOURCE column of the rule, follow “net” by a colon and a list of the host/subnet addresses as a comma-separated list.

```
net:<ip1>,<ip2>,...
```

Example 4. Example:

```
ACCEPT net:192.0.2.16/28,192.0.2.44 fw tcp 22
```

(FAQ 26) When I try to use any of the SYN options in nmap on or behind the firewall, I get “operation not permitted”. How can I use nmap with Shorewall?"

Temporarily remove and rejNotSyn, dropNotSyn and dropInvalid rules from `/etc/shorewall/rules` and restart Shorewall.

(FAQ 27) I'm compiling a new kernel for my firewall. What should I look out for?

First take a look at the Shorewall kernel configuration page. You probably also want to be sure that you have selected the “ **NAT of local connections (READ HELP)** ” on the Netfilter Configuration menu. Otherwise, DNAT rules with your firewall as the source zone won't work with your new kernel.

(FAQ 27a) I just built (or downloaded or otherwise acquired) and installed a new kernel and now Shorewall won't start. I know that my kernel options are correct.

The last few lines of a startup trace are these:

```
+ run_iptables2 -t nat -A eth0_masq -s 192.168.2.0/24 -d 0.0.0.0/0 -j
MASQUERADE
+ '[' 'x-t nat -A eth0_masq -s 192.168.2.0/24 -d 0.0.0.0/0 -j
MASQUERADE' = 'x-t nat -A eth0_masq -s 192.168.2.0/24 -d 0.0.0.
0/0 -j MASQUERADE' ']'
+ run_iptables -t nat -A eth0_masq -s 192.168.2.0/24 -d 0.0.0.0/0 -j
MASQUERADE
+ iptables -t nat -A eth0_masq -s 192.168.2.0/24 -d 0.0.0.0/0 -j
MASQUERADE
iptables: Invalid argument
+ '[' -z '' ']'
+ stop_firewall
+ set +x
```

Answer: Your new kernel contains headers that are incompatible with the ones used to compile your **iptables** utility. You need to rebuild **iptables** using your new kernel source.

(FAQ 28) How do I use Shorewall as a Bridging Firewall?

Shorewall Bridging Firewall support is available — check here for details.

(FAQ 39) How do I block connections to a particular domain name?

I tried this rule to block Google's Adsense that you'll find on everyone's site. Adsense is a Javascript that people add to their Web pages. So I entered the rule:

```
#ACTION    SOURCE        DEST                                    PROTO
REJECT     fw            net:pagead2.googlesyndication.com       all
```

However, this also sometimes restricts access to "google.com". Why is that? Using dig, I found these IPs for domain googlesyndication.com:

```
216.239.37.99
216.239.39.99
```

And this for google.com:

```
216.239.37.99
216.239.39.99
216.239.57.99
```

So my guess is that you are not actually blocking the domain, but rather the IP being called. So how in the world do you block an actual domain name?

Answer: Packet filters like Netfilter base their decisions on the contents of the various protocol headers at the front of each packet. Stateful packet filters (of which Netfilter is an example) use a combination of header contents and state created when the packet filter processed earlier packets. Netfilter (and Shorewall's use of netfilter) also consider the network interface(s) where each packet entered and/or where the packet will leave the firewall/router.

When you specify a domain name in a Shorewall rule, the iptables program resolves that name to one or more IP addresses and the actual netfilter rules that are created are expressed in terms of those IP addresses. So the rule that you entered was equivalent to:

```
#ACTION   SOURCE        DEST                  PROTO
REJECT    fw            net:216.239.37.99     all
REJECT    fw            net:216.239.39.99     all
```

Given that name-based multiple hosting is a common practice (another example: lists.shorewall.net and www1.shorewall.net are both hosted on the same system with a single IP address), it is not possible to filter connections to a particular name by examiniation of protocol headers alone. While some protocols such as FTP require the firewall to examine and possibly modify packet payload, parsing the payload of individual packets doesn't always work because the application-level data stream can be split across packets in arbitrary ways. This is one of the weaknesses of the 'string match' Netfilter extension available in Patch-O-Matic. The only sure way to filter on packet content is to proxy the connections in question -- in the case of HTTP, this means running something like Squid. Proxying allows the proxy process to assemble complete application-level messages which can then be accurately parsed and decisions can be made based on the result.

(FAQ 42) How can I tell which features my kernel and iptables support?

Answer: Use the **shorewall show capabilities** command at a root prompt.

```
gateway:~# shorewall show capabilities
Loading /usr/share/shorewall/functions...
Processing /etc/shorewall/params ...
Processing /etc/shorewall/shorewall.conf...
Loading Modules...
Shorewall has detected the following iptables/netfilter capabilities:
   NAT: Available
   Packet Mangling: Available
   Multi-port Match: Available
   Extended Multi-port Match: Available
   Connection Tracking Match: Available
   Packet Type Match: Available
   Policy Match: Available
   Physdev Match: Available
   IP range Match: Available
```

```
   Recent Match: Available
   Owner Match: Available
   Ipset Match: Available
   ROUTE Target: Available
   Extended MARK Target: Available
   CONNMARK Target: Available
   Connmark Match: Available
   Raw Table: Available
gateway:~#
```

Using Shorewall with Squid

Tom Eastep

Copyright © 2003-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled " GNU Free Documentation License ".

2005-09-12

Table of Contents

Squid as a Transparent Proxy
Configurations
- Squid (transparent) Running on the Firewall
- Squid (transparent) Running in the local network
- Squid (transparent) Running in the DMZ

Squid as a Manual Proxy

This page covers Shorewall configuration to use with Squid running as a Transparent Proxy or as a Manual Proxy.

Warning

This documentation assumes that you are running Shorewall 2.0.0 or later.

Squid as a Transparent Proxy

Important

This section gives instructions for transparent proxying of HTTP. HTTPS (normally TCP port 443) **cannot** be proxied transparently (stop and think about it for a minute; if HTTPS could be transparently proxied, then how secure would it be?).

Caution

Please observe the following general requirements:

- In all cases, Squid should be configured to run as a transrent proxy as described at http://www.tldp.org/HOWTO/mini/TransparentProxy.html.
- Depending on your distribution, other Squid configuration changes may be required. These changes typically consist of:
  1. Adding an ACL that represents the clients on your local network.

     Example:

     ```
     ACL my_networks src 192.168.1.0/24 192.168.2.0/24
     ```
  2. Allowing HTTP access to that ACL.

     Example:

     ```
     http_access allow my_networks
     ```

See your distribution's Squid documenation and http://www.squid-cache.org/ for details.

It is a good idea to get Squid working as a manual proxy first before you try transparent proxying.

- The following instructions mention the files /etc/shorewall/start and /etc/shorewall/init -- if you don't have those files, siimply create them.
- When the Squid server is in the DMZ zone or in the local zone, that zone must be defined ONLY by its interface -- no /etc/shorewall/hosts file entries. That is because the packets being routed to the Squid server still have their original destination IP addresses.
- You must have iptables installed on your Squid server.

Caution

In the instructions below, only TCP Port 80 is opened from the system running Squid to the Internet. If your users require browsing sites that use a port other than 80 (e.g., http://www.domain.tld:**8080**) then you must open those ports as well.

Configurations

Three different configurations are covered:

Squid (transparent) Running on the Firewall
Squid (transparent) Running in the local Network
Squid (transparent) Running in a DMZ

Squid (transparent) Running on the Firewall

You want to redirect all local www connection requests EXCEPT those to your own http server (206.124.146.177) to a Squid transparent proxy running on the firewall and listening on port 3128. Squid will of course require access to remote web servers.

In `/etc/shorewall/rules`:

```
#ACTION   SOURCE     DEST     PROTO    DEST PORT(S)     SOURCE     ORIGINAL
#                                                       PORT(S)    DEST
REDIRECT  loc        3128     tcp      www              -          !206.124.146.177
ACCEPT    $FW        net      tcp      www
```

There may be a requirement to exclude additional destination hosts or networks from being redirected. For example, you might also want requests destined for 130.252.100.0/24 to not be routed to Squid.

If needed, you may just add the additional hosts/networks to the ORIGINAL DEST column in your REDIRECT rule.

`/etc/shorewall/rules`:

```
#ACTION   SOURCE     DEST     PROTO    DEST PORT(S)     SOURCE     ORIGINAL
#                                                       PORT(S)    DEST
REDIRECT  loc        3128     tcp      www              -
!206.124.146.177,130.252.100.0/24
```

Squid (transparent) Running in the local network

You want to redirect all local www connection requests to a Squid transparent proxy running in your local zone at 192.168.1.3 and listening on port 3128. Your local interface is eth1. There may also be a web server running on 192.168.1.3. It is assumed that web access is already enabled from the local zone to the internet.

If you are running a Shorewall version earlier than 2.3.2 then:

1. On your firewall system, issue the following command

```
echo 202 www.out >> /etc/iproute2/rt_tables
```

2. Create `/etc/shorewall/addroutes` as follows:

```
#!/bin/sh

if [ -z "`ip rule list | grep www.out`" ] ; then
        ip rule add fwmark 0xCA table www.out # Note 0xCA = 202
        ip route add default via 192.168.1.3 dev eth1 table www.out
        ip route flush cache
        echo 0 > /proc/sys/net/ipv4/conf/eth1/send_redirects
fi
```

3. Make `/etc/shorewall/addroutes` executable via:

```
chmod +x /etc/shorewall/addroutes
```

4. In /etc/shorewall/init, put:

```
run_and_save_command "/etc/shorewall/addroutes"
```

If you are running Shorewall 2.3.2 or later:

1. Add this entry to your /etc/shorewall/providers file.

```
#NAME   NUMBER  MARK    DUPLICATE       INTERFACE       GATEWAY         OPTIONS
Squid   1       202     -               eth1            192.168.1.3     loose
```

Regardless of your Shorewall version, you need the following:

1. In `/etc/shorewall/start` add:

```
iptables -t mangle -A PREROUTING -i eth1 -s ! 192.168.1.3 -p tcp --dport 80 -j MARK --
set-mark 202
```

2. In `/etc/shorewall/interfaces` :

```
#ZONE   INTERFACE    BROADCAST    OPTIONS
loc     eth1         detect       routeback
```

3. In /etc/shorewall/rules:

```
#ACTION   SOURCE    DEST     PROTO   DEST PORT(S)
ACCEPT    loc       loc      tcp     www
```

   a. Alternatively, you can have the following policy in place of the above rule.

   `/etc/shorewall/policy`

```
#SOURCE   DESTINATION   POLICY
loc       loc           ACCEPT
```

4. On 192.168.1.3, arrange for the following command to be executed after networking has come up

```
iptables -t nat -A PREROUTING -i eth0 -d ! 192.168.1.3 -p tcp --dport 80 -j REDIRECT -
-to-ports 3128
```

   If you are running RedHat on the server, you can simply execute the following commands after you have typed the iptables command above:

```
iptables-save > /etc/sysconfig/iptables
 chkconfig --level 35 iptables on
```

Squid (transparent) Running in the DMZ

You have a single system in your DMZ with IP address 192.0.2.177. You want to run both a web server and Squid on that system.

In `/etc/shorewall/rules`:

```
#ACTION   SOURCE   DEST                  PROTO    DEST PORT(S)    SOURCE     ORIGINAL
#                                                                 PORT(S)    DEST
DNAT      loc      dmz:192.0.2.177:3128 tcp      80              -
!192.0.2.177
```

Squid as a Manual Proxy

Assume that Squid is running in zone SZ and listening on port SP; all web sites that are to be accessed through Squid are in the “net” zone. Then for each zone Z that needs access to the Squid server.

`/etc/shorewall/rules`:

```
#ACTION   SOURCE   DEST   PROTO   DEST PORT(S)
ACCEPT    Z        SZ     tcp     SP
ACCEPT    SZ       net    tcp     80,443
```

Example 1. Squid on the firewall listening on port 8080 with access from the “loc” zone:

`/etc/shorewall/rules:`

```
#ACTION   SOURCE   DEST   PROTO    DEST PORT(S)
ACCEPT    loc      $FW    tcp      8080
ACCEPT    $FW      net    tcp      80,443
```

Shorewall and UPnP

Tom Eastep

Copyright © 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-10-31

Table of Contents

UPnP
linux-igd Configuration
Shorewall Configuration

UPnP

In Shorewall 2.2.4, support was added for UPnP (Universal Plug and Play) using linux-igd (http://linux-igd.sourceforge.net). UPnP is required by a number of popular applications including MSN IM.

Warning

From a security architecture viewpoint, UPnP is a disaster. It assumes that:

a. All local systems and their users are completely trustworthy.
b. No local system is infected with any worm or trojan.

If either of these assumptions are not true then UPnP can be used to totally defeat your firewall and to allow incoming connections to arbitrary local systems on any port whatsoever. In short: USE UPnP **AT YOUR OWN RISK.**

Warning

The linux-igd project appears to be inactive and the web site does not display correctly on any open source browser that I've tried. Building and installing linux-igd is not for the faint of heart. You must download the source from CVS and be prepared to do quite a bit of fiddling with the include files from libupnp (which is required to build and/or run linux-igd).

Warning

Before building liunx-igd, you must apply all patches found at http://shorewall.net/pub/shorewall/contrib/linux-igd.

linux-igd Configuration

In /etc/upnpd.conf, you will want:

```
insert_forward_rules = yes
prerouting_chain_name = UPnP
forward_chain_name = forwardUPnP
```

Shorewall Configuration

In `/etc/shorewall/interfaces`, you need the 'upnp' option on your external interface.

Example:

```
#ZONE   INTERFACE       BROADCAST       OPTIONS
net     eth1            detect          dhcp,routefilter,norfc1918,tcpflags,upnp
```

If your fw->loc policy is not ACCEPT then you need this rule:

```
#ACTION             SOURCE  DEST
allowoutUPnP        $FW     loc
```

Note

To use 'allowoutUPnP', your iptables and kernel must support the 'owner match' feature (see the output of "shorewall show capabilities") and you may not be running kernel version 2.6.14 or later. If you are running 2.6.14 or later, then replace the above rule with:

```
#ACTION             SOURCE  DEST    PROTO     DEST PORT(S)     SOURCE     ORIGINAL
RATE     USER/
#                                                              PORT(S)    DESTINATION
LIMIT    GROUP
ACCEPT              $FW     loc     all       -                -          -            -
root
```

If your loc->fw policy is not ACCEPT then you need this rule:

```
#ACTION             SOURCE  DEST
allowinUPnP         loc     $FW
```

You MUST have this rule:

```
#ACTION             SOURCE  DEST
forwardUPnP         net     loc
```

You must also ensure that you have a route to 224.0.0.0/4 on your internal (local) interface as described in the linux-igd documentation.

ICMP Echo-request (Ping)

Tom Eastep

Copyright © 2001-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-09-12

Table of Contents

'Ping' Management
1. Revision History

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Note

Enabling "ping" will also enable ICMP-based *traceroute*. For UDP-based traceroute, see the port information page.

'Ping' Management

In Shorewall , ICMP echo-request's are treated just like any other connection request.

In order to accept ping requests from zone z1 to zone z2 where the policy for z1 to z2 is not ACCEPT, you need a rule in `/etc/shorewall/rules` of the form:

```
#ACTION       SOURCE     DEST      PROTO     DEST PORT(S)
Ping/ACCEPT     z1         z2
```

Example 1. Ping from local zone to firewall

To permit ping from the local zone to the firewall:

```
#ACTION       SOURCE    DEST      PROTO    DEST PORT(S)
Ping/ACCEPT    loc      $FW
```

If you would like to accept “ping” by default even when the relevant policy is DROP or REJECT, copy `/usr/share/shorewall/action.Drop` or `/usr/share shorewall/action.Reject` respectively to `/etc/shorewall` and simply add this line to the copy:

```
Ping/ACCEPT
```

With that rule in place, if you want to ignore “ping” from z1 to z2 then you need a rule of the form:

```
#ACTION       SOURCE    DEST      PROTO    DEST PORT(S)
Ping/DROP      z1        z2
```

Example 2. Silently drop pings from the Internet

To drop ping from the internet, you would need this rule in `/etc/shorewall/rules`:

```
#ACTION     SOURCE    DEST      PROTO    DEST PORT(S)
Ping/DROP   net       $FW
```

Note that the above rule may be used without changing the action files to prevent your log from being flooded by messages generated from remote pinging.

1. Revision History

Revision History		
Revision 1.3	2005-08-31	CR
Updated for Shorewall 3		
Revision 1.2	2004-01-03	TE
Add traceroute reference		
Revision 1.1	2003-08-23	TE
Initial version converted to Docbook XML		

Ports Required for Various Services/Applications

Tom Eastep

Copyright © 2001-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-11-17

Abstract

In addition to those applications described in the /etc/shorewall/rules documentation, here are some other services/applications that you may need to configure your firewall to accommodate.

Table of Contents

Important Notes
Auth (identd)
DNS
Emule
FTP
Gnutella
ICQ/AIM
IMAP
IPSEC
NFS
NTP (Network Time Protocol)
PCAnywhere™
Pop3
PPTP
rdate
rsync
SSH/SFTP
SMB/NMB (Samba/Windows Browsing/File Sharing)
SMTP
SNMP
Telnet
TFTP
Traceroute
Usenet (NNTP)
VNC
Vonage™
Web Access
X/XDMCP
Other Source of Port Information
1. Revision History

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release

Important Notes

Note

Shorewall distribution contains a library of user-defined macros that allow for easily allowing or blocking a particular application. **ls `/usr/share/shorewall/macro.*`** for the list of macros in your distribution. If you find what you need, you simply use the macro in a rule. For example, to allow DNS queries from the **dmz** zone to the **net** zone:

```
#ACTION          SOURCE         DESTINATION
DNS/ACCEPT       dmz            net
```

Note

In the rules that are shown in this document, the ACTION is shown as ACCEPT. You may need to use DNAT (see FAQ 30) or you may want DROP or REJECT if you are trying to block the application.

Example: You want to port forward FTP from the net to your server at 192.168.1.4 in your DMZ. The FTP section below gives you:

```
#ACTION          SOURCE     DESTINATION       PROTO       DEST PORT(S)
FTP/ACCEPT       <source>   <destination>
```

You would code your rule as follows:

```
#ACTION          SOURCE     DESTINATION       PROTO       DEST PORT(S)
FTP/DNAT         net        dmz:192.168.1.4
```

Auth (identd)

Caution

It is now the 21st Century **; don't use identd in production anymore.**

```
#ACTION           SOURCE     DESTINATION       PROTO       DEST PORT(S)
Auth/ACCEPT       <source>   <destination>
```

DNS

```
#ACTION           SOURCE     DESTINATION       PROTO       DEST PORT(S)
DNS/ACCEPT        <source>   <destination>
```

Note that if you are setting up a DNS server that supports recursive resolution, the server is the <*destination*> for resolution requests (from clients) and is also the <*source*> of recursive resolution requests (usually to other servers in the 'net' zone). So

for example, if you have a public DNS server in your DMZ that supports recursive resolution for local clients then you would need:

```
#ACTION      SOURCE     DESTINATION       PROTO       DEST PORT(S)
DNS/ACCEPT   all        dmz
DNS/ACCEPT   dmz        net
```

Note

Recursive Resolution means that if the server itself can't resolve the name presented to it, the server will attempt to resolve the name with the help of other servers.

Emule

In contrast to how the rest of this article is organized, for emule I will give you the rules necessary to run emule on a single machine in your loc network (since that's what 99.99% of you want to do). Assume that:

1. The internal machine running emule has IP address 192.168.1.4.
2. You use Masquerading or SNAT for the local network.
3. The zones are named as they are in the two- and three-interface QuickStart guides).
4. Your loc->net policy is ACCEPT

`/etc/shorewall/rules:`

```
#ACTION      SOURCE    DESTINATION           PROTO          DEST PORT(S)
DNAT         net       loc:192.168.1.4       tcp            4662
DNAT         net       loc:192.168.1.4       udp            4672
DNAT         net       loc:192.168.1.4       tcp            4711
```

FTP

```
#ACTION         SOURCE     DESTINATION       PROTO       DEST PORT(S)
FTP/ACCEPT      <source>   <destination>
```

Look here for much more information.

Gnutella

1. The internal machine running a Gnutella Client has IP address 192.168.1.4.
2. You use Masquerading or SNAT for the local network.
3. The zones are named as they are in the two- and three-interface QuickStart guides).
4. Your loc->net policy is ACCEPT

```
#ACTION               SOURCE    DESTINATION       PROTO       DEST PORT(S)
Gnutella/DNAT         net       loc:192.168.1.4
```

ICQ/AIM

```
#ACTION      SOURCE     DESTINATION       PROTO       DEST PORT(S)
ICQ/ACCEPT   <source>   net
```

IMAP

Caution

When accessing you mail from the internet,use **only IMAP over SSL**

```
#ACTION          SOURCE    DESTINATION      PROTO      DEST PORT(S)
IMAP/ACCEPT      <source>  <destination> #Secure & Unsecure IMAP
```

IPSEC

```
#ACTION    SOURCE          DESTINATION      PROTO      DEST PORT(S)
ACCEPT     <source>        <destination>    50
ACCEPT     <source>        <destination>    51
ACCEPT     <source>        <destination>    udp        500
ACCEPT     <destination>   <source>         50
ACCEPT     <destination>   <source>         51
ACCEPT     <destination>   <source>         udp        500
```

Lots more information here and here.

NFS

```
#ACTION    SOURCE                          DESTINATION      PROTO      DEST PORT(S)
ACCEPT     <z1>:<list of client IPs>       <z2>:a.b.c.d     tcp        111
ACCEPT     <z1>:<list of client IPs>       <z2>:a.b.c.d     udp
```

For more NFS information, see http://sourceforge.net/mailarchive/forum.php?thread_id=8972145&forum_id=2270.

NTP (Network Time Protocol)

```
#ACTION         SOURCE    DESTINATION      PROTO      DEST PORT(S)
NTP/ACCEPT      <source>  <destination>
```

PCAnywhere™

```
#ACTION         SOURCE    DESTINATION      PROTO      DEST PORT(S)
PCA/ACCEPT      <source>  <destination>
```

Pop3

Caution

If Possible , **Avoid this protocol** , use **IMAP** instead.

TCP Port 110 (Secure Pop3 is TCP Port 995)

```
#ACTION          SOURCE    DESTINATION      PROTO      DEST PORT(S)
POP3/ACCEPT      <source>  <destination> # Secure & Unsecure Pop3
```

PPTP

```
#ACTION    SOURCE    DESTINATION      PROTO      DEST PORT(S)
ACCEPT     <source>  <destination>    47
ACCEPT     <source>  <destination>    tcp        1723
```

Lots more information here and here.

rdate

```
#ACTION          SOURCE    DESTINATION      PROTO      DEST PORT(S)
Rdate/ACCEPT     <source>  <destination>
```

rsync

```
#ACTION          SOURCE    DESTINATION      PROTO      DEST PORT(S)
Rsync/ACCEPT     <source>  <destination>
```

SSH/SFTP

```
#ACTION    SOURCE    DESTINATION      PROTO      DEST PORT(S)
SSH/ACCEPT <source>  <destination>
```

SMB/NMB (Samba/Windows Browsing/File Sharing)

```
#ACTION        SOURCE         DESTINATION      PROTO      DEST PORT(S)
SMB/ACCEPT     <source>       <destination>
SMB/ACCEPT     <destination>  <source>
```

Also, see this page.

SMTP

```
#ACTION    SOURCE    DESTINATION      PROTO      DEST PORT(S)
SMTP/ACCEPT<source>  <destination>                     #Insecure SMTP
ACCEPT     <source>  <destination>    tcp        465    #SMTP over SSL (TLS)
```

SNMP

```
#ACTION          SOURCE    DESTINATION      PROTO      DEST PORT(S)
SNMP/ACCEPT      <source>  <destination>
```

Telnet

```
#ACTION           SOURCE    DESTINATION      PROTO      DEST PORT(S)
Telnet/ACCEPT     <source>  <destination>
```

TFTP

You must have TFTP connection tracking support in your kernel. If modularized, the modules are **ip_conntrack_tftp** (and **ip_nat_tftp** if any form of NAT is involved) These modules may be loaded using entries in `/etc/shorewall/modules`. The **ip_conntrack_tftp** module must be loaded first. Note that the `/etc/shorewall/modules` file released with recent Shorewall versions contains entries for these modules.

```
#ACTION    SOURCE    DESTINATION      PROTO      DEST PORT(S)
ACCEPT     <source>  <destination>    udp        69
```

Traceroute

```
#ACTION          SOURCE    DESTINATION      PROTO      DEST PORT(S)
Trcrt/ACCEPT     <source>  <destination>  #Good for 10 hops
```

UDP traceroute uses ports 33434 through 33434+<max number of hops>-1. Note that for the firewall to respond with a TTL expired ICMP reply, you will need to allow ICMP 11 outbound from the firewall. The standard Shorewall sample configurations all set this up for you automatically since those sample configurations enable all ICMP packet types originating on the firewall itself.

```
#ACTION    SOURCE    DESTINATION      PROTO      DEST PORT(S)
ACCEPT     fw        net              icmp
ACCEPT     fw        loc              icmp
ACCEPT     fw        ...
```

Usenet (NNTP)

```
#ACTION         SOURCE    DESTINATION      PROTO      DEST PORT(S)
NNTP/ACCEPT     <source>  <destination>
```

TCP Port 119

VNC

Vncviewer to Vncserver -- TCP port 5900 + <display number>.

Vncviewer to Vncserver -- TCP port 5900 + <display number>.

```
#ACTION    SOURCE    DESTINATION      PROTO      DEST PORT(S)
ACCEPT     <source>  <destination>    tcp        5901               #Display Number 1
ACCEPT     <source>  <destination>    tcp        5902               #Display Number 2
...
```

Vncserver to Vncviewer in listen mode -- TCP port 5500.

```
#ACTION         SOURCE    DESTINATION      PROTO      DEST PORT(S)
VNCL/ACCEPT     <source>  <destination>
```

Vonage™

The standard Shorewall loc->net ACCEPT policy is all that is required for Vonage™ IP phone service to work, provided that you have loaded the tftp helper modules (add the following entries to /etc/shorewall/modules if they are not there already):

```
    loadmodule ip_conntrack_tftp
    loadmodule ip_nat_tftp
```

Web Access

```
#ACTION        SOURCE    DESTINATION      PROTO      DEST PORT(S)
Web/ACCEPT     <source>  <destination> #Insecure HTTP& Secure HTTP
```

X/XDMCP

Assume that the Choser and/or X Server are running at <*chooser*> and the Display Manager/X applications are running at <*apps*>.

```
#ACTION    SOURCE    DESTINATION      PROTO      DEST PORT(S)
ACCEPT     <chooser> <apps>           udp        177         #XDMCP
ACCEPT     <apps>    <chooser>        tcp        6000:6009   #X Displays 0-9
```

Other Source of Port Information

Didn't find what you are looking for -- have you looked in your own /etc/services file?

Still looking? Try http://www.networkice.com/advice/Exploits/Ports

1. Revision History

Revision History		
Revision 1.17	2005-09-20	TE
More 3.0 Updates		
Revision 1.16	2005-09-02	CR
Updated for Shorewall v3.0		
Revision 1.15	2005-05-02	TE
Added Emule		
Revision 1.14	2004-10-01	TE
Add rsync.		
Revision 1.13	2004-09-21	TE
Add note about ICMP type 11 to Traceroute.		
Revision 1.12	2004-09-09	TE
Add note about Vonage™.		
Revision 1.11	2004-05-28	TE
Corrected directory for actions.std and enhanced the DNS section.		
Revision 1.10	2004-05-09	TE

Added TFTP.		
Revision 1.9	2004-04-24	TE
Revised ICQ/AIM.		
Revision 1.8	2004-04-23	TE
Added SNMP.		
Revision 1.7	2004-02-18	TE
Make NFS work for everyone.		
Revision 1.6	2004-02-14	TE
Add PCAnywhere.		
Revision 1.5	2004-02-05	TE
Added information about VNC viewers in listen mode.		
Revision 1.4	2004-01-26	TE
Correct ICQ.		
Revision 1.3	2004-01-04	TE
Alphabetize		
Revision 1.2	2004-01-03	TE
Add rules file entries.		
Revision 1.1	2002-07-30	TE
Initial version converted to Docbook XML		

Shorewall and FTP

Tom Eastep

Copyright © 2003, 2004, 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-08-31

Table of Contents

FTP Protocol
Linux FTP connection-tracking
FTP on Non-standard Ports
Rules

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

FTP Protocol

FTP transfers involve two TCP connections. The first **control** connection goes from the FTP client to port 21 on the FTP server. This connection is used for logon and to send commands and responses between the endpoints. Data transfers (including the output of "ls" and "dir" commands) requires a second data connection. The **data** connection is dependent on the **mode** that the client is operating in:

Passive Mode

(often the default for web browsers) -- The client issues a PASV command. Upon receipt of this command, the server listens on a dynamically-allocated port then sends a PASV reply to the client. The PASV reply gives the IP address and port number that the server is listening on. The client then opens a second connection to that IP address and port number.

Active Mode

(often the default for line-mode clients) -- The client listens on a dynamically-allocated port then sends a PORT command to the server. The PORT command gives the IP address and port number that the client is listening on. The server then opens a connection to that IP address and port number; the **source port** for this connection is 20 (ftp-data in /etc/services).

You can see these commands in action using your linux ftp command-line client in debugging mode. Note that my ftp client defaults to passive mode and that I can toggle between passive and active mode by issuing a "passive" command:

```
[teastep@wookie Shorewall]$ ftp ftp1.shorewall.net
Connected to lists.shorewall.net.
220-=(<*>)=-.:. (( Welcome to PureFTPd 1.0.12 )) .:.-=(<*>)=-
220-You are user number 1 of 50 allowed.
```

```
220-Local time is now 10:21 and the load is 0.14. Server port: 21.
220 You will be disconnected after 15 minutes of inactivity.
500 Security extensions not implemented
500 Security extensions not implemented
KERBEROS_V4 rejected as an authentication type
Name (ftp1.shorewall.net:teastep): ftp
331-Welcome to ftp.shorewall.net
331-
331 Any password will work
Password:
230 Any password will work
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> debug
Debugging on (debug=1).
ftp> ls
---> PASV
227 Entering Passive Mode (192,168,1,193,195,210)
---> LIST
150 Accepted data connection
drwxr-xr-x    5 0        0            4096 Nov  9  2002 archives
drwxr-xr-x    2 0        0            4096 Feb 12  2002 etc
drwxr-sr-x    6 0        50           4096 Feb 19 15:24 pub
226-Options: -l
226 3 matches total
ftp> passive
Passive mode off.
ftp> ls
---> PORT 192,168,1,3,142,58
200 PORT command successful
---> LIST
150 Connecting to port 36410
drwxr-xr-x    5 0        0            4096 Nov  9  2002 archives
drwxr-xr-x    2 0        0            4096 Feb 12  2002 etc
drwxr-sr-x    6 0        50           4096 Feb 19 15:24 pub
226-Options: -l
226 3 matches total
ftp>
```

Things to notice:

1. The commands that I issued are **strongly emphasized**.
2. Commands sent by the client to the server are preceded by --->
3. Command responses from the server over the control connection are numbered.
4. FTP uses a comma as a separator between the bytes of the IP address; and
5. When sending a port number, FTP sends the MSB then the LSB and separates the two bytes by a comma. As shown in the PORT command, port 142,58 translates to 142*256+58 = 36410.

Linux FTP connection-tracking

Given the normal loc->net policy of ACCEPT, passive mode access from local clients to remote servers will always work but active mode requires the firewall to dynamically open a “hole” for the server's connection back to the client. Similarly, if you are running an FTP server in your local zone then active mode should always work but passive mode requires the firewall to dynamically open a “hole” for the client's second connection to the server. This is the role of FTP connection-tracking support in the Linux kernel.

Where any form of NAT (SNAT, DNAT, Masquerading) on your firewall is involved, the PORT commands and PASV

responses may also need to be modified by the firewall. This is the job of the FTP nat support kernel function.

Including FTP connection-tracking and NAT support normally means that the modules “ip_conntrack_ftp” and “ip_nat_ftp” need to be loaded. Shorewall automatically loads these “helper” modules from /lib/modules/<*kernel-version*>/kernel/net/ipv4/netfilter/ and you can determine if they are loaded using the “lsmod” command. The <*kernel-version*> may be obtained by typing

```
uname -r
```

Example 1.

```
[root@lists etc]# lsmod
Module                  Size  Used by    Not tainted
autofs                 12148   0  (autoclean) (unused)
ipt_TOS                 1560  12  (autoclean)
ipt_LOG                 4120   5  (autoclean)
ipt_REDIRECT            1304   1  (autoclean)
ipt_REJECT              3736   4  (autoclean)
ipt_state               1048  13  (autoclean)
ip_nat_irc              3152   0  (unused)
ip_nat_ftp              3888   0  (unused)
ip_conntrack_irc        3984   1
ip_conntrack_ftp        5008   1
ipt_multiport           1144   2  (autoclean)
ipt_conntrack           1592   0  (autoclean)
iptable_filter          2316   1  (autoclean)
iptable_mangle          2680   1  (autoclean)
iptable_nat            20568   3  (autoclean) [ipt_REDIRECT ip_nat_irc ip_nat_ftp]
ip_conntrack           26088   5  (autoclean) [ipt_REDIRECT ipt_state ip_nat_irc
                                               ip_nat_ftp ip_conntrack_irc
ip_conntrack_ftp
                                               ipt_conntrack iptable_nat]
ip_tables              14488  12  [ipt_TOS ipt_LOG ipt_REDIRECT ipt_REJECT ipt_state
                                   ipt_multiport ipt_conntrack iptable_filter
                                   iptable_mangle iptable_nat]
tulip                  42464   0  (unused)
e100                   50596   1
keybdev                 2752   0  (unused)
mousedev                5236   0  (unused)
hid                    20868   0  (unused)
input                   5632   0  [keybdev mousedev hid]
usb-uhci               24684   0  (unused)
usbcore                73280   1  [hid usb-uhci]
ext3                   64704   2
jbd                    47860   2  [ext3]
[root@lists etc]#
```

If you want Shorewall to load these modules from an alternate directory, you need to set the MODULESDIR variable in /etc/shorewall/shorewall.conf to point to that directory.

FTP on Non-standard Ports

The above discussion about commands and responses makes it clear that the FTP connection-tracking and NAT helpers must scan the traffic on the control connection looking for PASV and PORT commands as well as PASV responses. If you run an FTP server on a nonstandard port or you need to access such a server, you must therefore let the helpers know by specifying the

port in /etc/shorewall/modules entries for the helpers.

Caution

You must have modularized FTP connection tracking support in order to use FTP on a non-standard port.

Example 2. if you run an FTP server that listens on port 49 or you need to access a server on the internet that listens on that port then you would have:

```
loadmodule ip_conntrack_ftp ports=21,49
loadmodule ip_nat_ftp ports=21,49          # NOTE: This is not necessary with kernel
2.6.11 and later!
```

Note

you MUST include port 21 in the ports list or you may have problems accessing regular FTP servers.

If there is a possibility that these modules might be loaded before Shorewall starts, then you should include the port list in /etc/modules.conf:

```
options ip_conntrack_ftp ports=21,49
options ip_nat_ftp ports=21,49          # NOTE: This is not necessary with kernel
2.6.11 and later!
```

Important

Once you have made these changes to /etc/shorewall/modules and/or /etc/modules.conf, you must either:

1. Unload the modules and restart shorewall:

   ```
   rmmod ip_nat_ftp; rmmod ip_conntrack_ftp; shorewall restart
   ```
2. Reboot

Rules

Warning

If you run an FTP server behind your firewall and your server offers a method of specifying the external IP address of your firewall, DON'T USE THAT FEATURE OF YOUR SERVER. Using that option will defeat the purpose of the ftp helper modules and can result in a server that doesn't work.

If the policy from the source zone to the destination zone is ACCEPT and you don't need DNAT (see FAQ 30) then **you need no rule**.

Otherwise, for FTP you need exactly **one** rule:

```
#ACTION      SOURCE     DESTINATION    PROTO     PORT(S)    SOURCE      ORIGINAL
#                                                           PORT(S)     DESTINATION
ACCEPT or    <source>   <destination>  tcp       21         -           <external IP
addr> if
DNAT                                                                     ACTION = DNAT
```

You need an entry in the ORIGINAL DESTINATION column only if the ACTION is DNAT, you have multiple external IP addresses and you want a specific IP address to be forwarded to your server.

Note that you do **NOT** need a rule with 20 (ftp-data) in the PORT(S) column. If you post your rules on the mailing list and they show 20 in the PORT(S) column, I will know that you haven't read this article and I will either ignore your post or tell you to RTFM.

Shorewall includes an FTP macro that simplifies creation of FTP rules. The macro source is in `/usr/share/shorewall/macro.FTP`. Using the macro is the preferred way to generate the rules described above. Here are a couple of examples.

Example 3. Server running behind a Masquerading Gateway

Suppose that you run an FTP server on 192.168.1.5 in your local zone using the standard port (21). You need this rule:

```
#ACTION      SOURCE     DESTINATION     PROTO     PORT(S)    SOURCE      ORIGINAL
#                                                            PORT(S)     DESTINATION
FTP/DNAT      net       loc:192.168.1.5
```

Example 4. Allow your DMZ FTP access to the Internet

```
#ACTION      SOURCE     DESTINATION     PROTO     PORT(S)    SOURCE      ORIGINAL
#                                                            PORT(S)     DESTINATION
FTP/ACCEPT   dmz        net
```

Note that the FTP connection tracking in the kernel cannot handle cases where a PORT command (or PASV reply) is broken across two packets. When such cases occur, you will see a console message similar to this one:

```
Apr 28 23:55:09 gateway kernel: conntrack_ftp: partial PORT 715014972+1
```

I see this problem occasionally with the FTP server in my DMZ. My solution is to add the following rule:

```
#ACTION      SOURCE     DESTINATION     PROTO     PORT(S)    SOURCE      ORIGINAL
#                                                            PORT(S)     DESTINATION
ACCEPT:info  dmz        net             tcp       -          20
```

The above rule accepts and logs all active mode connections from my DMZ to the net.

IPSEC Tunnels

Tom Eastep

Copyright © 2001-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-09-30

Table of Contents

Preliminary Reading
Configuring FreeS/Wan and Derivatives Such as OpenS/Wan
IPSec Gateway on the Firewall System
VPN Hub using Kernel 2.4
Mobile System (Road Warrior) Using Kernel 2.4
Dynamic RoadWarrior Zones
Limitations of Dynamic Zones

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Important

The information in this article is only applicable if you plan to have IPSEC end-points on the same system where Shorewall is used.

Warning

This documentation is incomplete regarding using IPSEC and the 2.6 Kernel. Netfilter currently lacks full support for the 2.6 kernel's implementation of IPSEC. Until that implementation is complete, only a simple network-network tunnel is described for 2.6.

UPDATE: Some distributions such as SuSE™ are now shipping Kernels and iptables with the IPSEC-Netfilter patches and policy match support. Check this article for information concerning this support and Shorewall.

Preliminary Reading

I recommend reading the VPN Basics article if you plan to implement any type of VPN.

Configuring FreeS/Wan and Derivatives Such as OpenS/Wan

There is an excellent guide to configuring IPSEC tunnels at http://jixen.tripod.com/. I highly recommend that you consult that site for information about configuring FreeS/Wan.

Important

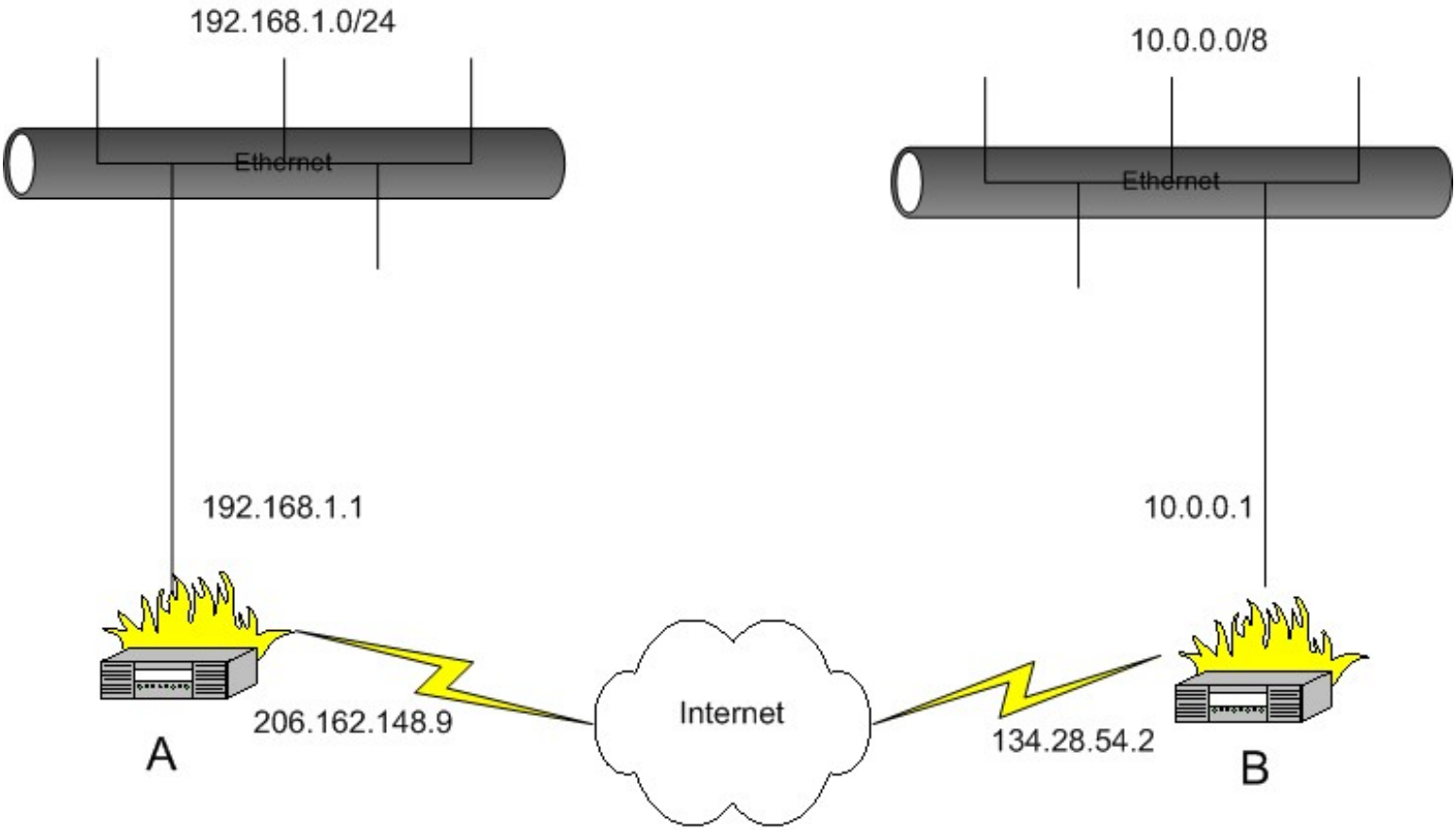
The documentation below assumes that you have disabled opportunistic encryption feature in FreeS/Wan 2.0 using the following additional entries in ipsec.conf:

```
conn block
        auto=ignore

conn private
        auto=ignore

conn private-or-clear
        auto=ignore

conn clear-or-private
        auto=ignore

conn clear
        auto=ignore

conn packetdefault
        auto=ignore
```

For further information see http://www.freeswan.org/freeswan_trees/freeswan-2.03/doc/policygroups.html.

IPSec Gateway on the Firewall System

Suppose that we have the following sutuation:



We want systems in the 192.168.1.0/24 sub-network to be able to communicate with systems in the 10.0.0.0/8 network. We assume that on both systems A and B, eth0 is the internet interface.

To make this work, we need to do two things:

a. Open the firewall so that the IPSEC tunnel can be established (allow the ESP and AH protocols and UDP Port 500).
b. Allow traffic through the tunnel.

Opening the firewall for the IPSEC tunnel is accomplished by adding an entry to the /etc/shorewall/tunnels file.

In /etc/shorewall/tunnels on system A, we need the following

```
#TYPE      ZONE        GATEWAY          GATEWAY ZONE
ipsec      net         134.28.54.2
```

In /etc/shorewall/tunnels on system B, we would have:

```
#TYPE      ZONE        GATEWAY          GATEWAY ZONE
ipsec      net         206.161.148.9
```

Note

If either of the endpoints is behind a NAT gateway then the tunnels file entry on the **other** endpoint should specify a tunnel type of ipsecnat rather than ipsec and the GATEWAY address should specify the external address of the NAT gateway.

You need to define a zone for the remote subnet or include it in your local zone. In this example, we'll assume that you have created a zone called “vpn” to represent the remote subnet. Note that you should define the vpn zone before the net zone.

/etc/shorewall/zones (both systems):

```
#ZONE          TYPE         OPTIONS
vpn            ipv4
net            ipv4
```

If you are running kernel 2.4:

At both systems, ipsec0 would be included in /etc/shorewall/interfaces as a “vpn” interface:

```
#ZONE          INTERFACE          BROADCAST        OPTIONS
vpn            ipsec0
```

If you are running kernel 2.6:

It is essential that the *vpn* zone be declared before the *net* zone in `/etc/shorewall/zones`.

Remember the assumption that both systems A and B have eth0 as their internet interface.

You must define the vpn zone using the /etc/shorewall/hosts file.

/etc/shorewall/hosts - System A

```
#ZONE         HOSTS                    OPTIONS
vpn           eth0:10.0.0.0/8
```

/etc/shorewall/hots - System B

```
#ZONE         HOSTS                    OPTIONS
vpn           eth0:192.168.1.0/24
```

In addition, **if you are using Masquerading or SNAT** on your firewalls, you need to elmiinate the remote network from Masquerade/SNAT. These entries **replace** your current masquerade/SNAT entries for the local networks.

/etc/shorewall/masq - System A

```
#INTERFACE              SUBNET                  ADDRESS
eth0:!10.0.0.0/8        192.168.1.0/24
```

/etc/shorewall/masq - System B

```
#INTERFACE              SUBNET                  ADDRESS
eth0:!192.168.1.0/24    10.0.0.0/8
```

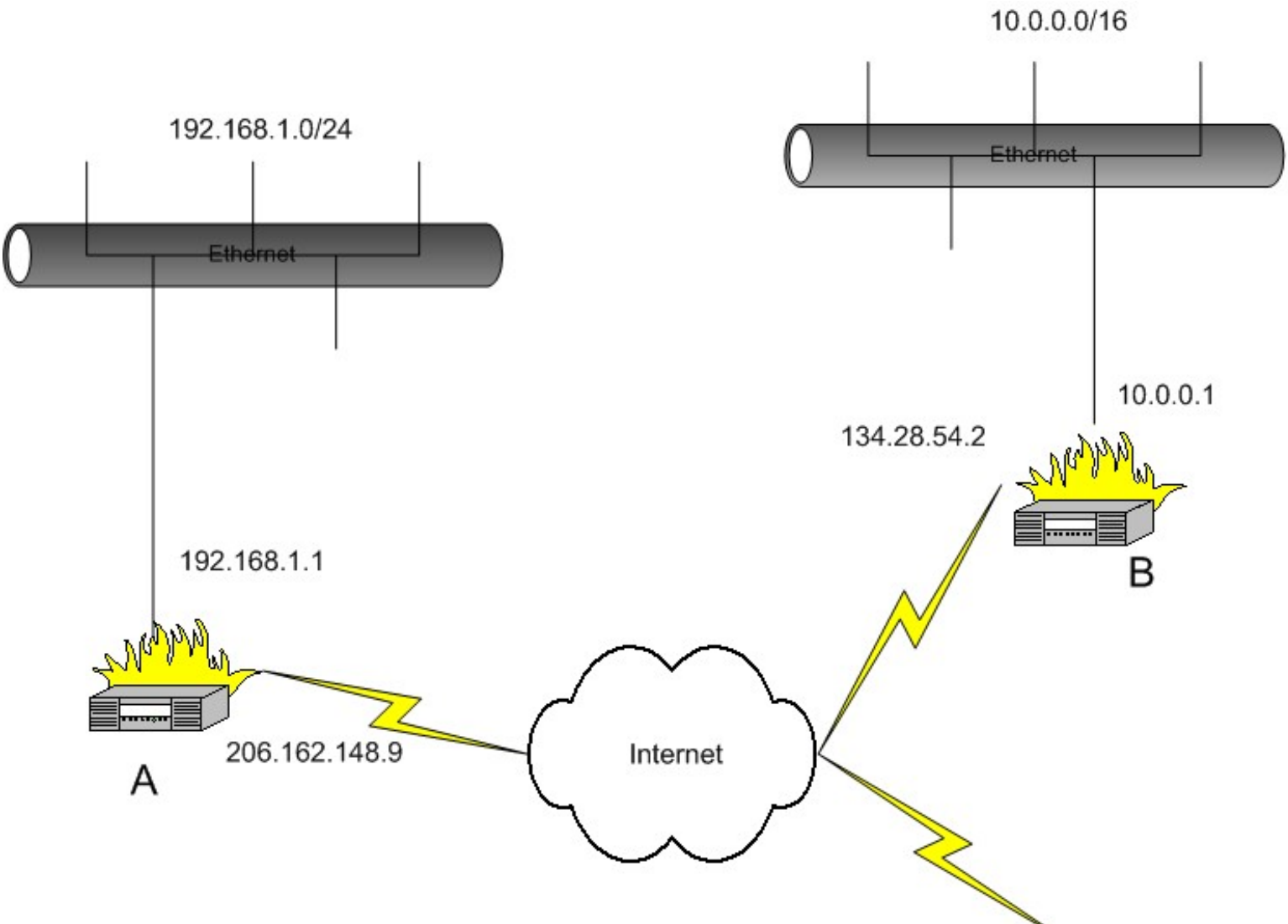
You will need to allow traffic between the “vpn” zone and the “loc” zone -- if you simply want to admit all traffic in both directions, you can use the policy file:

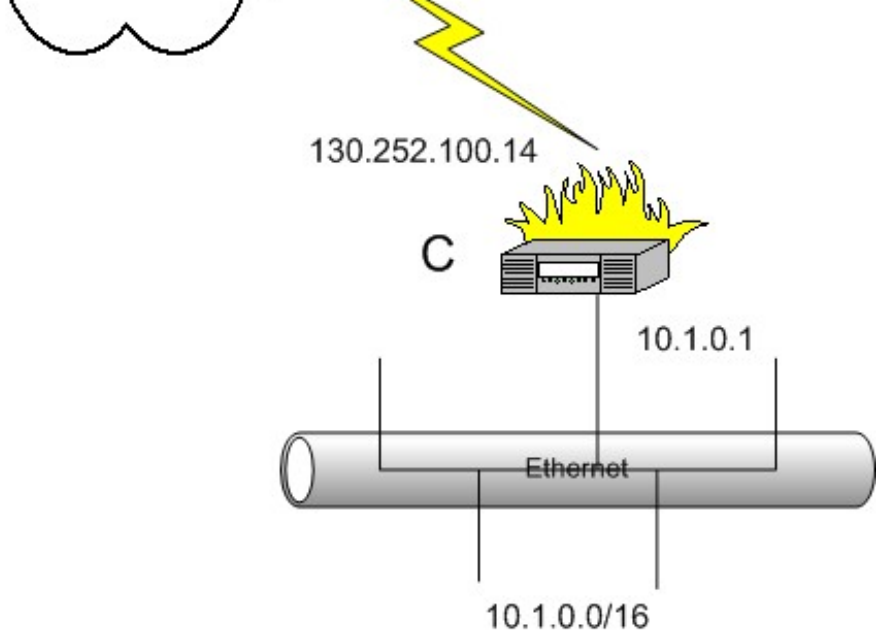
```
#SOURCE         DEST          POLICY        LOG LEVEL
loc             vpn           ACCEPT
vpn             loc           ACCEPT
```

Once you have these entries in place, restart Shorewall (type shorewall restart); you are now ready to configure the tunnel in FreeS/WAN.

VPN Hub using Kernel 2.4

Shorewall can be used in a VPN Hub environment where multiple remote networks are connected to a gateway running Shorewall. This environment is shown in this diatram.





We want systems in the 192.168.1.0/24 sub-network to be able to communicate with systems in the 10.0.0.0/16 and 10.1.0.0/16 networks and we want the 10.0.0.0/16 and 10.1.0.0/16 networks to be able to communicate.

To make this work, we need to do several things:

a. Open the firewall so that two IPSEC tunnels can be established (allow the ESP and AH protocols and UDP Port 500).
b. Allow traffic through the tunnels two/from the local zone (192.168.1.0/24).
c. Deny traffic through the tunnels between the two remote networks.

Opening the firewall for the IPSEC tunnels is accomplished by adding two entries to the /etc/shorewall/tunnels file.

In /etc/shorewall/tunnels on system A, we need the following

```
#TYPE          ZONE          GATEWAY          GATEWAY ZONE
ipsec          net           134.28.54.2
ipsec          net           130.252.100.14
```

In /etc/shorewall/tunnels on systems B and C, we would have:

```
#TYPE          ZONE          GATEWAY          GATEWAY ZONE
ipsec          net           206.161.148.9
```

Note

If either of the endpoints is behind a NAT gateway then the tunnels file entry on the **other** endpoint should specify a tunnel type of *ipsecnat* rather than ipsec and the GATEWAY address should specify the external address of the NAT gateway.

On each system, we will create a zone to represent the remote networks. On System A:

```
#ZONE        TYPE          OPTIONS
vpn1         ipv4
vp2          ipv4
```

On systems B and C:

```
#ZONE        TYPE          OPTIONS
vpn          ipv4
```

At system A, ipsec0 represents two zones so we have the following in /etc/shorewall/interfaces:

```
#ZONE        INTERFACE    BROADCAST       OPTIONS
-            ipsec0
```

The /etc/shorewall/hosts file on system A defines the two VPN zones:

```
#ZONE        HOSTS                        OPTIONS
vpn1         ipsec0:10.0.0.0/16
vpn2         ipsec0:10.1.0.0/16
```

At systems B and C, ipsec0 represents a single zone so we have the following in /etc/shorewall/interfaces:

```
#ZONE        INTERFACE       BROADCAST    OPTIONS
vpn          ipsec0
```

On systems A, you will need to allow traffic between the “vpn1” zone and the “loc” zone as well as between “vpn2” and the “loc” zone -- if you simply want to admit all traffic in both directions, you can use the following policy file entries on all three gateways:

```
#SOURCE      DEST        POLICY           LOG LEVEL
loc          vpn1        ACCEPT
vpn1         loc         ACCEPT
loc          vpn2        ACCEPT
vpn2         loc         ACCEPT
```

On systems B and C, you will need to allow traffic between the “vpn” zone and the “loc” zone -- if you simply want to admit all traffic in both directions, you can use the following policy file entries on all three gateways:

/etc/shorewall/policy -- Systems B & C

```
#SOURCE      DEST        POLICY           LOG LEVEL
loc          vpn         ACCEPT
vpn          loc         ACCEPT
```

Once you have the Shorewall entries added, restart Shorewall on each gateway (type shorewall restart); you are now ready to configure the tunnels in FreeS/WAN.

Note

to allow traffic between the networks attached to systems B and C, it is necessary to simply add two additional entries to the /etc/shorewall/policy file on system A.

```
#SOURCE      DEST        POLICY           LOG LEVEL
vpn1         vpn2        ACCEPT
vpn2         vpn1        ACCEPT
```

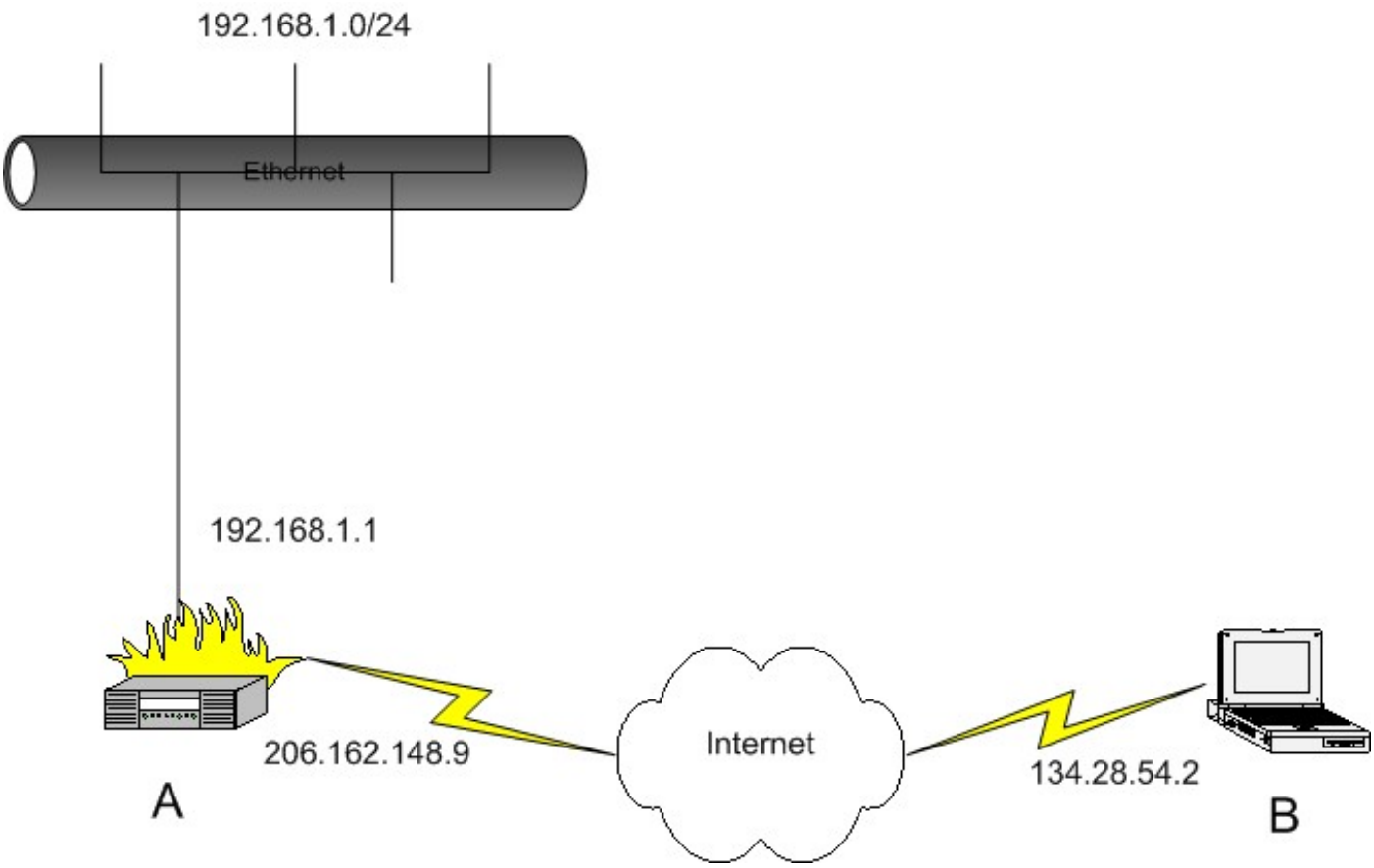
Note

If you find traffic being rejected/dropped in the OUTPUT chain, place the names of the remote VPN zones as a comma-separated list in the GATEWAY ZONE column of the /etc/shorewall/tunnels file entry.

Mobile System (Road Warrior) Using Kernel 2.4

Suppose that you have a laptop system (B) that you take with you when you travel and you want to be able to establish a secure

connection back to your local network.



Example 1. Road Warrior VPN

You need to define a zone for the laptop or include it in your local zone. In this example, we'll assume that you have created a zone called “vpn” to represent the remote host.

/etc/shorewall/zones - System A

```
#ZONE       TYPE        OPTIONS
vpn         ipv4
```

In this instance, the mobile system (B) has IP address 134.28.54.2 but that cannot be determined in advance. In the /etc/shorewall/tunnels file on system A, the following entry should be made:

```
#TYPE        ZONE        GATEWAY         GATEWAY ZONE
ipsec        net         0.0.0.0/0
```

Note

the GATEWAY ZONE column contains the name of the zone corresponding to peer subnetworks. This indicates that the gateway system itself comprises the peer subnetwork; in other words, the remote gateway is a standalone system.

You will need to configure /etc/shorewall/interfaces and establish your “through the tunnel” policy as shown under the first example above.

Dynamic RoadWarrior Zones

Beginning with Shorewall release 1.3.10, you can define multiple VPN zones and add and delete remote endpoints dynamically using /sbin/shorewall. With Shorewall 2.0.2 Beta 1 and later versions, this capability must be enabled by setting DYNAMIC_ZONES=Yes in shorewall.conf.

In /etc/shorewall/zones:

```
#ZONE       TYPE       OPTIONS
vpn1        ipv4
vpn2        ipv4
vpn3        ipv4
```

In /etc/shorewall/tunnels:

```
#TYPE       ZONE       GATEWAY         GATEWAY ZONE
ipsec       net        0.0.0.0/0       vpn1,vpn2,vpn3
```

When Shorewall is started, the zones vpn[1-3] will all be empty and Shorewall will issue warnings to that effect. These warnings may be safely ignored. FreeS/Wan may now be configured to have three different Road Warrior connections with the choice of connection being based on X-509 certificates or some other means. Each of these connectioins will utilize a different updown script that adds the remote station to the appropriate zone when the connection comes up and that deletes the remote station when the connection comes down. For example, when 134.28.54.2 connects for the vpn2 zone the “up” part of the script will issue the command:

```
/sbin/shorewall add ipsec0:134.28.54.2 vpn2
```

and the “down” part will:

```
/sbin/shorewall delete ipsec0:134.28.54.2 vpn2
```

Limitations of Dynamic Zones

If you include a dynamic zone in the exclude list of a DNAT rule, the dynamically-added hosts are not excluded from the rule.

```
#ACTION      SOURCE        DEST            PROTO     DEST PORT(S)
DNAT         z!dyn         loc:192.168.1.3 tcp       80
```

Example 2. dyn=dynamic zone

Dynamic changes to the zone **dyn** will have no effect on the above rule.

IPSEC using Linux Kernel 2.6

Tom Eastep

Copyright © 2004, 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-09-30

Table of Contents

Shorewall 3.0 and Kernel 2.6 IPSEC
IPSec Gateway on the Firewall System
Mobile System (Road Warrior)
Transport Mode
IPSEC and Windows™ XP
Source of Additional Samples

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Important

The information in this article is only applicable if you plan to have IPSEC end-points on the same system where Shorewall is used.

Warning

To use the features described in this article, your kernel and iptables must include the Netfilter+ipsec patches and policy match support. The Netfilter patches are available from Netfilter Patch-O-Matic-NG and are also included in some commercial distributions (most notably SuSE™ 9.1 through 9.3).

Important

You must have **BOTH** the Netfilter+ipsec patches and the policy match patch. **One without the other will not work**.

Here's a combination of components that I know works:

1. Kernel 2.6.11 from kernel.org. Patched with:
   - The five patches in http://shorewall.net/pub/shorewall/contrib/IPSEC/2.6.11
   - The "policy match" extension from the Patch-o-matic-ng CVS snapshot from 2005-May-04 (be sure to NOT try to apply the ipsec-NN patches from patch-o-matic-ng).
2. iptables 1.3.1 patched with the "policy match" extension from the Patch-o-matic-ng CVS snapshot from 2005-May-04.
3. ipsec-tools 0.5.2 compiled from source. I've also had success with:
   - ipsec-tools 0.5.2 and racoon 0.5.2 from Debian Sarge/testing
   - The ipsec-tools 0.5 rpm from SuSE 9.3.

Warning

As of this writing, the Netfilter+ipsec and policy match support are broken when used with a bridge device. The problem has been reported to the responsible Netfilter developer who has confirmed the problem.

Shorewall 3.0 and Kernel 2.6 IPSEC

This is **not** a HOWTO for Kernel 2.6 IPSEC -- for that, please see http://www.ipsec-howto.org/.

The 2.6 Linux Kernel introduces new facilities for defining encrypted communication between hosts in a network. The network administrator defines a set of *Security Policies* which are stored in the kernel as a *Security Policy Database* (SPD). Security policies determine which traffic is subject to encryption. *Security Associations* are created between pairs of hosts in the network (one SA for traffic in each direction); these SAs define how traffic is to be encrypted. Outgoing traffic that is to be encrypted according to the contents of the SPD requires an appropriate SA to exist. SAs may be created manually using **setkey**(8) but most often, they are created by a cooperative process involving the ISAKMP protocol and daemons such as **racoon** or **isakmpd**. Incoming traffic is verified against the SPD to ensure that no unencrypted traffic is accepted in violation of the administrator's policies.

There are three ways in which IPSEC traffic can interact with Shorewall policies and rules:

1. Traffic that is encrypted on the firewall system. The traffic passes through Netfilter twice -- first as unencrypted then encrypted.
2. Traffic that is decrypted on the firewall system. The traffic passes through Netfilter twice -- first as encrypted then as unencrypted.
3. Encrypted traffic that is passed through the firewall system. The traffic passes through Netfilter once.

In cases 1 and 2, the encrypted traffic is handled by entries in `/etc/shorewall/tunnels` (don't be mislead by the name of the file -- *transport mode* encrypted traffic is also handled by entries in that file). The unencrypted traffic is handled by normal rules and policies.

Under the 2.4 Linux Kernel, the association of unencrypted traffic and zones was made easy by the presense of IPSEC pseudo-interfaces with names of the form `ipsecn` (e.g. `ipsec0`). Outgoing unencrypted traffic (case 1.) was send through an `ipsecn` device while incoming unencrypted traffic (case 2) arrived from an `ipsecn` device. The 2.6 kernel-based implementation does away with these pseudo-interfaces. Outgoing traffic that is going to be encrypted and incoming traffic that has been decrypted must be matched against policies in the SPD and/or the appropriate SA.

Shorewall provides support for policy matching in three ways:

1. In `/etc/shorewall/masq`, traffic that will later be encrypted is exempted from MASQUERADE/SNAT using existing entries. If you want to MASQUERADE/SNAT outgoing traffic that will later be encrypted, you must include the appropriate indication in the new IPSEC column in that file.
2. The `/etc/shorewall/zones` file allows you to associate zones with traffic that will be encrypted or that has been decrypted.
3. A new option (**ipsec**) has been provided for entries in `/etc/shorewall/hosts`. When an entry has this option specified, traffic to/from the hosts described in the entry is assumed to be encrypted.

In summary, Shorewall provides the facilities to replace the use of ipsec pseudo-interfaces in zone and MASQUERADE/SNAT definition.

There are two cases to consider:

1. Encrypted communication is used to/from all hosts in a zone.

   The value **ipsec** is placed in the TYPE column of the `/etc/shorewall/zones` entry for the zone.
2. By default, encrypted communication is not used to communicate with the hosts in a zone.

   The value **ipv4** is placed in the TYPE column of the `/etc/shorewall/zones` entry for the zone and the new **ipsec** option is specified in `/etc/shorewall/hosts` for any hosts requiring secure communication.

Note

For simple zones such as are shown in the following examples, the two techniques are equivalent and are used

interchangably.

Note

It is redundent to have **ipsec** in the TYPE column of the `/etc/shorewall/zones` entry for a zone and to also have the **ipsec** option in `/etc/shorewall/hosts` entries for that zone.

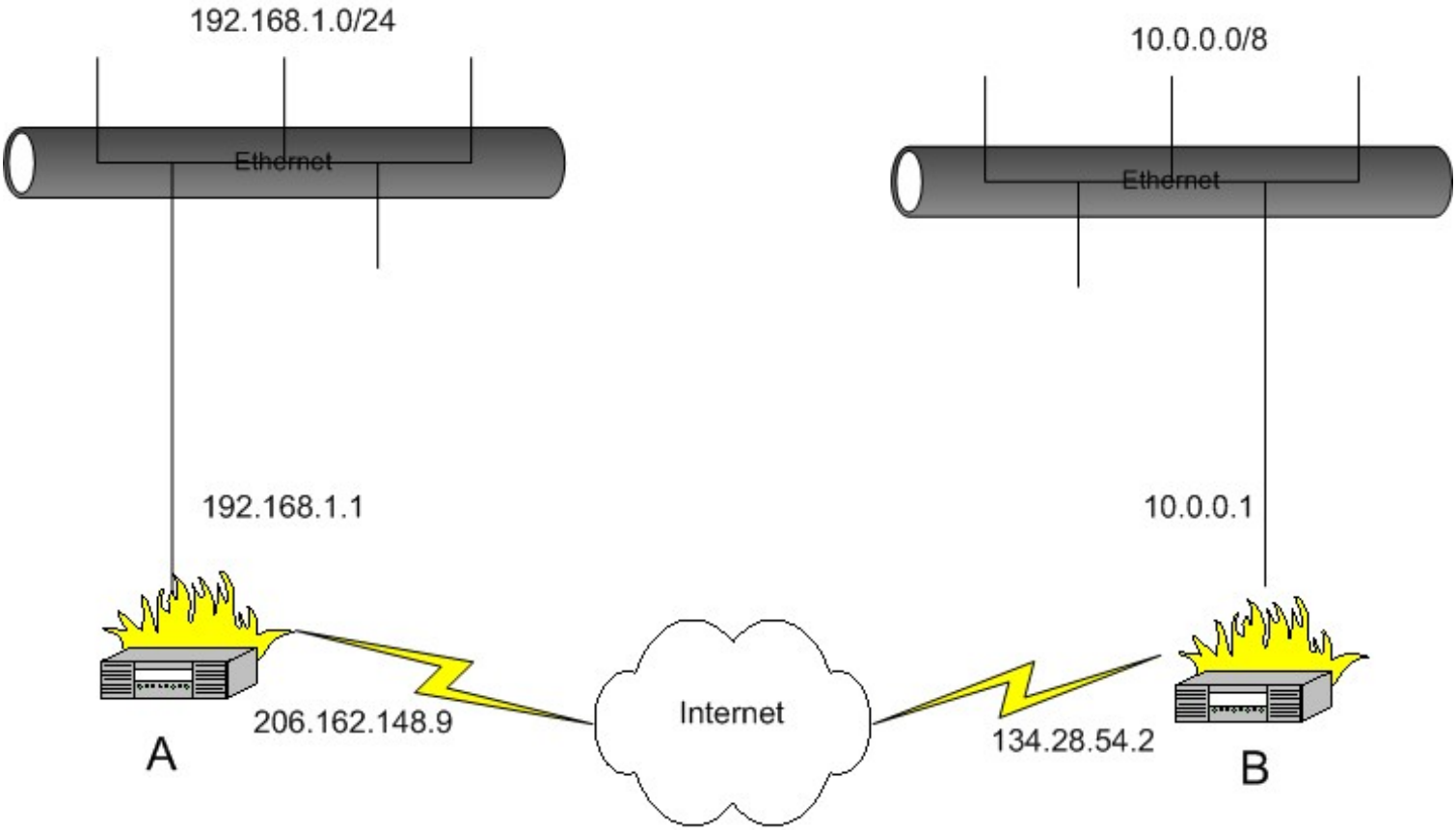
Finally, the OPTIONS, IN OPTIONS and OUT OPTIONS columns in /etc/shorewall/zones can be used to match the zone to a particular (set of) SA(s) used to encrypt and decrypt traffic to/from the zone and the security policies that select which traffic to encrypt/decrypt.

This article assumes the use of ipsec-tools (http://ipsec-tools.sourceforge.net). As of this writing, I recommend that you run at least version 0.5.2. Debian users, please note that there are separate Debian packages for ipsec-tools and racoon although the ipsec-tools project releases them as a single package.

For more information on IPSEC, Kernel 2.6 and Shorewall see my presentation on the subject given at LinuxFest NW 2005. Be warned though that the presentation is based on Shorewall 2.2 and there are some differences in the details of how IPSEC is configured.

IPSec Gateway on the Firewall System

Suppose that we have the following sutuation:



We want systems in the 192.168.1.0/24 sub-network to be able to communicate with systems in the 10.0.0.0/8 network. We assume that on both systems A and B, eth0 is the internet interface.

To make this work, we need to do two things:

a. Open the firewall so that the IPSEC tunnel can be established (allow the ESP and AH protocols and UDP Port 500).
b. Allow traffic through the tunnel.

Opening the firewall for the IPSEC tunnel is accomplished by adding an entry to the `/etc/shorewall/tunnels` file.

In `/etc/shorewall/tunnels` on system A, we need the following

`/etc/shorewall/tunnels` — System A:

```
#TYPE         ZONE        GATEWAY             GATEWAY ZONE
ipsec         net         134.28.54.2
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

`/etc/shorewall/tunnels` — System B:

```
#TYPE         ZONE        GATEWAY             GATEWAY ZONE
ipsec         net         206.162.148.9
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

Note

If either of the endpoints is behind a NAT gateway then the tunnels file entry on the **other** endpoint should specify a tunnel type of ipsecnat rather than ipsec and the GATEWAY address should specify the external address of the NAT gateway.

You need to define a zone for the remote subnet or include it in your local zone. In this example, we'll assume that you have created a zone called “vpn” to represent the remote subnet.

`/etc/shorewall/zones` — Systems A and B:

```
#ZONE          TYPE             OPTIONS             IN           OUT
#                                                    OPTIONS      OPTIONS
vpn            ipv4
net            ipv4
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

Remember the assumption that both systems A and B have eth0 as their internet interface.

You must define the vpn zone using the `/etc/shorewall/hosts` file. The hosts file entries below assume that you want the remote gateway to be part of the vpn zone — If you don't wish the remote gateway included, simply omit it's IP address from the HOSTS column.

/etc/shorewall/hosts — System A

```
#ZONE             HOSTS                                OPTIONS
vpn               eth0:10.0.0.0/8,134.28.54.2          ipsec
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

/etc/shorewall/hosts — System B

```
#ZONE             HOSTS                                OPTIONS
vpn               eth0:192.168.1.0/24,206.162.148.9    ipsec
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

Assuming that you want to give each local network free access to the remote network and vice versa, you would need the following `/etc/shorewall/policy` entries on each system:

```
#SOURCE          DESTINATION            POLICY          LEVEL       BURST:LIMIT
loc              vpn                    ACCEPT
vpn              loc                    ACCEPT
```

Once you have these entries in place, restart Shorewall (type shorewall restart); you are now ready to configure IPSEC.

For full encrypted connectivity in this configuration (between the subnets, between each subnet and the opposite gateway, and between the gateways), you will need eight policies in `/etc/racoon/setkey.conf`. For example, on gateway A:

```
# First of all flush the SPD and SAD databases
spdflush;
flush;

# Add some SPD rules

spdadd 192.168.1.0/24   10.0.0.0/8       any -P out ipsec esp/tunnel/206.162.148.9-
134.28.54.2/require;
spdadd 192.168.1.0/24   134.28.54.2/32   any -P out ipsec esp/tunnel/206.162.148.9-
134.28.54.2/require;
spdadd 206.162.148.9/32 134.28.54.2/32   any -P out ipsec esp/tunnel/206.162.148.9-
134.28.54.2/require;
spdadd 206.162.148.9/32 10.0.0.0/8       any -P out ipsec esp/tunnel/206.162.148.9-
134.28.54.2/require;
spdadd 10.0.0.0/8       192.168.1.0/24   any -P in  ipsec esp/tunnel/134.28.54.2-
206.162.148.9/require;
spdadd 10.0.0.0/8       206.162.148.9/32 any -P in  ipsec esp/tunnel/134.28.54.2-
206.162.148.9/require;
spdadd 134.28.54.2/32   192.168.1.0/24   any -P in  ipsec esp/tunnel/134.28.54.2-
206.162.148.9/require;
spdadd 134.28.54.2/32   206.162.148.9/32 any -P in  ipsec esp/tunnel/134.28.54.2-
206.162.148.9/require;
```

The `setkey.conf` file on gateway B would be similar.

A sample `/etc/racoon/racoon.conf` file using X.509 certificates might look like:

```
path certificates "/etc/certs" ;

listen
{
        isakmp 206.162.148.9;
}

remote 134.28.54.2
{
        exchange_mode main ;
        certificate_type x509 "GatewayA.pem" "GatewayA_key.pem" ;
        verify_cert on;
        my_identifier asn1dn ;
        peers_identifier asn1dn ;
        verify_identifier on ;
        lifetime time 24 hour ;
        proposal {
                encryption_algorithm blowfish;
                hash_algorithm sha1;
                authentication_method rsasig ;
                dh_group 2 ;
        }
}

sainfo address 192.168.1.0/24 any address 10.0.0.0/8 any
{
        pfs_group 2;
        lifetime time 12 hour ;
        encryption_algorithm blowfish ;
        authentication_algorithm hmac_sha1, hmac_md5 ;
        compression_algorithm deflate ;
}

sainfo address 206.162.148.9/32 any address 10.0.0.0/8 any
```

```
{
        pfs_group 2;
        lifetime time 12 hour ;
        encryption_algorithm blowfish ;
        authentication_algorithm hmac_sha1, hmac_md5 ;
        compression_algorithm deflate ;
}

sainfo address 206.162.148.9/32 any address 134.28.54.2/32 any
{
        pfs_group 2;
        lifetime time 12 hour ;
        encryption_algorithm blowfish ;
        authentication_algorithm hmac_sha1, hmac_md5 ;
        compression_algorithm deflate ;
}

sainfo address 192.168.1.0/24 any address 134.28.54.2/32 any
{
        pfs_group 2;
        lifetime time 12 hour ;
        encryption_algorithm blowfish ;
        authentication_algorithm hmac_sha1, hmac_md5 ;
        compression_algorithm deflate ;
}
```

Warning

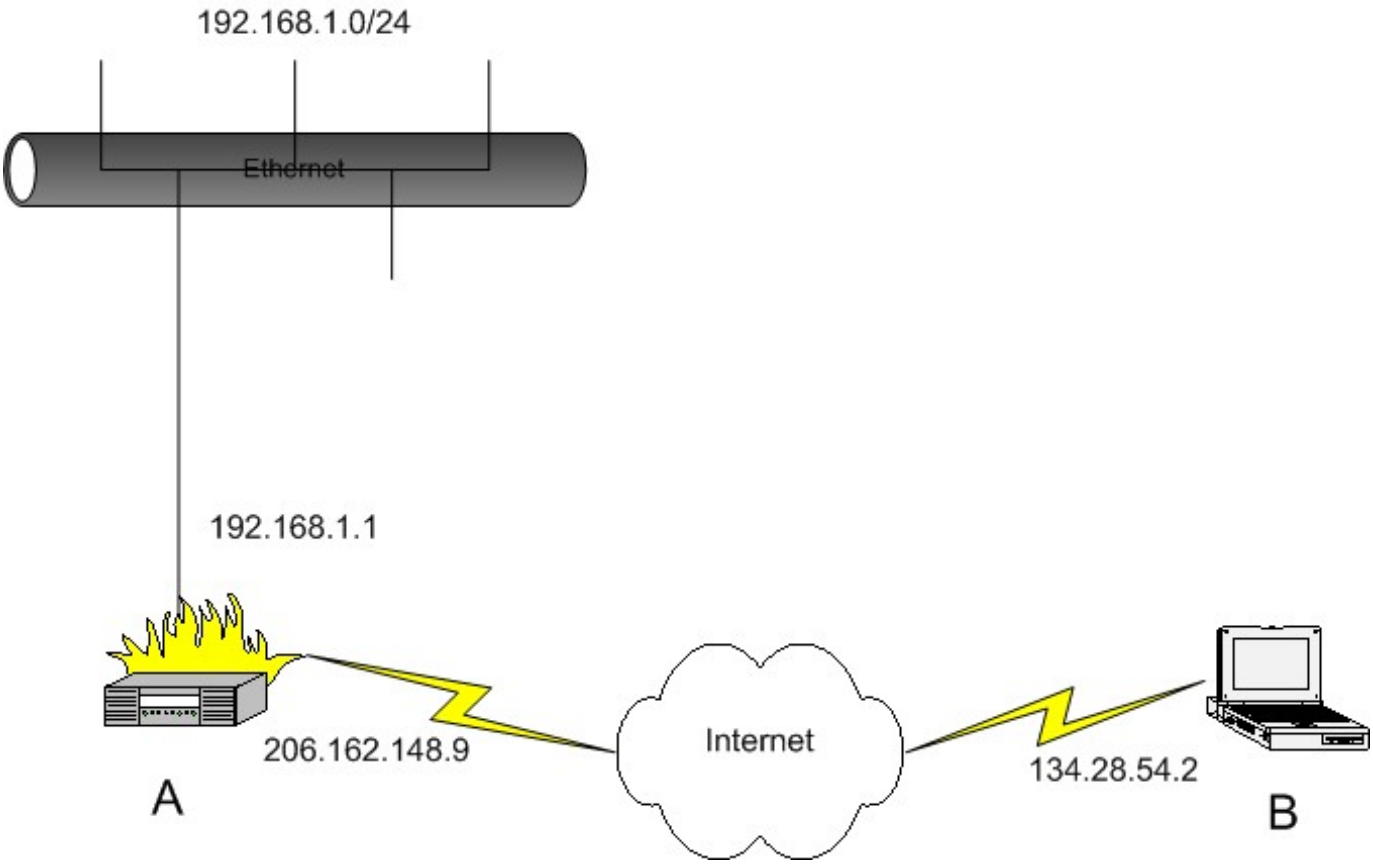
If you have hosts that access the internet through an IPSEC tunnel, then it is a good idea to set the MSS value for traffic from those hosts explicitly in the `/etc/shorewall/zones` file. For example, if hosts in the **sec** zone access the internet through an ESP tunnel then the following entry would be appropriate:

```
#ZONE   TYPE    OPTIONS                 IN                      OUT
#                                       OPTIONS                 OPTIONS
sec     ipsec   mode=tunnel             mss=1400
```

Note that CLAMPMSS=Yes in `shorewall.conf` isn't effective with the 2.6 native IPSEC implementation because there is no separate ipsec device with a lower mtu as there was under the 2.4 and earlier kernels.

Mobile System (Road Warrior)

Suppose that you have a laptop system (B) that you take with you when you travel and you want to be able to establish a secure connection back to your local network.



Example 1. Road Warrior VPN

You need to define a zone for the laptop or include it in your local zone. In this example, we'll assume that you have created a zone called “vpn” to represent the remote host.

/etc/shorewall/zones — System A

```
#ZONE          TYPE             OPTIONS             IN           OUT
#                                                    OPTIONS      OPTIONS
vpn            ipsec
net            ipv4
loc            ipv4
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

In this instance, the mobile system (B) has IP address 134.28.54.2 but that cannot be determined in advance. In the `/etc/shorewall/tunnels` file on system A, the following entry should be made:

```
#TYPE         ZONE        GATEWAY             GATEWAY ZONE
ipsec         net         0.0.0.0/0           vpn
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

Note

the GATEWAY ZONE column contains the name of the zone corresponding to peer subnetworks. This indicates that the gateway system itself comprises the peer subnetwork; in other words, the remote gateway is a standalone system.

The VPN zone is defined using the /etc/shorewall/hosts file:

/etc/shorewall/hosts — System A:

```
#ZONE             HOSTS                   OPTIONS
vpn               eth0:0.0.0.0/0
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

You will need to configure your “through the tunnel” policy as shown under the first example above.

On the laptop:

/etc/shorewall/zones - System B:

```
#ZONE          TYPE             OPTIONS             IN           OUT
#                                                    OPTIONS      OPTIONS
vpn            ipsec
net            ipv4
loc            ipv4
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

/etc/shorewall/tunnels - System B:

```
#TYPE         ZONE        GATEWAY             GATEWAY ZONE
ipsec         net         206.162.148.9       vpn
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

/etc/shorewall/hosts - System B:

```
#ZONE             HOSTS                  OPTIONS
vpn               eth0:0.0.0.0/0
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

On system A, here are the IPSEC files:

/etc/racoon/racoon.conf - System A:

```
path certificate "/etc/certs" ;

listen
{
        isakmp 206.162.148.9;
}

remote anonymous
{
        exchange_mode main ;
        generate_policy on ;
        passive on ;
        certificate_type x509 "GatewayA.pem" "GatewayA_key.pem" ;
        verify_cert on;
        my_identifier asn1dn ;
        peers_identifier asn1dn ;
        verify_identifier on ;
        lifetime time 24 hour ;
        proposal {
                encryption_algorithm blowfish ;
                hash_algorithm sha1;
                authentication_method rsasig ;
                dh_group 2 ;
        }
}

sainfo anonymous
{
        pfs_group 2;
        lifetime time 12 hour ;
```

```
        encryption_algorithm blowfish ;
        authentication_algorithm hmac_sha1, hmac_md5 ;
        compression_algorithm deflate ;
}
```

/etc/racoon/setkey.conf - System A:

```
flush;
spdflush;
```

If system A is running kernel 2.6.10 or later then it must also be running ipsec-tools (racoon) 0.5rc1 or later.

On the mobile system (system B), it is not possible to create a static IPSEC configuration because the IP address of the laptop's internet connection isn't static. I have created an 'ipsecvpn' script and included in the tarball and in the RPM's documentation directory; this script can be used to start and stop the connection.

The ipsecvpn script has some variable assignments at the top -- in the above case, these would be as follows:

```
#
# External Interface
#
INTERFACE=eth0
#
# Remote IPSEC Gateway
#
GATEWAY=206.162.148.9
#
# Networks behind the remote gateway
#
NETWORKS="192.168.1.0/24"
#
# Directory where X.509 certificates are stored.
#
CERTS=/etc/certs
#
# Certificate to be used for this connection. The cert
# directory must contain:
#
#     ${CERT}.pem     - the certificate
#     ${CERT}_key.pem - the certificates's key
#
CERT=roadwarrior
#
#     The setkey binary
#
SETKEY=/usr/sbin/setkey
#
#     The racoon binary
#
RACOON=/usr/sbin/racoon
```

The ipsecvpn script can be installed in /etc/init.d/ but it is probably best installed in /usr/local/sbin and run manually:

ipsecvpn start # Starts the tunnel

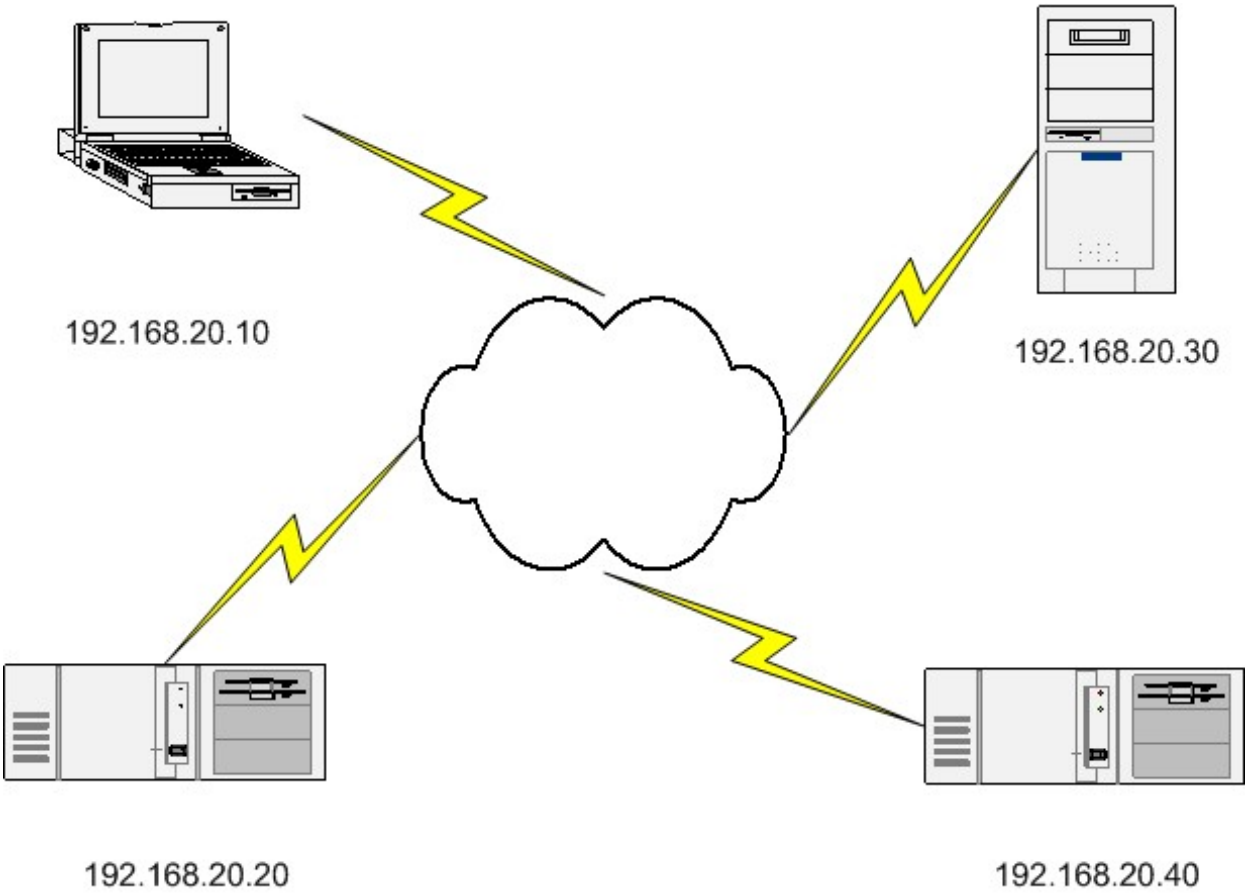
ipsecvpn stop # Stops the tunnel

Warning

Although the ipsecvpn script allows you to specify multiple remote NETWORKS as a space-separated list, SAs are created on the gateway only during ISAKMP negotiation. So in practice, only the first remote network accessed will be accessible from the roadwarrior.

Transport Mode

In today's wireless world, it is often the case that individual hosts in a network need to establish secure connections with the other hosts in that network. In that case, IPSEC transport mode is an appropriate solution.



Here's an example using the ipsec-tools package. The files shown are from host 192.168.20.10; the configuration of the other nodes is similar.

`/etc/racoon/racoon.conf`:

```
path pre_shared_key "/etc/racoon/psk.txt" ;

remote anonymous
{
        exchange_mode main ;
        my_identifier address ;
        lifetime time 24 hour ;
        proposal {
                encryption_algorithm blowfish ;
                hash_algorithm sha1;
                authentication_method pre_shared_key ;
                dh_group 2 ;
        }
}

sainfo anonymous
{
        pfs_group 2;
        lifetime time 12 hour ;
```

```
        encryption_algorithm blowfish ;
        authentication_algorithm hmac_sha1, hmac_md5 ;
        compression_algorithm deflate ;
}
```

`/etc/racoon/setkey.conf`:

```
# First of all flush the SPD database
spdflush;

# Add some SPD rules

spdadd 192.168.20.10/32 192.168.20.20/32 any -P out ipsec esp/transport/192.168.20.10-
192.168.20.20/require;
spdadd 192.168.20.20/32 192.168.20.10/32 any -P in  ipsec esp/transport/192.168.20.20-
192.168.20.10/require;
spdadd 192.168.20.10/32 192.168.20.30/32 any -P out ipsec esp/transport/192.168.20.10-
192.168.20.30/require;
spdadd 192.168.20.30/32 192.168.20.10/32 any -P in  ipsec esp/transport/192.168.20.30-
192.168.20.10/require;
spdadd 192.168.20.10/32 192.168.20.40/32 any -P out ipsec esp/transport/192.168.20.10-
192.168.20.40/require;
spdadd 192.168.20.40/32 192.168.20.10/32 any -P in  ipsec esp/transport/192.168.20.40-
192.168.20.10/require;
```

/etc/racoon/psk.txt:

```
192.168.20.20             <key for 192.168.20.10<->192.168.20.20>
192.168.20.30             <key for 192.168.20.10<->192.168.20.30>
192.168.20.40             <key for 192.168.20.10<->192.168.20.40>
```

Note that the **same key**must be used in both directions.

Shorewall configuration goes as follows:

`/etc/shorewall/interfaces`:

```
#ZONE   INTERFACE       BROADCAST       OPTIONS
net     eth0            detect          routefilter,dhcp,tcpflags
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

/etc/shorewall/tunnels:

```
#TYPE          ZONE             GATEWAY         GATEWAY
#                                               ZONE
ipsec:noah     net              192.168.20.0/24 loc
```

/etc/shorewall/zones:

```
#ZONE          TYPE             OPTIONS             IN            OUT
#                                                   OPTIONS       OPTIONS
loc            ipsec            mode=transport
net            ipv4
```

`/etc/shorewall/hosts`:

```
#ZONE           HOST(S)                         OPTIONS
loc             eth0:192.168.20.0/24
```

```
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS LINE -- DO NOT REMOVE
```

It is worth noting that although *loc* is a sub-zone of *net*, because *loc* is an IPSEC-only zone it does not need to be defined before *net* in */etc/shorewall/zones*.

`/etc/shorewall/policy`:

```
#SOURCE         DEST            POLICY          LOG LEVEL       LIMIT:BURST
$FW             all             ACCEPT
loc             $FW             ACCEPT
net             loc             NONE
loc             net             NONE
net             all             DROP            info
# The FOLLOWING POLICY MUST BE LAST
all             all             REJECT          info
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

Since there are no cases where net<->loc traffic should occur, NONE policies are used.

IPSEC and Windows™ XP

I have successfully configured my work laptop to use IPSEC with X.509 certificates for wireless IP communication when it is undocked at home. I looked at dozens of sites and the one I found most helpful was http://ipsec.math.ucla.edu/services/ipsec-windows.html. The instructions on that site are directed to students at UCLA but they worked fine for me (once I followed them very carefully).

Warning

The instructions found on the UCLA site are complex and do not include any information on the generation of X.509 certificates. There are lots of sites however that can tell you how to generate certificates, including http://www.ipsec-howto.org/.

One piece of information that may not be so easy to find is "How do I generate a PKCS#12 certificate to import into Windows?". Here's the openssl command that I used:

```
openssl pkcs12 -export -in eastepnc6000.pem -inkey eastepnc6000_key.pem -out
eastepnc6000.pfx -name "IPSEC Cert for Home Wireless"
```

I was prompted for a password to associate with the certificate. This password is entered on the Windows system during import.

In the above command:

- `eastepnc6000.pem` was the laptop's certificate in PEM format.
- `eastepnc6000_key.pem` was the laptop's private key (actually, it's the original signing request which includes the private key).
- `eastepnc6000.pfx` is the PKCS#12 output file.
- "IPSEC Cert for Home Wireless" is the friendly name for the certificate.

I started to write an article about how to do this, complete with graphics captured from my laptop. I gave up. I had captured 12 images and hadn't really started yet. The Windows interface for configuring IPSEC is the worst GUI that I have ever used. What can be displayed on one split Emacs screen (racoon.conf plus setkey.conf) takes 20+ different dialog boxes on Windows XP!!!

Source of Additional Samples

Be sure to check out the `src/racoon/samples` subdirectory in the ipsec-tools source tree. It has a wide variety of sample racoon configuration files.

VPN, Netfilter and Shorewall — The Basics

Tom Eastep

Copyright © 2004, 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-09-30

Table of Contents

Gateway-to-gateway traffic vs. Host-to-host traffic.
Relationship to Netfilter
What does this mean with Shorewall?
Defining Remote Zones
Eliminating the /etc/shorewall/tunnels file
    IPSEC
    PPTP
    OpenVPN

Gateway-to-gateway traffic vs. Host-to-host traffic.

The purpose of a *Virtual Private Network* (VPN) is to provide for secure communication between a set of hosts. Communication between a pair of hosts connected by a VPN occurs in stages:

1. **Local-host-to-local-gateway**. This communication is not encrypted; in the case where the traffic originates on the gateway itself, the communication is local to that system.
2. **Local-gateway-to-remote-gateway**. This communication is encrypted and can use a tunneling protocol such as GRE, AH or ESP or a standard protocol such as UDP or TCP. Some VPNs use multiple protocols; for example PPTP uses TCP port 1723 and GRE while IPSEC uses UDP port 500 together with ESP or AH.
3. **Remote-gateway-to-remote-host**. This is just the unencrypted traffic described in the first item as it is delivered to its destination.

Of course, one-way communication generally isn't useful so we need traffic in the other direction as well.

1. **Remote-host-to-remote-gateway**.
2. **Remote-gateway-to-local-gateway**.
3. **Local-gateway-to-local-host**.

Relationship to Netfilter

When Netfilter is configured on a VPN gateway, each VPN packet goes through Netfilter twice! Let's first consider outbound traffic:

1. **Local-host-to-local-gateway**. This traffic has a source address in the local network or on the gateway itself. The destination IP address is that of a remote host; either the remote gateway itself or a host behind that gateway.
2. **Local-gateway-to-remote-gateway.** This (encrypted) traffic has a source IP address on the gateway and is addressed to the remote gateway.

Incoming traffic is similar.

What does this mean with Shorewall?

When Shorewall is installed on a VPN gateway system, it categorizes the VPN-related traffic slightly differently:

1. **Local-host-to-remote-host** — same as **Local-host-to-local-gateway** above.
2. **Local-gateway-to-remote-gateway**.
3. **Remote-gateway-to-local-gateway**.
4. **Remote-host-to-local-host** — same as **Local-gateway-to-local-host** above.

Shorewall implements a set of features for dealing with VPN.

1. The `/etc/shorewall/tunnels` file. This file is used to define remote gateways and the type of encrypted traffic that will be passed between the Shorewall system and those remote gateways. In other words, the tunnels file deals with **Local-gateway-to-remote-gateway** and **Remote-gateway-to-local-gateway** traffic.
2. The `/etc/shorewall/zones` file. An entry in this file allows you to associated a name with the set of hosts behind the remote gateway (or to the remote gateway itself if it is a standalone system).
3. The `/etc/shorewall/interfaces` and `/etc/shorewall/hosts` files. These files are used to associate a set of remote hosts with the zone name defined in `/etc/shorewall/zones`.
4. The `/etc/shorewall/policy` and `/etc/shorewall/rules files`. These files are used to define the connections that are permitted between the remote and local hosts -- in other words, the **Local-host-to-remote-host** and **Remote-host-to-local-host** traffic.

Defining Remote Zones

Most VPN types are implemented using a virtual network device such as pppN (e.g., ppp0), tunN (e.g., tun0), etc. This means that in most cases, remote zone definition is similar to zones that you have already defined.

`/etc/shorewall/zones`:

```
#ZONE           DISPLAY             COMMENT
net             Internet            The big bad net
loc             Local               Local LAN
```

```
rem             Remote                  Remote LAN
```

/etc/shorewall/interfaces:

```
#ZONE           INTERFACE           BROADCAST           OPTION
net             eth0                detect              norft1918,routefilter
loc             eth1                detect
rem             tun0                192.168.10.0/24
```

The /etc/shorewall/hosts file comes into play when:

1. You have a number of remote networks.
2. The remote networks have different firewall requirements and you want to divide them into multiple zones.
3. There is no fixed relationship between the remote networks and virtual network devices (for example, the VPN uses PPTP and remote gateways connect on demand).

In this case, your configuration takes the following approach:

etc/shorewall/zones:

```
#ZONE           DISPLAY             COMMENT
net             Internet            The big bad net
loc             Local               Local LAN
rem1            Remote1             Remote LAN 1
rem2            Remote2             Remote LAN 2
```

/etc/shorewall/interfaces:

```
#ZONE           INTERFACE           BROADCAST           OPTION
net             eth0                detect              norft1918,routefilter
loc             eth1                detect
-               tun+                -
```

/etc/shorewall/hosts:

```
#ZONE           HOST(S)             OPTIONS
rem1            tun+:10.0.0.0/24
rem2            tun+:10.0.1.0/24
```

The `/etc/shorewall/hosts` file is also used with kernel 2.6 native IPSEC.

Eliminating the /etc/shorewall/tunnels file

The `/etc/shorewall/tunnels` file provides no functionality that could not be implemented using entries in `/etc/shorewall/rules` and I have elimination of the `/etc/shorewall/tunnels` file as a long-

term goal. The following sections show how entries in `/etc/shorewall/tunnels` can be replaced by rules for some common tunnel types.

IPSEC

`/etc/shorewall/tunnels`:

```
#TYPE           ZONE          GATEWAY          GATEWAY ZONE
ipsec           Z1            1.2.3.4          Z2
```

`/etc/shorewall/rules`:

```
#ACTION   SOURCE         DEST            PROTO   DEST    SOURCE
#                                                PORT    PORT(S)
ACCEPT    $FW            Z1:1.2.3.4      udp     500
ACCEPT    Z1:1.2.3.4     $FW             udp     500
ACCEPT    $FW            Z1:1.2.3.4      50
ACCEPT    Z1:1.2.3.4     $FW             50
ACCEPT    $FW            Z1:1.2.3.4      51
ACCEPT    Z1:1.2.3.4     $FW             51
ACCEPT    $FW            Z2:1.2.3.4      udp     500
ACCEPT    Z2:1.2.3.4     $FW             udp     500
```

The "noah" option causes the rules for protocol 51 to be eliminated. The "ipsecnat" causes UDP port 4500 to be accepted in both directions. If no GATEWAY ZONE is given then the last two rules above are omitted.

PPTP

`/etc/shorewall/tunnels`:

```
#TYPE           ZONE          GATEWAY          GATEWAY ZONE
pptpserver      Z1            1.2.3.4
```

`/etc/shorewall/rules`:

```
#ACTION   SOURCE         DEST            PROTO   DEST    SOURCE
#                                                PORT    PORT(S)

ACCEPT    Z1:1.2.3.4     $FW             tcp     1723
ACCEPT    $FW            Z1:1.2.3.4      47
ACCEPT    Z1:1.2.3.4     $FW             47
```

Tunnel type "pptpclient" simply reverses the direction of the tcp port 1723 rule.

OpenVPN

`/etc/shorewall/tunnels`:

```
#TYPE           ZONE          GATEWAY          GATEWAY ZONE
openvpn:P       Z1            1.2.3.4
```

/etc/shorewall/rules:

```
#ACTION   SOURCE         DEST            PROTO   DEST    SOURCE
#                                                PORT    PORT(S)

ACCEPT    Z1:1.2.3.4     $FW             udp     P
ACCEPT    $FW            Z1:1.2.3.4      udp     P
```

/etc/shorewall/tunnels:

```
#TYPE           ZONE          GATEWAY          GATEWAY ZONE
openvpnclient:P Z1            1.2.3.4
```

/etc/shorewall/rules:

```
#ACTION   SOURCE         DEST            PROTO   DEST    SOURCE
#                                                PORT    PORT(S)

ACCEPT    Z1:1.2.3.4     $FW             udp     -       P
ACCEPT    $FW            Z1:1.2.3.4      udp     P
```

/etc/shorewall/tunnels:

```
#TYPE           ZONE          GATEWAY          GATEWAY ZONE
openvpnserver:P Z1            1.2.3.4
```

/etc/shorewall/rules:

```
#ACTION   SOURCE         DEST            PROTO   DEST    SOURCE
#                                                PORT    PORT(S)

ACCEPT    Z1:1.2.3.4     $FW             udp     P
ACCEPT    $FW            Z1:1.2.3.4      udp     -       P
```

VPN

Tom Eastep

Copyright © 2002, 2004, 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-03-08

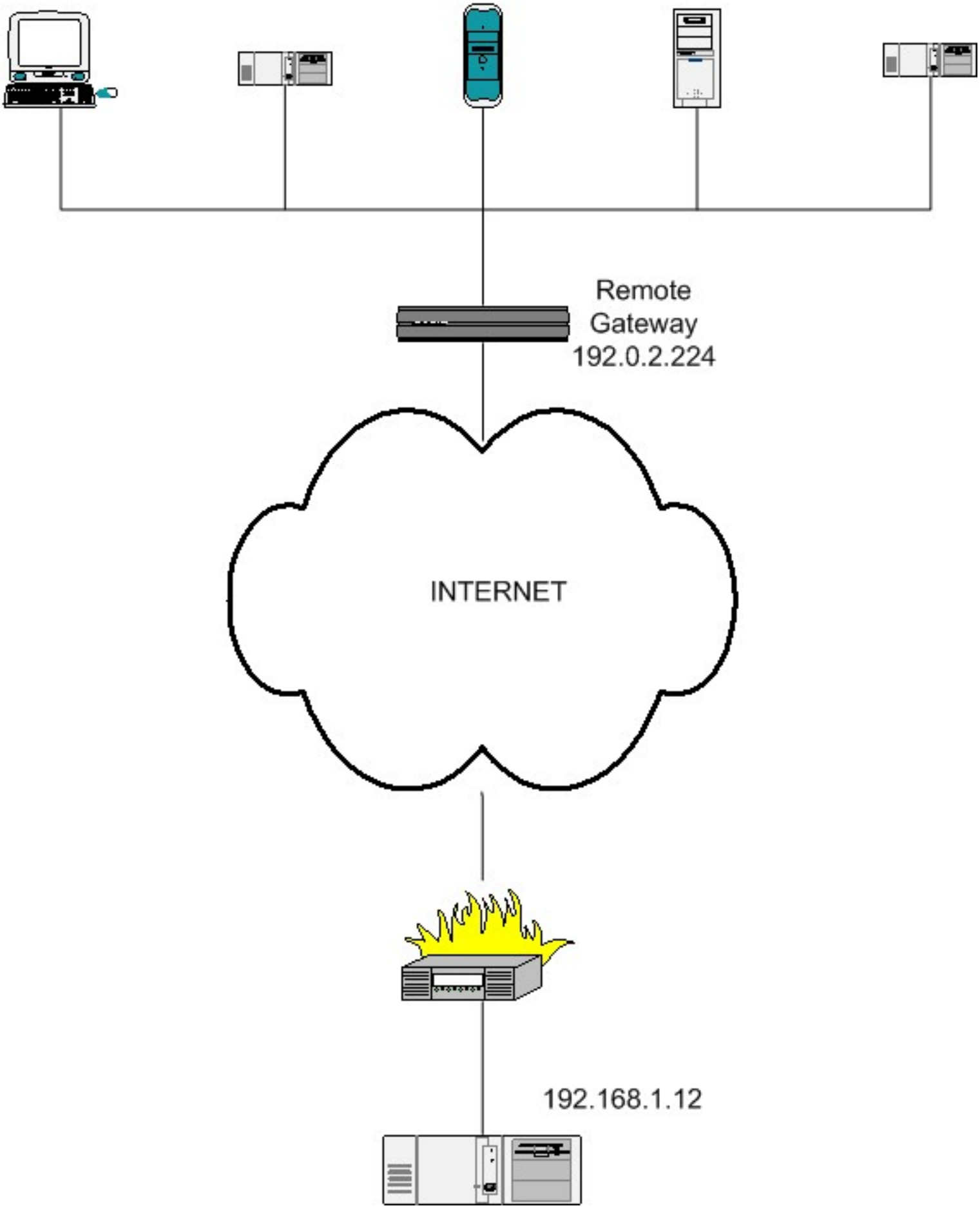
Table of Contents

Virtual Private Networking (VPN)

Virtual Private Networking (VPN)

It is often the case that a system behind the firewall needs to be able to access a remote network through Virtual Private Networking (VPN). The two most common means for doing this are IPSEC and PPTP. The basic setup is shown in the following diagram:

Remote Network



A system with an RFC 1918 address needs to access a remote network through a remote gateway. For this example, we will assume that the local system has IP address 192.168.1.12 and that the remote gateway has IP address 192.0.2.224.

If PPTP is being used, there are no firewall requirements beyond the default loc->net ACCEPT policy. There is one restriction however: Only one local system at a time can be connected to a single remote gateway unless you patch your kernel from the “Patch-o-matic” patches available at http://www.netfilter.org.

If IPSEC is being used, you should configure IPSEC to use *NAT Traversal* -- Under NAT traversal the IPSEC packets (protocol 50 or 51) are encapsulated in UDP packets with destination port 4500. Additionally, *keep-alive messages* are sent frequently so that NATing gateways between the end-points will retain their connection-tracking entries. This is the way that I connect to the HP Intranet and it works flawlessly without anything in Shorewall other than my ACCEPT loc->net policy. NAT traversal is available as a patch for Windows 2K and is a standard feature of Windows XP -- simply select "L2TP IPSec VPN" from the "Type of VPN" pulldown.

Alternatively, if IPSEC is being used then you can try the following: only one system may connect to the remote gateway and there are firewall configuration requirements as follows:

Table 1. /etc/shorewall/rules

ACTION	SOURCE	DESTINATION	PROTOCOL	PORT	CLIENT PORT	ORIGINAL DEST
DNAT	net:192.0.2.224	loc:192.168.1.12	50			
DNAT	net:192.0.2.224	loc:192.168.1.12	udp	500		

The above may or may not work — your milage may vary. NAT Traversal is definitely a better solution.

If you want to be able to give access to all of your local systems to the remote network, you should consider running a VPN client on your firewall. As starting points, see http://www.shorewall.net/Documentation.htm#Tunnels or http://www.shorewall.net/PPTP.htm.

PPTP - Unmaintained

Tom Eastep

Copyright © 2001, 2002, 2003, 2004, 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-04-13

Revision History		
Revision 1.4	2004-11-02	TE
Added link to Greg Kops's tutorial.		
Revision 1.3	2004-05-22	TE
Warning about PPTP conntrack patch and GRE tunnels.		
Revision 1.2	2004-04-15	TE
Revised instructions regarding PPTP conntrack patch.		
Revision 1.1	2003-12-23	TE
Added note about PPTP module support in Bering 1.2		

Abstract

Shorewall easily supports PPTP in a number of configurations.

Table of Contents

Overview
Preliminary Reading
PPTP Server Running on your Firewall
- Patching and building pppd
- Patching and building your Kernel
- Configuring Samba
- Configuring pppd
- Configuring pptpd
- Configuring Shorewall
  - Basic Setup
  - Remote Users in a Separate Zone
  - Multiple Remote Networks

PPTP Server Running Behind your Firewall
PPTP Clients Running Behind your Firewall
PPTP Client Running on your Firewall
PPTP Client running on your Firewall with PPTP Server in an ADSL Modem

Warning

This document is no longer maintained. Any volunteers?

Overview

Note

I am no longer attempting to maintain MPPE patches for current Linux kernel's and pppd. I recommend that you refer to the following URLs for information about installing MPPE into your kernel and pppd.

The Linux PPTP client project has a nice GUI for configuring and managing VPN connections where your Linux system is the PPTP client. This is what I currently use. I am no longer running PoPToP but rather I use the PPTP Server included with XP Professional (see PPTP Server running behind your Firewall below).

http://pptpclient.sourceforge.net

Everything you need to run a PPTP client.
http://www.poptop.org

The “kernelmod” package can be used to quickly install MPPE into your kernel without rebooting.
http://devel.elucid8design.com/el8/devel/tutorials/pptp.php

A nice tutorial for installing a PPTP server on Fedora.

I am leaving the instructions for building MPPE-enabled kernels and pppd in the text below for those who may wish to obtain the relevant current patches and “roll their own”.

Preliminary Reading

I recommend reading the VPN Basics article if you plan to implement any type of VPN.

PPTP Server Running on your Firewall

I will try to give you an idea of how to set up a PPTP server on your firewall system. This isn't a detailed HOWTO but rather an example of how I have set up a working PPTP server on my own firewall.

The steps involved are:

1. the section called “Patching and building pppd”
2. the section called “Patching and building your Kernel”
3. the section called “Configuring Samba”
4. the section called “Configuring pppd”
5. the section called “Configuring pptpd”
6. the section called “Configuring Shorewall”

Patching and building pppd

To run pppd on a 2.4 kernel, you need the pppd 2.4.1 or later. The primary site for releases of pppd is ftp://ftp.samba.org/pub/ppp.

You will need the following patches:

http://www.shorewall.net/pub/shorewall/pptp/ppp-2.4.1-openssl-0.9.6-mppe-patch.gz
http://www.shorewall.net/pub/shorewall/pptp/ppp-2.4.1-MSCHAPv2-fix.patch.gz

You may also want the following patch if you want to require remote hosts to use encryption:

ftp://ftp.shorewall.net/pub/shorewall/pptp/require-mppe.diff

Un-tar the pppd source and uncompress the patches into one directory (the patches and the ppp-2.4.1 directory are all in a single parent directory):

```
cd ppp-2.4.1
patch -p1 < ../ppp-2.4.0-openssl-0.9.6-mppe.patch
patch -p1 < ../ppp-2.4.1-MSCHAPv2-fix.patch
(Optional) patch -p1 < ../require-mppe.diff
./configure
make
```

You will need to install the resulting binary on your firewall system. To do that, I NFS mount my source filesystem and use “make install” from the ppp-2.4.1 directory.

Patching and building your Kernel

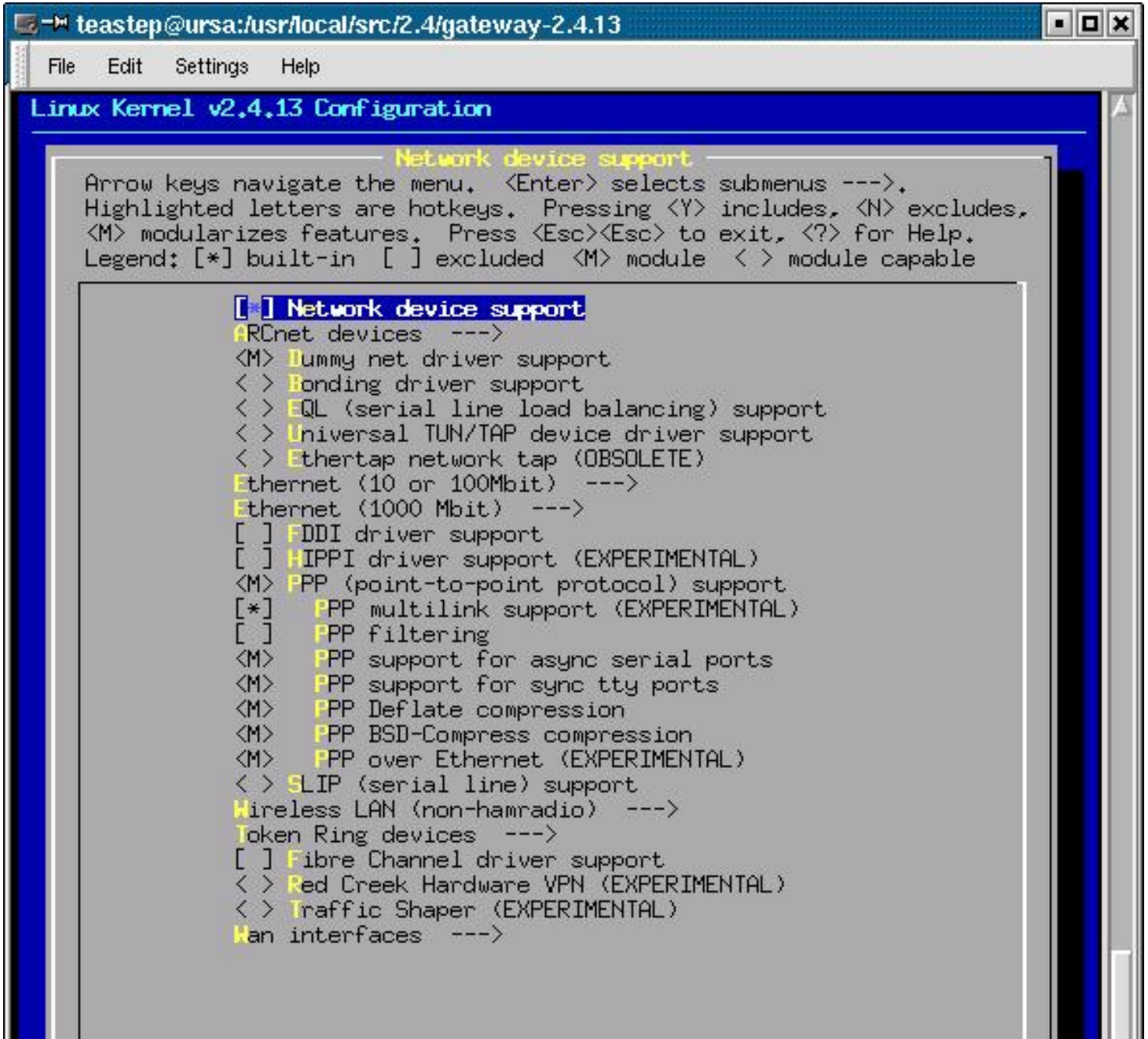
You will need one of the following patches depending on your kernel version:

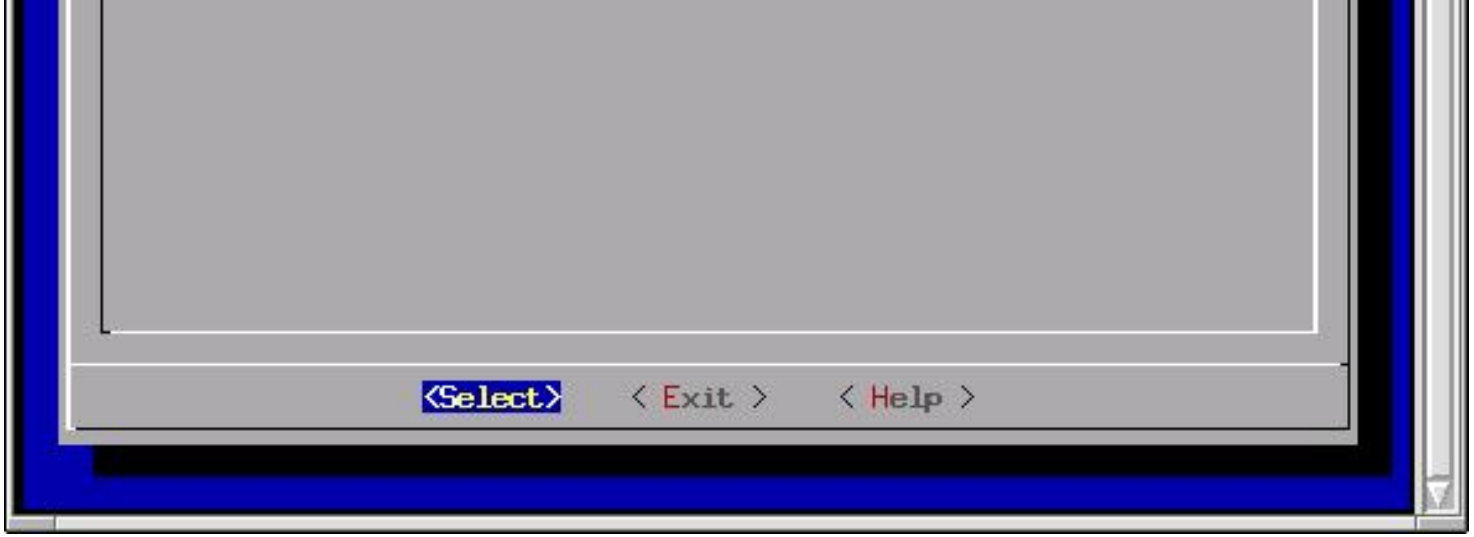
http://www.shorewall.net/pub/shorewall/pptp/linux-2.4.4-openssl-0.9.6a-mppe-patch.gz
http://www.shorewall/net/pub/shorewall/pptp/linux-2.4.16-openssl-0.9.6b-mppe-patch.gz

Uncompress the patch into the same directory where your top-level kernel source is located and:

```
cd <your GNU/Linux source top-level directory>
patch -p1 < ../linux-2.4.16-openssl-0.9.6b-mppe.patch
```

Now configure your kernel. Here is my ppp configuration:





Configuring Samba

You will need a WINS server (Samba configured to run as a WINS server is fine). Global section from /etc/samba/smb.conf on my WINS server (192.168.1.3) is:

```
[global]
     workgroup = TDM-NSTOP
     netbios name = WOOKIE
     server string = GNU/Linux Box
     encrypt passwords = Yes
     log file = /var/log/samba/%m.log
     max log size = 0
     socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
     os level = 65
     domain master = True
     preferred master = True
     dns proxy = No
     wins support = Yes
     printing = lprng

[homes]
     comment = Home Directories
     valid users = %S
```

```
     read only = No
     create mask = 0664
     directory mask = 0775

[printers]
     comment = All Printers
     path = /var/spool/samba
     printable = Yes
```

Configuring pppd

Here is a copy of my /etc/ppp/options.poptop file:

```
ipparam PoPToP
lock
mtu 1490
mru 1490
ms-wins 192.168.1.3
ms-dns 206.124.146.177
multilink
proxyarp
auth
+chap
+chapms
+chapms-v2
ipcp-accept-local
ipcp-accept-remote
lcp-echo-failure 30
lcp-echo-interval 5
deflate 0
mppe-128
mppe-stateless
require-mppe
require-mppe-stateless
```

Note

- System 192.168.1.3 acts as a WINS server so I have included that IP as the “ms-wins” value.
- I have pointed the remote clients at my DNS server -- it has external address 206.124.146.177.
- I am requiring 128-bit stateless compression (my kernel is built with the “require-mppe.diff” patch mentioned above.

Here's my /etc/ppp/chap-secrets:

```
Secrets for authentication using CHAP
# client        server    secret    IP addresses
CPQTDM\\TEastep *         <shhhhhh> 192.168.1.7
TEastep         *         <shhhhhh> 192.168.1.7
```

I am the only user who connects to the server but I may connect either with or without a domain being specified. The system I connect from is my laptop so I give it the same IP address when tunneled in at it has when I use its wireless LAN card around the house.

You will also want the following in /etc/modules.conf:

```
alias ppp-compress-18 ppp_mppe
alias ppp-compress-21 bsd_comp
alias ppp-compress-24 ppp_deflate
alias ppp-compress-26 ppp_deflate
```

Configuring pptpd

PoPTop (pptpd) is available from http://www.poptop.org/.

Here is a copy of my /etc/pptpd.conf file:

```
option /etc/ppp/options.poptop
speed 115200
localip 192.168.1.254
remoteip 192.168.1.33-38
```

Note

- I specify the /etc/ppp/options.poptop file as my ppp options file (I have several).
- The local IP is the same as my internal interface's (192.168.1.254).
- I have assigned a remote IP range that overlaps my local network. This, together with “proxyarp” in my /etc/ppp/options.poptop file make the remote hosts look like they are part of the local subnetwork.

I use this file to start/stop pptpd -- I have this in /etc/init.d/pptpd:

```
#!/bin/sh
#
# /etc/rc.d/init.d/pptpd
#
# chkconfig: 5 12 85
# description: control pptp server
#

case "$1" in
start)
    echo 1 > /proc/sys/net/ipv4/ip_forward
    modprobe ppp_async
    modprobe ppp_generic
    modprobe ppp_mppe
    modprobe slhc
    if /usr/local/sbin/pptpd; then
        touch /var/lock/subsys/pptpd
    fi
    ;;
stop)
    killall pptpd
    rm -f /var/lock/subsys/pptpd
    ;;
restart)
    killall pptpd
    if /usr/local/sbin/pptpd; then
```

```
        touch /var/lock/subsys/pptpd
    fi
    ;;
status)
    ifconfig
    ;;
*)
    echo "Usage: $0 {start|stop|restart|status}"
    ;;
esac
```

Configuring Shorewall

Basic Setup

Here' a basic setup that treats your remote users as if they were part of your **loc** zone. Note that if your primary internet connection uses ppp0, then be sure that **loc** follows **net** in /etc/shorewall/zones.

`/etc/shorewall/tunnels`:

```
#TYPE           ZONE            GATEWAY          GATEWAY ZONE
pptpserver      net             0.0.0.0/0
```

`/etc/shorewall/interfaces`:

```
#ZONE          INTERFACE        BROADCAST        OPTIONS
loc            ppp+
```

Remote Users in a Separate Zone

If you want to place your remote users in their own zone so that you can control connections between these users and the local network, follow this example. Note that if your primary internet connection uses ppp0 then be sure that **vpn** follows **net** in /etc/shorewall/zones as shown below.

`/etc/shorewall/tunnels`:

```
#TYPE           ZONE            GATEWAY          GATEWAY ZONE
pptpserver      net             0.0.0.0/0
```

/etc/shorewall/zones:

```
#ZONE           DISPLAY         COMMENTS
net             Internet        The Internet
loc             Local           Local Network
vpn             VPN             Remote Users
```

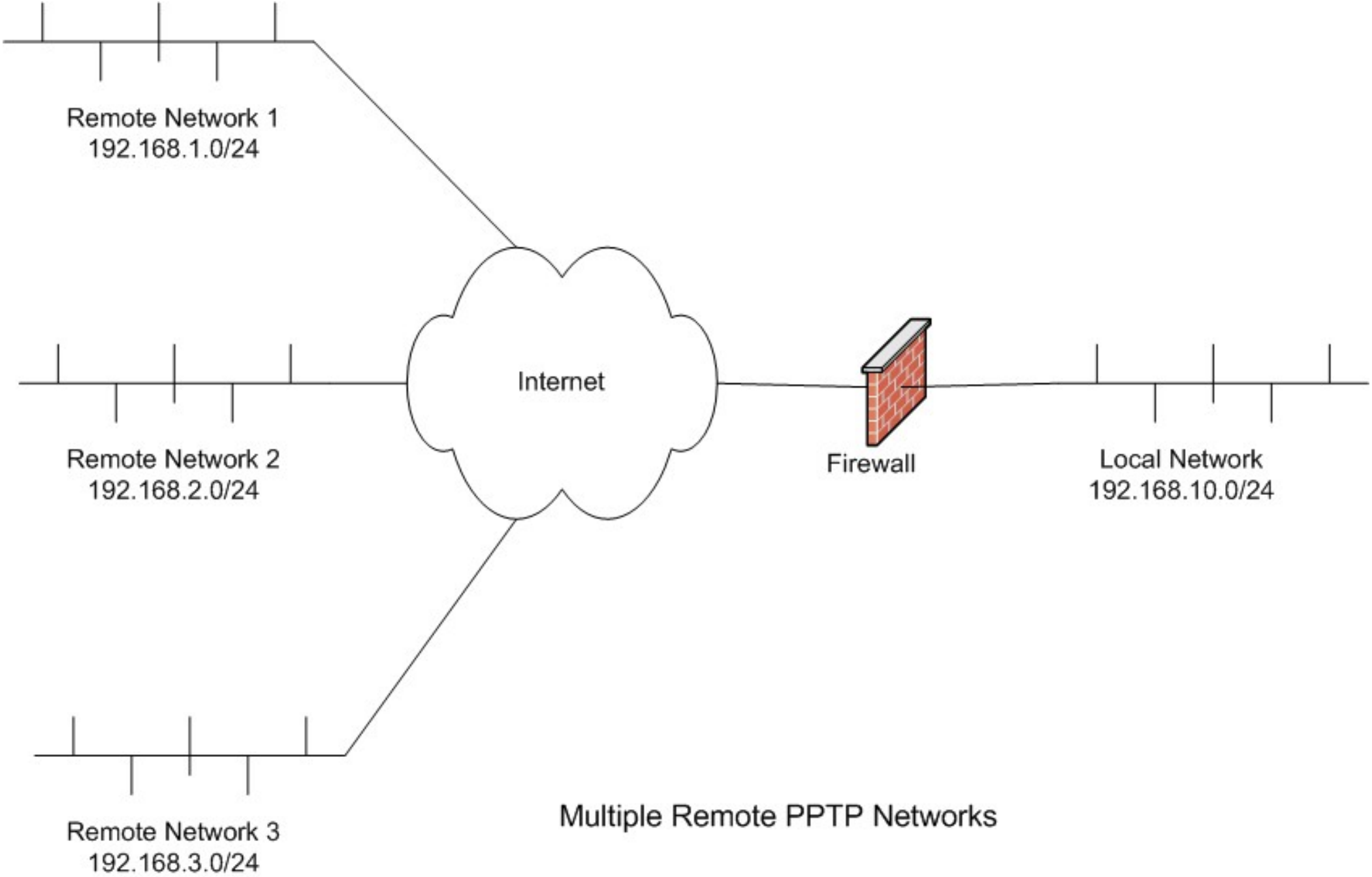
/etc/shorewall/interfaces:

```
#ZONE          INTERFACE        BROADCAST        OPTIONS
net            eth0             206.124.146.255  norfc1918
loc            eth2             192.168.10.255
vpn            ppp+
```

Your policies and rules may now be configured for traffic to/from the **vpn** zone.

Multiple Remote Networks

Often there will be situations where you want multiple connections from remote networks with these networks having different firewalling requirements.



Here's how you configure this in Shorewall. Note that if your primary internet connection uses ppp0 then be sure that the **vpn{1-3}** zones follows **net** in /etc/shorewall/zones as shown below.

/etc/shorewall/tunnels:

```
#TYPE           ZONE            GATEWAY           GATEWAY ZONE
pptpserver      net             0.0.0.0/0
```

/etc/shorewall/zones:

```
#ZONE           DISPLAY         COMMENTS
net             Internet        The Internet
loc             Local           Local Network
vpn1            Remote1         Remote Network 1
vpn2            Remote2         Remote Network 2
vpn3            Remote3         Remote Network 3
```

/etc/shorewall/interfaces:

```
#ZONE          INTERFACE        BROADCAST        OPTIONS
net            eth0             206.124.146.255  norfc1918
loc            eth2             192.168.10.255
-              ppp+
```

/etc/shorewall/hosts:

```
#ZONE          HOST(S)                  OPTIONS
vpn1           ppp+:192.168.1.0/24
vpn2           ppp+:192.168.2.0/24
vpn3           ppp+:192.168.3.0/24
```

Your policies and rules can now be configured using separate zones (vpn1, vpn2, and vpn3) for the three remote network.

PPTP Server Running Behind your Firewall

If you have a single external IP address, add the following to your /etc/shorewall/rules file:

/etc/shorewall/rules:

```
#ACTION      SOURCE         DEST                  PROTO       DEST PORT(S)
DNAT         net            loc:<server address>  tcp         1723
DNAT         net            loc:<server address>  47
```

If you have multiple external IP address and you want to forward a single <*external address*>, add the following to your /etc/shorewall/rules file:

/etc/shorewall/rules:

```
#ACTION      SOURCE         DEST                  PROTO       DEST PORT(S)     SOURCE
ORIGINAL
#
PORT(S)         DEST
DNAT         net            loc:<server address>  tcp         1723             -
<external address>
DNAT         net            loc:<server address>  47          -                -
<external address>
```

PPTP Clients Running Behind your Firewall

You shouldn't have to take any special action for this case unless you wish to connect multiple clients to the same external server. In that case, you must install the PPTP connection/tracking and NAT patch from Netfilter Patch-O-Matic (some distributions are now shipping with this patch installed). I recommend that you also add these four lines to your /etc/shorewall/modules file:

```
loadmodule ip_conntrack_proto_gre
loadmodule ip_conntrack_pptp
loadmodule ip_nat_pptp
loadmodule ip_nat_proto_gre
```

For LEAF/Bering users, the 2.4.20 kernel as already been patched as described at the URL above and the three modules are included in the Bering 1.2 modules tarball.

Warning

Installing the above modules will prevent any GRE tunnels that you have from working correctly.

PPTP Client Running on your Firewall

The PPTP GNU/Linux client is available at http://sourceforge.net/projects/pptpclient/. Rather than use the configuration script that comes with the client, I built my own. I also build my own kernel as described above rather than using the mppe package that is available with the client. My /etc/ppp/options file is mostly unchanged from what came with the client (see below).

The key elements of this setup are as follows:

1. Define a zone for the remote network accessed via PPTP.
2. Associate that zone with a ppp interface.
3. Define rules for PPTP traffic to/from the firewall.
4. Define rules for traffic two and from the remote zone.

Here are examples from one of my old setups:

`/etc/shorewall/zones`:

```
#ZONE           DISPLAY         COMMENTS
cpq             Compaq          Compaq Intranet
```

`/etc/shorewall/interfaces`:

```
#ZONE           INTERFACE       BROADCAST           OPTIONS
-               ppp+
```

/etc/shorewall/hosts:

```
#ZONE           HOST(S)                             OPTIONS
```

```
cpq             ppp+:!192.168.1.0/24
```

/etc/shorewall/tunnels:

```
#TYPE           ZONE              GATEWAY            GATEWAY ZONE
pptpclient      net               0.0.0.0/0
```

I use the combination of interface and hosts file to define the “cpq” zone because I also run a PPTP server on my firewall (see above). Using this technique allows me to distinguish clients of my own PPTP server from arbitrary hosts at Compaq; I assign addresses in 192.168.1.0/24 to my PPTP clients and Compaq doesn't use that RFC1918 Class C subnet.

I use this script in /etc/init.d to control the client. The reason that I disable ECN when connecting is that the Compaq tunnel servers don't do ECN yet and reject the initial TCP connection request if I enable ECN :-(

```
#!/bin/sh
#
# /etc/rc.d/init.d/pptp
#
# chkconfig: 5 60 85
# description: PPTP Link Control
#
NAME="Tandem"
ADDRESS=tunnel-tandem.compaq.com
USER='Tandem\tommy'
ECN=0
DEBUG=

start_pptp() {
    echo $ECN > /proc/sys/net/ipv4/tcp_ecn
    if /usr/sbin/pptp $ADDRESS user $USER noauth $DEBUG; then
        touch /var/lock/subsys/pptp
        echo "PPTP Connection to $NAME Started"
    fi
}
```

```
stop_pptp() {
    if killall /usr/sbin/pptp 2> /dev/null; then
        echo "Stopped pptp"
    else
        rm -f /var/run/pptp/*
    fi

    # if killall pppd; then
    # echo "Stopped pppd"
    # fi

    rm -f /var/lock/subsys/pptp

    echo 1 > /proc/sys/net/ipv4/tcp_ecn
}


case "$1" in
start)
    echo "Starting PPTP Connection to ${NAME}..."
    start_pptp
    ;;
stop)
    echo "Stopping $NAME PPTP Connection..."
    stop_pptp
    ;;
restart)
    echo "Restarting $NAME PPTP Connection..."
    stop_pptp
    start_pptp
    ;;
status)
    ifconfig
    ;;
*)
    echo "Usage: $0 {start|stop|restart|status}"
    ;;
esac
```

Here's my /etc/ppp/options file:

```
#
# Identify this connection
#
ipparam Compaq
#
# Lock the port
#
lock
#
# We don't need the tunnel server to authenticate itself
#
noauth

+chap
+chapms
+chapms-v2

multilink
mrru 1614
#
# Turn off transmission protocols we know won't be used
#
nobsdcomp
nodeflate

#
# We want MPPE
#
mppe-128
mppe-stateless

#
# We want a sane mtu/mru
```

```
#
mtu 1000
mru 1000

#
# Time this thing out of it goes poof
#
lcp-echo-failure 10
lcp-echo-interval 10
```

My /etc/ppp/ip-up.local file sets up the routes that I need to route Compaq traffic through the PPTP tunnel:

```
#/bin/sh

case $6 in
Compaq)
    route add -net 16.0.0.0 netmask 255.0.0.0 gw $5 $1
    route add -net 130.252.0.0 netmask 255.255.0.0 gw $5 $1
    route add -net 131.124.0.0 netmask 255.255.0.0 gw $5 $1
    ...
    ;;
esac
```

Finally, I run the following script every five minutes under crond to restart the tunnel if it fails:

```
#!/bin/sh
restart_pptp() {
    /sbin/service pptp stop
    sleep 10
    if /sbin/service pptp start; then
        /usr/bin/logger "PPTP Restarted"
    fi
}

if [ -n "`ps ax | grep /usr/sbin/pptp | grep -v grep`" ]; then
    exit 0
```

```
fi

echo "Attempting to restart PPTP"

restart_pptp > /dev/null 2>&1 &
```

Here's a script and corresponding ip-up.local from Jerry Vonau <jvonau@home.com> that controls two PPTP connections.

PPTP Client running on your Firewall with PPTP Server in an ADSL Modem

Some ADSL systems in Europe (most notably in Austria) feature a PPTP server built into an ADSL “Modem”. In this setup, an ethernet interface is dedicated to supporting the PPTP tunnel between the firewall and the “Modem” while the actual internet access is through PPTP (interface ppp0). If you have this type of setup, you need to modify the sample configuration that you downloaded as described in this section. **These changes are in addition to those described in the QuickStart Guides.**

Lets assume the following:

- ADSL Modem connected through eth0
- Modem IP address = 192.168.1.1
- eth0 IP address = 192.168.1.2

The changes you need to make are as follows:

1. Add this entry to /etc/shorewall/zones:

   ```
   #ZONE           DISPLAY         COMMENTS
   modem           Modem           ADSL Modem
   ```

   That entry defines a new zone called “modem” which will contain only your ADSL modem.
2. Add the following entry to /etc/shorewall/interfaces:

   ```
   #ZONE           INTERFACE       BROADCAST          OPTIONS
   ```

```
modem           eth0            192.168.1.255       dhcp
```

You will of course modify the “net” entry in /etc/shorewall/interfaces to specify “ppp0” as the interface as described in the QuickStart Guide corresponding to your setup.

3. Add the following to /etc/shorewall/tunnels:

```
#TYPE           ZONE            GATEWAY             GATEWAY ZONE
pptpclient      modem           192.168.1.1
```

That entry allows a PPTP tunnel to be established between your Shorewall system and the PPTP server in the modem.

Samba/SMB

Tom Eastep

Copyright © 2002-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-09-30

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

If you wish to run Samba on your firewall and access shares between the firewall and local hosts, you need the following rules:

```
#ACTION    SOURCE    DEST    PROTO     DEST PORT(S)    SOURCE
#                                                      PORT(S)
SMB/ACCEPT  $FW       loc
SMB/ACCEPT  loc       $FW
```

To pass traffic SMB/Samba traffic between zones Z1 and Z2:

```
#ACTION    SOURCE    DEST    PROTO     DEST PORT(S)    SOURCE
#                                                      PORT(S)
SMB/ACCEPT  Z1        Z2
SMB/ACCEPT  Z2        Z1
```

To make network browsing ("Network Neighborhood") work properly between Z1 and Z2 **requires a Windows Domain Controller and/or a WINS server.** I have run Samba on my firewall to handle browsing between two zones connected to my firewall.

When debugging Samba/SMB problems, I recommend that you do the following:

1. Copy `action.Drop` and `action.Reject` from `/usr/share/shorewall` to `/etc/shorewall`.
2. Edit the copies and remove the **SMB/DROP** and **SMB/REJECT** lines.
3. **shorewall restart**

The above steps will cause SMB traffic that is dropped or rejected by policy to be logged rather than handled silently.

If you are using Windows XP™ to test your setup,make you sure you have a properly configured client firewall .

You can just remove the copies and **shorewall restart** when you are finished debugging.

Shorewall Logging

Tom Eastep

Copyright © 2001 - 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-10-10

Table of Contents

How to Log Traffic Through a Shorewall Firewall
Where the Traffic is Logged and How to Change the Destination
- Syslog Levels
- Configuring a Separate Log for Shorewall Messages (ulogd)

Syslog-ng
Understanding the Contents of Shorewall Log Messages

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

How to Log Traffic Through a Shorewall Firewall

The disposition of packets entering a Shorewall firewall is determined by one of a number of Shorewall facilities. Only some of these facilities permit logging.

1. The packet is part of an established connecection. While the packet can be logged using LOG rules in the ESTABLISHED section of /etc/shorewall/rules, that is not recommended because of the large amount of information that may be logged.
2. The packet represents a connection request that is related to an established connection (such as a data connection associated with an FTP control connection). These packets may be logged using LOG rules in the RELATED section of /etc/shorewall/rules.
3. The packet is rejected because of an option in /etc/shorewall/shorewall.conf or /etc/shorewall/interfaces. These packets can be logged by setting the appropriate logging-related option in /etc/shorewall/shorewall.conf.
4. The packet matches a rule in /etc/shorewall/rules. By including a syslog level (see below) in the ACTION column of a rule (e.g., "ACCEPT**:info** net $FW tcp 22"), the connection attempt will be logged at that level.
5. The packet doesn't match a rule so it is handled by a policy defined in /etc/shorewall/policy. These may be logged by specifying a syslog level in the LOG LEVEL column of the policy's entry (e.g., "loc net ACCEPT **info**").

Where the Traffic is Logged and How to Change the Destination

By default, Shorewall directs NetFilter to log using syslog (8). Syslog classifies log messages by a *facility* and a *priority* (using

the notation *facility.priority*).

The facilities defined by syslog are *auth, authpriv, cron, daemon, kern, lpr, mail, mark, news, syslog, user, uucp* and *local0* through *local7*.

Throughout the Shorewall documentation, I will use the term *level* rather than *priority* since *level* is the term used by NetFilter. The syslog documentation uses the term *priority*.

Syslog Levels

Syslog levels are a method of describing to syslog (8) the importance of a message. A number of Shorewall parameters have a syslog level as their value.

Valid levels are:

7 - **debug** (Debug-level messages)
6 - **info** (Informational)
5 - **notice** (Normal but significant Condition)
4 - **warning** (Warning Condition)
3 - **err** (Error Condition)
2 - **crit** (Critical Conditions)
1 - **alert** (must be handled immediately)
0 - **emerg** (System is unusable)

For most Shorewall logging, a level of 6 (info) is appropriate. Shorewall log messages are generated by NetFilter and are logged using the *kern* facility and the level that you specify. If you are unsure of the level to choose, 6 (info) is a safe bet. You may specify levels by name or by number.

Syslogd writes log messages to files (typically in /var/log/*) based on their facility and level. The mapping of these facility/level pairs to log files is done in /etc/syslog.conf (5). If you make changes to this file, you must restart syslogd before the changes can take effect.

Syslog may also write to your system console. See Shorewall FAQ 16 for ways to avoid having Shorewall messages written to the console.

Configuring a Separate Log for Shorewall Messages (ulogd)

There are a couple of limitations to syslogd-based logging:

1. If you give, for example, kern.info it's own log destination then that destination will also receive all kernel messages of levels 5 (notice) through 0 (emerg).
2. All kernel.info messages will go to that destination and not just those from NetFilter.

Beginning with Shorewall version 1.3.12, if your kernel has ULOG target support (and most vendor-supplied kernels do), you may also specify a log level of ULOG (must be all caps). When ULOG is used, Shorewall will direct netfilter to log the related messages via the ULOG target which will send them to a process called “ulogd”. The ulogd program is included in most distributions and is also available from http://www.gnumonks.org/projects/ulogd. Ulogd can be configured to log all Shorewall messages to their own log file.

Note

The ULOG logging mechanism is completely separate from syslog. Once you switch to ULOG, the settings in /etc/syslog.conf have absolutely no effect on your Shorewall logging (except for Shorewall status messages which

still go to syslog).

Once you have installed ulogd, edit /etc/ulogd.conf (`/usr/local/etc/ulogd.conf` if you built ulogd yourself) and set:

1. syslogfile *<the file that you wish to log to>*
2. syslogsync 1

Also on the firewall system:

touch *<the file that you wish to log to>*

Your distribution's ulogd package may include a logrotate file in /etc/logrotate.d. If you change the log file location, be sure to change that logrotate file accordingly.

You will need to change all instances of log levels (usually “info”) in your Shorewall configuration files to “ULOG” - this includes entries in the policy, rules and shorewall.conf files. Here's what I had at one time:

```
gateway:/etc/shorewall# grep -v ^\# * | egrep '\$LOG|ULOG|LOGFILE'
params:LOG=ULOG
policy:loc              $FW             REJECT          $LOG
policy:net              all             DROP            $LOG            10/sec:40
policy:all              all             REJECT          $LOG
rules:REJECT:$LOG       loc                             net                     tcp
25
rules:REJECT:$LOG       loc                             net                     udp
1025:1031
rules:REJECT:$LOG       dmz                             net                     udp
1025:1031
rules:ACCEPT:$LOG       dmz                             net                     tcp
1024:                                   20
rules:REJECT:$LOG       $FW                             net                     udp
1025:1031
shorewall.conf:LOGFILE=/var/log/shorewall
shorewall.conf:LOGUNCLEAN=$LOG
shorewall.conf:MACLIST_LOG_LEVEL=$LOG
shorewall.conf:TCP_FLAGS_LOG_LEVEL=$LOG
shorewall.conf:RFC1918_LOG_LEVEL=$LOG
gateway:/etc/shorewall#
```

Finally edit /etc/shorewall/shorewall.conf and set LOGFILE=*<file that you wish to log to>*. This tells the /sbin/shorewall program where to look for the log when processing its “show log”, “logwatch” and “monitor” commands.

Syslog-ng

Here is a post describing configuring syslog-ng to work with Shorewall.

Understanding the Contents of Shorewall Log Messages

For general information on the contents of Netfilter log messages, see http://logi.cc/linux/netfilter-log-format.php3.

For Shorewall-specific information, see FAQ #17.

Shorewall and Routing

Tom Eastep

Copyright © 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-11-16

Table of Contents

Routing vs. Firewalling.
Routing and Netfilter
- Packets Entering the Firewall from Outside
- Packets Originating on the Firewall

Alternate Routing Table Configuration
Routing and Proxy ARP
Multiple Internet Connection Support in Shorewall 2.4.2 and Later

Routing vs. Firewalling.

One of the most misunderstood aspects of Shorewall is its relationship with routing. This article attempts to clear some of the fog that surrounds this issue.

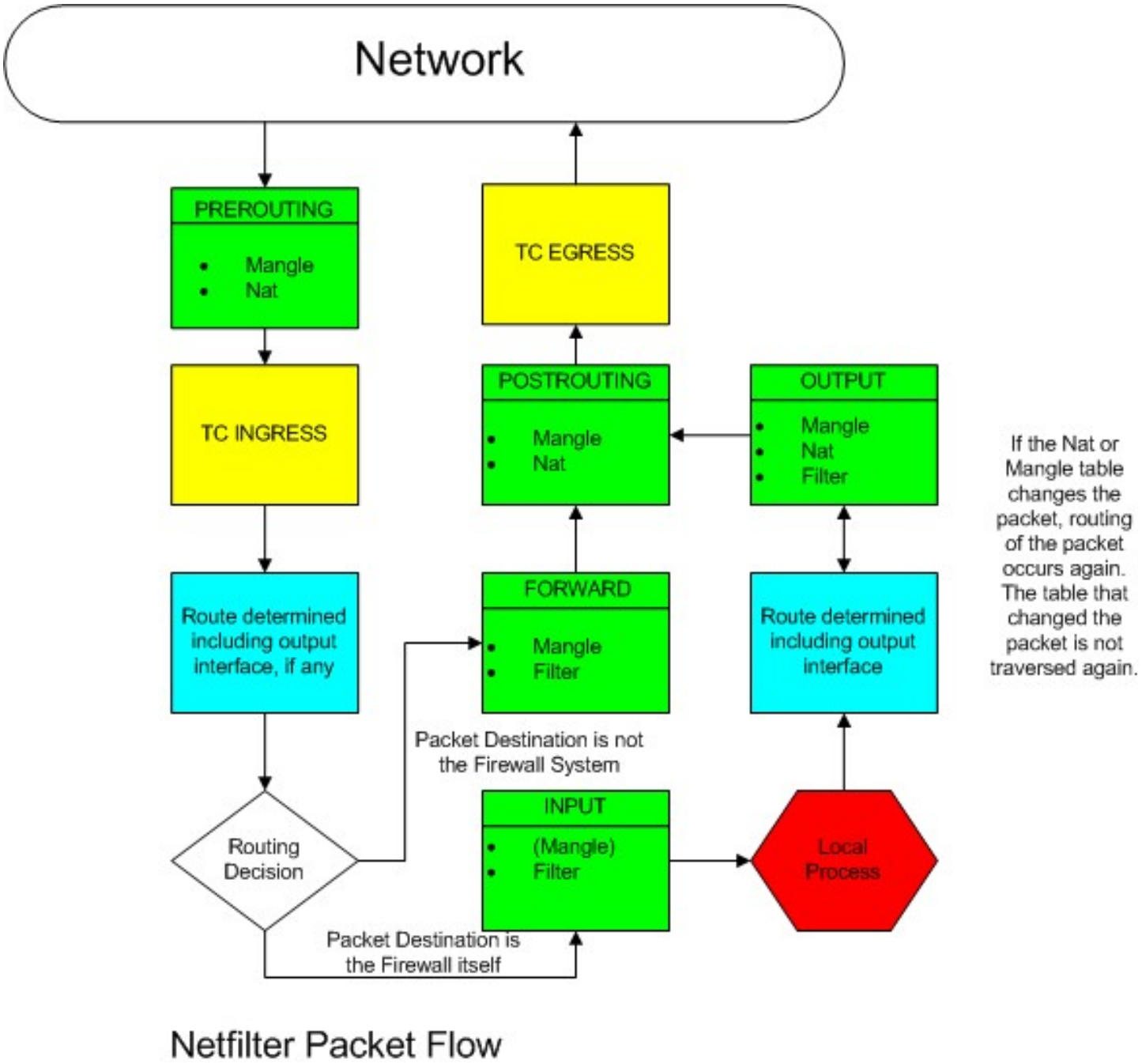
As a general principle:

1. Routing determines where packets are to be sent.
2. Once routing determines where the packet is to go, the firewall (Shorewall) determines if the packet is allowed to go there.

There are ways that Shorewall can affect routing which are described in the following sections.

Routing and Netfilter

The following diagram shows the relationship between routing decisions and Netfilter.



The light blue boxes indicate where routing decisions are made. Upon exit from one of these boxes, if the packet is being sent to another system then the interface and the next hop have been uniquely determined.

The green boxes show where Netfilter processing takes place (as directed by Shorewall). You will notice that there are two different paths through this maze, depending on where the packet originates. We will look at each of these separately.

Packets Entering the Firewall from Outside

When a packet arrives from outside, it first undergoes Netfilter PREROUTING processing. In Shorewall terms:

1. Packets may be marked using entries in the /etc/shorewall/tcrules file. Entries in that file containing ":P" in the mark column are applied here as are rules that default to the MARK_IN_FORWARD_CHAIN=No setting in `/etc/shorewall/shorewall.conf`. These marks may be used to specify that the packet should be routed using an *alternate routing table*; see the Shorewall Squid documentation for examples.

Caution

Marking packets then using the *fwmark* selector in your "**ip rule add**" commands should NOT be your first choice. In most cases, you can use the *from* or *dev* selector instead.

2. The destination IP address may be rewritten as a consequence of:
   - DNAT[-] rules.
   - REDIRECT[-] rules.
   - Entries in `/etc/shorewall/nat`.

So the only influence that Shorewall has over where these packets go is via NAT or by marking them so that they may be routed using an alternate routing table.

Packets Originating on the Firewall

Processing of packets that originate on the firewall itself are initially routed using the default routing table then passed through the OUTPUT chains. Shorewall can influence what happens here:

1. Packets may be marked using entries in the /etc/shorewall/tcrules file (rules with "$FW" in the SOURCE column). These marks may be used to specify that the packet should be re-routed using an alternate routing table.
2. The destination IP address may be rewritten as a consequence of:
   - DNAT[-] rules that specify $FW as the SOURCE.
   - Entries in `/etc/shorewall/nat` that have "Yes" in LOCAL column.

So again in this case, the only influence that Shorewall has over the packet destination is NAT or marking.

Alternate Routing Table Configuration

The Shorewall Squid documentation shows how alternate routing tables can be created and used. That documentation shows how you can use logic in `/etc/shorewall/init` to create and populate an alternate table and to add a routing rule for its use. It is fine to use that technique so long as you understand that you are basically just using the Shorewall init script (`/etc/init.d/shorewall`) to configure your alternate routing table at boot time and that **other than as described in the previous section, there is no connection between Shorewall and routing when using Shorewall versions prior to 2.3.2.**

Routing and Proxy ARP

There is one instance where Shorewall creates main routing table entries. When an entry in `/etc/shorewall/proxyarp` contains "No" in the HAVEROUTE column then Shorewall will create a host route to the IP address listed in the ADDRESS column through the interface named in the INTERFACE column. **This is the only case where Shorewall directly manipulates the main routing table**.

Example:

`/etc/shorewall/proxyarp`:

```
#ADDRESS        INTERFACE       EXTERNAL        HAVEROUTE       PERSISTENT
206.124.146.177 eth1            eth0            No
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

The above entry will cause Shorewall to execute the following command:

```
ip route add 206.124.146.177 dev eth1
```

Multiple Internet Connection Support in Shorewall 2.4.2 and Later

Beginning with Shorewall 2.3.2, support is included for multiple internet connections. If you wish to use this feature, we recommend strongly that you upgrade to version 2.4.2 or later.

Shorewall multi-ISP support is now covered in a separate article.

Shorewall and Multiple Internet Connections

Tom Eastep

Copyright © 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-11-16

Table of Contents

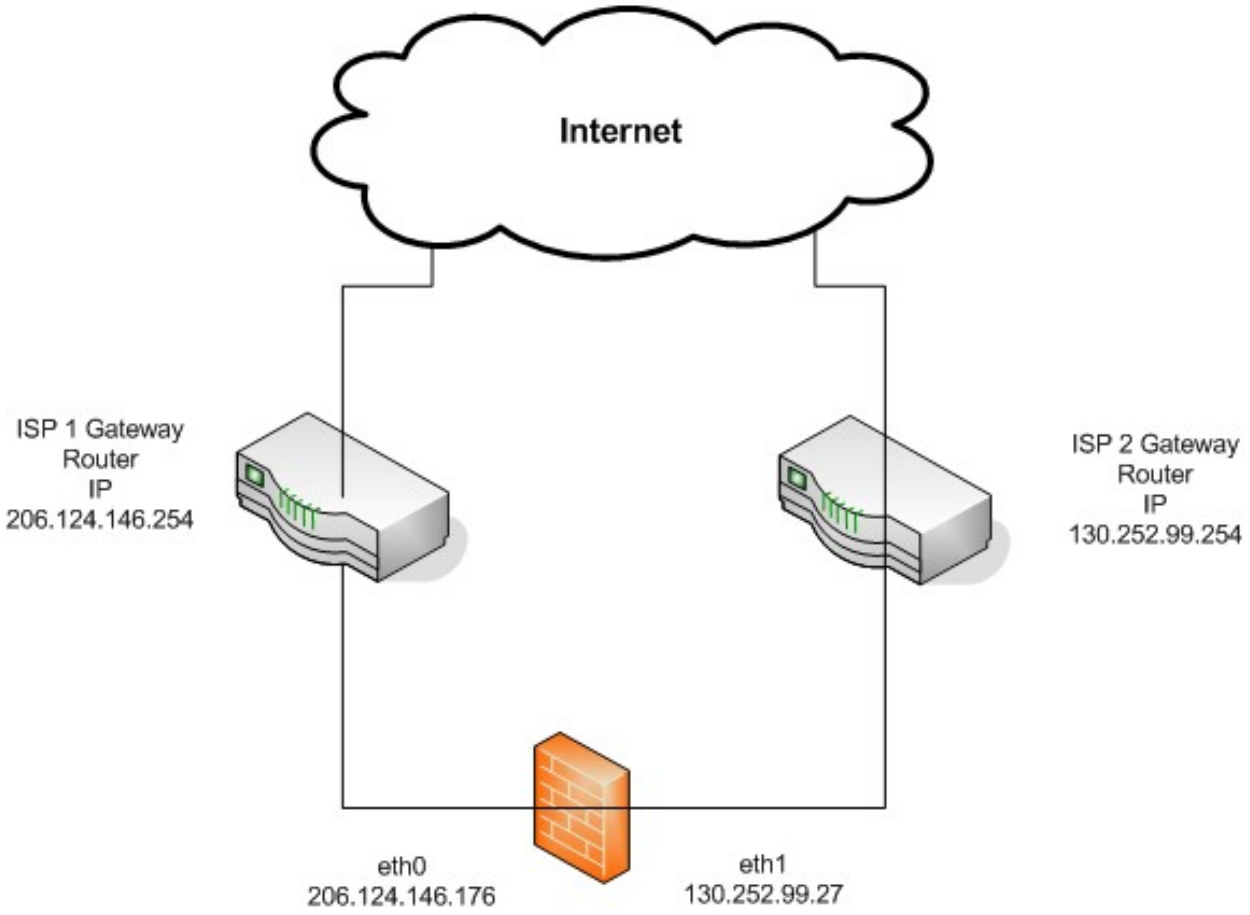
Multiple Internet Connection Support in Shorewall 2.4.2 and Later
- Overview
- /etc/shorewall/providers File
- What an entry in the Providers File Does
- Example

Multiple Internet Connection Support in Shorewall 2.4.2 and Later

Beginning with Shorewall 2.3.2, support is included for multiple internet connections. If you wish to use this feature, we recommend strongly that you upgrade to version 2.4.2 or later. This section assumes that you have so upgraded.

Overview

Let's assume that a firewall is connected via two separate ethernet interfaces to two different ISPs as in the following diagram.



- eth0 connects to ISP1. The IP address of eth0 is 206.124.146.176 and the ISP's gateway router has IP address 206.124.146.254.
- eth1 connects to ISP 2. The IP address of eth1 is 130.252.99.27 and the ISP's gateway router has IP address 130.252.99.254.

Each of these *providers* is described in an entry in the file `/etc/shorewall/providers`.

Entries in `/etc/shorewall/providers` can specify that outgoing connections are to be load-balanced between the two ISPs. Entries in `/etc/shorewall/tcrules` can be used to direct particular outgoing connections to one ISP or the other. Use of `/etc/shorewall/tcrules` is not required for `/etc/shorewall/providers` to work, but you must select a unique MARK value for each provider so Shorewall can set up the correct marking rules for you.

When using `/etc/shorewall/providers`, connections from the internet are automatically routed back out of the correct interface and through the correct ISP gateway. This works whether the connection is handled by the firewall itself or if it is routed or port-forwarded to a system behind the firewall.

Shorewall will set up the routing and will update the `/etc/iproute2/rt_tables` to include the table names and number of the tables that it adds.

Caution

This feature uses packet marking to control the routing. As a consequence, there are some restrictions concerning entries in `/etc/shorewall/tcrules`:

- Packet marking for traffic control purposes may not be done in the PREROUTING table for connections involving providers with 'track' specified (see below).
- You may not use the SAVE or RESTORE options.

- You may not use connection marking.

Use of this feature requires that your kernel and iptables support CONNMARK target and conntrack match support. It does NOT require the ROUTE target extension.

Warning

The current version of iptables (1.3.1) is broken with respect to CONNMARK and iptables-save/iptables-restore. This means that if you configure multiple ISPs, **shorewall restore** may fail. If it does, you may patch your iptables using the patch at http://shorewall.net/pub/shorewall/contrib/iptables/CONNMARK.diff.

The `/etc/shorewall/providers` file can also be used in other routing scenarios. See the Squid documentation for an example.

/etc/shorewall/providers File

Entries in this file have the following columns. As in all Shorewall configuration files, enter "-" in a column if you don't want to enter any value.

NAME

The provider name. Must begin with a letter and consist of letters and digits. The provider name becomes the name of the generated routing table for this provider.

NUMBER

A number between 1 and 252. This becomes the routing table number for the generated table for this provider.

MARK

A mark value used in your /etc/shorewall/tcrules file to direct packets to this provider. Shorewall will also mark connections that have seen input from this provider with this value and will restore the packet mark in the PREROUTING CHAIN.

DUPLICATE

Gives the name or number of a routing table to duplicate. May be 'main' or the name or number of a previously declared provider. For most applications, you want to specify 'main' here.

INTERFACE

The name of the interface to the provider.

GATEWAY

The IP address of the provider's Gateway router.

You can enter **detect** here and Shorewall will attempt to automatically determine the gateway IP address.

Hint: "detect" is appropriate for use in cases where the interface named in the INTERFACE column is dynamically configured via DHCP etc.

OPTIONS

A comma-separated list from the following:

track

If specified, connections FROM this interface are to be tracked so that responses may be routed back out this same interface.

You want specify 'track' if internet hosts will be connecting to local servers through this provider. Any time that you

specify 'track', you will also want to specify 'balance' (see below).

balance

The providers that have 'balance' specified will get outbound traffic load-balanced among them. Balancing will not be perfect, as it is route based, and routes are cached. This means that routes to often-used sites will always be over the same provider.

By default, each provider is given the same weight (1) . Beginning with 2.4.0-RC3, you can change the weight of a given provider by following *balance* with "=" and the desired weight (e.g., balance=2). The weights reflect the relative bandwidth of the providers connections and should be small numbers since the kernel actually creates additional default routes for each weight increment.

loose

Do not include routing rules that force traffic whose source IP is an address of the INTERFACE to be routed to this provider. Useful for defining providers that are to be used only when the appropriate packet mark is applied.

COPY

When you specify an existing table in the DUPLICATE column, Shorewall copies all routes through the interface specified in the INTERFACE column plus the interfaces listed in this column. At a minumum, you should list all interfaces on your firewall in this column except those internet interfaces specified in the INTERFACE column of entries in this file.

What an entry in the Providers File Does

Adding another entry in the providers file simply creates an alternate routing table for you. In addition:

1. Unless **loose** is specified, an ip rule is generated for each IP address on the INTERFACE that routes traffic from that address through the associated routing table.
2. If you specify **track**, then connections which have had at least one packet arrive on the interface listed in the INTERFACE column have their connection mark set to the value in the MARK column. In the PREROUTING chain, packets with that connmark have their packet mark set to that value; packets so marked then bypass any prerouting rules that you create in `/etc/shorewall/tcrules`. This ensures that packets associated with connections from outside are always routed out of the correct interface.
3. If you specify **balance**, then Shorewall will replace the 'default' route with weight 100 in the 'main' routing table with a load-balancing route among those gateways where **balance** was specified. So if you configure default routes, be sure that their weight is less than 100 or the route added by Shorewall will not be used.

That's **all** that these entries do. You still have to follow the principle stated at the top of this article:

1. Routing determines where packets are to be sent.
2. Once routing determines where the packet is to go, the firewall (Shorewall) determines if the packet is allowed to go there.

The bottom line is that if you want traffic to go out through a particular provider then you *must* mark that traffic with the provider's MARK value in `/etc/shorewall/tcrules` and you must do that marking in the PREROUTING chain.

Warning

Entries in `/etc/shorewall/providers` permanently alter your firewall/gateway's routing; that is, the effect of these changes is not reversed by **shorewall stop** or **shorewall clear**. To restore routing to its original state, you will have to restart your network. This can usually be done by **/etc/init.d/network restart** or **/etc/init.d/networking restart**. Check your distribution's networking documentation.

You can mitigate the effect of the Shorewall-generated changes to your routing table by specifying a *metric* for each default route that you configure. Shorewall will generate a load-balancing default route (assuming that **balance** has been specified for some of the providers) that does not include a metric and that will therefore not

replace any existing route that has a non-zero metric.

Example

The configuration in the figure at the top of this section would be specified in `/etc/shorewall/providers` as follows. Assume tht there is a single internal interface, `eth2`.

```
#NAME    NUMBER  MARK    DUPLICATE       INTERFACE       GATEWAY         OPTIONS
COPY
ISP1     1       1       main            eth0            206.124.146.254 track,balance
eth2
ISP2     2       2       main            eth1            130.252.99.254  track,balance
eth2
```

Other configuration files go something like this:

`/etc/shorewall/interfaces`:

```
#ZONE    INTERFACE    BROADCAST       OPTIONS
net      eth0         detect          …
net      eth1         detect          …
```

`/etc/shorewall/policy`:

```
#SOURCE    DESTINATION    POLICY     LIMIT:BURST
net        net            DROP
```

If you have masqueraded hosts, be sure to update `/etc/shorewall/masq` to masquerade to both ISPs. For example, if you masquerade all hosts connected to `eth2` then:

```
#INTERFACE       SUBNET            ADDRESS
eth0             eth2              206.124.146.176
eth1             eth2              130.252.99.27
```

Warning

Entries in `/etc/shorewall/masq` have no effect on which ISP a particular connection will be sent through. That is rather the purpuse of entries in `/etc/shorewall/tcrules`.

Now suppose that you want to route all outgoing SMTP traffic from your local network through ISP 2. You would make this entry in /etc/shorewall/tcrules (and if you are running a version of Shorewall earlier than 3.0.0, you would set TC_ENABLED=Yes in /etc/shorewall/shorewall.conf).

```
#MARK            SOURCE          DEST            PROTO   PORT(S) CLIENT  USER    TEST
#                                                                PORT(S)
2:P              <local network> 0.0.0.0/0       tcp     25
```

Extension Scripts and Common Actions

Tom Eastep

Copyright © 2001-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-10-31

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Extension scripts are user-provided scripts that are invoked at various points during firewall start, restart, stop and clear. The scripts are placed in /etc/shorewall and are processed using the Bourne shell "source" mechanism.

Caution

1. Be sure that you actually need to use an extension script to do what you want. Shorewall has a wide range of features that cover most requirements.
2. DO NOT SIMPLY COPY RULES THAT YOU FIND ON THE NET INTO AN EXTENSION SCRIPT AND EXPECT THEM TO WORK AND TO NOT BREAK SHOREWALL. TO USE SHOREWALL EXTENSION SCRIPTS YOU MUST KNOW WHAT YOU ARE DOING WITH RESPECT TO iptables/Netfilter AND SHOREWALL.

The following scripts can be supplied:

- init -- invoked early in "shorewall start" and "shorewall restart"
- initdone -- invoked after Shorewall has flushed all existing rules but before any rules have been added to the builtin chains.
- start -- invoked after the firewall has been started or restarted.
- started -- invoked as a first step when the firewall is being started
- stop -- invoked as a first step when the firewall is being stopped.
- stopped -- invoked after the firewall has been stopped.
- clear -- invoked after the firewall has been cleared.
- refresh -- invoked while the firewall is being refreshed but before the common and/or blacklst chains have been rebuilt.
- continue -- invoked to allow you to insert special rules to allow traffic while Shorewall is [re]starting. Any rules added in this script should be deleted in your *start* script. This script is invoked earlier in the [re]start process than is the *initdone* script described above.

If your version of Shorewall doesn't have the file that you want to use from the above list, you can simply create the file yourself. You can also supply a script with the same name as any of the filter chains in the firewall and the script will be invoked after the /etc/shorewall/rules file has been processed but before the /etc/shorewall/policy file has been

processed.

There are a couple of special considerations for commands in extension scripts:

- When you want to run **iptables**, use the command **run_iptables** instead. **run_iptables** will run the iptables utility passing the arguments to **run_iptables** and if the command fails, the firewall will be stopped.
- If you wish to generate a log message, use **log_rule_limit**. Parameters are:
  - Log Level
  - Chain to insert the rule into
  - Chain name to display in the message (this can be different from the preceding argument — see the Port Knocking article for an example of how to use this).
  - Disposition to report in the message (ACCEPT, DROP, etc)
  - Rate Limit (if passed as "" then $LOGLIMIT is assumed — see the LOGLIMIT option in /etc/shorewall/shorewall.conf)
  - Log Tag ("" if none)
  - Command (-A or -I for append or insert).
  - The remaining arguments are passed "as is" to iptables
- if you run commands other than **iptables** that must be re-run in order to restore the firewall to its current state then you must save the commands to the *restore file*. The restore file is a temporary file in `/var/lib/shorewall` that will be renamed `/var/lib/shorewall/restore-base` at the successful completion of the Shorewall command. The **shorewall save** command combines `/var/lib/shorewall/restore-base` with the output of **iptables-save** to produce the `/var/lib/shorewall/restore` script.

  Here are three functions that are useful when running commands other than **iptables**:
  1. **save_command()** -- saves the passed command to the restore file.

     Example:

     ```
     save_command echo Operation Complete
     ```

     That command would simply write "echo Operation Complete" to the restore file.
  2. **run_and_save_command()** -- saves the passed command to the restore file then executes it. The return value is the exit status of the command. Example:

     ```
     run_and_save_command "echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all"
     ```

     Note that as in this example, when the command involves file redirection then the entire command must be enclosed in quotes. This applies to all of the functions described here.
  3. **ensure_and_save_command()** -- runs the passed command. If the command fails, the firewall is restored to it's prior saved state and the operation is terminated. If the command succeeds, the command is written to the restore file
- Many of the extension scripts get executed for both the shorewall start and shorewall restart commands. You can determine which command is being executed using the contents of $COMMAND.

  ```
  if [ $COMMAND = start ]; then
     ...
  ```

You can also define a *common action* to be performed immediately before a policy of ACCEPT, DROP or REJECT is applied. Separate actions can be assigned to each policy type so for example you can have a different common action for DROP and REJECT policies. The most common usage of common actions is to silently drop traffic that you don't wish to have logged by the policy.

As released, Shorewall defines a number of actions which are cataloged in the `/usr/share/shorewall/actions.std` file. That file is processed before /etc/shorewall/actions. Among the entries in `/usr/share/shorewall/actions.std` are:

```
Drop:DROP
Reject:REJECT
```

So the action named “Drop” is performed immediately before DROP policies are applied and the action called “Reject” is performed before REJECT policies are applied. These actions are defined in the files `/usr/share/shorewall/action.Drop` and `/usr/share/shorewall/action.Reject` respectively.

You can override these defaults with entries in your /etc/shorewall/actions file. For example, if that file were to contain “MyDrop:DROP” then the common action for DROP policies would become “MyDrop”.

One final note. The chain created to perform an action has the same name as the action. You can use an extension script by that name to add rules to the action's chain in the same way as you can any other chain. So if you create the new action “Dagger” and define it in `/etc/shorewall/action.Dagger`, you can also have an extension script named `/etc/shorewall/Dagger` that can add rules to the “Dagger” chain that can't be created using `/etc/shorewall/action.Dagger`.

Port Knocking

Tom Eastep

Copyright © 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-09-12

Table of Contents

What is Port Knocking?
Implementing Port Knocking in Shorewall

What is Port Knocking?

Port knocking is a technique whereby attempting to connect to port A enables access to port B from that same host. For the example on which this article is based, see http://www.soloport.com/iptables.html which should be considered to be part of this documentation.

Implementing Port Knocking in Shorewall

In order to implement this solution, your iptables and kernel must support the 'recent match' extension (see FAQ 42). These instructions also assume Shorewall version 2.2.0 or later.

In this example:

1. Attempting to connect to port 1600 enables SSH access. Access is enabled for 60 seconds.
2. Attempting to connect to port 1601 disables SSH access (note that in the article linked above, attempting to connect to port 1599 also disables access. This is an port scan defence as explained in the article).

To implement that approach:

1. Add an action named SSHKnock (see the Action documentation). Leave the `action.SSHKnock` file empty.
2. Create /etc/shorewall/SSHKnock with the following contents:

```
if [ -n "$LEVEL" ]; then
    log_rule_limit $LEVEL $CHAIN SSHKnock ACCEPT "" "$TAG" -A -p tcp --dport 22   -m
recent --rcheck --name SSH
    log_rule_limit $LEVEL $CHAIN SSHKnock DROP   "" "$TAG" -A -p tcp --dport ! 22
fi
run_iptables -A $CHAIN -p tcp --dport 22   -m recent --rcheck --seconds 60 --name SSH
-j ACCEPT
run_iptables -A $CHAIN -p tcp --dport 1599 -m recent                      --name SSH
--remove -j DROP
run_iptables -A $CHAIN -p tcp --dport 1600 -m recent                      --name SSH
--set    -j DROP
run_iptables -A $CHAIN -p tcp --dport 1601 -m recent                      --name SSH
--remove -j DROP
```

3. Now if you want to protect SSH access to the firewall from the Internet, add this rule in `/etc/shorewall/rules`:

```
#ACTION          SOURCE            DEST            PROTO       DEST PORT(S)
SSHKnock         net               $FW             tcp         22,1599,1600,1601
```

If you want to log the DROPs and ACCEPTs done by SSHKnock, you can just add a log level as in:

```
#ACTION          SOURCE            DEST            PROTO       DEST PORT(S)
SSHKnock:info    net               $FW             tcp         22,1599,1600,1601
```

4. If you wish to use SSHKnock with a forwarded connection, you must be using Shorewall 2.3.1 or later for fullest protection. Assume that you forward port 22 from external IP address 206.124.146.178 to internal system 192.168.1.5. In /etc/shorewall/rules:

```
#ACTION          SOURCE            DEST             PROTO       DEST PORT(S)  SOURCE
ORIGINAL
#                                                                              PORT(S)
DEST
DNAT-            net               loc:192.168.1.5 tcp         22            -
206.124.146.178
SSHKnock         net               $FW              tcp         1599,1600,1601
SSHKnock         net               loc:192.168.1.5 tcp         22            -
206.124.146.178
```

Note

You can use SSHKnock with DNAT on earlier releases provided that you omit the ORIGINAL DEST entry on the second SSHKnock rule. This rule will be quite secure provided that you specify 'norfc1918' on your external interface.

Actions

Tom Eastep

Copyright © 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-11-02

Table of Contents

What are Shorewall Actions?
Common Actions
Defining your own Actions
Actions and Logging
Creating an Action using an Extension Script

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

What are Shorewall Actions?

Shorewall actions allow a symbolic name to be associated with a series of one or more iptables rules. The symbolic name may appear in the ACTION column of an `/etc/shorewall/rules` file entry in which case, the traffic matching that rules file entry will be passed to the series of iptables rules named by the action.

Actions can be thought of as templates. When an action is invoked in an `/etc/shorewall/rules` entry, it may be qualified by a logging specification (log level and optionally a log tag). The presence of the log level/tag causes a modified series of rules to be generated in which each packet/rule match within the action causes a log message to be generated.

There are three types of Shorewall actions:

1. Built-in Actions. These actions are known by the Shorewall code itself. They are listed in the comments at the top of the file `/usr/share/shorewall/actions.std`.
2. Standard Actions. These actions are released as part of Shorewall. They are listed in the file `/usr/share/shorewall/actions.std` and are defined in the corresponding action.* files in `/usr/share/shorewall`. Each `action.*` file has a comment at the beginning of the file that describes what the action does. As an example, here is the definition of the *AllowSMB* standard action.

```
#
# Shorewall 2.2 /usr/share/shorewall/action.AllowSMB
#
#       Allow Microsoft SMB traffic. You need to invoke this action in
#       both directions.
#
###############################################################################
#TARGET   SOURCE         DEST            PROTO   DEST    SOURCE          RATE    USER/
#                                                PORT    PORT(S)         LIMIT   GROUP
ACCEPT   -              -               udp     135,445
ACCEPT   -              -               udp     137:139
```

```
ACCEPT   -               -               udp     1024:   137
ACCEPT   -               -               tcp     135,139,445
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

If you wish to modify one of the standard actions, do not modify the definition in /usr/share/shorewall. Rather, copy the file to `/etc/shorewall` (or somewhere else on your CONFIG_PATH) and modify the copy.

3. User-defined Actions. These actions are created by end-users. They are listed in the file /etc/shorewall/actions and are defined in action.* files in /etc/shorewall/actions or in another directory listed in your CONFIG_PATH (defined in /etc/shorewall/shorewall.conf).

Common Actions

Shorewall allows the association of a *common action* with policies. A separate common action may be associated with ACCEPT, DROP and REJECT policies. Common actions provide a way to invoke a set of common rules just before the policy is enforced. Common actions accomplish two goals:

1. Relieve log congestion. Common actions typically include rules to silently drop or reject traffic that would otherwise be logged when the policy is enforced.
2. Ensure correct operation. Common actions can also avoid common pitfalls like dropping connection requests on port TCP port 113. If these connections are dropped (rather than rejected) then you may encounter problems connecting to internet services that utilize the AUTH protocol of client authentication[1].

Shorewall provides common actions for the REJECT and DROP policies. The common action for REJECT is named *Reject* and the common action for DROP is named *Drop*. These associations are made through two entries in /usr/share/shorewall/actions.std:

```
Drop:DROP       #Common Action for DROP policy
Reject:REJECT   #Common Action for REJECT policy
```

These may be overridden by entries in your /etc/shorewall/actions file.

Warning

Entries in the DROP and REJECT common actions **ARE NOT THE CAUSE OF CONNECTION PROBLEMS**. Remember — common actions are only invoked immediately before the packet is going to be dropped or rejected anyway!!!

Defining your own Actions

Before defining a new action, you should evaluate whether your goal can be best accomplished using an action or a *macro*. See this article for details.

To define a new action:

1. Add a line to `/etc/shorewall/actions` that names your new action. Action names must be valid shell variable names ((must begin with a letter and be composed of letters, digits and underscore characters) as well as valid Netfilter chain names. If you intend to log from the action, the name must have a maximum of 11 characters. It is recommended that the name you select for a new action begins with a capital letter; that way, the name won't conflict with a Shorewall-defined chain name.

   The name of the action may be optionally followed by a colon (“:”) and ACCEPT, DROP or REJECT. When this is done, the named action will become the *common action* for policies of type ACCEPT, DROP or REJECT respectively. The common action is applied immediately before the policy is enforced (before any logging is done under that policy) and is used mainly to suppress logging of uninteresting traffic which would otherwise clog your logs. The same policy name can appear in multiple actions; the last such action for each policy name is the one which Shorewall will use.

   Shorewall includes pre-defined actions for DROP and REJECT -- see above.
2. Once you have defined your new action name (ActionName), then copy /usr/share/shorewall/action.template to `/etc/shorewall/action.ActionName` (for example, if your new action name is “Foo” then copy

`/usr/share/shorewall/action.template` to `/etc/shorewall/action.Foo`).
3. Now modify the new file to define the new action.

Columns in the action.template file are as follows:

- TARGET - Must be ACCEPT, DROP, REJECT, LOG, CONTINUE, QUEUE or <*action*> where <*action*> is a previously-defined action (that is, it must precede the action being defined in this file in your `/etc/shorewall/actions` file). These actions have the same meaning as they do in the `/etc/shorewall/rules` file (CONTINUE terminates processing of the current action and returns to the point where that action was invoked). The TARGET may optionally be followed by a colon (":") and a syslog log level (e.g, REJECT:info or ACCEPT:debugging). This causes the packet to be logged at the specified level. You may also specify ULOG (must be in upper case) as a log level. This will log to the ULOG target for routing to a separate log through use of ulogd (http://www.gnumonks.org/projects/ulogd).

  You may also use a macro in your action provided that the macro's expansion only results in the ACTIONs ACCEPT, DROP, REJECT, LOG, CONTINUE, or QUEUE. See `/usr/share/shorewall/Drop` for an example of an action that users macros extensively.
- SOURCE - Source hosts to which the rule applies. A comma-separated list of subnets and/or hosts. Hosts may be specified by IP or MAC address; mac addresses must begin with "~" and must use "-" as a separator.

  Alternatively, clients may be specified by interface name. For example, eth1 specifies a client that communicates with the firewall system through eth1. This may be optionally followed by another colon (":") and an IP/MAC/subnet address as described above (e.g., eth1:192.168.1.5).
- DEST - Location of Server. Same as above with the exception that MAC addresses are not allowed.

  Unlike in the SOURCE column, you may specify a range of up to 256 IP addresses using the syntax <*first ip*>-<*last ip*>.
- PROTO - Protocol - Must be "tcp", "udp", "icmp", a number, or "all".
- DEST PORT(S) - Destination Ports. A comma-separated list of Port names (from `/etc/services`), port numbers or port ranges; if the protocol is "icmp", this column is interpreted as the destination icmp-type(s).

  A port range is expressed as <*low port*>:<*high port*>.

  This column is ignored if PROTOCOL = all but must be entered if any of the following fields are supplied. In that case, it is suggested that this field contain "-".

  If your kernel contains multi-port match support, then only a single Netfilter rule will be generated if in this list and in the CLIENT PORT(S) list below:
  1. There are 15 or less ports listed.
  2. No port ranges are included.

  Otherwise, a separate rule will be generated for each port.

- SOURCE PORT(S) - Port(s) used by the client. If omitted, any source port is acceptable. Specified as a comma-separated list of port names, port numbers or port ranges.

  If you don't want to restrict client ports but need to specify an ADDRESS in the next column, then place "-" in this column.

  If your kernel contains multi-port match support, then only a single Netfilter rule will be generated if in this list and in the DEST PORT(S) list above:
  1. There are 15 or less ports listed.
  2. No port ranges are included.

  Otherwise, a separate rule will be generated for each port.

- RATE LIMIT - You may rate-limit the rule by placing a value in this column:

  ```
  <rate>/<interval>[:<burst>]
  ```

  where <*rate*> is the number of connections per <*interval*> ("sec" or "min") and <*burst*> is the largest burst permitted. If no <*burst*> is given, a value of 5 is assumed. There may be no whitespace embedded in the specification.

```
     Example: 10/sec:20
```

- USER/GROUP - For output rules (those with the firewall as their source), you may control connections based on the effective UID and/or GID of the process requesting the connection. This column can contain any of the following:

  [!]<*user number*>[:]
  [!]<*user name*>[:]
  [!]:<*group number*>
  [!]:<*group name*>
  [!]<*user number*>:<*group number*>
  [!]<*user name*>:<*group number*>
  [!]<*user inumber*>:<*group name*>
  [!]<*user name*>:<*group name*>
  [!]+<*program name*> (Note: support for this form was removed from Netfilter in kernel version 2.6.14).

Omitted column entries should be entered using a dash ("-:).

Example:

`/etc/shorewall/actions`:

```
     LogAndAccept
```

`/etc/shorewall/action.LogAndAccept`

```
     LOG:info
     ACCEPT
```

To use your action, in `/etc/shorewall/rules` you might do something like:

```
#ACTION      SOURCE      DEST        PROTO    DEST PORT(S)
LogAndAccept loc         $FW         tcp      22
```

Actions and Logging

Specifying a log level in a rule that specifies a user- or Shorewall-defined action will cause each rule in the action to be logged with the specified level (and tag).

The extent to which logging of action rules occur is governed by the following:

1. When you invoke an action and specify a log level, only those rules in the action that have no log level will be changed to log at the level specified at the action invocation.

   Example:

   /etc/shorewall/action.foo

   ```
   #TARGET      SOURCE     DEST     PROTO    DEST PORT(S)
   ACCEPT       -          -        tcp      22
   bar:info
   ```

   /etc/shorewall/rules:

   ```
   #ACTION      SOURCE     DEST     PROTO    DEST PORT(S)
   foo:debug    $FW         net
   ```

Logging in the invoke 'foo' action will be as if foo had been defined as:

```
#TARGET       SOURCE      DEST      PROTO     DEST PORT(S)
ACCEPT:debug  -           -         tcp       22
bar:info
```

2. If you follow the log level with "!" then logging will be at that level for all rules recursively invoked by the action.

   Example:

   /etc/shorewall/action.foo

```
#TARGET       SOURCE      DEST      PROTO     DEST PORT(S)
ACCEPT        -           -         tcp       22
bar:info
```

   /etc/shorewall/rules:

```
#ACTION       SOURCE      DEST      PROTO     DEST PORT(S)
foo:debug!    $FW         net
```

   Logging in the invoke 'foo' action will be as if foo had been defined as:

```
#TARGET       SOURCE      DEST      PROTO     DEST PORT(S)
ACCEPT:debug  -           -         tcp       22
bar:debug
```

The change in Shorewall 2.1.2 has an effect on extension scripts used with user-defined actions. If you define an action 'acton' and you have an `/etc/shorewall/acton` script then when that script is invoked, the following three variables will be set for use by the script:

- $CHAIN = the name of the chain where your rules are to be placed. When logging is used on an action invocation, Shorewall creates a chain with a slightly different name from the action itself.
- $LEVEL = Log level. If empty, no logging was specified.
- $TAG = Log Tag.

Example:

`/etc/shorewall/rules`:

```
#ACTION           SOURCE            DEST
acton:info:test   $FW               net
```

Your /etc/shorewall/acton file will be run with:

- $CHAIN="%acton1"
- $LEVEL="info"
- $TAG="test"

For an example of how to use these variables, see this article.

Creating an Action using an Extension Script

There may be cases where you wish to create a chain with rules that can't be constructed using the tools defined in the action.template. In that case, you can use an extension script.

Note

If you actually need an action to drop broadcast packets, use the **dropBcast** standard action rather than create one like this.

Example 1. An action to drop all broadcast packets

/etc/shorewall/actions

```
DropBcasts
```

/etc/shorewall/action.DropBcasts

```
# This file is empty
```

/etc/shorewall/DropBcasts

```
run_iptables -A DropBcasts -m pkttype --pkttype broadcast -j DROP
```

For a richer example, see this article.

---

[1] AUTH is actually pretty silly on today's internet but it's amazing how many servers still employ it.

Macros

Tom Eastep

Cristian Rodríguez

Copyright © 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-11-18

Table of Contents

Overview of Shorewall Macros?
Defining your own Macros
Macros and Logging
How do I know if I should create an Action or a Macro?

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Overview of Shorewall Macros?

Shorewall macros allow a symbolic name to be associated with a series of one or more iptables rules. The symbolic name may appear in the ACTION column of an `/etc/shorewall/rules` file entry and in the TARGET column of an action in which case, the traffic matching that rules file entry will be passed to the series of iptables rules named by the macro.

Macros can be thought of as templates. When a macro is invoked in an `/etc/shorewall/rules` entry, it may be qualified by a logging specification (log level and optionally a log tag). The presence of the log level/tag causes a modified series of rules to be generated in which each packet/rule match within the macro causes a log message to be generated.

There are two types of Shorewall macros:

1. Standard Macros. These macros are released as part of Shorewall. They are defined in macros.* files in `/usr/share/shorewall`. Each `macros.*` file has a comment at the beginning of the file that describes what the macro does. As an example, here is the definition of the *SMB* standard macro.

```
#
# Shorewall 3.0 /usr/share/shorewall/macro.SMB
#
#       Handle Microsoft SMB traffic. You need to invoke this macro in
#       both directions.
#
###############################################################################
#TARGET  SOURCE         DEST            PROTO   DEST    SOURCE          RATE    USER/
#                                               PORT    PORT(S)         LIMIT   GROUP
PARAM    -              -               udp     135,445
PARAM    -              -               udp     137:139
PARAM    -              -               udp     1024:   137
```

```
PARAM    -               -                tcp     135,139,445
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

If you wish to modify one of the standard macros, do not modify the definition in /usr/share/shorewall. Rather, copy the file to `/etc/shorewall` (or somewhere else on your CONFIG_PATH) and modify the copy.

2. User-defined Macros. These macros are created by end-users. They are defined in macros.* files in /etc/shorewall or in another directory listed in your CONFIG_PATH (defined in /etc/shorewall/shorewall.conf).

Most Standard Macros are *parameterized*. That means that you specify what you want to do (ACCEPT, DROP, REJECT, etc.) when you invoke the macro. The SMB macro shown above is parameterized (note PARAM in the TARGET column). When invoking a parameterized macro, you follow the name of the macro with a slash ("/") and the action that you want to substitute for PARAM.

Example:

/etc/shorewall/rules:

```
#ACTION      SOURCE         DEST          PROTO   DEST PORT(S)
SMB/ACCEPT   loc            fw
```

The above is equivalent to coding the following series of rules:

```
#TARGET      SOURCE         DEST          PROTO   DEST PORT(s)
ACCEPT       loc            fw            udp     135,445
ACCEPT       loc            fw            udp     137:139
ACCEPT       loc            fw            udp     1024:   137
ACCEPT       loc            fw            tcp     135,139,445
```

Logging is covered in a following section. The other columns are treated as follows:

SOURCE and DEST

If a value other than "-" appears in both the macro body and in the invocation of the macro, then the value in the invocation is examined and the appropriate action is taken. If the value in the invocation appears to be an address (IP or MAC) or the name of an ipset, then it is placed after the value in the macro body. Otherwise, it is placed before the value in the macro body.

Example 1:

/etc/shorewall/macro.SMTP

```
#TARGET          SOURCE   DEST            PROTO    DEST PORT(S)
PARAM            -        loc             tcp      25
```

/etc/shorewall/rules

```
#ACTION          SOURCE   DEST            PROTO    DEST PORT(S)
SMTP/DNAT:info   net      192.168.1.5
```

This would be equivalent to coding the following directly in /etc/shorewall/rules

```
#ACTION          SOURCE   DEST            PROTO    DEST PORT(S)
DNAT:info        net      loc:192.168.1.5 tcp      25
```

Example 2:

/etc/shorewall/macro.SMTP

```
#TARGET          SOURCE   DEST            PROTO    DEST PORT(S)
PARAM            -        192.168.1.5     tcp      25
```

/etc/shorewall/rules

```
#ACTION          SOURCE   DEST            PROTO    DEST PORT(S)
SMTP/DNAT:info   net      loc
```

This would be equivalent to coding the following directly in /etc/shorewall/rules

```
#ACTION          SOURCE   DEST            PROTO    DEST PORT(S)
DNAT:info        net      loc:192.168.1.5 tcp      25
```

Remaining columns

Any value in the invocation replaces the value in the rule in the macro.

One remaining restriction must be mentioned: macros that are invoked from actions cannot themselves invoke other actions.

Defining your own Macros

To define a new macro:

1. Macro names must be valid shell variable names ((must begin with a letter and be composed of letters, digits and underscore characters) as well as valid Netfilter chain names.
2. Copy /usr/share/shorewall/macro.template to `/etc/shorewall/macro.MacroName` (for example, if your new macro name is “Foo” then copy `/usr/share/shorewall/macro.template` to `/etc/shorewall/macro.Foo`).
3. Now modify the new file to define the new macro.

Columns in the macro.template file are as follows:

- ACTION - ACCEPT, DROP, REJECT, DNAT, DNAT-, REDIRECT, CONTINUE, LOG, QUEUE, PARAM or an action name. Note that a macro may not invoke another macro.

  ACCEPT - allow the connection request

  ACCEPT+ - like ACCEPT but also excludes the connection from any subsequent DNAT[-] or REDIRECT[-] rules.

  NONAT - Excludes the connection from any subsequent DNAT[-] or REDIRECT[-] rules but doesn't generate a rule to accept the traffic.

  DROP - ignore the request

  REJECT - disallow the request and return an icmp unreachable or an RST packet.

  DNAT - Forward the request to another address (and optionally another port).

  DNAT- - Advanced users only. Like DNAT but only generates the DNAT iptables rule and not the companion ACCEPT rule.

  SAME - Similar to DNAT except that the port may not be remapped and when multiple server addresses are listed, all requests from a given remote system go to the same server.

  SAME- - Advanced users only. Like SAME but only generates the SAME iptables rule and not the companion ACCEPT rule.

  REDIRECT - Redirect the request to a local port on the firewall.

  REDIRECT- - Advanced users only. Like REDIRECT but only generates the REDIRECT iptables rule and not the companion ACCEPT rule.

  CONTINUE - (For experts only). Do not process any of the following rules for this (source zone,destination zone). If The source and/or destination If the address falls into a zone defined later in /etc/shorewall/zones, this connection request will be passed to the rules defined for that (those) zone(s).

  LOG - Simply log the packet and continue.

  QUEUE - Queue the packet to a user-space application such as ftwall (http://p2pwall.sf.net).

  The ACTION may optionally be followed by ":" and a syslog log level (e.g, REJECT:info or DNAT:debug). This causes the

packet to be logged at the specified level.

- SOURCE - Source hosts to which the rule applies. A comma-separated list of subnets and/or hosts. Hosts may be specified by IP or MAC address; mac addresses must begin with “~” and must use “-” as a separator.

  Alternatively, clients may be specified by interface name. For example, eth1 specifies a client that communicates with the firewall system through eth1. This may be optionally followed by another colon (“:”) and an IP/MAC/subnet address as described above (e.g. eth1:192.168.1.5).
- DEST - Location of Server. Same as above with the exception that MAC addresses are not allowed.

  Unlike in the SOURCE column, you may specify a range of up to 256 IP addresses using the syntax *<first ip>-<last ip>*.
- PROTO - Protocol - Must be “tcp”, “udp”, “icmp”, a number, or “all”.
- DEST PORT(S) - Destination Ports. A comma-separated list of Port names (from `/etc/services`), port numbers or port ranges; if the protocol is “icmp”, this column is interpreted as the destination icmp-type(s).

  A port range is expressed as *<low port>*:*<high port>*.

  This column is ignored if PROTOCOL = all but must be entered if any of the following fields are supplied. In that case, it is suggested that this field contain “-”.

  If your kernel contains multi-port match support, then only a single Netfilter rule will be generated if in this list and in the CLIENT PORT(S) list below:
  1. There are 15 or less ports listed.
  2. No port ranges are included.

  Otherwise, a separate rule will be generated for each port.

- SOURCE PORT(S) - Port(s) used by the client. If omitted, any source port is acceptable. Specified as a comma-separated list of port names, port numbers or port ranges.

  If you don't want to restrict client ports but need to specify an ADDRESS in the next column, then place "-" in this column.

  If your kernel contains multi-port match support, then only a single Netfilter rule will be generated if in this list and in the DEST PORT(S) list above:
  1. There are 15 or less ports listed.
  2. No port ranges are included.

  Otherwise, a separate rule will be generated for each port.

- RATE LIMIT - You may rate-limit the rule by placing a value in this column:

```
<rate>/<interval>[:<burst>]
```

  where *<rate>* is the number of connections per *<interval>* (“sec” or “min”) and *<burst>* is the largest burst permitted. If no *<burst>* is given, a value of 5 is assumed. There may be no whitespace embedded in the specification.

```
Example: 10/sec:20
```

- USER/GROUP - For output rules (those with the firewall as their source), you may control connections based on the effective UID and/or GID of the process requesting the connection. This column can contain any of the following:

  [!]*<user number>*[:]

  [!]*<user name>*[:]

  [!]:*<group number>*

  [!]:*<group name>*

  [!]*<user number>*:*<group number>*

  [!]*<user name>*:*<group number>*

  [!]*<user inumber>*:*<group name>*

  [!]*<user name>*:*<group name>*

[!]+<*program name*> (Note: support for this form was removed from Netfilter in kernel version 2.6.14).

Omitted column entries should be entered using a dash ("-:).

Example:

`/etc/shorewall/macro.LogAndAccept`

```
     LOG:info
     ACCEPT
```

To use your macro, in `/etc/shorewall/rules` you might do something like:

```
#ACTION      SOURCE      DEST        PROTO    DEST PORT(S)
LogAndAccept loc         $FW         tcp      22
```

Macros and Logging

Specifying a log level in a rule that invokes a user- or Shorewall-defined action will cause each rule in the macro to be logged with the specified level (and tag).

The extent to which logging of macro rules occur is governed by the following:

1. When you invoke a macro and specify a log level, only those rules in the macro that have no log level will be changed to log at the level specified at the action invocation.

   Example:

   /etc/shorewall/macro.foo

   ```
   #ACTION      SOURCE     DEST     PROTO    DEST PORT(S)
   ACCEPT       -          -        tcp      22
   bar:info
   ```

   /etc/shorewall/rules:

   ```
   #ACTION      SOURCE     DEST     PROTO    DEST PORT(S)
   foo:debug    $FW        net
   ```

   Logging in the invoked 'foo' macro will be as if foo had been defined as:

   ```
   #ACTION      SOURCE     DEST     PROTO    DEST PORT(S)
   ACCEPT:debug -          -        tcp      22
   bar:info
   ```
2. If you follow the log level with "!" then logging will be at that level for all rules recursively invoked by the macro.

   Example:

   /etc/shorewall/macro.foo

   ```
   #ACTION      SOURCE     DEST     PROTO    DEST PORT(S)
   ACCEPT       -          -        tcp      22
   bar:info
   ```

   /etc/shorewall/rules:

```
#ACTION       SOURCE      DEST      PROTO     DEST PORT(S)
foo:debug!    $FW         net
```

Logging in the invoked 'foo' macro will be as if foo had been defined as:

```
#ACTION       SOURCE      DEST      PROTO     DEST PORT(S)
ACCEPT:debug  -           -         tcp       22
bar:debug
```

How do I know if I should create an Action or a Macro?

While actions and macros perform similar functions, in any given case you will generally find that one is more appropriate than the other.

1. You can not associate an Extension Script with a macro the way that you can with an Action. So if you need access to iptables features not directly supported by Shorewall then you must use an action.
2. Macros are expanded in-line while each action is it's own chain. So if there are a lot of rules involved in your new action/macro then it is generally better to use an action than a macro. Only the packets selected when you invoke the action are directed to the corresponding chain. On the other hand, if there are only one or two rules involved in what you want to do then a macro is more efficient.

Shorewall Requirements

Tom Eastep

Copyright © 2001-2005 Thomas M Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-09-14

Table of Contents

Shorewall Requires:

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Shorewall Requires:

- A **Linux** kernel that supports netfilter (No, it won't work on BSD or Solaris). I've tested with 2.4.2 - 2.6.13. Check here for kernel configuration information.
- iptables 1.2 or later (but I recommend at least version 1.2.9)
- Iproute ("ip" and "tc" utilities). The iproute package is included with most distributions but may not be installed by default. The official download site is http://developer.osdl.org/dev/iproute2/download/.
- A Bourne shell or derivative such as bash or ash. This shell must have correct support for variable expansion formats ${*variable%pattern*}, ${*variable%%pattern*}, ${*variable#pattern*} and ${*variable##pattern*}.
- Your shell must produce a sensible result when a number n (128 <= n <= 255) is left shifted by 24 bits. You can check this at a shell prompt by:
  - echo $((128 << 24))
  - The result must be either 2147483648 or -2147483648.
- The firewall monitoring display is greatly improved if you have awk (gawk) installed.

Kernel Configuration

Tom Eastep

Copyright © 2001-2004 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2004-11-01

Table of Contents

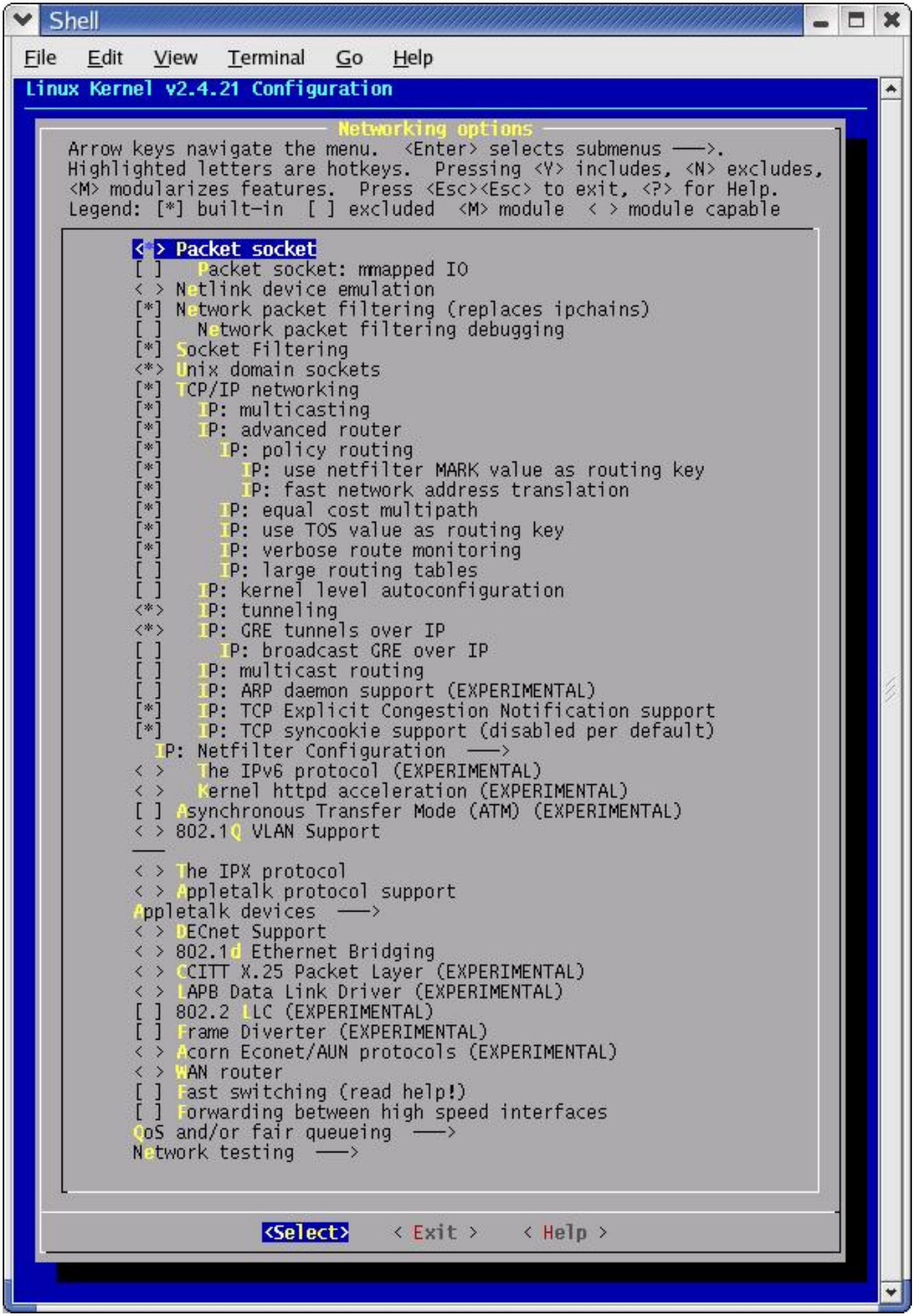
Network Options Configuration
Netfilter Configuration
Kernel 2.6 Netfilter Options

Note

For information regarding configuring and building GNU/Linux kernels, see http://www.kernelnewbies.org.

Network Options Configuration

Here's a screen shot of my Network Options Configuration:



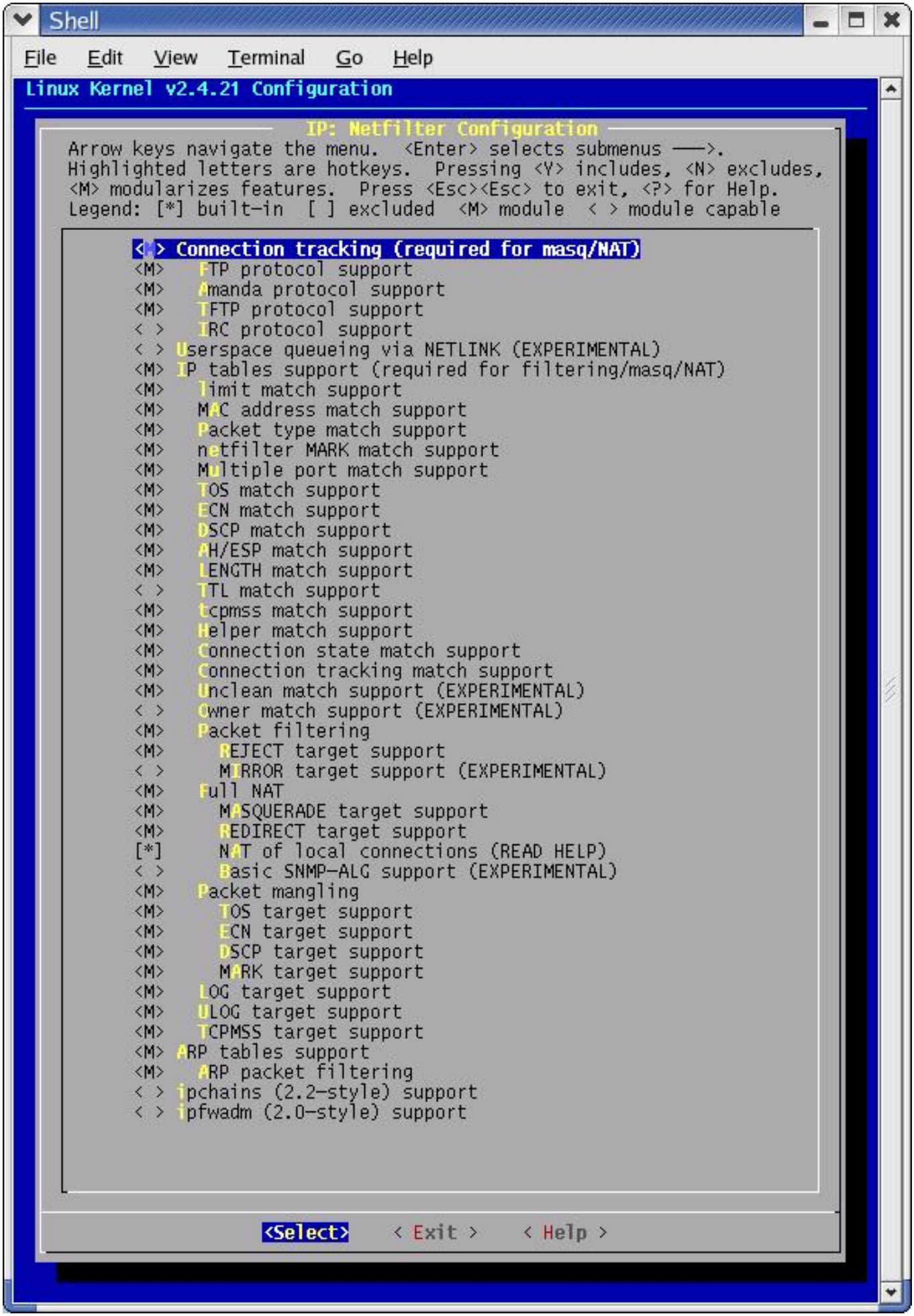
While not all of the options that I've selected are required, they should be sufficient for most applications. Here's an excerpt from the corresponding .config file (Note: If you are running a kernel older than 2.4.17, be sure to select

CONFIG_NETLINK and CONFIG_RTNETLINK):

```
#
 # Networking options
 #
 CONFIG_PACKET=y
 # CONFIG_PACKET_MMAP is not set
 # CONFIG_NETLINK_DEV is not set
 CONFIG_NETFILTER=y
 # CONFIG_NETFILTER_DEBUG is not set
 CONFIG_FILTER=y
 CONFIG_UNIX=y
 CONFIG_INET=y
 CONFIG_IP_MULTICAST=y
 CONFIG_IP_ADVANCED_ROUTER=y
 CONFIG_IP_MULTIPLE_TABLES=y
 CONFIG_IP_ROUTE_FWMARK=y
 CONFIG_IP_ROUTE_NAT=y
 CONFIG_IP_ROUTE_MULTIPATH=y
 CONFIG_IP_ROUTE_TOS=y
 CONFIG_IP_ROUTE_VERBOSE=y
 # CONFIG_IP_ROUTE_LARGE_TABLES is not set
 # CONFIG_IP_PNP is not set
 CONFIG_NET_IPIP=y
 CONFIG_NET_IPGRE=y
 # CONFIG_NET_IPGRE_BROADCAST is not set
 # CONFIG_IP_MROUTE is not set
 # CONFIG_ARPD is not set
 CONFIG_INET_ECN=y
 CONFIG_SYN_COOKIES=y
```

Netfilter Configuration

Here's a screen shot of my Netfilter configuration:



Note that I have built everything I need as modules. You can also build everything into your kernel but if you want to be able to deal with FTP running on a non-standard port then you **must** modularize FTP Protocol support.

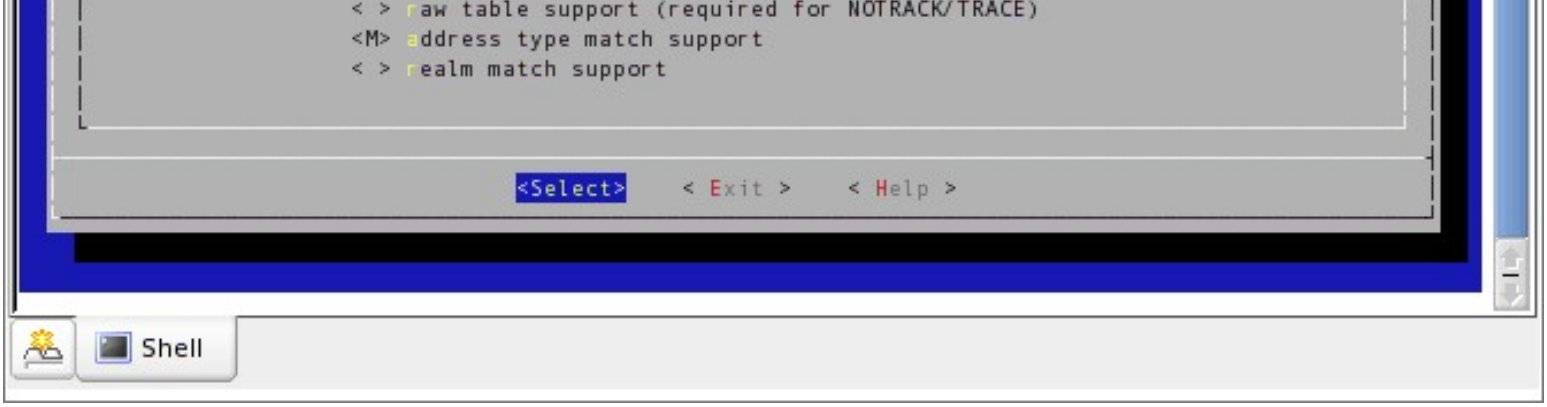
Here's the corresponding part of my .config file:

```
#
#   IP: Netfilter Configuration
#
CONFIG_IP_NF_CONNTRACK=m
CONFIG_IP_NF_FTP=m
CONFIG_IP_NF_AMANDA=m
CONFIG_IP_NF_TFTP=m
# CONFIG_IP_NF_IRC is not set
# CONFIG_IP_NF_QUEUE is not set
CONFIG_IP_NF_IPTABLES=m
CONFIG_IP_NF_MATCH_LIMIT=m
CONFIG_IP_NF_MATCH_MAC=m
CONFIG_IP_NF_MATCH_PKTTYPE=m
CONFIG_IP_NF_MATCH_MARK=m
CONFIG_IP_NF_MATCH_MULTIPORT=m
CONFIG_IP_NF_MATCH_TOS=m
CONFIG_IP_NF_MATCH_ECN=m
CONFIG_IP_NF_MATCH_DSCP=m
CONFIG_IP_NF_MATCH_AH_ESP=m
CONFIG_IP_NF_MATCH_LENGTH=m
# CONFIG_IP_NF_MATCH_TTL is not set
CONFIG_IP_NF_MATCH_TCPMSS=m
CONFIG_IP_NF_MATCH_HELPER=m
CONFIG_IP_NF_MATCH_STATE=m
CONFIG_IP_NF_MATCH_CONNTRACK=m
CONFIG_IP_NF_MATCH_UNCLEAN=m
# CONFIG_IP_NF_MATCH_OWNER is not set
CONFIG_IP_NF_FILTER=m
CONFIG_IP_NF_TARGET_REJECT=m
# CONFIG_IP_NF_TARGET_MIRROR is not set
CONFIG_IP_NF_NAT=m
CONFIG_IP_NF_NAT_NEEDED=y
CONFIG_IP_NF_TARGET_MASQUERADE=m
CONFIG_IP_NF_TARGET_REDIRECT=m
CONFIG_IP_NF_NAT_AMANDA=m
CONFIG_IP_NF_NAT_LOCAL=y
# CONFIG_IP_NF_NAT_SNMP_BASIC is not set
CONFIG_IP_NF_NAT_FTP=m
CONFIG_IP_NF_NAT_TFTP=m
CONFIG_IP_NF_MANGLE=m
CONFIG_IP_NF_TARGET_TOS=m
CONFIG_IP_NF_TARGET_ECN=m
CONFIG_IP_NF_TARGET_DSCP=m
CONFIG_IP_NF_TARGET_MARK=m
CONFIG_IP_NF_TARGET_LOG=m
CONFIG_IP_NF_TARGET_ULOG=m
CONFIG_IP_NF_TARGET_TCPMSS=m
CONFIG_IP_NF_ARPTABLES=m
CONFIG_IP_NF_ARPFILTER=m
# CONFIG_IP_NF_COMPAT_IPCHAINS is not set
# CONFIG_IP_NF_COMPAT_IPFWADM is not set
```

Kernel 2.6 Netfilter Options

Here's a screenshot of my modularized 2.6 Kernel config (Navigation: Device Drivers → Networking Support → Networking Options → Network Packet Filtering (replaces ipchains) → IP: Netfilter configuration):

```
Shell - Konsole
Session  Edit  View  Bookmarks  Settings  Help

Linux Kernel v2.6.8 Configuration

                    IP: Netfilter Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are hotkeys.
  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?>
  for Help.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                  <M> Connection tracking (required for masq/NAT)
                  <M>   FTP protocol support
                  <M>   IRC protocol support
                  <M>   TFTP protocol support
                  <M>   Amanda backup protocol support
                  <M> Userspace queueing via NETLINK
                  <M> IP tables support (required for filtering/masq/NAT)
                  <M>   limit match support
                  <M>   IP range match support
                  <M>   MAC address match support
                  <M>   Packet type match support
                  <M>   netfilter MARK match support
                  <M>   Multiple port match support
                  <M>   TOS match support
                  <M>   recent match support
                  <M>   ECN match support
                  <M>   DSCP match support
                  <M>   AH/ESP match support
                  <M>   LENGTH match support
                  <M>   TTL match support
                  <M>   tcpmss match support
                  <M>   Helper match support
                  <M>   Connection state match support
                  <M>   Connection tracking match support
                  <M>   Owner match support
                  <M>   Physdev match support
                  <M>   Packet filtering
                  <M>     REJECT target support
                  <M>   Full NAT
                  <M> MASQUERADE target support
                  <M> REDIRECT target support
                  <M> NETMAP target support
                  <M> SAME target support
                  [*] NAT of local connections (READ HELP)
                  <M> Basic SNMP-ALG support (EXPERIMENTAL)
                  <M> Packet mangling
                  <M>   TOS target support
                  <M>   ECN target support
                  <M>   DSCP target support
                  <M>   MARK target support
                  <M>   CLASSIFY target support
                  <M> LOG target support
                  <M> ULOG target support
                  <M> TCPMSS target support
                  < > ARP tables support
                  < > ipchains (2.2-style) support
                  < >   ipfwadm (2.0-style) support
                  < > raw table support (required for NOTRACK/TRACE)
                  <M> address type match support
                  < > realm match support
```



Here is the corresponding part of the .config file:

```
CONFIG_IP_NF_CONNTRACK=m
CONFIG_IP_NF_FTP=m
CONFIG_IP_NF_IRC=m
CONFIG_IP_NF_TFTP=m
CONFIG_IP_NF_AMANDA=m
CONFIG_IP_NF_QUEUE=m
CONFIG_IP_NF_IPTABLES=m
CONFIG_IP_NF_MATCH_LIMIT=m
CONFIG_IP_NF_MATCH_IPRANGE=m
CONFIG_IP_NF_MATCH_MAC=m
CONFIG_IP_NF_MATCH_PKTTYPE=m
CONFIG_IP_NF_MATCH_MARK=m
CONFIG_IP_NF_MATCH_MULTIPORT=m
CONFIG_IP_NF_MATCH_TOS=m
CONFIG_IP_NF_MATCH_RECENT=m
CONFIG_IP_NF_MATCH_ECN=m
CONFIG_IP_NF_MATCH_DSCP=m
CONFIG_IP_NF_MATCH_AH_ESP=m
CONFIG_IP_NF_MATCH_LENGTH=m
CONFIG_IP_NF_MATCH_TTL=m
CONFIG_IP_NF_MATCH_TCPMSS=m
CONFIG_IP_NF_MATCH_HELPER=m
CONFIG_IP_NF_MATCH_STATE=m
CONFIG_IP_NF_MATCH_CONNTRACK=m
CONFIG_IP_NF_MATCH_OWNER=m
CONFIG_IP_NF_MATCH_PHYSDEV=m
CONFIG_IP_NF_FILTER=m
CONFIG_IP_NF_TARGET_REJECT=m
CONFIG_IP_NF_NAT=m
CONFIG_IP_NF_NAT_NEEDED=y
CONFIG_IP_NF_TARGET_MASQUERADE=m
CONFIG_IP_NF_TARGET_REDIRECT=m
CONFIG_IP_NF_TARGET_NETMAP=m
CONFIG_IP_NF_TARGET_SAME=m
CONFIG_IP_NF_NAT_LOCAL=y
CONFIG_IP_NF_NAT_SNMP_BASIC=m
CONFIG_IP_NF_NAT_IRC=m
CONFIG_IP_NF_NAT_FTP=m
CONFIG_IP_NF_NAT_TFTP=m
CONFIG_IP_NF_NAT_AMANDA=m
CONFIG_IP_NF_MANGLE=m
CONFIG_IP_NF_TARGET_TOS=m
```

```
CONFIG_IP_NF_TARGET_ECN=m
CONFIG_IP_NF_TARGET_DSCP=m
CONFIG_IP_NF_TARGET_MARK=m
CONFIG_IP_NF_TARGET_CLASSIFY=m
CONFIG_IP_NF_TARGET_LOG=m
CONFIG_IP_NF_TARGET_ULOG=m
CONFIG_IP_NF_TARGET_TCPMSS=m
# CONFIG_IP_NF_ARPTABLES is not set
# CONFIG_IP_NF_COMPAT_IPCHAINS is not set
# CONFIG_IP_NF_COMPAT_IPFWADM is not set
# CONFIG_IP_NF_RAW is not set
CONFIG_IP_NF_MATCH_ADDRTYPE=m
# CONFIG_IP_NF_MATCH_REALM is not set
CONFIG_IP6_NF_QUEUE=m
CONFIG_IP6_NF_IPTABLES=m
CONFIG_IP6_NF_MATCH_LIMIT=m
CONFIG_IP6_NF_MATCH_MAC=m
CONFIG_IP6_NF_MATCH_RT=m
CONFIG_IP6_NF_MATCH_OPTS=m
CONFIG_IP6_NF_MATCH_FRAG=m
CONFIG_IP6_NF_MATCH_HL=m
CONFIG_IP6_NF_MATCH_MULTIPORT=m
CONFIG_IP6_NF_MATCH_OWNER=m
CONFIG_IP6_NF_MATCH_MARK=m
CONFIG_IP6_NF_MATCH_IPV6HEADER=m
CONFIG_IP6_NF_MATCH_AHESP=m
CONFIG_IP6_NF_MATCH_LENGTH=m
CONFIG_IP6_NF_MATCH_EUI64=m
CONFIG_IP6_NF_FILTER=m
CONFIG_IP6_NF_TARGET_LOG=m
CONFIG_IP6_NF_MANGLE=m
CONFIG_IP6_NF_TARGET_MARK=m
# CONFIG_IP6_NF_RAW is not set
CONFIG_DECNET_NF_GRABULATOR=m
CONFIG_BRIDGE_NF_EBTABLES=m
```

Shorewall Features

Tom Eastep

Copyright © 2001-2004 Thomas M Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-05-27

Table of Contents

Features

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Features

- Uses Netfilter's connection tracking facilities for stateful packet filtering.
- Can be used in **a wide range of router/firewall/gateway applications** .
  - Completely customizable using configuration files.
  - No limit on the number of network interfaces.
  - Allows you to partition the network into zones and gives you complete control over the connections permitted between each pair of zones.
  - Multiple interfaces per zone and multiple zones per interface permitted.
  - Supports nested and overlapping zones.
- QuickStart Guides (HOWTOs) to help get your first firewall up and running quickly
- A **GUI** is available via Webmin 1.060 and later (http://www.webmin.com)
- Extensive **documentation** in available in both XML and HTML formats.
- **Flexible address management/routing support** (and you can use all types in the same firewall):
  - Masquerading/SNAT.

    - Port Forwarding (DNAT).
    - One-to-one NAT.
    - Proxy ARP.
    - NETMAP (requires a 2.6 kernel or a patched 2.4 kernel).
    - Multiple ISP support
- **Blacklisting** of individual IP addresses and subnetworks is supported.
- Operational Support.
    - Commands to start, stop and clear the firewall
    - Supports status monitoring with an audible alarm when an “interesting” packet is detected.
    - Wide variety of informational commands.
- **VPN Support**.
    - IPSEC, GRE, IPIP and OpenVPN Tunnels.
    - PPTP clients and Servers.
- Support for **Traffic** Control/**Shaping**.
- Wide support for different **GNU/Linux Distributions**.
    - RPM and Debian packages available.
    - Includes automated install, upgrade, fallback and uninstall facilities for users who can't use or choose not to use the RPM or Debian packages.
    - Included as a standard part of LEAF/Bering (router/firewall on a floppy, CD or compact flash).
- Media Access Control (**MAC**) Address **Verification**.
- **Traffic Accounting.**
- **Bridge**/Firewall support (requires a 2.6 kernel or a patched 2.4 kernel).

Configuration Files

Tom Eastep

Copyright © 2001-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-10-20

Table of Contents

Files
Comments
Line Continuation
INCLUDE Directive
Using DNS Names
Comma-separated Lists
Complementing an Address or Subnet
Exclusion Lists
IP Address Ranges
Port Numbers/Service Names
Port Ranges
Port Lists
Using Shell Variables
Using MAC Addresses
Shorewall Configurations

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Caution

If you copy or edit your configuration files on a system running Microsoft Windows, you must run them through dos2unix before you use them with Shorewall.

Files

- `/etc/shorewall/shorewall.conf` - used to set several firewall parameters.
- `/etc/shorewall/params` - use this file to set shell variables that you will expand in other files.
- `/etc/shorewall/zones` - partition the firewall's view of the world into zones.
- `/etc/shorewall/policy` - establishes firewall high-level policy.
- `/etc/shorewall/interfaces` - describes the interfaces on the firewall system.
- `/etc/shorewall/hosts` - allows defining zones in terms of individual hosts and subnetworks.
- `/etc/shorewall/masq` - directs the firewall where to use many-to-one (dynamic) Network Address Translation

(a.k.a. Masquerading) and Source Network Address Translation (SNAT).

- `/etc/shorewall/modules` - directs the firewall to load kernel modules.
- `/etc/shorewall/rules` - defines rules that are exceptions to the overall policies established in /etc/shorewall/policy.
- `/etc/shorewall/nat` - defines one-to-one NAT rules.
- `/etc/shorewall/proxyarp` - defines use of Proxy ARP.
- `/etc/shorewall/routestopped` - defines hosts accessible when Shorewall is stopped.
- `/etc/shorewall/tcrules` - defines marking of packets for later use by traffic control/shaping or policy routing.
- `/etc/shorewall/tos` - defines rules for setting the TOS field in packet headers.
- `/etc/shorewall/tunnels` - defines tunnels (VPN) with end-points on the firewall system.
- `/etc/shorewall/blacklist` - lists blacklisted IP/subnet/MAC addresses.
- `/etc/shorewall/init` - commands that you wish to execute at the beginning of a “shorewall start” or “shorewall restart”.
- `/etc/shorewall/start` - commands that you wish to execute at the completion of a “shorewall start” or “shorewall restart”
- `/etc/shorewall/stop` - commands that you wish to execute at the beginning of a “shorewall stop”.
- `/etc/shorewall/stopped` - commands that you wish to execute at the completion of a “shorewall stop”.
- `/etc/shorewall/ecn` - disable Explicit Congestion Notification (ECN - RFC 3168) to remote hosts or networks.
- `/etc/shorewall/accounting` - define IP traffic accounting rules
- `/etc/shorewall/actions` and `/usr/share/shorewall/action.template` allow user-defined actions.
- `/etc/shorewall/providers` - defines an alternate routing table.
- `/usr/share/shorewall/actions.std` - Actions defined by Shorewall.
- `/usr/share/shorewall/action.*` - Details of actions defined by Shorewall.
- `/usr/share/shorewall/macro.*` - Details of macros defined by Shorewall.
- `/usr/share/rfc1918` — Defines the behavior of the 'norfc1918' interface option in `/etc/shorewall/interfaces`. **If you need to change this file, copy it to `/etc/shorewall` and modify the copy**.

Comments

You may place comments in configuration files by making the first non-whitespace character a pound sign (“#”). You may also place comments at the end of any line, again by delimiting the comment from the rest of the line with a pound sign.

Example 1. Comments in a Configuration File

```
# This is a comment
ACCEPT  net     $FW      tcp     www     #This is an end-of-line comment
```

Line Continuation

You may continue lines in the configuration files using the usual backslash (“\”) followed immediately by a new line character (Enter key).

Example 2. Line Continuation

```
ACCEPT  net     $FW      tcp \↵
smtp,www,pop3,imap  #Services running on the firewall
```

INCLUDE Directive

Any file may contain INCLUDE directives. An INCLUDE directive consists of the word INCLUDE followed by a path name and causes the contents of the named file to be logically included into the file containing the INCLUDE. Relative path names given in an INCLUDE directive are assumed to reside in /etc/shorewall or in an alternate configuration directory if one has been specified for the command.

INCLUDE's may be nested to a level of 3 -- further nested INCLUDE directives are ignored with a warning message.

Example 3. Use of INCLUDE

```
shorewall/params.mgmt:

   MGMT_SERVERS=1.1.1.1,2.2.2.2,3.3.3.3
    TIME_SERVERS=4.4.4.4
    BACKUP_SERVERS=5.5.5.5

   ----- end params.mgmt -----

shorewall/params:

   # Shorewall 1.3 /etc/shorewall/params
    [..]
    ########################################

    INCLUDE params.mgmt

  # params unique to this host here
  #LAST LINE - ADD YOUR ENTRIES ABOVE THIS ONE - DO NOT REMOVE

  ----- end params -----

shorewall/rules.mgmt:

  ACCEPT net:$MGMT_SERVERS   $FW                  tcp    22
  ACCEPT $FW                 net:$TIME_SERVERS    udp    123
  ACCEPT $FW                 net:$BACKUP_SERVERS  tcp    22

 ----- end rules.mgmt -----

shorewall/rules:

 # Shorewall version 1.3 - Rules File
  [..]
  ########################################

  INCLUDE rules.mgmt

  # rules unique to this host here
  #LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE

----- end rules -----
```

Using DNS Names

Caution

I personally recommend strongly against using DNS names in Shorewall configuration files. If you use DNS names and you are called out of bed at 2:00AM because Shorewall won't start as a result of DNS problems then don't say that you were not forewarned.

Host addresses in Shorewall configuration files may be specified as either IP addresses or DNS Names.

DNS names in iptables rules aren't nearly as useful as they first appear. When a DNS name appears in a rule, the iptables utility resolves the name to one or more IP addresses and inserts those addresses into the rule. So changes in the DNS->IP address relationship that occur after the firewall has started have absolutely no effect on the firewall's ruleset.

If your firewall rules include DNS names then:

- If your `/etc/resolv.conf` is wrong then your firewall won't start.
- If your `/etc/nsswitch.conf` is wrong then your firewall won't start.
- If your Name Server(s) is(are) down then your firewall won't start.
- If your startup scripts try to start your firewall before starting your DNS server then your firewall won't start.
- Factors totally outside your control (your ISP's router is down for example), can prevent your firewall from starting.
- You must bring up your network interfaces prior to starting your firewall.

Each DNS name must be fully qualified and include a minimum of two periods (although one may be trailing). This restriction is imposed by Shorewall to insure backward compatibility with existing configuration files.

Example 4. Valid DNS Names

- mail.shorewall.net
- shorewall.net. (note the trailing period).

Example 5. Invalid DNS Names

- mail (not fully qualified)
- shorewall.net (only one period)

DNS names may not be used as:

- The server address in a DNAT rule (/etc/shorewall/rules file)
- In the ADDRESS column of an entry in /etc/shorewall/masq.
- In the `/etc/shorewall/nat` file.

These restrictions are imposed by Netfilter and not by Shorewall.

Comma-separated Lists

Comma-separated lists are allowed in a number of contexts within the configuration files. A comma separated list:

- Must not have any embedded white space.

```
     Valid:   routefilter,dhcp,norfc1918
     Invalid: routefilter,     dhcp,     norfc1818
```

- If you use line continuation to break a comma-separated list, the continuation line(s) must begin in column 1 (or there would be embedded white space)
- Entries in a comma-separated list may appear in any order.

Complementing an Address or Subnet

Where specifying an IP address, a subnet or an interface, you can precede the item with “!” to specify the complement of the item. For example, !192.168.1.4 means “any host but 192.168.1.4”. There must be no white space following the “!”.

Exclusion Lists

Shorewall 3.0 differs from earlier versions in that in most contexts where a comma-separated list of addresses is accepted, an *exclusion list* may also be included. An exclusion list is a comma-separated list of addresses that begins with "!".

Example:

```
!192.168.1.3,192.168.1.12,192.168.1.32/27
```

The above list refers to "All addresses except 192.168.1.3, 192.168.1.12 and 192.168.1.32-192.168.1.63.

Exclusion lists can also be added after a network address.

Example:

```
192.168.1.0/24!192.168.1.3,192.168.1.12,192.168.1.32/27
```

The above list refers to "All addresses in 192.168.1.0-192.168.1.255 except 192.168.1.3, 192.168.1.12 and 192.168.1.32-192.168.1.63.

IP Address Ranges

If you kernel and iptables have iprange match support, you may use IP address ranges in Shorewall configuration file entries; IP address ranges have the syntax <*low IP address*>-<*high IP address*>. Example: 192.168.1.5-192.168.1.12.

To see if your kernel and iptables have the required support, use the **shorewall show capabilities** command:

```
>~ shorewall show capabilities
...
Shorewall has detected the following iptables/netfilter capabilities:
   NAT: Available
   Packet Mangling: Available
   Multi-port Match: Available
   Connection Tracking Match: Available
   Packet Type Match: Not available
   Policy Match: Available
   Physdev Match: Available
   IP range Match: Available <--------------
```

Port Numbers/Service Names

Unless otherwise specified, when giving a port number you can use either an integer or a service name from /etc/services.

Port Ranges

If you need to specify a range of ports, the proper syntax is <low port number>:<high port number>. For example, if you want to forward the range of tcp ports 4000 through 4100 to local host 192.168.1.3, the entry in /etc/shorewall/rules is:

```
#ACTION    SOURCE     DESTINATION     PROTO     DEST PORTS(S)
DNAT       net        loc:192.168.1.3 tcp       4000:4100
```

If you omit the low port number, a value of zero is assumed; if you omit the high port number, a value of 65535 is assumed.

Port Lists

In most cases where a port or port range may appear, a comma-separated list of ports or port ranges may also be entered. Shorewall will use the Netfilter **multiport** match capability if it is available (see the output of "**shorewall show capabilities**") and if its use is appropriate.

Shorewall can use multiport match if:

1. The list contains 15 or fewer port number; and
2. There are no port ranges listed OR your iptables/kernel support the Extended **multiport** match (again see the output of "**shorewall show capabilities**"). Where the Extended **multiport** match is available, each port range counts as two ports toward the maximum of 15.

Using Shell Variables

You may use the /etc/shorewall/params file to set shell variables that you can then use in some of the other configuration files.

It is suggested that variable names begin with an upper case letter to distinguish them from variables used internally within the Shorewall programs

Example:

```
/etc/shorewall/params

    NET_IF=eth0
    NET_BCAST=130.252.100.255
    NET_OPTIONS=routefilter,norfc1918

/etc/shorewall/interfaces record:

    net $NET_IF $NET_BCAST $NET_OPTIONS

The result will be the same as if the record had been written

    net eth0 130.252.100.255 routefilter,norfc1918
```

Variables may be used anywhere in the other configuration files.

Because the `/etc/shorewall/params` file is simply sourced into the shell, you can place arbitrary shell code in the file and it will be executed each time that the file is read. Any code included should follow these guidelines:

1. The code should not have side effects, especially on other shorewall configuration files.
2. The code should be safe to execute multiple times without producing different results.
3. Should not depend on where the code is called from (the params file is sourced by both /sbin/shorewall and /usr/lib/shorewall/firewall).
4. Should not assume anything about the state of Shorewall.
5. The names of any functions or variables declared should begin with an upper case letter.

One possible use of this feature is to compensate for recent Linux behavior in which the identity of network interfaces varies from boot to boot (what is `eth0` after one boot may be `eth1` after the next). SuSE™ users, for example, can take the following approach:

```
wookie:~ # lspci
0000:00:00.0 Host bridge: VIA Technologies, Inc. VT82C598 [Apollo MVP3] (rev 04)
```

```
0000:00:01.0 PCI bridge: VIA Technologies, Inc. VT82C598/694x [Apollo MVP3/Pro133x
AGP]
0000:00:03.0 Ethernet controller: Intel Corporation 82557/8/9 [Ethernet Pro 100] (rev
01)
0000:00:04.0 Ethernet controller: Lite-On Communications Inc LNE100TX (rev 20)
0000:00:05.0 Ethernet controller: Digital Equipment Corporation DECchip 21142/43 (rev
41)
0000:00:14.0 ISA bridge: VIA Technologies, Inc. VT82C586/A/B PCI-to-ISA [Apollo VP]
(rev 45)
0000:00:14.1 IDE interface: VIA Technologies, Inc.
VT82C586A/B/VT82C686/A/B/VT823x/A/C PIPC Bus Master IDE (rev 06)
0000:00:14.2 USB Controller: VIA Technologies, Inc. VT82xxxxx UHCI USB 1.1 Controller
(rev 02)
0000:00:14.3 Bridge: VIA Technologies, Inc. VT82C586B ACPI (rev 10)
0000:01:00.0 VGA compatible controller: ATI Technologies Inc 3D Rage LT Pro AGP-133
(rev dc)
wookie:~ #
```

If the firewall's external interface is the DECchip controllor at 0000:00:05.0 and the internal interface is the Ethernet Pro 100 at 0000:00:03.0, then the following entries in `/etc/shorewall/params` will set EXT_IF and INT_IF to the names of these two controllers respectively:

```
EXT_IF=$(getcfg-interface bus-pci-0000:00:05.0)
INT_IF=$(getcfg-interface bus-pci-0000:00:03.0)
```

Caution

The **shorewall save** and **shorewall restore** commands should be used carefully if you use the above workaround for unstable interface names. In particular, you should set OPTIONS="" in `/etc/default/shorewall` or `/etc/sysconfig/shorewall` so that the "-f" option will not be specified on startup at boot time.

Using MAC Addresses

Media Access Control (MAC) addresses can be used to specify packet source in several of the configuration files. In order to control traffic to/from a host by its MAC address, the host must be on the same network as the firewall.

To use this feature, your kernel must have MAC Address Match support (CONFIG_IP_NF_MATCH_MAC) included.

MAC addresses are 48 bits wide and each Ethernet Controller has a unique MAC address.

In GNU/Linux, MAC addresses are usually written as a series of 6 hex numbers separated by colons.

Example 6. MAC Address of an Ethernet Controller

```
[root@gateway root]# ifconfig eth0
eth0 Link encap:Ethernet HWaddr 02:00:08:E3:FA:55
inet addr:206.124.146.176 Bcast:206.124.146.255 Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:2398102 errors:0 dropped:0 overruns:0 frame:0
TX packets:3044698 errors:0 dropped:0 overruns:0 carrier:0
collisions:30394 txqueuelen:100
RX bytes:419871805 (400.4 Mb) TX bytes:1659782221 (1582.8 Mb)
Interrupt:11 Base address:0x1800
```

Because Shorewall uses colons as a separator for address fields, Shorewall requires MAC addresses to be written in another way. In Shorewall, MAC addresses begin with a tilde ("~") and consist of 6 hex numbers separated by hyphens. In Shorewall, the MAC address in the example above would be written **~02-00-08-E3-FA-55**.

Note

It is not necessary to use the special Shorewall notation in the `/etc/shorewall/maclist` file.

Shorewall Configurations

Shorewall allows you to have configuration directories other than `/etc/shorewall`. The shorewall check, start and restart commands allow you to specify an alternate configuration directory and Shorewall will use the files in the alternate directory rather than the corresponding files in /etc/shorewall. The alternate directory need not contain a complete configuration; those files not in the alternate directory will be read from `/etc/shorewall`.

This facility permits you to easily create a test or temporary configuration by

1. copying the files that need modification from /etc/shorewall to a separate directory;
2. modify those files in the separate directory; and
3. specifying the separate directory in a **shorewall start** or **shorewall restart** command (e.g., **shorewall restart /etc/testconfig**)

The try command allows you to attempt to restart using an alternate configuration and if an error occurs to automatically restart the standard configuration.

MAC Verification

Tom Eastep

Copyright © 2001-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-10-13

Table of Contents

Components
/etc/shorewall/maclist
Examples

All traffic from an interface or from a subnet on an interface can be verified to originate from a defined set of MAC addresses. Furthermore, each MAC address may be optionally associated with one or more IP addresses.

Important

MAC addresses are only visible within an ethernet segment so all MAC addresses used in verification must belong to devices physically connected to one of the LANs to which your firewall is connected.

This means what it says! MAC addresses are only used within a LAN and never go outside of that LAN so please don't post on the mailing list asking how to use MAC addresses of computers connected to remote networks. The only MAC address that your firewall is going to see from these hosts is the MAC address of your upstream router.

Important

Your kernel must include MAC match support (CONFIG_IP_NF_MATCH_MAC - module name ipt_mac.o).

Important

MAC verification is only applied to new incoming connection requests.

Important

DO NOT use MAC verification as your only security measure . MAC addresses can be easily spoofed. You can use it in combination with either IPSEC or OpenVPN.

Components

There are six components to this facility.

1. The **maclist** interface option in /etc/shorewall/interfaces. When this option is specified, all new connection requests arriving on the interface are subject to MAC verification.
2. The **maclist** option in /etc/shorewall/hosts. When this option is specified for a subnet, all new connection requests from that subnet are subject to MAC verification.
3. The /etc/shorewall/maclist file. This file is used to associate MAC addresses with interfaces and to optionally associate IP addresses with MAC addresses.
4. The **MACLIST_DISPOSITION** and **MACLIST_LOG_LEVEL** variables in /etc/shorewall/shorewall.conf. The MACLIST_DISPOSITION variable has the value DROP, REJECT or ACCEPT and determines the disposition of connection requests that fail MAC verification. The MACLIST_LOG_LEVEL variable gives the syslogd level at which connection requests that fail verification are to be logged. If set the empty value (e.g., MACLIST_LOG_LEVEL="") then failing connection requests are not logged.
5. Beginning with Shorewall 2.2.3, the **MACLIST_TTL** variable in /etc/shorewall/shorewall.conf. The performance of configurations with a large numbers of entries in /etc/shorewall/maclist can be improved by setting the MACLIST_TTL variable.

   If your iptables and kernel support the "Recent Match" (see the output of "shorewall check" near the top), you can cache the results of a 'maclist' file lookup and thus reduce the overhead associated with MAC Verification.

   When a new connection arrives from a 'maclist' interface, the packet passes through then list of entries for that interface in /etc/shorewall/maclist. If there is a match then the source IP address is added to the 'Recent' set for that interface. Subsequent connection attempts from that IP address occurring within $MACLIST_TTL seconds will be accepted without having to scan all of the entries. After $MACLIST_TTL from the first accepted connection request from an IP address, the next connection request from that IP address will be checked against the entire list.

   If MACLIST_TTL is not specified or is specified as empty (e.g, MACLIST_TTL="" or is specified as zero then 'maclist' lookups will not be cached).
6. Beginning with Shorewall 2.4.6, the **MACLIST_TABLE** variable in /etc/shorewall/shorewall.conf. Normally, MAC verification occurs in the filter table (INPUT and FORWARD) chains. When forwarding a packet from an interface with MAC verification to a bridge interface, that doesn't work.

   This problem can be worked around by setting MACLIST_TABLE=mangle which will cause MAC verification to occur out of the PREROUTING chain. Because REJECT isn't available in that environment, you may not specify MACLIST_DISPOSITION=REJECT with MACLIST_TABLE=mangle.

/etc/shorewall/maclist

The columns in /etc/shorewall/maclist are:

INTERFACE

The name of an ethernet interface on the Shorewall system.

MAC

The MAC address of a device on the ethernet segment connected by INTERFACE. It is not necessary to use the Shorewall MAC format in this column although you may use that format if you so choose.

IP Address

An optional comma-separated list of IP addresses for the device whose MAC is listed in the MAC column.

Examples

Example 1. Here are my files (look here for details about my setup)

/etc/shorewall/shorewall.conf:

```
MACLIST_DISPOSITION=REJECT
MACLIST_LOG_LEVEL=info
```

/etc/shorewall/interfaces:

```
#ZONE   INTERFACE       BROADCAST       OPTIONS
net     $EXT_IF         206.124.146.255
dhcp,norfc1918,routefilter,logmartians,blacklist,tcpflags,nosmurfs
loc     $INT_IF         192.168.1.255   dhcp
dmz     $DMZ_IF         -
vpn     tun+            -
Wifi    $WIFI_IF        -               maclist,dhcp
#LAST LINE - ADD YOUR ENTRIES ABOVE THIS ONE - DO NOT REMOVE
```

/etc/shorewall/maclist:

```
#INTERFACE              MAC                     IP ADDRESSES (Optional)
$WIFI_IF                00:04:5e:3f:85:b9                       #WAP11
$WIFI_IF                00:06:25:95:33:3c                       #WET11
$WIFI_IF                00:0b:4d:53:cc:97       192.168.3.8     #TIPPER
$WIFI_IF                00:1f:79:cd:fe:2e       192.168.3.6     #Work Laptop
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

As shown above, I use MAC Verification on my wireless zone.

Note

While marketed as a wireless bridge, the WET11 behaves like a wireless router with DHCP relay. When forwarding DHCP traffic, it uses the MAC address of the host (TIPPER) but for other forwarded traffic it uses it's own MAC address. Consequently, I list the IP addresses of both devices in /etc/shorewall/maclist.

Example 2. Router in Wireless Zone

Suppose now that I add a second wireless segment to my wireless zone and gateway that segment via a router with MAC address 00:06:43:45:C6:15 and IP address 192.168.3.253. Hosts in the second segment have IP addresses in the subnet 192.168.4.0/24. I would add the following entry to my /etc/shorewall/maclist file:

```
$WIFI_IF                    00:06:43:45:C6:15       192.168.3.253,192.168.4.0/24
```

This entry accommodates traffic from the router itself (192.168.3.253) and from the second wireless segment (192.168.4.0/24). Remember that all traffic being sent to my firewall from the 192.168.4.0/24 segment will be forwarded by the router so that traffic's MAC address will be that of the router (00:06:43:45:C6:15) and not that of the host sending the traffic.

OpenVPN Tunnels and Bridges

Simon Matter

Tom Eastep

Copyright © 2003, 2004, 2005 Simon Mater, Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-10-25

Table of Contents

Preliminary Reading
Bridging two Masqueraded Networks
Roadwarrior
Securing a Home Wireless Network with OpenVPN (OpenVPN Bridge)
- Configuring the Bridge
- Configuring OpenVPN
  - Firewall (Server) configuration.
  - Tipper Configuration
  - Eastepnc6000 (Windows XP) Configuration
- Configuring Shorewall
  - Firewall
  - Tipper

Caution

This article applies to Shorewall 3.0 and later and to OpenVPN 2.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

OpenVPN is a robust and highly configurable VPN (Virtual Private Network) daemon which can be used to securely link two or more private networks using an encrypted tunnel over the internet. OpenVPN is an Open Source project and is licensed under the GPL. OpenVPN can be downloaded from http://openvpn.net/.

Unless there are interoperability issues (the remote systems do not support OpenVPN), OpenVPN is my choice any time that I need a VPN.

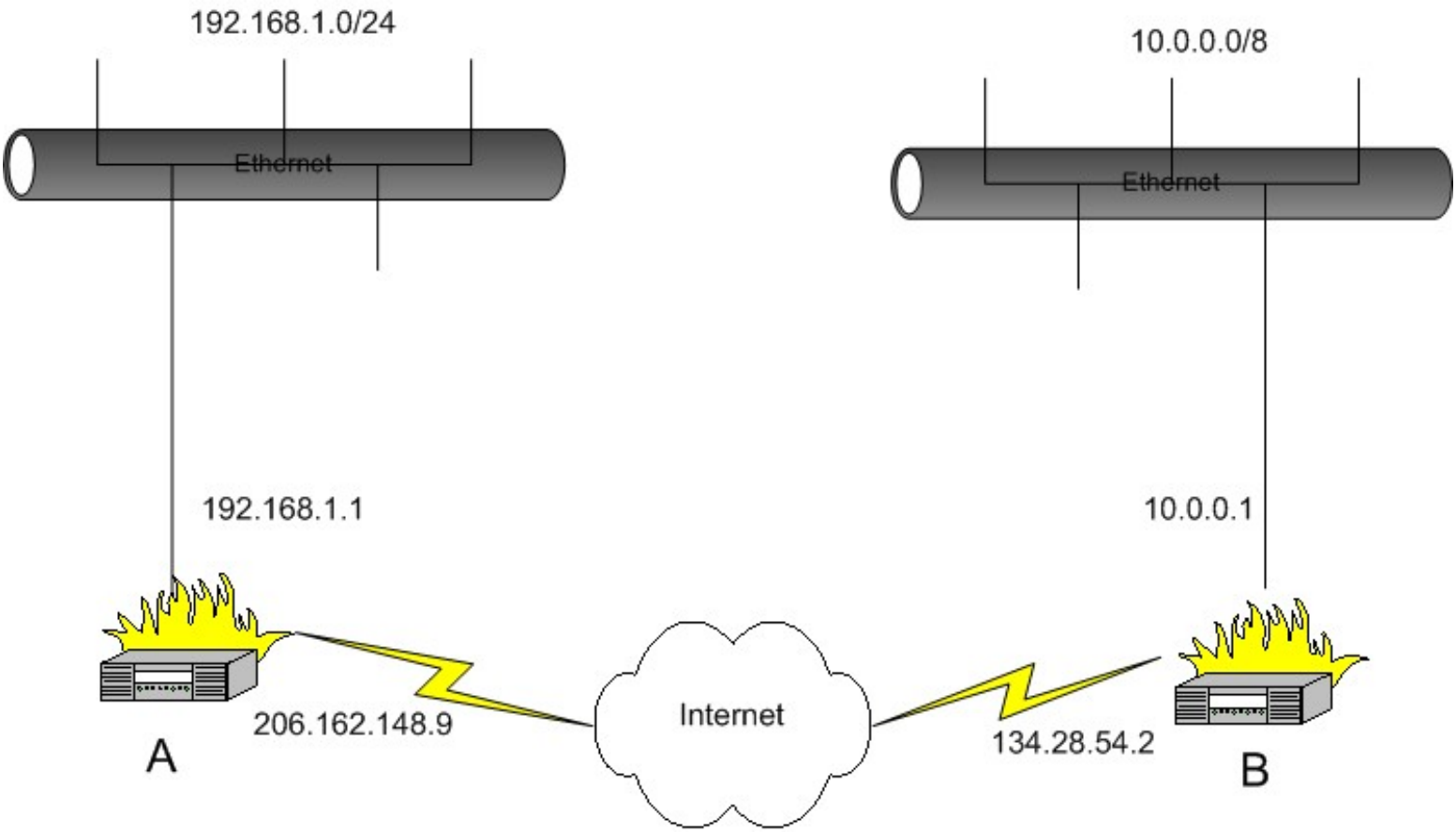
1. It is widely supported -- I run it on both Linux and Windows XP.
2. It requires no kernel patching.
3. It is very easy to configure.
4. It just works!

Preliminary Reading

I recommend reading the VPN Basics article if you plan to implement any type of VPN.

Bridging two Masqueraded Networks

Suppose that we have the following situation:



We want systems in the 192.168.1.0/24 subnetwork to be able to communicate with the systems in the 10.0.0.0/8 network. This is accomplished through use of the `/etc/shorewall/tunnels` file and the `/etc/shorewall/policy file` and OpenVPN.

While it was possible to use the Shorewall start and stop script to start and stop OpenVPN, I decided to use the init script of OpenVPN to start and stop it.

On each firewall, you will need to declare a zone to represent the remote subnet. We'll assume that this zone is called “vpn” and declare it in `/etc/shorewall/zones` on both systems as follows.

`/etc/shorewall/zones` — Systems A & B

```
#ZONE   TYPE   OPTIONS                    IN                          OUT
#                                         OPTIONS                     OPTIONS
vpn     ipv4
```

On system A, the 10.0.0.0/8 will comprise the **vpn** zone.

In `/etc/shorewall/interfaces` on system A:

```
#ZONE      INTERFACE        BROADCAST     OPTIONS
vpn        tun0
```

In `/etc/shorewall/tunnels` on system A, we need the following:

```
#TYPE         ZONE           GATEWAY        GATEWAY ZONE
openvpn       net            134.28.54.2
```

This entry in `/etc/shorewall/tunnels` opens the firewall so that OpenVPN traffic on the default port 1194/udp will be accepted to/from the remote gateway. If you change the port used by OpenVPN to 7777, you can define /etc/shorewall/tunnels like this:

/etc/shorewall/tunnels with port 7777:

```
#TYPE              ZONE            GATEWAY          GATEWAY ZONE
openvpn:7777       net             134.28.54.2
```

Similarly, if you want to use TCP for your tunnel rather than UDP (the default), then you can define /etc/shorewall/tunnels like this:

/etc/shorewall/tunnels using TCP:

```
#TYPE              ZONE            GATEWAY          GATEWAY ZONE
openvpn:tcp        net             134.28.54.2
```

Finally, if you want to use TCP and port 7777:

/etc/shorewall/tunnels using TCP port 7777:

```
#TYPE              ZONE            GATEWAY          GATEWAY ZONE
openvpn:tcp:7777   net             134.28.54.2
```

This is the OpenVPN config on system A:

```
dev tun
local 206.162.148.9
remote 134.28.54.2
ifconfig 192.168.99.1 192.168.99.2
up ./route-a.up
tls-server
dh dh1024.pem
ca ca.crt
cert my-a.crt
key my-a.key
comp-lzo
verb 5
```

Similarly, On system B the 192.168.1.0/24 subnet will comprise the **vpn** zone

In `/etc/shorewall/interfaces` on system B:

```
#ZONE       INTERFACE         BROADCAST     OPTIONS
vpn         tun0              192.168.1.255
```

In `/etc/shorewall/tunnels` on system B, we have:

```
#TYPE          ZONE            GATEWAY         GATEWAY ZONE
openvpn        net             206.191.148.9
```

And in the OpenVPN config on system B:

```
dev tun
local 134.28.54.2
remote 206.162.148.9
```

```
ifconfig 192.168.99.2 192.168.99.1
up ./route-b.up
tls-client
ca ca.crt
cert my-b.crt
key my-b.key
comp-lzo
verb 5
```

You will need to allow traffic between the “vpn” zone and the “loc” zone on both systems -- if you simply want to admit all traffic in both directions, you can use the policy file:

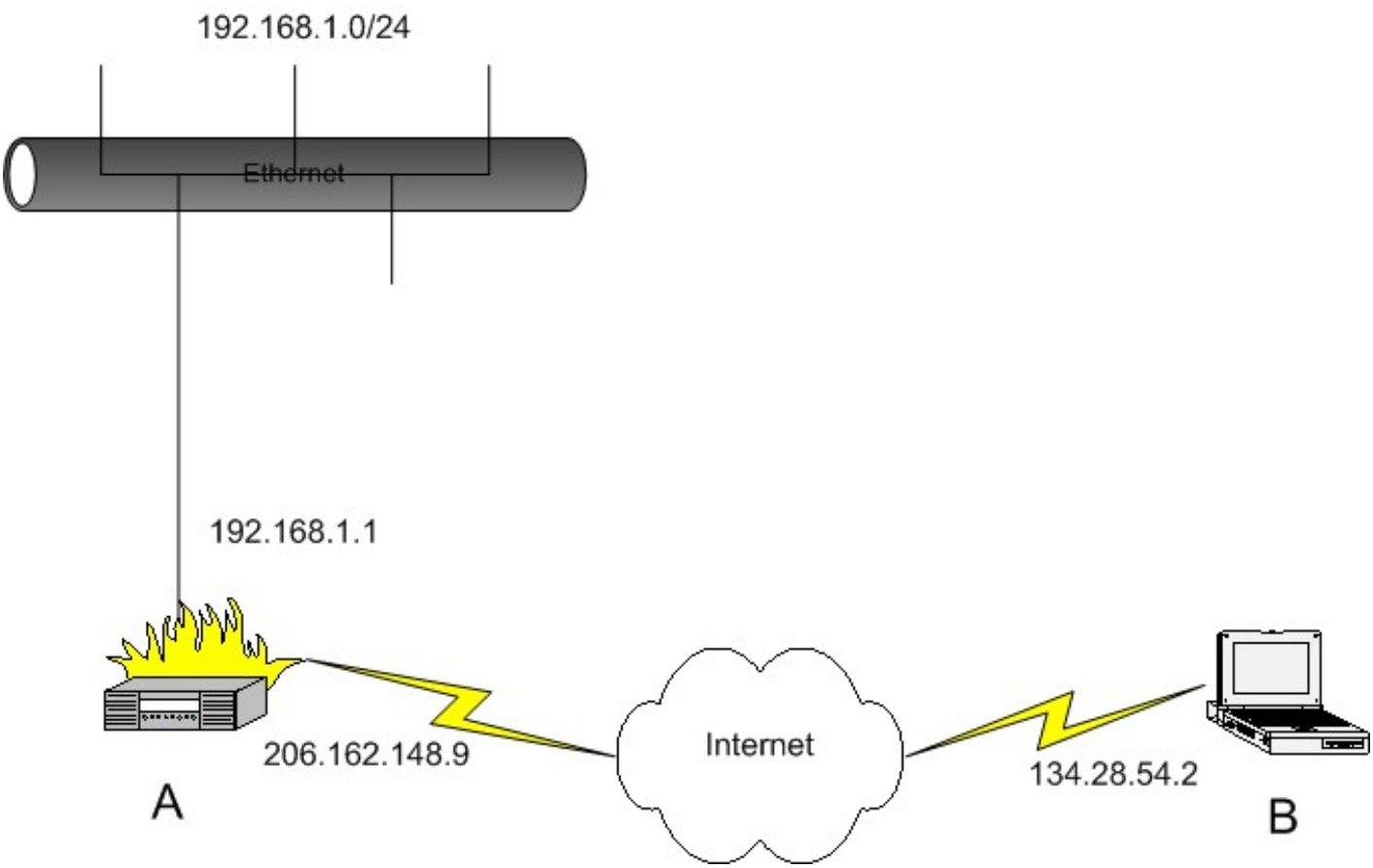
`/etc/shorewall/policy` on systems A & B

```
#SOURCE         DEST            POLICY          LOG LEVEL
loc             vpn             ACCEPT
vpn             loc             ACCEPT
```

On both systems, restart Shorewall and start OpenVPN. The systems in the two masqueraded subnetworks can now talk to each other.

Roadwarrior

OpenVPN 2.0 provides excellent support for roadwarriors. Consider the setup in the following diagram:



On the gateway system (System A), we need a zone to represent the remote clients — we'll call that zone “road”.

`/etc/shorewall/zones` — System A:

```
#ZONE   TYPE    OPTIONS                 IN                      OUT
#                                       OPTIONS                 OPTIONS
```

```
road    ipv4
```

On system A, the remote clients will comprise the **road** zone.

In `/etc/shorewall/interfaces` on system A:

```
#ZONE      INTERFACE        BROADCAST     OPTIONS
road       tun+
```

In `/etc/shorewall/tunnels` on system A, we need the following:

```
#TYPE         ZONE           GATEWAY        GATEWAY ZONE
openvpn:1194  net            0.0.0.0/0
```

If you are running Shorewall 2.4.3 or later, you might prefer the following in `/etc/shorewall/tunnels` on system A. Specifying the tunnel type as openvpnserver has the advantage that the VPN connection will still work if the client is behind a gateway/firewall that uses NAT.

```
#TYPE               ZONE           GATEWAY        GATEWAY ZONE
openvpnserver:1194  net            0.0.0.0/0
```

We want the remote systems to have access to the local LAN — we do that with an entry in `/etc/shorewall/policy` (assume that the local LAN comprises the zone “loc”).

```
#SOURCE      DESTINATION        POLICY
road         loc                ACCEPT
```

The OpenVPN configuration file on system A is something like the following:

```
dev tun

server 192.168.2.0 255.255.255.0

dh dh1024.pem

ca /etc/certs/cacert.pem

crl-verify /etc/certs/crl.pem

cert /etc/certs/SystemA.pem
key /etc/certs/SystemA_key.pem

port 1194

comp-lzo

user nobody

group nogroup

ping 15
ping-restart 45
ping-timer-rem
persist-tun
persist-key

verb 3
```

Configuration on the remote clients follows a similar line. We define a zone to represent the remote LAN:

`/etc/shorewall/zones` — System B:

```
#ZONE   TYPE   OPTIONS                 IN                      OUT
#                                      OPTIONS                 OPTIONS
home    ipv4
```

On system A, the hosts accessible through the tunnel will comprise the **home** zone.

In `/etc/shorewall/interfaces` on system B:

```
#ZONE      INTERFACE        BROADCAST     OPTIONS
home       tun0
```

In `/etc/shorewall/tunnels` on system B, we need the following:

```
#TYPE         ZONE           GATEWAY        GATEWAY ZONE
openvpn:1194  net            206.162.148.9
```

Again in you are running Shorewall 2.4.3 or later, in `/etc/shorewall/tunnels` on system B you might prefer:

```
#TYPE               ZONE           GATEWAY        GATEWAY ZONE
openvpnclient:1194  net            206.162.148.9
```

We want the remote client to have access to the local LAN — we do that with an entry in `/etc/shorewall/policy`.

```
#SOURCE      DESTINATION        POLICY
$FW          home               ACCEPT
```

The OpenVPN configuration on the remote clients is along the following line:

```
dev tun
remote 206.162.148.9
up /etc/openvpn/home.up

tls-client
pull

ca /etc/certs/cacert.pem

cert /etc/certs/SystemB.pem
key /etc/certs/SystemB_key.pem

port 1194

user nobody
group nogroup

comp-lzo

ping 15
ping-restart 45
ping-timer-rem
persist-tun
persist-key
```

```
verb 3
```

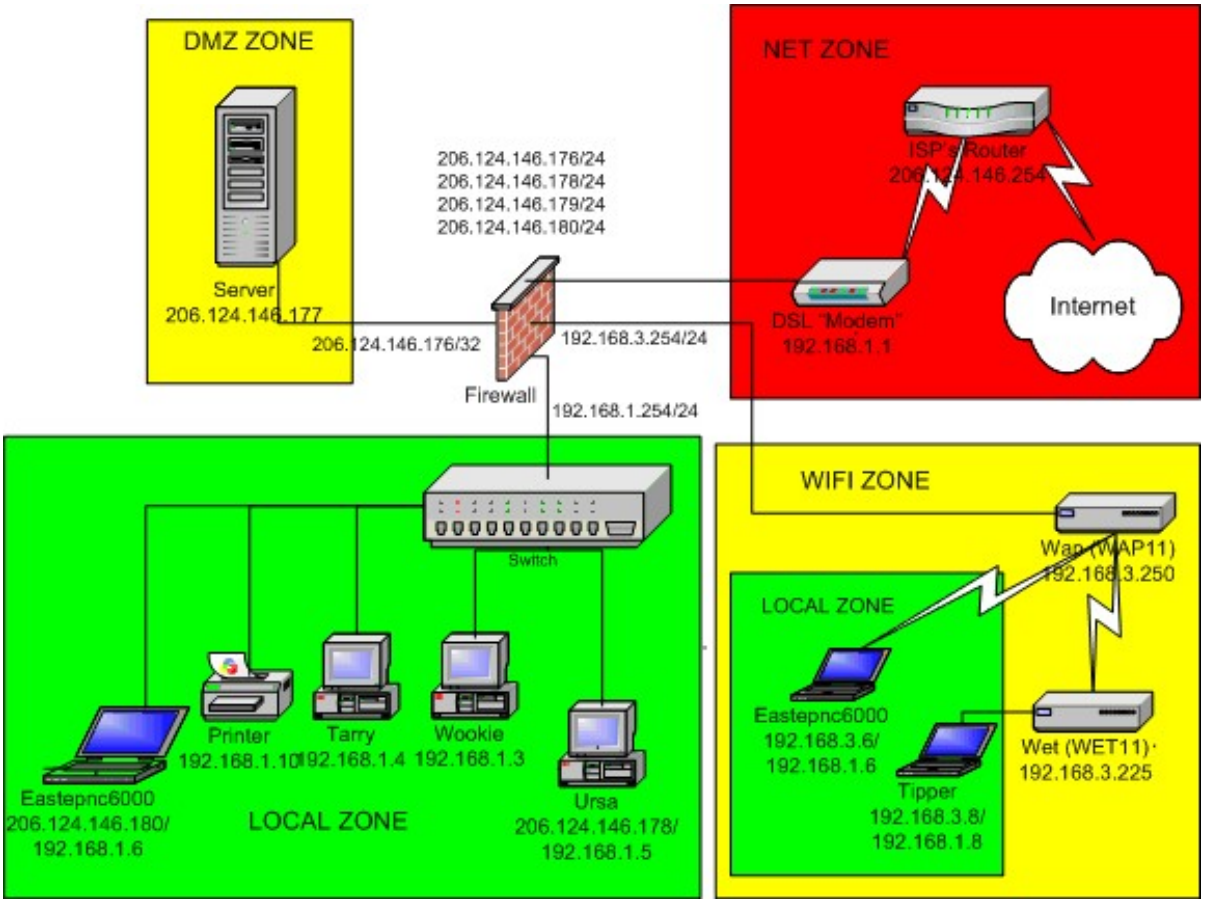
If you want multiple remote clients to be able to communicate openly with each other then you must:

1. Include the **client-to-client** directive in the server's OpenVPN configuration; and
2. Specify the **routeback** option on the `tun+` device in /etc/shorewall/interfaces.

If you want to selectively allow communication between the clients, then see this article by Marc Zonzon

Securing a Home Wireless Network with OpenVPN (OpenVPN Bridge)

This section will describe how we secured our home wireless network using OpenVPN. Our network is as shown in the following diagram.



The Wireless network is in the lower right of the diagram and consists of two laptops: Eastepnc6000 (Windows XP - SP1) and Tipper (SuSE 10.0). We use OpenVPN to bridge those two laptops with the local LAN shown in the lower left hand corner. The laptops are configured with addresses in the 192.168.3.0/24 network connected to the firewall's `eth0` interface which places them in the firewall's **Wifi** zone. OpenVPN bridging allows them to be assigned an additional IP address from the 192.168.1.0/24 network and to be securely bridged to the LAN on the lower left.

Note

Eastepnc6000 is shown in both the local LAN and in the Wifi zone with IP address 192.168.1.6 -- clearly, the computer can only be in one place or the other. Tipper can also be in either place and will have the IP address 192.168.1.8 regardless.

Configuring the Bridge

The firewall runs Debian Sarge so the bridge is defined in `/etc/network/interfaces`.

```
# LAN interface
auto br0
iface br0 inet static
        address 192.168.1.254
        netmask 255.255.255.0
        pre-up /usr/sbin/openvpn --mktun --dev tap0
        pre-up /sbin/ip link set tap0 up
        pre-up /sbin/ip link set eth3 up
        pre-up /usr/sbin/brctl addbr br0
        pre-up /usr/sbin/brctl addif br0 eth3
        pre-up /usr/sbin/brctl addif br0 tap0
        post-down /usr/sbin/brctl delif br0 eth3
        post-down /usr/sbin/brctl delif br0 tap0
        post-down /usr/sbin/brctl delbr br0
        post-down /usr/sbin/openvpn --rmtun --dev tap0
```

Note that the IP address assigned to the bridge is 192.168.1.254 -- that is the default gateway address for hosts in the local zone.

Configuring OpenVPN

We use X.509 certificates for authentication.

Firewall (Server) configuration.

/etc/openvpn/server-bridge.conf defines a bridge and reserves IP addresses 192.168.1.64-192.168.1.71 for VPN clients. Note that the bridge server only uses local IP address 192.168.3.254. We run two instances of OpenVPN; this one and a second tunnel-mode instance for remote access (see this article).

```
dev tap0

local 192.168.3.254

server-bridge 192.168.1.254 255.255.255.0 192.168.1.64 192.168.1.71

client-to-client

dh dh1024.pem

ca /etc/certs/cacert.pem

crl-verify /etc/certs/crl.pem

cert /etc/certs/gateway.pem
key /etc/certs/gateway_key.pem

port 1194

comp-lzo

user nobody
group nogroup

keepalive 15 45
ping-timer-rem
persist-tun
persist-key

client-config-dir /etc/openvpn/bridge-clients
ccd-exclusive
```

```
verb 3
```

The files in `/etc/openvpn/bridge-clients` are used to assign a fixed IP address to each laptop. For example, tipper.shorewall.net:

```
ifconfig-push 192.168.1.8 255.255.255.0
```

Tipper Configuration

/etc/openvpn/wireless.conf:

```
dev tap

remote 192.168.3.254
tls-remote gateway.shorewall.net

client

redirect-gateway

ca /etc/certs/cacert.pem

cert /etc/certs/tipper.pem
key /etc/certs/tipper_key.pem

port 1194

comp-lzo

ping 15
ping-restart 45
ping-timer-rem
persist-tun
persist-key

mute-replay-warnings

verb 3
```

Eastepnc6000 (Windows XP) Configuration

C:\Program Files\Openvpn\config\homewireless.ovpn:

```
dev tap
remote 192.168.3.254
tls-remote gateway.shorewall.net

tls-client
pull

ca "/Program Files/OpenVPN/certs/cacert.pem"

cert "/Program Files/OpenVPN/certs/eastepnc6000.pem"
key "/Program Files/OpenVPN/certs/eastepnc6000_key.pem"

redirect-gateway

port 1194

comp-lzo
```

```
ping 15
ping-restart 45
ping-timer-rem
persist-tun
persist-key

verb 3
```

Configuring Shorewall

In this configuration, we don't need any firewalling between the laptops and the local LAN so we set BRIDGING=No in shorewall.conf. The configuration of the bridge then becomes as described in the Simple Bridge documentation. If you need to control the traffic allowed through the VPN bridge then you will want to configure Shorewall as shown in the Bridge/Firewall documentation.

Firewall

/etc/shorewall/interfaces

Note that the bridge (br0) is defined as the interface to the local zone and has the **routeback** option.

```
#ZONE   INTERFACE       BROADCAST               OPTIONS
net     eth2            206.124.146.255
dhcp,norfc1918,logmartians,blacklist,tcpflags,nosmurfs
loc     br0             192.168.1.255           dhcp,routeback
dmz     eth1            -                       logmartians
Wifi    eth0            192.168.3.255           dhcp,maclist
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

/etc/shorewall/tunnels

```
#TYPE                   ZONE    GATEWAY         GATEWAY
#                                               ZONE
openvpnserver:1194      Wifi    192.168.3.0/24
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

Tipper

Wireless networks pose a threat to all systems that are connected to them and we therefore run Firewalls on the two Laptops. Eastepnc6000 runs Sygate™ Security Agent and Tipper runs a Shorewall-based Netfilter firewall.

/etc/shorewall/zones

```
#ZONE   TYPE    OPTIONS                 IN                      OUT
#                                       OPTIONS                 OPTIONS
lan     ipv4    #Wired LAN at our home
net     ipv4
#LAST LINE - ADD YOUR ENTRIES ABOVE THIS ONE - DO NOT REMOVE
```

/etc/shorewall/interfaces

```
#ZONE    INTERFACE      BROADCAST       OPTIONS
#
net      eth0           detect          routefilter,dhcp,tcpflags
lan      tap0           192.168.1.255
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

/etc/shorewall/policy

Since we don't expect any traffic between the **net** zone and the **lan** zone, we use NONE policies for that traffic. If any such traffic should occur, it will be handled according to the all->all policy.

```
#SOURCE         DEST            POLICY          LOG             LIMIT:BURST
#                                               LEVEL
fw              net             ACCEPT
fw              lan             ACCEPT
lan             fw              ACCEPT
net             lan             NONE
lan             net             NONE
net             all             DROP            info
# The FOLLOWING POLICY MUST BE LAST
all             all             REJECT          info
#LAST LINE -- DO NOT REMOVE
```

About My Network

Tom Eastep

Copyright © 2001-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-11-01

Table of Contents

My Current Network
Firewall Configuration
- Shorewall.conf
- Params File (Edited)
- Zones File
- Interfaces File
- Routestopped File
- Providers File
- Blacklist File
- RFC1918 File
- Policy File
- Masq File
- NAT File
- Proxy ARP File
- Tunnels
- Actions File
- action.Mirrors File
- Rules File (The shell variables are set in /etc/shorewall/params)
- /etc/shorewall/tcdevices
- /etc/shorewall/tcclasses
- /etc/shorewall/tcrules
- /etc/network/interfaces
- /etc/openvpn/server.conf

Tipper and Eastepnc6000 Configuration in the Wireless Network
Tipper Configuration while on the Road
- zones
- policy
- interfaces
- rules
- /etc/openvpn/home.conf
- /etc/openvpn/home.up

My Current Network

Caution

I use a combination of One-to-one NAT and Proxy ARP, neither of which are relevant to a simple configuration with a single public IP address. If you have just a single public IP address, most of what you see here won't apply to your setup so beware of copying parts of this configuration and expecting them to work for you. What you copy may or may not work in your environment.

Caution

The configuration shown here corresponds to Shorewall version 3.0.0. My configuration uses features not available in earlier Shorewall releases.

I have DSL service with 5 static IP addresses (206.124.146.176-180). My DSL “modem” (Westell 2200) is connected to eth2 and has IP address 192.168.1.1 (factory default). The modem is configured in “bridge” mode so PPPoE is not involved. I have a local network connected to eth3 which is bridged to interface tun0 via bridge br0 (subnet 192.168.1.0/24), a wireless network (192.168.3.0/24) connected to eth0, and a DMZ connected to eth1 (206.124.146.176/32). Note that I configure the same IP address on both `eth1` and `eth2`.

In this configuration:

- I use one-to-one NAT for *"Ursa"* (my personal system that run SuSE 10.0) - Internal address 192.168.1.5 and external address 206.124.146.178.
- I use one-to-one NAT for *"Eastepnc6000*" (My work system -- Windows XP SP1/SuSE 10.0). Internal address 192.168.1.6 and external address 206.124.146.180.
- I use SNAT through 206.124.146.179 for my Wife's Windows XP system “*Tarry*”, my *crash and burn* system "*Wookie*", our SuSE 10.0 laptop “*Tipper*” which connects through the Wireless Access Point (wap) via a Wireless Bridge (wet), and my work laptop (*eastepnc6000*) when it is not docked in my office.

Note

While the distance between the WAP and where I usually use the laptop isn't very far (50 feet or so), using a WAC11 (CardBus wireless card) has proved very unsatisfactory (lots of lost connections). By replacing the WAC11 with the WET11 wireless bridge, I have virtually eliminated these problems (Being an old radio tinkerer (K7JPV), I was also able to eliminate the disconnects by hanging a piece of aluminum foil on the family room wall. Needless to say, my wife Tarry rejected that as a permanent solution :-).

- Squid runs on the DMZ server and is configured as a transparent proxy.

The firewall runs on a P-II/233 with Debian Sarge (testing).

Ursa runs Samba for file sharing with the Windows systems and is configured as a Wins server.

The wireless network connects to the firewall's eth0 via a LinkSys WAP11. In additional to using the rather weak WEP 40-bit encryption (64-bit with the 24-bit preamble), I use MAC verification and OpenVPN in bridge mode.

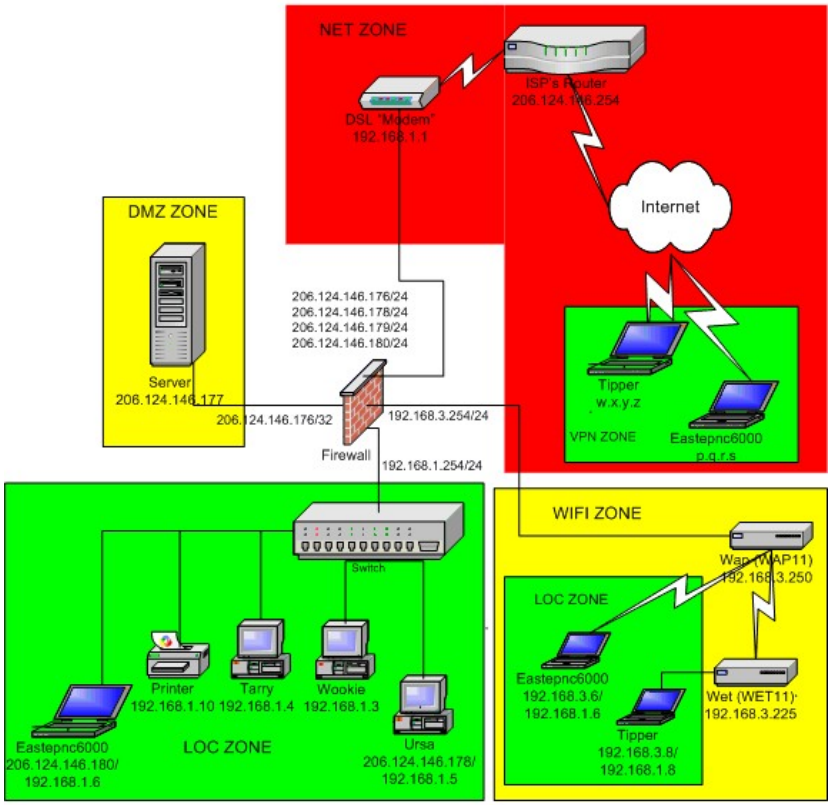
The single system in the DMZ (address 206.124.146.177) runs Postfix, Courier IMAP (imap and imaps), DNS (Bind 9), a Web server (Apache) and an FTP server (Pure-ftpd) under Fedora Core 4. The system also runs fetchmail to fetch our email from our old and current ISPs. That server is accessible from the Internet through Proxy ARP.

The firewall system itself runs a DHCP server that serves the local and wireless networks.

All administration and publishing is done using ssh/scp. I have a desktop environment installed on the firewall but I usually don't start it. X applications tunnel through SSH to *Ursa* or one of the laptops. The server also has a desktop environment installed but it is seldom started either. For the most part, X tunneled through SSH is used for server administration and the server runs at run level 3 (multi-user console mode on Fedora).

The ethernet interface in the Server is configured with IP address 206.124.146.177, netmask 255.255.255.0. The server's default gateway is 206.124.146.254 (Router at my ISP. This is the same default gateway used by the firewall itself). On the firewall, an entry in my /etc/network/interfaces file (see below) adds a host route to 206.124.146.177 through eth1 when that interface is brought up.

In addition to the OpenVPN bridge, the firewall hosts an OpenVPN Tunnel server for VPN access from our second home in Omak, Washington or when we are otherwise out of town.



Note

Eastepnc6000 is shown in both the local LAN and in the Wifi zone with IP address 192.168.1.6 -- clearly, the computer can only be in one place or the other. *Tipper* can also be in either place and will have the IP address 192.168.1.8 regardless.

Firewall Configuration

Shorewall.conf

```
STARTUP_ENABLED=Yes
LOGFILE=/var/log/messages
LOGFORMAT="Shorewall:%s:%s:"
LOGTAGONLY=No
LOGRATE=
LOGBURST=
LOGALLNEW=
BLACKLIST_LOGLEVEL=
MACLIST_LOG_LEVEL=$LOG
TCP_FLAGS_LOG_LEVEL=$LOG
RFC1918_LOG_LEVEL=$LOG
SMURF_LOG_LEVEL=$LOG
LOG_MARTIANS=No
IPTABLES=
PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin:/usr/local/sbin
SHOREWALL_SHELL=/bin/dash
SUBSYSLOCK=
MODULESDIR=
CONFIG_PATH=/etc/shorewall:/usr/share/shorewall
```

```
RESTOREFILE=standard
IPSECFILE=zones
FW=
IP_FORWARDING=On
ADD_IP_ALIASES=Yes
ADD_SNAT_ALIASES=Yes
RETAIN_ALIASES=Yes
TC_ENABLED=Internal
CLEAR_TC=Yes
MARK_IN_FORWARD_CHAIN=Yes
CLAMPMSS=Yes
ROUTE_FILTER=No
DETECT_DNAT_IPADDRS=Yes
MUTEX_TIMEOUT=60
ADMINISABSENTMINDED=Yes
BLACKLISTNEWONLY=Yes
DELAYBLACKLISTLOAD=No
MODULE_SUFFIX=
DISABLE_IPV6=Yes
BRIDGING=No
DYNAMIC_ZONES=No
PKTTYPE=No
RFC1918_STRICT=Yes
MACLIST_TTL=60
SAVE_IPSETS=Yes
MAPOLDACTIONS=No
FASTACCEPT=No
BLACKLIST_DISPOSITION=DROP
MACLIST_TABLE=mangle
MACLIST_DISPOSITION=DROP
TCP_FLAGS_DISPOSITION=DROP
```

Params File (Edited)

```
NTPSERVERS=<list of NTP server IP addresses>
POPSERVERS=<list of external POP3 servers accessed by fetchmail running on the DMZ
server>
LOG=info
WIFI_IF=eth0
EXT_IF=eth2
INT_IF=br0
DMZ_IF=eth1
OMAK=<ip address of the gateway at our second home>
```

Zones File

```
#ZONE   TYPE   OPTTIONS                IN                  OUT
#                                       OPTIONS             OPTIONS
fw      firewall
net     ipv4
dmz     ipv4
loc     ipv4
vpn     ipv4
Wifi    ipv4
#LAST LINE - ADD YOUR ENTRIES ABOVE THIS ONE - DO NOT REMOVE
```

Interfaces File

```
#ZONE   INTERFACE       BROADCAST               OPTIONS
net     $EXT_IF         206.124.146.255
dhcp,norfc1918,logmartians,blacklist,tcpflags,nosmurfs
loc     $INT_IF         detect                  dhcp,routeback
```

```
dmz     $DMZ_IF         -
vpn     tun+            -
Wifi    $WIFI_IF        -                       dhcp,maclist
#LAST LINE - ADD YOUR ENTRIES ABOVE THIS ONE - DO NOT REMOVE
```

Routestopped File

```
#INTERFACE      HOST(S)         OPTIONS
$DMZ_IF         206.124.146.177 source
$INT_IF         -               source,dest
$WIFI_IF        -               source,dest
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS ONE - DO NOT REMOVE
```

Providers File

This entry isn't necessary but it allows me to smoke test parsing of the providers file.

```
#NAME   NUMBER  MARK    DUPLICATE       INTERFACE       GATEWAY         OPTIONS
COPY
Blarg   1       1       main            $EXT_IF         206.124.146.254
track,balance=1         $INT_IF,$DMZ_IF,$WIFI_IF,tun0
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

Blacklist File

I use ipsets to represent my blacklist.

```
#ADDRESS/SUBNET         PROTOCOL        PORT
+Blacklistports[dst]
+Blacklistnets[src,dst]
+Blacklist[src,dst]
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS ONE - DO NOT REMOVE
```

RFC1918 File

Because my DSL modem has an RFC 1918 address (192.168.1.1) and is connected to eth0, I need to make an exception for that address in my rfc1918 file. I copied /usr/share/shorewall/rfc1918 to /etc/shorewall/rfc1918 and changed it as follows:

```
#SUBNET           TARGET
192.168.1.1       RETURN
172.16.0.0/12     logdrop        # RFC 1918
192.168.0.0/16    logdrop        # RFC 1918
10.0.0.0/8        logdrop        # RFC 1918
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

Policy File

```
#SOURCE         DESTINATION     POLICY          LOG LEVEL       BURST:LIMIT
$FW             $FW             ACCEPT
loc             net             ACCEPT
$FW             vpn             ACCEPT
vpn             net             ACCEPT
vpn             loc             ACCEPT
fw              Wifi            ACCEPT
loc             vpn             ACCEPT
$FW             loc             ACCEPT          #Firewall to Local
loc             $FW             REJECT          $LOG
net             all             DROP            $LOG            10/sec:40
all             all             REJECT          $LOG
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS ONE - DO NOT REMOVE
```

Masq File

Although most of our internal systems use one-to-one NAT, my wife's system (192.168.1.4) uses IP Masquerading (actually SNAT) as do our wireless network systems and visitors with laptops.

The first entry allows access to the DSL modem and uses features introduced in Shorewall 2.1.1. The leading plus sign ("+_") causes the rule to be placed before rules generated by the /etc/shorewall/nat file below. The double colons ("::") cause the entry to be exempt from ADD_SNAT_ALIASES=Yes in my shorewall.conf file above.

```
#INTERFACE              SUBNET          ADDRESS         PROTO   PORT
+$EXT_IF::192.168.1.1   0.0.0.0/0       192.168.1.254
$EXT_IF:2               192.168.0.0/22  206.124.146.179
$DMZ_IF::               206.124.146.176 192.168.1.254   tcp     80
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

NAT File

```
#EXTERNAL               INTERFACE       INTERNAL        ALL INTERFACES  LOCAL
206.124.146.178         $EXT_IF:0       192.168.1.5     No              No
206.124.146.180         $EXT_IF:1       192.168.1.6     No              No
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

Proxy ARP File

I configure the host route to 206.124.146.177 on `eth1` in /etc/network/interfaces.

```
#ADDRESS                INTERFACE       EXTERNAL        HAVEROUTE          PERSISTENT
206.124.146.177         $DMZ_IF         $EXT_IF         yes
192.168.1.1             $EXT_IF         $INT_IF         yes # Allow access to DSL
modem from the local zone
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

Tunnels

```
#TYPE                   ZONE    GATEWAY         GATEWAY ZONE    PORT
openvpnserver:1194      net     0.0.0.0/0
openvpnserver:1194      Wifi    192.168.3.0/24
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

Actions File

```
#ACTION
Mirrors             #Accept traffic from the Shorewall Mirror sites
SSHKnock            #Port Knocking
#LAST LINE - ADD YOUR ENTRIES ABOVE THIS ONE - DO NOT REMOVE
```

action.Mirrors File

The *Mirrors* and *Mirrornets* ipsets define the set of Shorewall mirrors.

```
#TARGET  SOURCE         DEST            PROTO   DEST    SOURCE     ORIGINAL     RATE
#                                               PORT    PORT(S)    DEST         LIMIT
ACCEPT   +Mirrors
ACCEPT   +Mirrornets
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

Rules File (The shell variables are set in /etc/shorewall/params)

```
#################################################################################################################################################
#ACTION         SOURCE                          DEST                    PROTO   DEST
SOURCE          ORIGINAL        RATE    USER/
#                                                                               PORT
PORT(S)         DEST            LIMIT   GROUP
#################################################################################################################################################
SECTION NEW
REJECT:$LOG     loc                             net                     tcp     25
REJECT:$LOG     loc                             net                     udp
1025:1031
#
# Stop NETBIOS crap
#
REJECT          loc                             net                     tcp
137,445
REJECT          loc                             net                     udp
137:139

#
# Stop my idiotic work laptop from sending to the net with an HP source/dest IP
address
#
DROP            loc:!192.168.0.0/22             net
DROP            Wifi                            net:15.0.0.0/8
DROP            Wifi                            net:16.0.0.0/8
#################################################################################################################################################
# Local Network to Firewall
#
DROP            loc:!192.168.0.0/22             fw                      # Silently
drop traffic with an HP source IP from my XP box
ACCEPT          loc                             fw                      tcp
ssh,time,631,8080
ACCEPT          loc                             fw                      udp
161,ntp,631
DROP            loc                             fw                      tcp     3185
#SuSE Meta pppd
Ping/ACCEPT     loc                             fw
#################################################################################################################################################
# Roadwarriors to Firewall
#
ACCEPT          vpn                             fw                      tcp
ssh,time,631,8080
ACCEPT          vpn                             fw                      udp
161,ntp,631
Ping/ACCEPT     vpn                             fw
#################################################################################################################################################
# Local Network to DMZ
#
DNAT-           loc                             dmz:206.124.146.177:3128 \
                                                                        tcp     www
-               !206.124.146.177,192.168.1.1
DROP            loc:!192.168.0.0/22             dmz
ACCEPT          loc                             dmz                     udp
domain,xdmcp
ACCEPT          loc                             dmz                     tcp
www,smtp,smtps,domain,ssh,imap,https,imaps,ftp,10023,pop3,3128                  -
Ping/ACCEPT     loc                             dmz
#################################################################################################################################################
# Insecure Wireless to DMZ
#

ACCEPT          Wifi                            dmz                     udp
domain
ACCEPT          Wifi                            dmz                     tcp
domain
```

```
################################################################################################################################
# Insecure Wireless to Internet
#
ACCEPT          Wifi                            net                     udp     500
ACCEPT          Wifi                            net                     udp     4500
Ping/ACCEPT     Wifi                            net
################################################################################################################################
# Road Warriors to DMZ
#
ACCEPT          vpn                             dmz                     udp
domain
ACCEPT          vpn                             dmz                     tcp
www,smtp,smtps,domain,ssh,imap,https,imaps,ftp,10023,pop3       -
Ping/ACCEPT     vpn                             dmz
################################################################################################################################
# Internet to ALL -- drop NewNotSyn packets
#
dropNotSyn      net             fw              tcp
dropNotSyn      net             loc             tcp
dropNotSyn      net             dmz             tcp
################################################################################################################################
# Internet to DMZ
#
ACCEPT          net                             dmz                     udp
domain
ACCEPT          net                             dmz                     tcp
smtps,www,ftp,imaps,domain,https        -
ACCEPT          net                             dmz                     tcp     smtp
-               206.124.146.177,206.124.146.178
ACCEPT          net                             dmz                     udp
33434:33454
Mirrors         net                             dmz                     tcp     rsync
ACCEPT          net                             dmz                     tcp     22
Ping/ACCEPT     net                             dmz
################################################################################################################################
#
# Net to Local
#
# When I'm "on the road", the following two rules allow me VPN access back home using
PPTP.
#
DNAT            net                             loc:192.168.1.4         tcp     1729
DNAT            net                             loc:192.168.1.4         gre
ACCEPT          net:$OMAK                       loc:192.168.1.5         tcp     22
#
# Auth for IRC
#
ACCEPT          net                             loc:192.168.1.5         tcp     113
#
# Real Audio
#
ACCEPT          net                             loc:192.168.1.5         udp
6970:7170
#
# Overnet
#
#ACCEPT         net                             loc:192.168.1.5         tcp     4662
#ACCEPT         net                             loc:192.168.1.5         udp     12112
#
# OpenVPN
#
ACCEPT          net                             loc:192.168.1.5         udp     1194
#
# Silently Handle common probes
#
```

```
REJECT          net                             loc                     tcp
www,ftp,https
DROP            net                             loc                     icmp    8
#############################################################################################################################
# DMZ to Internet
#
ACCEPT          dmz                             net                     udp
domain,ntp
ACCEPT          dmz                             net                     tcp
smtp,domain,www,81,https,whois,echo,2702,21,2703,ssh,8080,cvspserver
REJECT:$LOG     dmz                             net                     udp
1025:1031
ACCEPT          dmz                             net:$POPSERVERS         tcp     pop3
#
#
# OpenVPN
#
ACCEPT          net                             loc:192.168.1.5         udp     1194
#
# Silently Handle common probes
#
REJECT          net                             loc                     tcp
www,ftp,https
DROP            net                             loc                     icmp    8
#############################################################################################################################
# DMZ to Firewall -- ntp & snmp, Silently reject Auth
#
ACCEPT          dmz                             fw                      udp     ntp
ntp
ACCEPT          dmz                             fw                      tcp
161,ssh
ACCEPT          dmz                             fw                      udp     161
REJECT          dmz                             fw                      tcp     auth
Ping/ACCEPT     dmz                             fw
#############################################################################################################################
# DMZ to Local Network
#
ACCEPT          dmz                             loc                     tcp
smtp,6001:6010
ACCEPT          dmz:206.124.146.177             loc:192.168.1.5,192.168.1.3 \
                                                                        tcp     111
ACCEPT          dmz:206.124.146.177             loc:192.168.1.5,192.168.1.3 \
                                                                        udp
Ping/ACCEPT     dmz                             loc
#############################################################################################################################
# Internet to Firewall
#
REJECT          net                             fw                      tcp
www,ftp,https
DROP            net                             fw                      icmp    8
ACCEPT          net                             fw                      udp
33434:33454
ACCEPT          net:$OMAK                       fw                      udp     ntp
ACCEPT          net                             fw                      tcp     auth
SSHKnock:info   net                             fw                      tcp
22,4320,4321,4322
#############################################################################################################################
# Firewall to Internet
#
ACCEPT          fw                              net:$NTPSERVERS         udp     ntp
ntp
#ACCEPT         fw                              net:$POPSERVERS         tcp     pop3
ACCEPT          fw                              net                     udp
domain
ACCEPT          fw                              net                     tcp
```

```
domain,www,https,ssh,1723,whois,1863,ftp,2702,2703,7
ACCEPT          fw                              net                     udp
33435:33535
ACCEPT          fw                              net                     icmp
REJECT:$LOG     fw                              net                     udp
1025:1031
DROP            fw                              net                     udp     ntp
Ping/ACCEPT     fw                              net
#################################################################################################################################################################
# Firewall to DMZ
#
ACCEPT          fw                              dmz                     tcp
www,ftp,ssh,smtp,993,465
ACCEPT          fw                              dmz                     udp
domain
REJECT          fw                              dmz                     udp
137:139
Ping/ACCEPT     fw                              dmz
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE

#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

/etc/shorewall/tcdevices

```
#INTERFACE      IN-BANDWITH     OUT-BANDWIDTH
$EXT_IF         1.5mbit         384kbit
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

/etc/shorewall/tcclasses

My traffic shaping configuration is basically the "WonderShaper" example from tc4shorewall with a little tweaking.

```
#INTERFACE      MARK    RATE            CEIL            PRIORITY        OPTIONS
$EXT_IF         10      full            ful             1               tcp-ack,tos-
minimize-delay
$EXT_IF         20      9*full/10       9*full/10       2               default
$EXT_IF         30      6*full/10       6*full/10       3
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

/etc/shorewall/tcrules

I give full bandwidth to my local systems -- the server gets throttled and rsync gets throttled even more.

Note

The class id for tc4shorewall-generated classes is *<device number>:<100 + mark value>* where the first device in `/etc/shorewall/tcdevices` is device number 1, the second is device number 2 and so on. The rules below are using the Netfilter CLASSIFY target to classify the traffic directly without having to first mark then classify based on the marks.

```
#MARK           SOURCE                  DEST            PROTO   PORT(S) CLIENT  USER
TEST
#                                                                       PORT(S)
1:110           192.168.0.0/22          $EXT_IF
1:130           206.124.146.177         $EXT_IF         tcp     -       873 #Rsync to
the Mirrors
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

Here is the output of **shorewall show tc** while the Shorewall mirrors were receiving updates via rsync and the link was otherwise idle. Note the rate limiting imposed by the 1:30 Class.

```
Shorewall-3.0.0-RC2 Traffic Control at gateway - Sat Oct 22 09:11:26 PDT 2005
```

```
...

Device eth2:
qdisc htb 1: r2q 10 default 120 direct_packets_stat 2 ver 3.17
 Sent 205450106 bytes 644093 pkts (dropped 0, overlimits 104779)
 backlog 20p
qdisc ingress ffff: ----------------
 Sent 160811382 bytes 498294 pkts (dropped 37, overlimits 0)
qdisc sfq 110: parent 1:110 limit 128p quantum 1514b flows 128/1024 perturb 10sec
 Sent 81718034 bytes 417516 pkts (dropped 0, overlimits 0)
qdisc sfq 120: parent 1:120 limit 128p quantum 1514b flows 128/1024 perturb 10sec
 Sent 61224535 bytes 177773 pkts (dropped 0, overlimits 0)
qdisc sfq 130: parent 1:130 limit 128p quantum 1514b flows 128/1024 perturb 10sec
 Sent 62507157 bytes 48802 pkts (dropped 0, overlimits 0)
 backlog 20p
class htb 1:110 parent 1:1 leaf 110: prio 1 quantum 4915 rate 384000bit ceil
384000bit burst 1791b/8 mpu 0b overhead 0b cburst 1791b/8 mpu 0b overhead 0b level 0
 Sent 81718034 bytes 417516 pkts (dropped 0, overlimits 0)
 rate 424bit
 lended: 417516 borrowed: 0 giants: 0
 tokens: 36864 ctokens: 36864

class htb 1:1 root rate 384000bit ceil 384000bit burst 1791b/8 mpu 0b overhead 0b
cburst 1791b/8 mpu 0b overhead 0b level 7
 Sent 205422474 bytes 644073 pkts (dropped 0, overlimits 0)
 rate 231568bit 19pps
 lended: 0 borrowed: 0 giants: 0
 tokens: -26280 ctokens: -26280

class htb 1:130 parent 1:1 leaf 130: prio 3 quantum 2944 rate 230000bit ceil
230000bit burst 1714b/8 mpu 0b overhead 0b cburst 1714b/8 mpu 0b overhead 0b level 0
 Sent 62507157 bytes 48802 pkts (dropped 0, overlimits 0)
 rate 230848bit 19pps backlog 18p
 lended: 48784 borrowed: 0 giants: 0
 tokens: -106401 ctokens: -106401

class htb 1:120 parent 1:1 leaf 120: prio 2 quantum 4416 rate 345000bit ceil
345000bit burst 1771b/8 mpu 0b overhead 0b cburst 1771b/8 mpu 0b overhead 0b level 0
 Sent 61224535 bytes 177773 pkts (dropped 0, overlimits 0)
 rate 1000bit
 lended: 177773 borrowed: 0 giants: 0
 tokens: 41126 ctokens: 41126

...
```

/etc/network/interfaces

This file is Debian-specific and defines the configuration of the network interfaces.

```
# The loopback network interface
auto lo
iface lo inet loopback

# DMZ interface
auto eth1
iface eth1 inet static
        address 206.124.146.176
        netmask 255.255.255.255
        broadcast 0.0.0.0
        up ip route add 206.124.146.177 dev eth1

# Internet interface
auto eth2
iface eth2 inet static
        address 206.124.146.176
```

```
        netmask 255.255.255.0
        gateway 206.124.146.254
        up ip route add 192.168.1.1 dev eth2

# Wireless network
auto eth0
iface eth0 inet static
        address 192.168.3.254
        netmask 255.255.255.0

# LAN interface
auto br0
iface br0 inet static
        address 192.168.1.254
        netmask 255.255.255.0
        pre-up /usr/sbin/openvpn --mktun --dev tap0
        pre-up /sbin/ip link set tap0 up
        pre-up /sbin/ip link set eth3 up
        pre-up /usr/sbin/brctl addbr br0
        pre-up /usr/sbin/brctl addif br0 eth3
        pre-up /usr/sbin/brctl addif br0 tap0
        up ip route add 224.0.0.0/4 dev br0
        post-down /usr/sbin/brctl delif br0 eth3
        post-down /usr/sbin/brctl delif br0 tap0
        post-down /usr/sbin/brctl delbr br0
        post-down /usr/sbin/openvpn --rmtun --dev tap0

# Unbridged LAN interface (Not started automatically)
iface eth3 inet static
        address 192.168.1.254
        netmask 255.255.255.0
        up ip route add 224.0.0.0/4 dev eth3

# Second Internet interface (Not started automatically -- address is duplicate of one
added by Shorewall to eth2)
iface eth4 inet static
        pre-up modprobe ne io=0x300 irq=10
        address 206.124.146.179
        netmask 255.255.255.0
```

/etc/openvpn/server.conf

Only the tunnel-mode OpenVPN configuration is described here -- the bridge is described in the OpenVPN documentation.

```
dev tun

local 206.124.146.176

server 192.168.2.0 255.255.255.0

dh dh1024.pem

ca /etc/certs/cacert.pem

crl-verify /etc/certs/crl.pem

cert /etc/certs/gateway.pem
key /etc/certs/gateway_key.pem

port 1194

comp-lzo

user nobody
group nogroup
```

```
keepalive 15 45
ping-timer-rem
persist-tun
persist-key

client-config-dir /etc/openvpn/clients
ccd-exclusive
client-to-client

verb 3
```

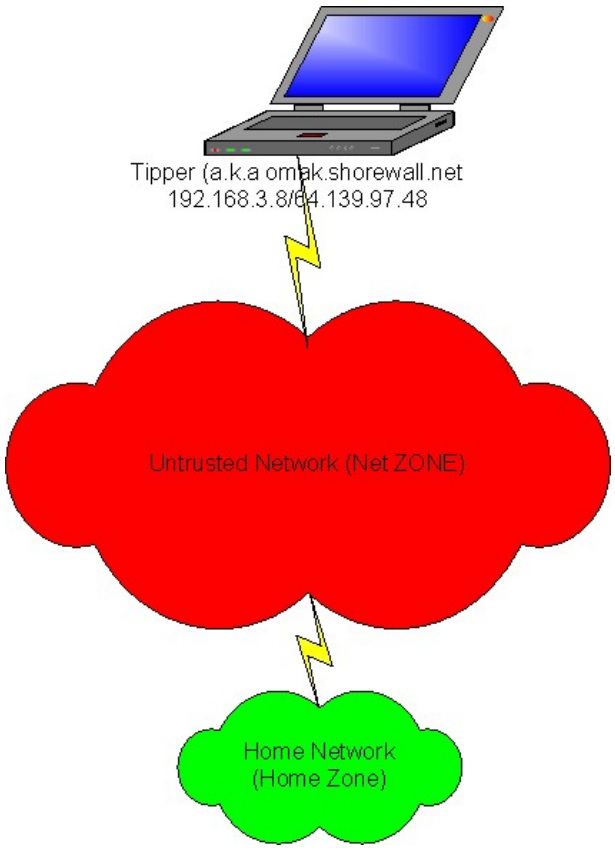
Tipper and Eastepnc6000 Configuration in the Wireless Network

Please find this information in the OpenVPN bridge mode documentation.

Tipper Configuration while on the Road

This laptop is either configured on our wireless network (192.168.3.8) or as a standalone system on the road.

Tipper's view of the world is shown in the following diagram:



zones

```
#ZONE   DISPLAY         COMMENTS
home    Home            Shorewall Network
net     Net             Internet
```

```
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

policy

```
#SOURCE         DEST            POLICY          LOG LEVEL       LIMIT:BURST
$FW             net             ACCEPT
$FW             home            ACCEPT
home            $FW             ACCEPT
net             home            NONE
home            net             NONE
net             all             DROP            info
# The FOLLOWING POLICY MUST BE LAST
all             all             REJECT          info
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

interfaces

```
#ZONE   INTERFACE       BROADCAST       OPTIONS
net     eth0            detect          dhcp,tcpflags
home    tun0            -
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

rules

```
#ACTION         SOURCE                  DEST    PROTO   DEST    SOURCE  ORIGINAL
RATE    USER/
#                                                       PORT    PORT(S) DEST
LIMIT   GROUP
ACCEPT          net                     $FW     icmp    8
ACCEPT          net                     $FW     tcp     22
ACCEPT          net                     $FW     tcp     4000:4100
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

/etc/openvpn/home.conf

```
dev tun
remote gateway.shorewall.net
up /etc/openvpn/home.up

tls-client
pull

ca /etc/certs/cacert.pem

cert /etc/certs/tipper.pem
key /etc/certs/tipper_key.pem

port 1194

user nobody
group nogroup

comp-lzo

ping 15
ping-restart 45
ping-timer-rem
persist-tun
persist-key

verb 3
```

/etc/openvpn/home.up

```
#!/bin/bash

ip route add 192.168.1.0/24 via $5     #Access to Home Network
ip route add 206.124.146.177/32 via $5 #So that DNS names will resolve in my
                                       #Internal Bind 9 view because the source IP
will
                                       #be in 192.168.2.0/24
```

Proxy ARP

Tom Eastep

Copyright © 2001-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-10-16

Table of Contents

Example
ARP cache

Proxy ARP (RFC 1027) is a way to make a machine physically located on one network appear to be logically part of a different physical network connected to the same router/firewall. Typically it allows us to hide a machine with a public IP address on a private network behind a router, and still have the machine appear to be on the public network "in front of" the router. The router "proxys" ARP requests and all network traffic to and from the hidden machine to make this fiction possible.

Consider a router with two interface cards, one connected to a public network PUBNET and one connected to a private network PRIVNET. We want to hide a server machine on the PRIVNET network but have it accessible from the PUBNET network. The IP address of the server machine lies in the PUBNET network, even though we are placing the machine on the PRIVNET network behind the router.

By enabling proxy ARP on the router, any machine on the PUBNET network that issues an ARP "who has" request for the server's MAC address will get a proxy ARP reply from the router containing the router's MAC address. This tells machines on the PUBNET network that they should be sending packets destined for the server via the router. The router forwards the packets from the machines on the PUBNET network to the server on the PRIVNET network.

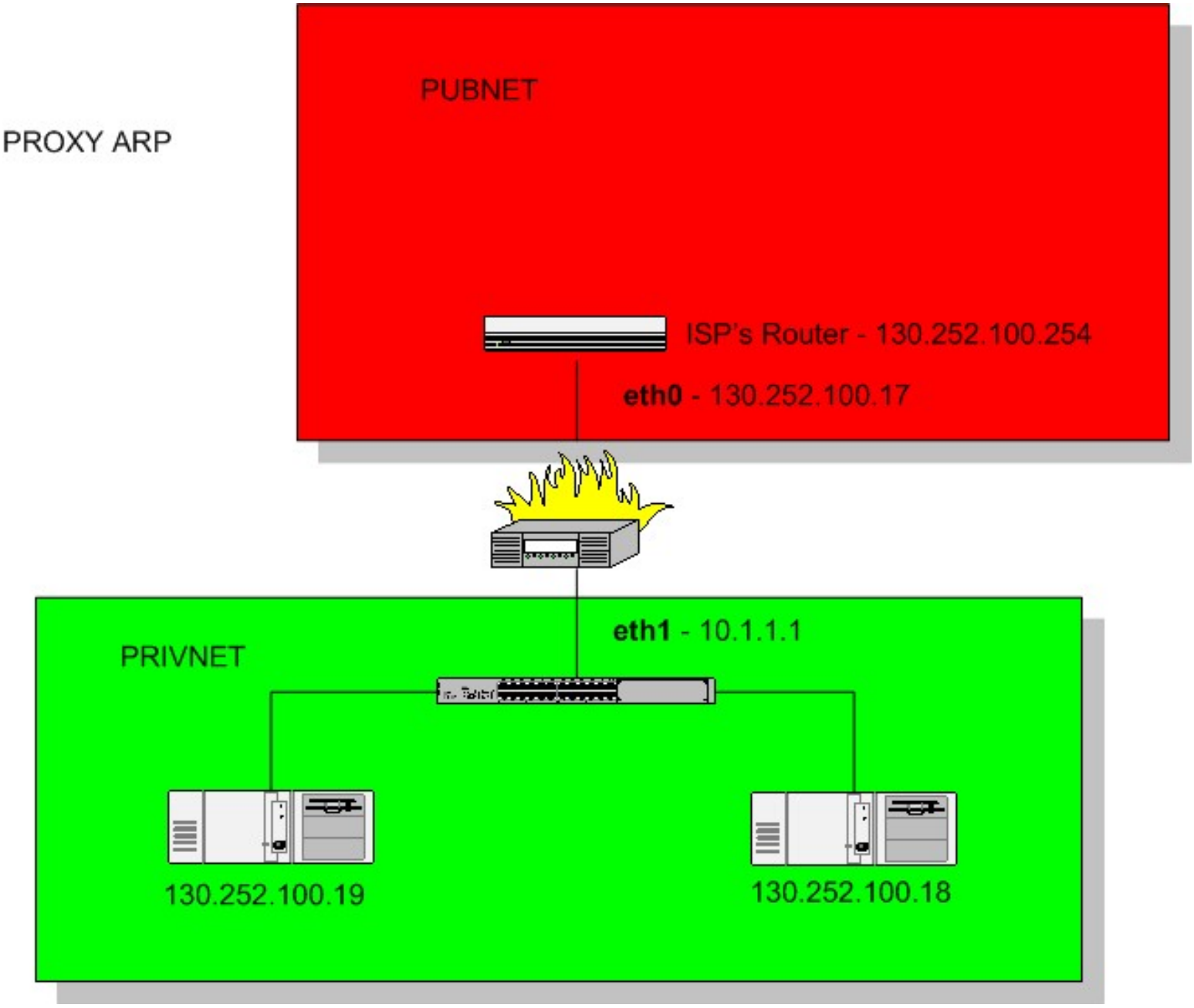
Similarly, when the server on the PRIVNET network issues a "who has" request for any machines on the PUBNET network, the router provides its own MAC address via proxy ARP. This tells the server to send packets for machines on the PUBNET network via the router. The router forwards the packets from the server on the PRIVNET network to the machines on the PUBNET network.

The proxy ARP provided by the router allows the server on the PRIVNETnetwork to appear to be on the PUBNET network. It lets the router pass ARP requests and other network packets in both directions between the server machine and the PUBNET network, making the server machine appear to be connected to the PUBNET network even though it is on the PRIVNET network hidden behind the router.

Before you try to use this technique, I strongly recommend that you read the Shorewall Setup Guide.

Example

The following figure represents a Proxy ARP environment.



Proxy ARP can be used to make the systems with addresses 130.252.100.18 and 130.252.100.19 appear to be on the upper (130.252.100.*) subnet. Assuming that the upper firewall interface is eth0 and the lower interface is eth1, this is accomplished using the following entries in `/etc/shorewall/proxyarp`:

```
#ADDRESS          INTERFACE   EXTERNAL  HAVEROUTE  PERSISTENT
130.252.100.18    eth1        eth0      no         yes
130.252.100.19    eth1        eth0      no         yes
```

Be sure that the internal systems (130.242.100.18 and 130.252.100.19 in the above example) are not included in any specification in `/etc/shorewall/masq` or `/etc/shorewall/nat`.

Note

I've used an RFC1918 IP address for eth1 - that IP address is largely irrelevant (see below).

The lower systems (130.252.100.18 and 130.252.100.19) should have their subnet mask and default gateway configured exactly the same way that the Firewall system's eth0 is configured. In other words, they should be configured just like they would be if they were parallel to the firewall rather than behind it.

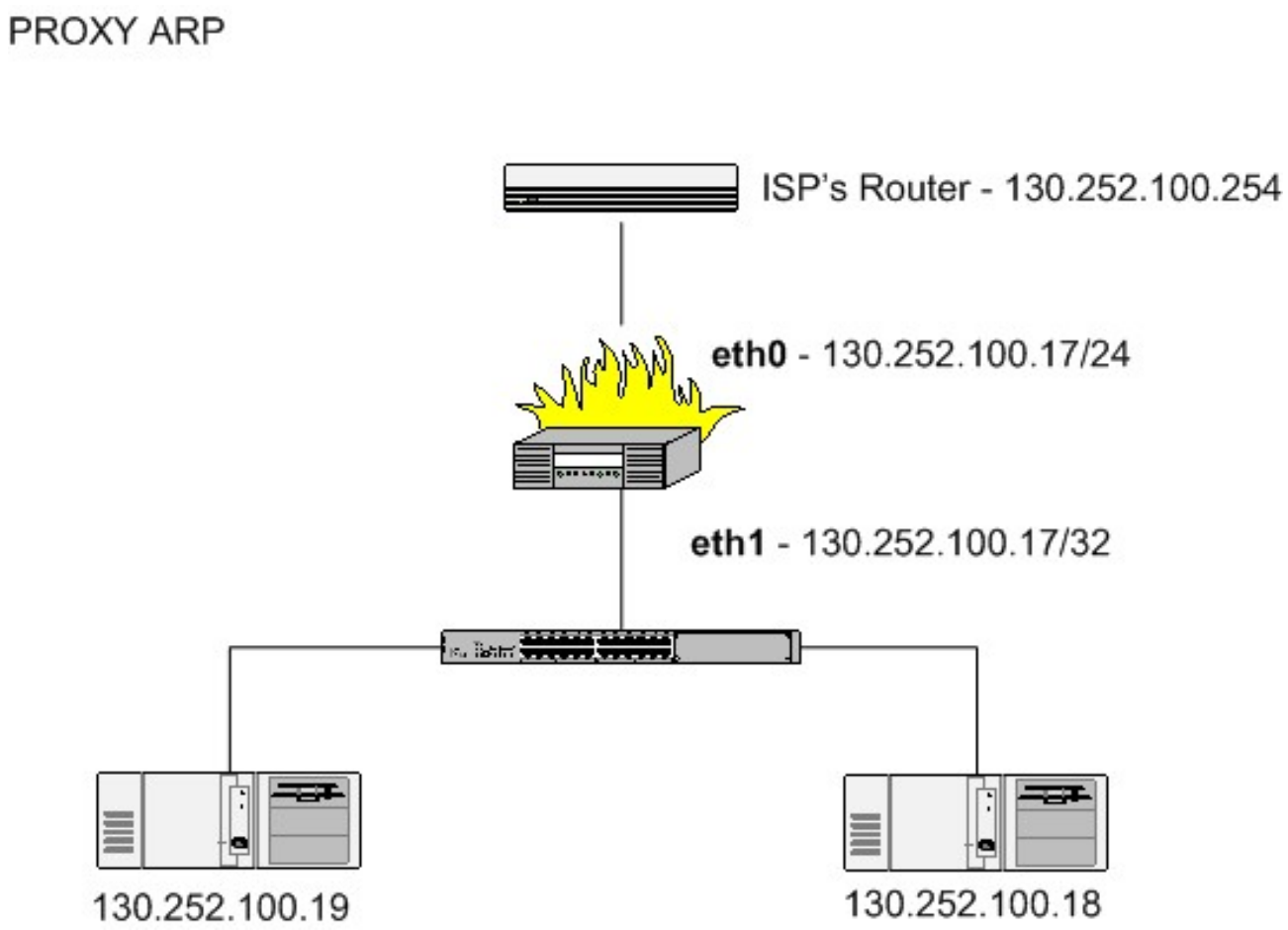
Warning

Do not add the Proxy ARP'ed address(es) (130.252.100.18 and 130.252.100.19 in the above example) to the external interface (eth0 in this example) of the firewall.

Note

It should be stressed that entries in the proxyarp file do not automatically enable traffic between the external network and the internal host(s) — such traffic is still subject to your policies and rules.

While the address given to the firewall interface is largely irrelevant, one approach you can take is to make that address the same as the address of your external interface!



In the diagram above, `eth1` has been given the address 130.252.100.17, the same as `eth0`. Note though that the VLSM is 32 so there is no network associated with this address. This is the approach that I take with my DMZ.

To permit internet hosts to connect to the local systems, you use ACCEPT rules. For example, if you run a web server on 130.252.100.19 which you have configured to be in the **loc** zone then you would need this entry in /etc/shorewall/rules:

```
#ACTION SOURCE          DEST                     PROTO   DEST
#                                                        PORT
ACCEPT  net             loc:130.252.100.19       tcp     80
```

Warning

Your distribution's network configuration GUI may not be capable of configuring a device in this way. It may complain about the duplicate address or it may configure the address incorrectly. Here is what the above configuration should look like when viewed using **ip** (the line "inet 130.252.100.17/32 scope global eth1" is the most important):

```
gateway:~# ip addr ls eth1
3: eth1: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:a0:cc:d1:db:12 brd ff:ff:ff:ff:ff:ff
    inet 130.252.100.17/32 scope global eth1
gateway:~#
```

Note in particular that there is no broadcast address. Here is an `ifcfg-eth-id-00:a0:cc:d1:db:12` file from SuSE that produces this result (Note: SuSE ties the configuration file to the card by embedding the card's MAC address in the file name):

```
BOOTPROTO='static'
BROADCAST='130.252.100.17'
IPADDR='130.252.100.17'
MTU=''
NETMASK='255.255.255.255'
NETWORK='130.252.100.17'
REMOTE_IPADDR=''
STARTMODE='onboot'
UNIQUE='8otl.IPwRm6bNMRD'
_nm_name='bus-pci-0000:00:04.0'
```

Here is an excerpt from a Debian /etc/network/interfaces file that does the same thing:

```
...
auto eth1
iface eth1 inet static
        address 130.252.100.17
        netmask 255.255.255.255
        broadcast 0.0.0.0
...
```

ARP cache

A word of warning is in order here. ISPs typically configure their routers with a long ARP cache timeout. If you move a system from parallel to your firewall to behind your firewall with Proxy ARP, it will probably be HOURS before that system can communicate with the internet.

If you sniff traffic on the firewall's external interface, you can see incoming traffic for the internal system(s) but the

traffic is never sent out the internal interface.

You can determine if your ISP's gateway ARP cache is stale using ping and tcpdump. Suppose that we suspect that the gateway router has a stale ARP cache entry for 130.252.100.19. On the firewall, run tcpdump as follows:

```
tcpdump -nei eth0 icmp
```

Now from 130.252.100.19, ping the ISP's gateway (which we will assume is 130.252.100.254):

```
ping 130.252.100.254
```

We can now observe the tcpdump output:

```
13:35:12.159321 0:4:e2:20:20:33 0:0:77:95:dd:19 ip 98: 130.252.100.19 >
130.252.100.254: icmp: echo request (DF)
13:35:12.207615 0:0:77:95:dd:19 0:c0:a8:50:b2:57 ip 98: 130.252.100.254 >
130.252.100.177 : icmp: echo reply
```

Notice that the source MAC address in the echo request is different from the destination MAC address in the echo reply!! In this case 0:4:e2:20:20:33 was the MAC of the firewall's eth0 NIC while 0:c0:a8:50:b2:57 was the MAC address of the system on the lower left. In other words, the gateway's ARP cache still associates 130.252.100.19 with the NIC in that system rather than with the firewall's eth0.

If you have this problem, there are a couple of things that you can try:

1. A reading of *TCP/IP Illustrated, Vol 1* by Stevens reveals[1] that a “gratuitous” ARP packet should cause the ISP's router to refresh their ARP cache (section 4.7). A gratuitous ARP is simply a host requesting the MAC address for its own IP; in addition to ensuring that the IP address isn't a duplicate...

   > if the host sending the gratuitous ARP has just changed its hardware address..., this packet causes any other host...that has an entry in its cache for the old hardware address to update its ARP cache entry accordingly.

   Which is, of course, exactly what you want to do when you switch a host from being exposed to the Internet to behind Shorewall using proxy ARP (or one-to-one NAT for that matter). Happily enough, recent versions of Redhat's iputils package include “arping”, whose “-U” flag does just that:

   ```
   arping -U -I <net if> <newly proxied IP>
   arping -U -I eth0 66.58.99.83             # for example
   ```

   Stevens goes on to mention that not all systems respond correctly to gratuitous ARPs, but googling for “arping -U” seems to support the idea that it works most of the time.

   To use arping with Proxy ARP in the above example, you would have to:

   ```
   shorewall clear
   ip addr add 130.252.100.18 dev eth0
   ip addr add 130.252.100.19 dev eth0
   arping -U -c 10 -I eth0 130.252.100.18
   ```

```
arping -U -c 10 -I eth0 130.252.100.19
ip addr del 130.252.100.18 dev eth0
ip addr del 130.252.100.19 dev eth0
shorewall start
```

2. You can call your ISP and ask them to purge the stale ARP cache entry but many either can't or won't purge individual entries.

Warning

There are two distinct versions of **arping** available:

1. **arping** by Thomas Habets (Debian package *arping*).
2. **arping** as part of the iputils package by Alexey Kuznetsov (Debian package *iputils-arping*).

You want the second one by Alexey Kuznetsov.

---

[1] Courtesy of Bradey Honsinger

Shorewall and Ipsets

Tom Eastep

Copyright © 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-09-30

Table of Contents

What are Ipsets?
Shorewall Support for Ipsets
Defining Dynamic Zones using Ipsets

What are Ipsets?

Ipsets are an extention to Netfilter/iptables that are currently available in Patch-O-Matic-ng (http://www.netfilter.org). Using ipsets requires that you patch your kernel and iptables and that you build and install the ipset utility from http://people.netfilter.org/kadlec/ipset/.

Ipset allows you to create one or more named sets of addresses then use those sets to define Netfilter/iptables rules. Possible uses of ipsets include:

1. Blacklists. Ipsets provide an effecient way to represent large sets of addresses and you can maintain the lists without the need to restart or even refresh your Shorewall configuration.
2. Zone definition. Using the /etc/shorewall/hosts file, you can define a zone based on the (dynamic) contents of an ipset. Again, you can then add or delete addresses to the ipset without restarting Shorewall.

See the ipsets site (URL above) for additional information about ipsets.

Shorewall Support for Ipsets

Support for ipsets was introduced in Shorewall version 2.3.0. In most places where a host or network address may be used, you may also use the name of an ipset prefaced by "+".

Example: "+Mirrors"

The name of the set may optionally followed by:

a. a number from 1 to 6 enclosed in square brackets ([]) -- this number indicates the maximum number of ipset binding levels that are to be matched. Depending on the context where the ipset name is used, either all "src" or all "dst" matches will be used.

   Example: "+Mirrors[4]"
b. a series of "src" and "dst" options separated by commas and inclosed in square brackets ([]). These will be passed directly to iptables in the generated --set clause. See the ipset documentation for details.

   Example: "+Mirrors[src,dst,src]"

   Note that "+Mirrors[4]" used in the SOURCE column of the rules file is equivalent to "+Mirrors[src,src,src,src]".

To generate a negative match, prefix the "+" with "!" as in "!+Mirrors".

Example 1: Blacklist all hosts in an ipset named "blacklist"

`/etc/shorewall/blacklist`

```
#ADDRESS/SUBNET          PROTOCOL         PORT
+blacklist
```

Example 2: Allow SSH from all hosts in an ipset named "sshok:

`/etc/shorewall/rules`

```
#ACTION       SOURCE       DEST      PROTO     DEST PORT(S)
ACCEPT        +sshok       $FW       tcp       22
```

Shorewall can automatically manage the contents of your ipsets for you. If you specify SAVE_IPSETS=Yes in /etc/shorewall/shorewall.conf then "shorewall save" will save the contents of your ipsets. The file where the sets are saved is formed by taking the name where the Shorewall configuration is stored and appending "-ipsets". So if you enter the command "shorewall save standard" then your Shorewall

Regardless of the setting of SAVE_IPSETS, the **shorewall -f start** and **shorewall restore** commands will restore the ipset contents corresponding to the Shorewall configuration restored provided that the saved Shorewall configuration specified exists.

For example, **shorewall restore standard** would restore the ipset contents from `/var/lib/shorewall/standard-ipsets` provided that `/var/lib/shorewall/standard` exists and is executable and that `/var/lib/shorewall/standard-ipsets` exists and is executable.

Also regardless of the setting of SAVE_IPSETS, the **shorewall forget** command will purge the saved ipset information (if any) associated with the saved shorewall configuration being removed.

You can also associate ipset contents with Shorewall configuration directories using the following command:

```
ipset -S > <config directory>/ipsets
```

Example:

```
ipset -S > /etc/shorewall/ipsets
```

When you start or restart Shorewall (including using the **try** command) from the configuration directory, your ipsets will be configured from the saved ipsets file. Once again, this behavior is independent of the setting of SAVE_IPSETS.

As mentioned above, ipsets are well suited for large blacklists. You can maintain your blacklist using the 'ipset' utility without ever having to restart or refresh Shorewall. If you use the SAVE_IPSETS=Yes feature just be sure to "shorewall save" after altering the blacklist ipset(s). Example:

`/etc/shorewall/blacklist`:

```
#ADDRESS/SUBNET         PROTOCOL        PORT
+Blacklist[src,dst]
+Blacklistnets[src,dst]
```

Create the blacklist ipsets using:

```
ipset -N Blacklist iphash
ipset -N Blacklistnets nethash
```

Add entries:

```
ipset -A Blacklist 206.124.146.177
ipset -A Blacklistnets 206.124.147.0/24
```

To allow entries for individual ports:

```
ipset -N SMTP portmap --from 1 --to 31
ipset -A SMTP 25

ipset -A Blacklist 206.124.146.177
ipset -B Blacklist 206.124.146.177 -b SMTP
```

Now only port 25 will be blocked from 206.124.146.177.

Defining Dynamic Zones using Ipsets

The use of ipsets provides a much better way to define dynamic zones than is provided by the native Shorewall implementation. To define a dynamic zone of hosts **dyn** that interface through interface eth3, use:

/etc/shorewall/zones:

```
#ZONE          TYPE          OPTIONS             IN OPTIONS        OUT OPTIONS
dyn            ipv4
```

/etc/shorewall/interfaces:

```
#ZONE          INTERFACE      OPTIONS
-              eth3           …
```

/etc/shorewall/hosts:

```
#ZONE          HOSTS          OPTIONS
dyn            eth3:+Dyn
```

Now create an ipmap named **Dyn** and you're all set. You can add and delete addresses from Dyn without having to touch Shorewall.

Shorewall and the 2.6 Linux Kernel

Tom Eastep

Copyright © 2003, 2004, 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-01-14

Table of Contents

General
IPSEC

General

Shorewall is compatible with the Linux 2.6 kernel series and contains support for the following features that are added in that series:

1. NETMAP Target Support.
2. Bridge/Firewall Support (physdev match support).
3. CLASSIFY Target Support.

IPSEC

The 2.6 Linux kernel introduces a new implementation of IPSEC which eliminates the `ipsecN` device names. Netfilter/iptables support for this new implementation is incomplete unless your kernel has been patched. For unpatched kernels, see the Shorewall IPSEC documentation (Shorewall support for IPSEC with unpatched 2.6 kernels is very limited). For patched 2.6 kernels (including those supplied with SuSE™ 9.2) see the Kernel 2.6 IPSEC documentation.

Network Mapping

Tom Eastep

Copyright © 2004-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or mify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-05-19

Table of Contents

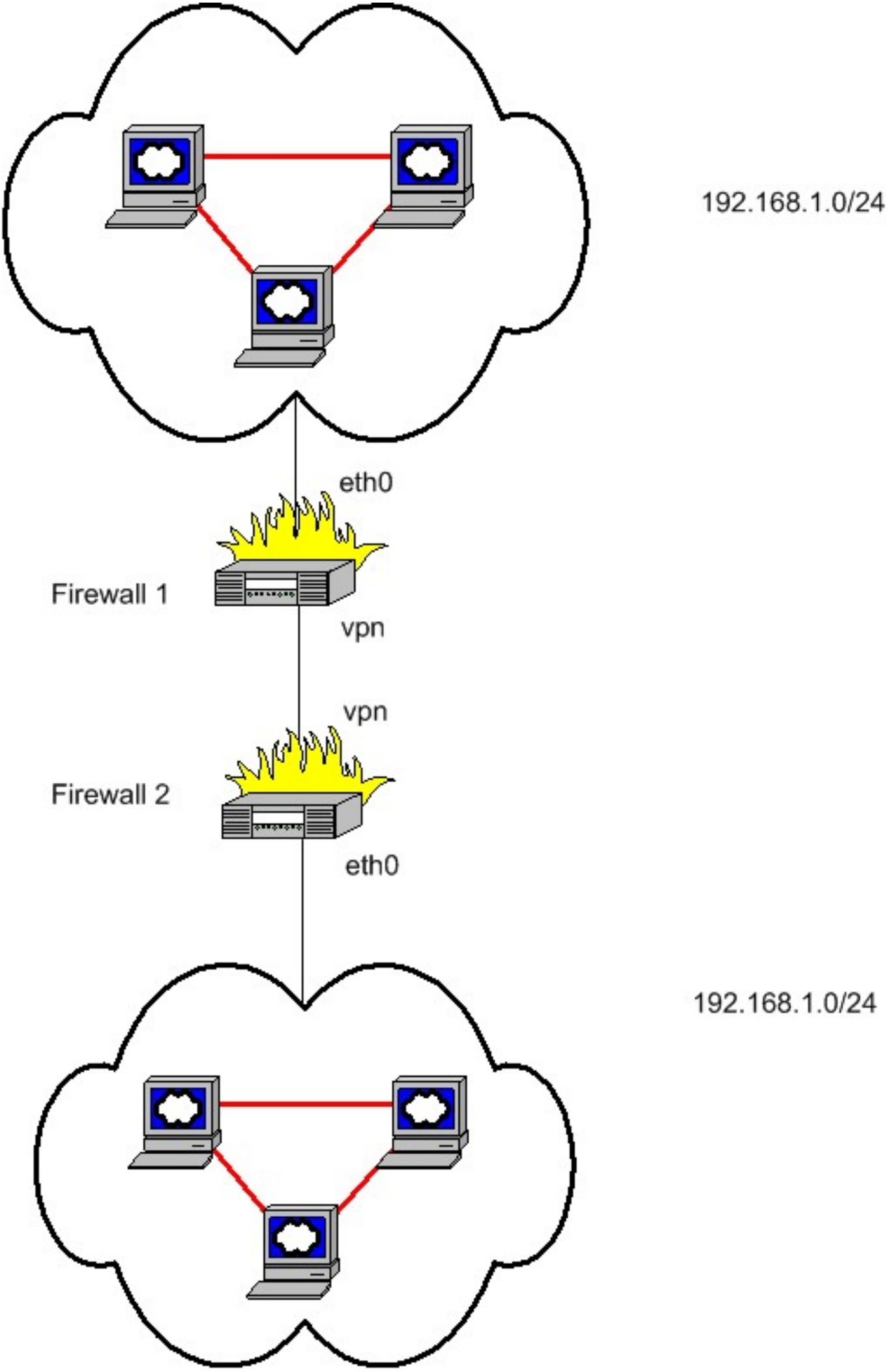
Why use Network Mapping
Solution
Author's Notes
Can't I do this with one router? Why do I need two?

Why use Network Mapping

Network Mapping is most often used to resolve IP address conflicts. Suppose that two organizations, A and B, need to be linked and that both organizations have allocated the 192.168.1.0/24 subnetwork. There is a need to connect the two networks so that all systems in A can access the 192.168.1.0/24 network in B and vice versa without any re-addressing.

Solution

Shorewall NETMAP support is designed to supply a solution. The basic situation is as shown in the following diagram.


192.168.1.0/24
eth0
Firewall 1
vpn
vpn
Firewall 2
eth0
192.168.1.0/24

While the link between the two firewalls is shown here as a VPN, it could be any type of interconnection that allows routing of RFC 1918 traffic.

The systems in the top cloud will access the 192.168.1.0/24 subnet in the lower cloud using addresses in another unused /24. Similarly, the systems in the bottom cloud will access the 192.168.1.0/24 subnet in the upper cloud using a second unused /24.

In order to apply this solution:

- You must be running Shorewall 2.0.1 Beta 2 or later.
- Your kernel must have NETMAP support. 2.6 Kernels have NETMAP support without patching while 2.4 kernels must be patched using Patch-O-Matic from netfilter.org.
- NETMAP support must be enabled in your kernel (CONFIG_IP_NF_TARGET_NETMAP=m or CONFIG_IP_NF_TARGET_NETMAP=y).
- Your iptables must have NETMAP support. NETMAP support is available in iptables 1.2.9 and later.

Network mapping is defined using the `/etc/shorewall/netmap` file. Columns in this file are:

TYPE

Must be DNAT or SNAT.

If DNAT, traffic entering INTERFACE and addressed to NET1 has it's destination address rewritten to the corresponding address in NET2.

If SNAT, traffic leaving INTERFACE with a source address in NET1 has it's source address rewritten to the corresponding address in NET2.

NET1

Must be expressed in CIDR format (e.g., 192.168.1.0/24).

INTERFACE

A firewall interface. This interface must have been defined in `/etc/shorewall/interfaces`.

NET2

A second network expressed in CIDR format.

Referring to the figure above, lets suppose that systems in the top cloud are going to access the 192.168.1.0/24 network in the bottom cloud using addresses in 10.10.10.0/24 and that systems in the bottom could will access 192.168.1.0/24 in the top could using addresses in 10.10.11.0.

Important

You must arrange for routing as follows:

- Traffic from the top cloud to 10.10.10.0/24 must be routed to eth0 on firewall 1.
- Firewall 1 must route traffic to 10.10.10.0/24 through firewall 2.
- Traffic from the bottom cloud to 10.10.11.0/24 must be routed to eth0 on firewall 2.
- Firewall 2 must route traffic to 10.10.11.0/24 through firewall 1.

The entries in `/etc/shorewall/netmap` in firewall1 would be as follows:

```
#TYPE NET1            INTERFACE         NET2
SNAT  192.168.1.0/24 vpn                10.10.11.0/24        #RULE 1A
DNAT  10.10.11.0/24  vpn                192.168.1.0/24       #RULE 1B
```

The entry in `/etc/shorewall/netmap` in firewall2 would be:

```
#TYPE NET1            INTERFACE         NET2
DNAT  10.10.10.0/24  vpn                192.168.1.0/24       #RULE 2A
SNAT  192.168.1.0/24 vpn                10.10.10.0/24        #RULE 2B
```

Example 1. 192.168.1.4 in the top cloud connects to 192.168.1.27 in the bottom cloud

In order to make this connection, the client attempts a connection to 10.10.10.27. The following table shows how the source and destination IP addresses are modified as requests are sent and replies are returned. The RULE column refers to the above `/etc/shorewall/netmap` entries and gives the rule which transforms the source and destination IP addresses to those shown on the next line.

FROM	TO	SOURCE IP ADDRESS	DESTINATION IP ADDRESS	RULE
192.168.1.4 in upper cloud	Firewall 1	192.168.1.4	10.10.10.27	1A
Firewall 1	Firewall 2	10.10.11.4	10.10.10.27	2A
Filrewall 2	192.168.1.27 in lower cloud	10.10.11.4	192.168.1.27	
192.168.1.27 in the lower cloud	Firewall 2	192.168.1.27	10.10.11.4	2B
Firewall 2	Firewall 1	10.10.10.27	10.10.11.4	1B
Firewall 1	192.168.1.4 in upper cloud	10.10.10.27	192.168.1.4	

Author's Notes

This could all be made a bit simpler by eliminating the TYPE field and have Shorewall generate both the SNAT and DNAT rules from a single entry. I have chosen to include the TYPE in order to make the implementation a bit more flexible. If you find cases where you can use an SNAT or DNAT entry by itself, please let me know and I'll add the example to this page.

In the previous section, the table in the example contains a bit of a lie. Because of Netfilter's connection tracking, rules 2B and 1B aren't needed to handle the replies. They ARE needed though for hosts in the bottom cloud to be able to establish connections with the 192.168.1.0/24 network in the top cloud.

Can't I do this with one router? Why do I need two?

The single router would have to be able to route to two different 192.168.1.0/24 networks. In Netfilter parlance, that would mean that the destination IP address would have to be rewritten after the packet had been routed; Netfilter doesn't have that capability.

Note that if you do it with two routers, then adding a third is easy. There's no reason why you can't have yet another network that is 192.168.1.0/24 on the inside, but you can allocated it 10.10.12.0/24 for everybody else.

Shorewall Traffic Accounting

Tom Eastep

Copyright © 2003-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-11-02

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Shorewall accounting rules are described in the file /etc/shorewall/accounting. By default, the accounting rules are placed in a chain called "accounting" and can thus be displayed using "shorewall show accounting". All traffic passing into, out of or through the firewall traverses the accounting chain including traffic that will later be rejected by interface options such as "tcpflags" and "maclist". If your kernel doesn't support the connection tracking match extension (Kernel 2.4.21) then some traffic rejected under "norfc1918" will not traverse the accounting chain.

The columns in the accounting file are as follows:

- **ACTION** - What to do when a match is found. Possible values are:
  - COUNT- Simply count the match and continue trying to match the packet with the following accounting rules
  - DONE- Count the match and don't attempt to match any following accounting rules.
  - *<chain>* - The name of a chain to jump to. Shorewall will create the chain automatically. If the name of the chain is followed by ":COUNT" then a COUNT rule matching this rule will automatically be added to <chain>. Chain names must start with a letter, must be composed of letters and digits, and may contain underscores ("_") and periods (."). Beginning with Shorewall version 1.4.8, chain names man also contain embedded dashes ("-") and are not required to start with a letter.
- **CHAIN** - The name of the chain where the accounting rule is to be added. If empty or "-" then the "accounting" chain is assumed.
- **SOURCE** - Packet Source. The name of an interface, an address (host or net) or an interface name followed by ":" and a host or net address.
- **DESTINATION** - Packet Destination Format the same as the SOURCE column.
- **PROTOCOL** - A protocol name (from `/etc/protocols`), a protocol number or "ipp2p". For "ipp2p", your kernel and iptables must have ipp2p match support from Netfilter Patch_o_matic_ng.
- **DEST PORT** - Destination Port number. Service name from `/etc/services` or port number. May only be specified if the protocol is TCP or UDP (6 or 17). If the PROTOCOL is "ipp2p", then this column is interpreted as an ipp2p option without the leading "--" (default "ipp2p"). For a list of value ipp2p options, as root type **iptables -m ipp2p --help**.
- **SOURCE PORT**- Source Port number. Service name from /etc/services or port number. May only be specified if the protocol is TCP or UDP (6 or 17).
- **USER/GROUP** - This column may only be non-empty if the CHAIN is OUTPUT. The column may contain:

```
[!][<user name or number>][:<group name or number>][+<program name>]
```

When this column is non-empty, the rule applies only if the program generating the output is running under the effective

<user> and/or <group> specified (or is NOT running under that id if "!" is given).

Examples:

joe #program must be run by joe
:kids #program must be run by a member of the 'kids' group.
!:kids #program must not be run by a member of the 'kids' group
+upnpd #program named upnpd (This feature was removed from Netfilter in kernel version 2.6.14).

In all columns except ACTION and CHAIN, the values “-”,“any” and “all” are treated as wild-cards.

The accounting rules are evaluated in the Netfilter “filter” table. This is the same environment where the “rules” file rules are evaluated and in this environment, DNAT has already occurred in inbound packets and SNAT has not yet occurred on outbound ones.

Accounting rules are not stateful -- each rule only handles traffic in one direction. For example, if eth0 is your internet interface and you have a web server in your DMZ connected to eth1 then to count HTTP traffic in both directions requires two rules:

```
        #ACTION CHAIN   SOURCE  DESTINATION     PROTOCOL        DEST
SOURCE
        #                                                       PORT            PORT
        DONE    -       eth0    eth1            tcp             80
        DONE    -       eth1    eth0            tcp             -               80
```

Associating a counter with a chain allows for nice reporting. For example:

```
        #ACTION         CHAIN   SOURCE  DESTINATION     PROTOCOL        DEST
SOURCE
        #                                                               PORT
PORT
        web:COUNT       -       eth0    eth1            tcp             80
        web:COUNT       -       eth1    eth0            tcp             -
80
        web:COUNT       -       eth0    eth1            tcp             443
        web:COUNT       -       eth1    eth0            tcp             -
443
        DONE            web
```

Now “shorewall show web” will give you a breakdown of your web traffic:

```
     [root@gateway shorewall]# shorewall show web
     Shorewall-1.4.6-20030821 Chain web at gateway.shorewall.net - Wed Aug 20
09:48:56 PDT 2003

     Counters reset Wed Aug 20 09:48:00 PDT 2003

     Chain web (4 references)
     pkts bytes target     prot opt in     out     source               destination
       11  1335            tcp  --  eth0   eth1    0.0.0.0/0            0.0.0.0/0
tcp dpt:80
       18  1962            tcp  --  eth1   eth0    0.0.0.0/0            0.0.0.0/0
tcp spt:80
        0     0            tcp  --  eth0   eth1    0.0.0.0/0            0.0.0.0/0
tcp dpt:443
        0     0            tcp  --  eth1   eth0    0.0.0.0/0            0.0.0.0/0
```

```
tcp spt:443
       29  3297 RETURN     all  --  *      *       0.0.0.0/0           0.0.0.0/0
       [root@gateway shorewall]#
```

Here is a slightly different example:

```
        #ACTION         CHAIN   SOURCE  DESTINATION     PROTOCOL        DEST
SOURCE
        #                                                               PORT
PORT
        web             -       eth0    eth1            tcp             80
        web             -       eth1    eth0            tcp             -
80
        web             -       eth0    eth1            tcp             443
        web             -       eth1    eth0            tcp             -
443
        COUNT           web     eth0    eth1
        COUNT           web     eth1    eth0
```

Now “shorewall show web” simply gives you a breakdown by input and output:

```
     [root@gateway shorewall]# shorewall show accounting web
     Shorewall-1.4.6-20030821 Chains accounting web at gateway.shorewall.net - Wed
Aug 20 10:27:21 PDT 2003

     Counters reset Wed Aug 20 10:24:33 PDT 2003

     Chain accounting (3 references)
         pkts bytes target     prot opt in     out     source
destination
         8767  727K web        tcp  --  eth0   eth1    0.0.0.0/0           0.0.0.0/0
tcp dpt:80
            0     0 web        tcp  --  eth0   eth1    0.0.0.0/0           0.0.0.0/0
tcp dpt:443
        11506   13M web        tcp  --  eth1   eth0    0.0.0.0/0           0.0.0.0/0
tcp spt:80
            0     0 web        tcp  --  eth1   eth0    0.0.0.0/0           0.0.0.0/0
tcp spt:443

     Chain web (4 references)
         pkts bytes target     prot opt in     out     source
destination
         8767  727K            all  --  eth0   eth1    0.0.0.0/0           0.0.0.0/0
        11506   13M            all  --  eth1   eth0    0.0.0.0/0           0.0.0.0/0
     [root@gateway shorewall]#
```

Here's how the same example would be constructed on an HTTP server with only one interface (eth0).

Caution

READ THE ABOVE CAREFULLY -- IT SAYS **SERVER**. If you want to account for web browsing, you have to reverse the rules below.

```
        #ACTION         CHAIN   SOURCE  DESTINATION     PROTOCOL        DEST
SOURCE
        #                                                               PORT
```

```
PORT
        web             -       eth0    -               tcp             80
        web             -       -       eth0            tcp             -
80
        web             -       eth0    -               tcp             443
        web             -       -       eth0            tcp             -
443
        COUNT           web     eth0
        COUNT           web     -       eth0
```

Note that with only one interface, only the SOURCE (for input rules) or the DESTINATION (for output rules) is specified in each rule.

Here's the output:

```
     [root@mail shorewall]# shorewall show accounting web Shorewall-1.4.7
     Chains accounting web at mail.shorewall.net - Sun Oct 12 10:27:21 PDT 2003

     Counters reset Sat Oct 11 08:12:57 PDT 2003

     Chain accounting (3 references)
      pkts bytes target     prot opt in     out     source               destination
      8767  727K web        tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/0
tcp dpt:80
     11506   13M web        tcp  --  *      eth0    0.0.0.0/0            0.0.0.0/0
tcp spt:80
         0     0 web        tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/0
tcp dpt:443
         0     0 web        tcp  --  *      eth0    0.0.0.0/0            0.0.0.0/0
tcp spt:443

     Chain web (4 references)
      pkts bytes target     prot opt in     out     source               destination
      8767  727K            all  --  eth0   *       0.0.0.0/0            0.0.0.0/0
     11506   13M            all  --  *      eth0    0.0.0.0/0            0.0.0.0/0
     [root@mail shorewall]#
```

For an example of integrating Shorewall Accounting with MRTG, see http://www.nightbrawler.com/code/shorewall-stats/.

Shorewall and Aliased Interfaces

Tom Eastep

Copyright © 2001-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-09-30

Table of Contents

Background
Adding Addresses to Interfaces
So how do I handle more than one address on an interface?
- Separate Rules
- DNAT
- SNAT
- One-to-one NAT
- MULTIPLE SUBNETS

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Background

The traditional net-tools contain a program called *ifconfig* which is used to configure network devices. ifconfig introduced the concept of *aliased* or *virtual* interfaces. These virtual interfaces have names of the form *interface:integer* (e.g., `eth0:0`) and ifconfig treats them more or less like real interfaces.

Example 1. ifconfig

```
[root@gateway root]# ifconfig eth0:0
eth0:0    Link encap:Ethernet  HWaddr 02:00:08:3:FA:55
          inet addr:206.124.146.178  Bcast:206.124.146.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:11 Base address:0x2000
[root@gateway root]#
```

The ifconfig utility is being gradually phased out in favor of the ip utility which is part of the *iproute* package. The ip utility does not use the concept of aliases or virtual interfaces but rather treats additional addresses on an interface as objects in their own right. The ip utility does provide for interaction with ifconfig in that it allows addresses to be *labeled* where these labels take the form of ipconfig virtual interfaces.

Example 2. ip

```
[root@gateway root]# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc htb qlen 100
    link/ether 02:00:08:e3:fa:55 brd ff:ff:ff:ff:ff:ff
    inet 206.124.146.176/24 brd 206.124.146.255 scope global eth0
    inet 206.124.146.178/24 brd 206.124.146.255 scope global secondary eth0:0
[root@gateway root]#
```

Note

One **cannot** type "**ip addr show dev eth0:0**" because "`eth0:0`" is a label for a particular address rather than a device name.

```
[root@gateway root]# ip addr show dev eth0:0
Device "eth0:0" does not exist.
[root@gateway root]#
```

The iptables program doesn't support virtual interfaces in either it's "-i" or "-o" command options; as a consequence, Shorewall does not allow them to be used in the /etc/shorewall/interfaces file or anywhere else except as described in the discussion below.

Adding Addresses to Interfaces

Most distributions have a facility for adding additional addresses to interfaces. If you have already used your distribution's capability to add your required addresses, you can skip this section.

Shorewall provides facilities for automatically adding addresses to interfaces as described in the following section. It is also easy to add them yourself using the **ip** utility. The above alias was added using:

```
ip addr add 206.124.146.178/24 brd 206.124.146.255 dev eth0 label eth0:0
```

You probably want to arrange to add these addresses when the device is started rather than placing commands like the above in one of the Shorewall extension scripts. For example, on RedHat systems, you can place the commands in /sbin/ifup-local:

```
#!/bin/sh

case $1 in
    eth0)
        /sbin/ip addr add 206.124.146.178 dev eth0 label eth0:0
        ;;
esac
```

RedHat systems also allow adding such aliases from the network administration GUI (which only works well if you have a graphical environment on your firewall).

On Debian and LEAF/Bering systems, it is as simple as adding the command to the interface definition as follows:

```
# Internet interface
auto eth0
iface eth0 inet static
        address 206.124.146.176
```

```
        netmask 255.255.255.0
        gateway 206.124.146.254
        up ip addr add 206.124.146.178/24 brd 206.124.146.255 dev eth0 label eth0:0
```

So how do I handle more than one address on an interface?

The answer depends on what you are trying to do with the interfaces. In the sub-sections that follow, we'll take a look at common scenarios.

Separate Rules

If you need to make a rule for traffic to/from the firewall itself that only applies to a particular IP address, simply qualify the $FW zone with the IP address.

Example 3. allow SSH from net to eth0:0 above

[`/etc/shorewall/rules`]

```
#ACTION   SOURCE     DEST                  PROTO      DEST PORT(S)
ACCEPT    net        $FW:206.124.146.178   tcp        22
```

DNAT

Suppose that I had set up eth0:0 as above and I wanted to port forward from that virtual interface to a web server running in my local zone at 192.168.1.3. That is accomplised by a single rule in the `/etc/shorewall/rules` file:

```
#ACTION   SOURCE     DEST                  PROTO      DEST PORT(S)   SOURCE
ORIGINAL
#                                                                    PORT(S)   DEST
DNAT      net        loc:192.168.1.3       tcp        80             -
206.124.146.178
```

SNAT

If you wanted to use eth0:0 as the IP address for outbound connections from your local zone (eth1), then in `/etc/shorewall/masq`:

```
#INTERFACE              SUBNET          ADDRESS
eth0                    eth1            206.124.146.178
```

Shorewall can create the alias (additional address) for you if you set ADD_SNAT_ALIASES=Yes in `/etc/shorewall/shorewall.conf`.

Warning

Addresses added by ADD_SNAT_ALIASES=Yes are deleted and re-added during **shorewall restart**. As a consequence, connections using those addresses may be severed.

Beginning with Shorewall 1.3.14, Shorewall can actually create the “label” (virtual interface) so that you can see the created address using ifconfig. In addition to setting ADD_SNAT_ALIASES=Yes, you specify the virtual interface name in the INTERFACE column as follows.

/etc/shorewall/masq

```
#INTERFACE              SUBNET          ADDRESS
eth0:0                  eth1            206.124.146.178
```

Shorewall can also set up SNAT to round-robin over a range of IP addresses. Do do that, you specify a range of IP addresses in the ADDRESS column. If you specify a label in the INTERFACE column, Shorewall will use that label for the first address of the range and will increment the label by one for each subsequent label.

/etc/shorewall/masq

```
#INTERFACE               SUBNET          ADDRESS
eth0:0                   eth1            206.124.146.178-206.124.146.180
```

The above would create three IP addresses:

```
eth0:0 = 206.124.146.178
eth0:1 = 206.124.146.179
eth0:2 = 206.124.146.180
```

One-to-one NAT

If you wanted to use one-to-one NAT to link `eth0:0` with local address 192.168.1.3, you would have the following in `/etc/shorewall/nat`:

```
#EXTERNAL           INTERFACE          INTERNAL      ALL INTERFACES    LOCAL
206.124.146.178     eth0               192.168.1.3   no                no
```

Shorewall can create the alias (additional address) for you if you set ADD_IP_ALIASES=Yes in /etc/shorewall/shorewall.conf.

Warning

Addresses added by ADD_IP_ALIASES=Yes are deleted and re-added during **shorewall restart**. As a consequence, connections using those addresses may be severed.

Beginning with Shorewall 1.3.14, Shorewall can actually create the “label” (virtual interface) so that you can see the created address using ifconfig. In addition to setting ADD_IP_ALIASES=Yes, you specify the virtual interface name in the INTERFACE column as follows.

/etc/shorewall/nat

```
#EXTERNAL           INTERFACE          INTERNAL      ALL INTERFACES    LOCAL
206.124.146.178     eth0:0             192.168.1.3   no                no
```

In either case, to create rules in `/etc/shorewall/rules` that pertain only to this NAT pair, you simply qualify the local zone with the internal IP address.

Example 4. You want to allow SSH from the net to 206.124.146.178 a.k.a. 192.168.1.3.

```
#ACTION     SOURCE     DEST               PROTO     DEST PORT(S)
ACCEPT      net        loc:192.168.1.3    tcp       22
```

MULTIPLE SUBNETS

Sometimes multiple IP addresses are used because there are multiple subnetworks configured on a LAN segment. This technique does not provide for any security between the subnetworks if the users of the systems have administrative privileges because in that case, the users can simply manipulate their system's routing table to bypass your firewall/router. Nevertheless, there are cases where you simply want to consider the LAN segment itself as a zone and allow your firewall/router to route between the two subnetworks.

Example 5. Local interface eth1 interfaces to 192.168.1.0/24 and 192.168.20.0/24. The primary IP address of eth1 is 192.168.1.254 and eth1:0 is 192.168.20.254. You simply want your firewall to route between these two subnetworks.

This example applies to Shorewall 1.4.2 and later.

In `/etc/shorewall/zones`:

```
#ZONE        TYPE          OPTIONS
loc          ipv4
```

In `/etc/shorewall/interfaces`:

```
#ZONE       INTERFACE  BROADCAST                       OPTIONS
loc         eth1       192.168.1.255,192.168.20.255    routeback
```

In `/etc/shorewall/rules`, simply specify ACCEPT rules for the traffic that you want to permit.

Example 6. Local interface eth1 interfaces to 192.168.1.0/24 and 192.168.20.0/24. The primary IP address of eth1 is 192.168.1.254 and eth1:0 is 192.168.20.254. You want to make these subnetworks into separate zones and control the access between them (the users of the systems do not have administrative privileges).

In `/etc/shorewall/zones`:

```
#ZONE        TYPE                 OPTIONS
loc          ipv4
loc2         ipv4
```

In `/etc/shorewall/interfaces`:

```
#ZONE       INTERFACE  BROADCAST                       OPTIONS
-           eth1       192.168.1.255,192.168.20.255
```

In `/etc/shorewall/hosts`:

```
#ZONE        HOSTS                    OPTIONS
loc          eth1:192.168.1.0/24
loc2         eth1:192.168.20.0/24
```

In `/etc/shorewall/rules`, simply specify ACCEPT rules for the traffic that you want to permit.

Shorewall Blacklisting Support

Tom Eastep

Copyright © 2002-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-08-28

Table of Contents

Introduction
Static Blacklisting
Dynamic Blacklisting

Introduction

Shorewall supports two different forms of blacklisting; static and dynamic. The BLACKLISTNEWONLY option in /etc/shorewall/shorewall.conf controls the degree of blacklist filtering:

1. BLACKLISTNEWONLY=No -- All incoming packets are checked against the blacklist. New blacklist entries can be used to terminate existing connections.
2. BLACKLISTNEWONLY=Yes -- The blacklists are only consulted for new connection requests. Blacklists may not be used to terminate existing connections. Only the source address is checked against the blacklists.

Important

Only the source address is checked against the blacklists. Blacklists only stop blacklisted hosts from connecting to you — they do not stop you or your users from connecting to blacklisted hosts .

Important

Dynamic Shorewall blacklisting is not appropriate for blacklisting 1,000s of different addresses. Static Blacklisting can handle large blacklists but only if you use ipsets. Without ipsets, the blacklists will take forever to load and will have a very negative effect on firewall performance.

Static Blacklisting

Shorewall static blacklisting support has the following configuration parameters:

- You specify whether you want packets from blacklisted hosts dropped or rejected using the BLACKLIST_DISPOSITION setting in `/etc/shorewall/shorewall.conf`.
- You specify whether you want packets from blacklisted hosts logged and at what syslog level using the BLACKLIST_LOGLEVEL setting in `/etc/shorewall/shorewall.conf`.
- You list the IP addresses/subnets that you wish to blacklist in `/etc/shorewall/blacklist`. You may also specify PROTOCOL and Port numbers/Service names in the blacklist file.
- You specify the interfaces whose incoming packets you want checked against the blacklist using the "blacklist" option in `/etc/shorewall/interfaces`.
- The black list is refreshed from `/etc/shorewall/blacklist` by the "**shorewall refresh**" command.

Users with a large static black list may want to set the DELAYBLACKLISTLOAD option in shorewall.conf (added in Shorewall version 2.2.0). When DELAYBLACKLISTLOAD=Yes, Shorewall will enable new connections before loading the blacklist rules. While this may allow connections from blacklisted hosts to slip by during construction of the blacklist, it can substantially reduce the time that all new connections are disabled during "shorewall [re]start".

Beginning with Shorewall 2.4.0, you can use ipsets to define your static blacklist. Here's an example:

```
#ADDRESS/SUBNET         PROTOCOL        PORT
+Blacklistports[dst]
+Blacklistnets[src,dst]
+Blacklist[src,dst]
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

In this example, there is a portmap ipset *Blacklistports* that blacklists all traffic with destination ports included in the ipsec. There are also *Blacklistnets* (type *nethash*) and *Blacklist* (type *iphash*) ipsets that allow blacklisting networks and individual IP addresses. Note that [src,dst] is specified so that individual entries in the sets can be bound to other portmap ipsets to allow blacklisting (*source address*, *destination port*) combinations. For example:

```
ipset -N SMTP portmap --from 1 --to 31
ipset -A SMTP 25
```

```
ipset -A Blacklist 206.124.146.177
ipset -B Blacklist 206.124.146.177 -b SMTP
```

This will blacklist SMTP traffic from host 206.124.146.177.

Dynamic Blacklisting

Dynamic blacklisting doesn't use any configuration parameters but is rather controlled using /sbin/shorewall commands:

- drop <*ip address list*> - causes packets from the listed IP addresses to be silently dropped by the firewall.
- reject <*ip address list*> - causes packets from the listed IP addresses to be rejected by the firewall.
- allow <*ip address list*> - re-enables receipt of packets from hosts previously blacklisted by a *drop* or *reject* command.
- save - save the dynamic blacklisting configuration so that it will be automatically restored the next time that the firewall is restarted.
- show dynamic - displays the dynamic blacklisting configuration.

Dynamic blacklisting is not dependent on the “blacklist” option in `/etc/shorewall/interfaces`.

Example 1. Ignore packets from a pair of systems

```
    shorewall drop 192.0.2.124 192.0.2.125
```

Drops packets from hosts 192.0.2.124 and 192.0.2.125

Example 2. Re-enable packets from a system

```
    shorewall allow 192.0.2.125
```

Re-enables traffic from 192.0.2.125.

Controlling Output Traffic by UID/GID

Tom Eastep

Copyright © 2003 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2003-09-19

Table of Contents

Overview
User Sets
Restricting a rule to a particular user and/or group

Overview

This capability was added in Shorewall release 1.4.7.

Netfilter provides the capability to filter packets generated on the firewall system by User Id and/or Group Id. Shorewall provides two separate but related ways to use this Netfilter capability:

- Shorewall allows you to define collections of users called "User Sets" and then to restrict certain rules in /etc/shorewall/rules to a given User Set.
- Shorewall also allows you to restrict a given rule to a particular user and/or group.

Since only packets created by programs running on the Shorewall box itself, only rules whose SOURCE is the firewall ($FW) may be restricted using either of the facilities.

User Sets

Given the way that this facility is implemented in Shorewall, it is not possible to control logging of individual rules using a User Set and logging is rather specified on the User Set itself.

User Sets are defined in the /etc/shorewall/usersets file. Columns in that file include:

USERSET

The name of a User Set. Must be a legal shell identifier of no more than six (6) characters in length.

REJECT

Log level for connections rejected for this User Set.

ACCEPT

Log level for connections accepted for this User Set.

DROP

Log level for connections dropped for this User Set.

In the REJECT and ACCEPT columns, if you don't want to specify a value in the column but you want to specify a value in a following column, you may enter “-”.

Users and/or groups are added to User Sets using the /etc/shorewall/users file. Columns in that file are:

USERSET

The name of a User Set defined in /etc/shorewall/usersets.

USER

The name of a user defined on the system or a user number.

GROUP

The name of a group defined on the system or a number.

Only one of the USER and GROUP column needs to be non-empty. If you wish to specify a GROUP but not a USER, enter “-” in the user column.

If both USER and GROUP are specified then only programs running under that USER:GROUP pair will match rules specifying the User Set named in the USERSET column.

Once a user set has been defined, its name may be placed in the USER SET column of the /etc/shorewall/rules file.

Important

When the name of a user set is given in the USER SET column, you may not include a log level in the ACTION column; logging of such rules is governed solely by the user set's definition in the /etc/shorewall/userset file.

Example 1. You want members of the “admin” group and “root” to be able to use ssh on the firewall to connect to local systems. You want to log all connections accepted for these users using syslog at the “info” level.

/etc/shorewall/usersets

```
#USERSET        REJECT  ACCEPT  DROP
admins          -       info
```

/etc/shorewall/users

```
#USERSET        USER            GROUP
admins          -               admin
admins          root
```

/etc/shorewall/rules

```
#ACTION SOURCE  DESTINATION     PROTO   PORT    SOURCE  ORIGINAL        RATE    USER
#                                               PORT(S) DESTINATION             SET

ACCEPT  $FW     loc             tcp     22      -       -               -
admins
```

Restricting a rule to a particular user and/or group

In cases where you may want to restrict a rule to a particular user and/or group, the USER SET column in the rules file may be specified as:

```
[ <user name or number> ] : [ <group name or number> ]
```

When a user and/or group name is given in the USER SET column, it is OK to specify a log level in the ACTION column.

Example 2. You want user mail to be able to send email from the firewall to the local net zone

/etc/shorewall/rules (be sure to note the “:” in the USER SET column entry).

```
#ACTION SOURCE  DESTINATION     PROTO   PORT    SOURCE  ORIGINAL        RATE    USER
#                                               PORT(S) DESTINATION             SET

ACCEPT  $FW     loc             tcp     25      -       -               -       mail:
```

Corporate Network

Tom Eastep

Graeme Boyle

Copyright © 2003, 2005 Thomas M. Eastep and Graeme Boyle

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-09-13

Table of Contents

The Network
- Summary
- Some Mistakes I Made
- Lessons Learned
- Futures

Configuration Files
- Shorewall.conf
- Zones File
- Interfaces File
- Routestopped File
- Policy File
- Masq File
- NAT File
- Proxy ARP File
- Tunnels File
- Rules File (The shell variables are set in /etc/shorewall/params)
- Start File
- Stop File
- Init File

The Network

Note

- This configuration is used on a corporate network that has a Linux (RedHat 8.0) server with three interfaces, running Shorewall 1.4.5 release,
- Make sure you know what public IP addresses are currently being used and verify these *before* starting.
- Verify your DNS settings *before* starting any Shorewall configuration especially if you have split DNS.
- System names and Internet IP addresses have been changed to protect the innocent.

Warning

This configuration uses a combination of One-to-one NAT and Proxy ARP. This is generally not relevant to a simple configuration with a single public IP address. If you have just a single public IP address, most of what you see here won't apply to your setup so beware of copying parts of this configuration and expecting them to work for you. What you copy may or may not work in your configuration.

I have a T1 with 64 static IP addresses (192.0.18.65-127/26). The internet is connected to eth0. The local network is connected via eth1 (10.10.0.0/22) and the DMZ is connected to eth2 (192.168.21.0/24). I have an IPSec tunnel connecting our offices in Germany to our offices in the US. I host two Microsoft Exchange servers for two different companies behind the firewall hence, the two Exchange servers in the diagram below.

Summary

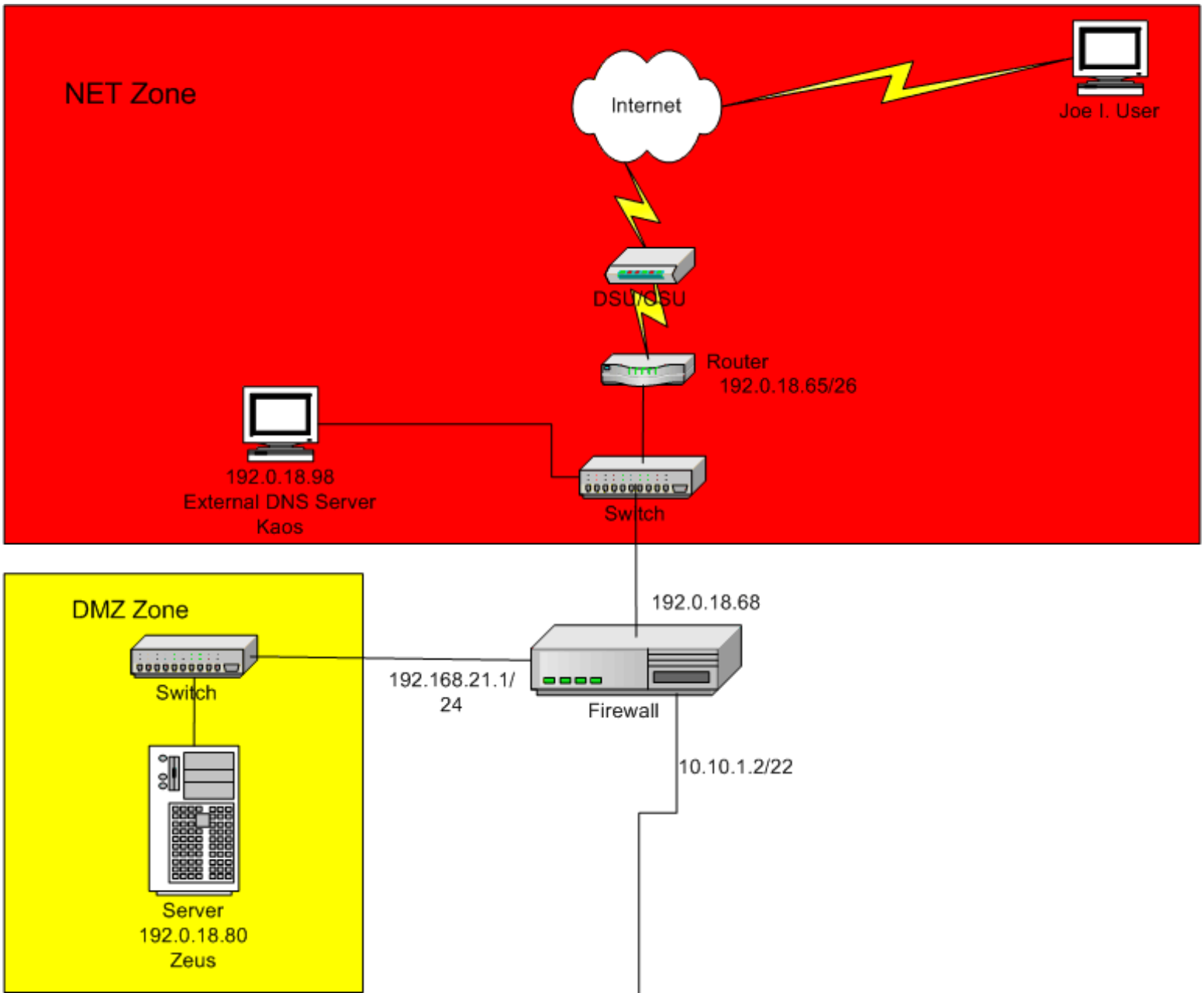
- SNAT for all systems connected to the LAN - Internal addresses 10.10.x.x to external address 192.0.18.127.
- One-to-one NAT for *Polaris* (Exchange Server #2). Internal address 10.10.1.8 and external address 192.0.18.70.
- One-to-one NAT for *Sims* (Inventory Management server). Internal address 10.10.1.56 and external address 192.0.18.75.
- One-to-one NAT for *Project* (Project Web Server). Internal address 10.10.1.55 and external address 192.0.18.84.
- One-to-one NAT for *Fortress* (Exchange Server). Internal address 10.10.1.252 and external address 192.0.18.93.
- One-to-one NAT for *BBSRV* (Blackberry Server). Internal address 10.10.1.230 and external address 192.0.18.97.
- One-to-one NAT for *Intweb* (Intranet Web Server). Internal address 10.10.1.60 and external address 192.0.18.115.

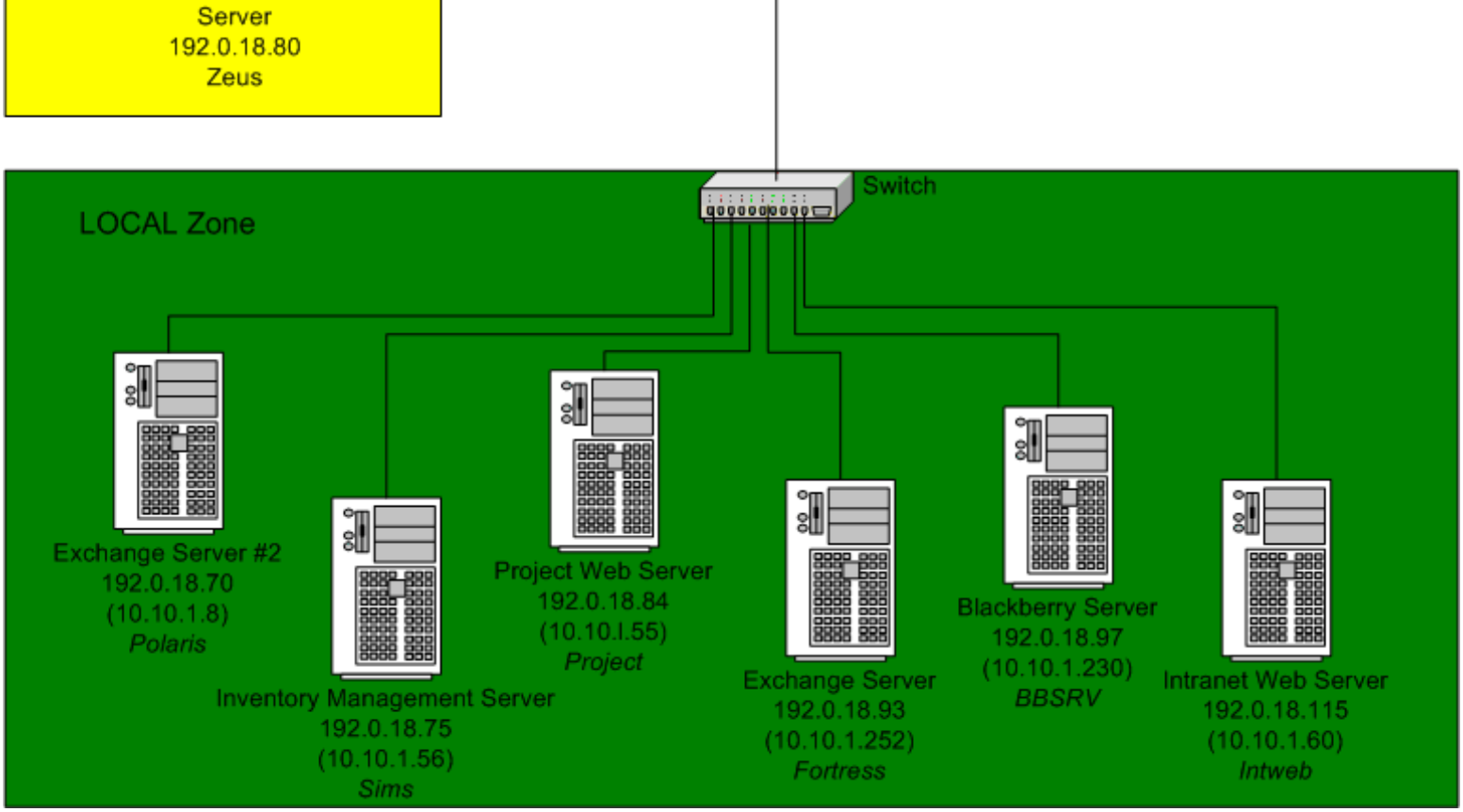
The firewall runs on a 2Gb, Dual PIV/2.8GHz, Intel motherboard with RH8.0.

The Firewall is also a proxy server running Privoxy 3.0.

The single system in the DMZ (address 192.0.18.80) runs sendmail, imap, pop3, DNS, a Web server (Apache) and an FTP server (vsFTPd 1.1.0). That server is managed through Proxy ARP.

All administration and publishing is done using ssh/scp. I have X installed on the firewall and the system in the DMZ. X applications tunnel through SSH to Hummingbird Exceed running on a PC located in the LAN. Access to the firewall using SSH is restricted to systems in the LAN, DMZ or the system Kaos which is on the Internet and managed by me.





The Ethernet 0 interface in the Server is configured with IP address 192.0.18.68, netmask 255.255.255.192. The server's default gateway is 192.0.18.65, the Router connected to my network and the ISP. This is the same default gateway used by the firewall itself. On the firewall, Shorewall automatically adds a host route to 192.0.18.80 through Ethernet 2 (192.168.21.1) because of the entry in /etc/shorewall/proxyarp (see below). I modified the start, stop and init scripts to include the fixes suggested when having an IPSec tunnel.

Some Mistakes I Made

Yes, believe it or not, I made some really basic mistakes when building this firewall. Firstly, I had the new firewall setup in parallel with the old firewall so that there was no interruption of service to my users. During my out-bound testing, I set up systems on the LAN to utilize the firewall which worked fine. When testing my NAT connections, from the outside, these would fail and I could not understand why. Eventually, I changed the default route on the internal system I was trying to access, to point to the new firewall and “bingo”, everything worked as expected. This oversight delayed my deployment by a couple of days not to mention level of frustration it produced.

Another problem that I encountered was in setting up the Proxyarp system in the DMZ. Initially I forgot to remove the entry for the eth2 from the /etc/shorewall/masq file. Once my file settings were correct, I started verifying that the ARP caches on the firewall, as well as the outside system “kaos”, were showing the correct Ethernet MAC address. However, in testing remote access, I could access the system in the DMZ only from the firewall and LAN but not from the Internet. The message I received was “connection denied” on all protocols. What I did not realize was that a “helpful” administrator that had turned on an old system and assigned the same address as the one I was using for Proxyarp without notifying me. How did I work this out. I shutdown the system in the DMZ, rebooted the router and flushed the ARP cache on the firewall and kaos. Then, from kaos, I started pinging that IP address and checked the updated ARP cache and lo-and-behold a different MAC address showed up. High levels of frustration etc., etc. The administrator will not be doing that again! :-)

Lessons Learned

- Read the documentation.
- Draw your network topology before starting.
- Understand what services you are going to allow in and out of the firewall, whether they are TCP or UDP packets and make a note of these port numbers.
- Try to get quiet time to build the firewall - you need to focus on the job at hand.
- When asking for assistance, be honest and include as much detail as requested. Don't try and hide IP addresses etc., you will probably screw up the logs and make receiving assistance harder.
- Read the documentation.

Futures

This is by no means the final configuration. In the near future, I will be moving more systems from the LAN to the DMZ. I will also be watching the logs for port scan programs etc. but, this should be standard security maintenance.

Configuration Files

Here are copies of my files. I have removed most of the internal documentation for the purpose of this space however, my system still has the original files with all the comments and I highly recommend you do the same.

Shorewall.conf

```
###############################################################################
# /etc/shorewall/shorewall.conf V1.4 - Change the following variables to
# match your setup
#
# This program is under GPL [http://www.gnu.org/copyleft/gpl.htm]
#
# This file should be placed in /etc/shorewall
#
# (c) 1999,2000,2001,2002,2003 - Tom Eastep (teastep@shorewall.net)
###############################################################################
# L O G G I N G
###############################################################################
LOGFILE=/var/log/messages
LOGFORMAT="Shorewall:%s:%s:"
LOGRATE=
LOGBURST=
LOGUNCLEAN=info
BLACKLIST_LOGLEVEL=
LOGNEWNOTSYN=
MACLIST_LOG_LEVEL=info
TCP_FLAGS_LOG_LEVEL=debug
RFC1918_LOG_LEVEL=debug
PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin:/usr/local/sbin
SUBSYSLOCK=/var/lock/subsys/shorewall
STATEDIR=/var/lib/shorewall
MODULESDIR=
FW=fw
NAT_ENABLED=Yes
MANGLE_ENABLED=Yes
IP_FORWARDING=On
ADD_IP_ALIASES=Yes
ADD_SNAT_ALIASES=Yes
TC_ENABLED=Yes
CLEAR_TC=No
MARK_IN_FORWARD_CHAIN=No
CLAMPMSS=No
ROUTE_FILTER=Yes
NAT_BEFORE_RULES=No
MULTIPORT=Yes
DETECT_DNAT_IPADDRS=Yes
MUTEX_TIMEOUT=60
NEWNOTSYN=Yes
BLACKLIST_DISPOSITION=DROP
MACLIST_DISPOSITION=REJECT
TCP_FLAGS_DISPOSITION=DROP
#LAST LINE -- DO NOT REMOVE
```

Zones File

```
#
# Shorewall 1.4 -- Sample Zone File For Two Interfaces
# /etc/shorewall/zones
#
# This file determines your network zones. Columns are:
#
# ZONE Short name of the zone
# DISPLAY Display name of the zone
# COMMENTS Comments about the zone
#
#ZONE          DISPLAY              COMMENTS
net            Net                  Internet
loc            Local                Local Networks
dmz            DMZ                  Demilitarized Zone
vpn1           VPN1                 VPN to Germany
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

Interfaces File

```
###############################################################################
#ZONE    INTERFACE     BROADCAST      OPTIONS
net      eth0          62.123.106.127 routefilter,norfc1918,blacklist,tcpflags
loc      eth1          detect         dhcp,routefilter
dmz      eth2          detect
vpn1     ipsec0
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

Routestopped File

```
#INTERFACE HOST(S)
eth1       -
eth2       -
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

Policy File

```
################################################################################
#SOURCE DEST POLICY       LOG LEVEL         LIMIT:BURST
loc     net  ACCEPT
loc     fw   ACCEPT
loc     dmz  ACCEPT
# If you want open access to the Internet from your Firewall
# remove the comment from the following line.
fw      net  ACCEPT
fw      loc  ACCEPT
fw      dmz  ACCEPT
dmz     fw   ACCEPT
dmz     loc  ACCEPT
dmz     net  ACCEPT
#
# Adding VPN Access
loc     vpn1 ACCEPT
dmz     vpn1 ACCEPT
fw      vpn1 ACCEPT
vpn1    loc  ACCEPT
vpn1    dmz  ACCEPT
vpn1    fw   ACCEPT
#
```

```
net     all  DROP         info
all     all  REJECT       info
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

Masq File

```
#INTERFACE SUBNET ADDRESS
eth0       eth1   192.0.18.126
#
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

NAT File

```
#EXTERNAL    INTERFACE INTERNAL    ALL INTERFACES LOCAL
#
# Intranet Web Server
192.0.18.115 eth0:0    10.10.1.60  No  No
#
# Project Web Server
192.0.18.84  eth0:1    10.10.1.55  No  No
#
# Blackberry Server
192.0.18.97  eth0:2    10.10.1.55  No  No
#
# Corporate Mail Server
192.0.18.93  eth0:3    10.10.1.252 No  No
#
# Second Corp Mail Server
192.0.18.70  eth0:4    10.10.1.8   No  No
#
# Sims Server
192.0.18.75  eth0:5    10.10.1.56  No  No
#
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

Proxy ARP File

```
#ADDRESS    INTERFACE EXTERNAL HAVEROUTE
#
# The Corporate email server in the DMZ
192.0.18.80 eth2      eth0     No
#
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

Tunnels File

```
# TYPE ZONE GATEWAY         GATEWAY ZONE PORT
ipsec  net  134.147.129.82
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

Rules File (The shell variables are set in /etc/shorewall/params)

```
###############################################################################
#ACTION   SOURCE             DEST            PROTO     DEST  SOURCE  ORIGINAL
#                                                      PORT  PORT(S) DEST
#
# Accept DNS connections from the firewall to the network
```

```
#
ACCEPT    fw                  net             tcp       53
ACCEPT    fw                  net             udp       53
#
# Accept SSH from internet interface from kaos only
#
ACCEPT    net:192.0.18.98     fw              tcp       22
#
# Accept connections from the local network for administration
#
ACCEPT    loc                 fw              tcp       20:22
ACCEPT    loc                 net             tcp       22
ACCEPT    loc                 fw              tcp       53
ACCEPT    loc                 fw              udp       53
ACCEPT    loc                 net             tcp       53
ACCEPT    loc                 net             udp       53
#
# Allow Ping To And From Firewall
#
ACCEPT    loc                 fw              icmp      8
ACCEPT    loc                 dmz             icmp      8
ACCEPT    loc                 net             icmp      8
ACCEPT    dmz                 fw              icmp      8
ACCEPT    dmz                 loc             icmp      8
ACCEPT    dmz                 net             icmp      8
DROP      net                 fw              icmp      8
DROP      net                 loc             icmp      8
DROP      net                 dmz             icmp      8
ACCEPT    fw                  loc             icmp      8
ACCEPT    fw                  dmz             icmp      8
DROP      fw                  net             icmp      8
#
# Accept proxy web connections from the inside
#
ACCEPT    loc                 fw              tcp       8118
#
# Forward PcAnywhere, Oracle and Web traffic from outside to the Demo systems
# From a specific IP Address on the Internet.
#
# ACCEPT  net:207.65.110.10  loc:10.10.3.151 tcp       1521,http
# ACCEPT  net:207.65.110.10  loc:10.10.2.32  tcp       5631:5632
#
# Intranet web server
ACCEPT    net                 loc:10.10.1.60  tcp       443
ACCEPT    dmz                 loc:10.10.1.60  tcp       443
#
# Projects web server
ACCEPT    net                 loc:10.10.1.55  tcp       80
ACCEPT    dmz                 loc:10.10.1.55  tcp       80
#
# Blackberry Server
ACCEPT    net                 loc:10.10.1.230 tcp       3101
#
# Corporate Email Server
ACCEPT    net                 loc:10.10.1.252 tcp       25,53,110,143,443
#
# Corporate #2 Email Server
ACCEPT    net                 loc:10.10.1.8   tcp       25,80,110,443
#
# Sims Server
ACCEPT    net                 loc:10.10.1.56  tcp       80,443
ACCEPT    net                 loc:10.10.1.56  tcp       7001:7002
ACCEPT    net:63.83.198.0/24 loc:10.10.1.56  tcp       5631:5632
```

```
#
# Access to DMZ
ACCEPT    loc                dmz             udp       53,177
ACCEPT    loc                dmz             tcp       80,25,53,22,143,443,993,20,110
ACCEPT    net                dmz             udp       53
ACCEPT    net                dmz             tcp       25,53,22,21,123
ACCEPT    dmz                net             tcp       25,53,80,123,443,21,22
ACCEPT    dmz                net             udp       53
#
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

Start File

```
###############################################################################
# Shorewall 1.4 -- /etc/shorewall/start
#
# Add commands below that you want to be executed after shorewall has
# been started or restarted.
#
qt service ipsec start
```

Stop File

```
###############################################################################
# Shorewall 1.4 -- /etc/shorewall/stop
#
# Add commands below that you want to be executed at the beginning of a
# “shorewall stop” command.
#
qt service ipsec stop
```

Init File

```
###############################################################################
# Shorewall 1.4 -- /etc/shorewall/init
#
# Add commands below that you want to be executed at the beginning of
# a “shorewall start” or “shorewall restart” command.
#
qt service ipsec stop
```

DHCP

Tom Eastep

Copyright © 2001, 2002, 2004 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2004-05-24

Table of Contents

If you want to Run a DHCP Server on your firewall
If a Firewall Interface gets its IP Address via DHCP

Note

For most operations, DHCP software interfaces to the Linux IP stack at a level below Netfilter. Hence, Netfilter (and therefore Shorewall) cannot be used effectively to police DHCP. The "dhcp" interface option described in this article allows for Netfilter to stay out of DHCP's way for those operations that can be controlled by Netfilter and prevents unwanted logging of DHCP-related traffic by Shorewall-generated Netfilter logging rules.

If you want to Run a DHCP Server on your firewall

- Specify the "dhcp" option on each interface to be served by your server in the `/etc/shorewall/interfaces` file. This will generate rules that will allow DHCP to and from your firewall system.
- When starting "dhcpd", you need to list those interfaces on the run line. On a RedHat system, this is done by modifying `/etc/sysconfig/dhcpd`.

If a Firewall Interface gets its IP Address via DHCP

- Specify the "dhcp" option for this interface in the `/etc/shorewall/interfaces` file. This will generate rules that will allow DHCP to and from your firewall system.

- If you know that the dynamic address is always going to be in the same subnet, you can specify the subnet address in the interface's entry in the `/etc/shorewall/interfaces` file.
- If you don't know the subnet address in advance, you should specify “detect” for the interface's subnet address in the `/etc/shorewall/interfaces` file and start Shorewall after the interface has started.
- In the event that the subnet address might change while Shorewall is started, you need to arrange for a “shorewall refresh” command to be executed when a new dynamic IP address gets assigned to the interface. Check your DHCP client's documentation.

ECN

Tom Eastep

Copyright © 2001, 2002, 2003, 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-01-17

Table of Contents

Explicit Congestion Notification (ECN)

Warning

2006-01-17. The ECN Netfilter target in recent 2.6 Linux Kernels is broken. Symptoms are that you will be unable to establish a TCP connection to hosts defined in the /etc/shorewall/ecn file.

Explicit Congestion Notification (ECN)

Explicit Congestion Notification (ECN) is described in RFC 3168 and is a proposed internet standard. Unfortunately, not all sites support ECN and when a TCP connection offering ECN is sent to sites that don't support it, the result is often that the connection request is ignored.

To allow ECN to be used, Shorewall allows you to enable ECN on your Linux systems then disable it in your firewall when the destination matches a list that you create (the /etc/shorewall/ecn file).

You enable ECN by

```
echo 1 > /proc/sys/net/ipv4/tcp_ecn
```

You must arrange for that command to be executed at system boot. Most distributions have a method for doing that -- on RedHat, you make an entry in /etc/sysctl.conf.

```
net.ipv4.tcp_ecn = 1
```

Entries in /etc/shorewall/ecn have two columns as follows:

INTERFACE

The name of an interface on your system

HOST(S)

An address (host or subnet) of a system or group of systems accessed through the interface in the first column. You may include a comma-separated list of such addresses in this column.

Example 1. Your external interface is eth0 and you want to disable ECN for tcp connections to 192.0.2.0/24:

Table 1. /etc/shorewall/ecn

INTERFACE	HOST(S)
eth0	192.0.2.0/24

Shorewall Error Messages

Tom Eastep

Copyright © 2004, 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-10-03

Table of Contents

Introduction
Messages Produced by /sbin/shorewall
Messages Produced by /usr/share/shorewall/firewall
Warnings
Iptables Error Messages

Introduction

Shorewall can produce a wide variety of error messages when a problem is detected with your configuration. This article attempts to explain the cause of and cures for some of these messages.

Messages Produced by /sbin/shorewall

Some error messages are produced by the /sbin/shorewall utility. These messages are detailed in this section.

ERROR: <label> must specify a simple file name: <name>

> This means that you have specified a restore file name with a "/". Restore files must be simple file names with no slashes.

ERROR: Shorewall is not properly installed

> The files `/usr/share/shorewall/firewall` and/or `/usr/share/shorewall/version` do not exist.

ERROR: <file name> exists and is not a saved Shorewall configuration

> The named file in `/var/lib/shorewall` exists but is not executable.

ERROR: Reserved file name: <file name>

You have specified either `save` or `restore-base` as the name of a restore file -- those names are reserved for use by Shorewall.

ERROR: Currently-running Configuration Not Saved

During processing of a **shorewall save** command, the **iptables-save** command failed.

ERROR: /var/lib/shorewall/restore-base does not exist

The **shorewall start** and **shorewall restart** commands create a file called `/var/lib/shorewall/restore-base` which forms the basis for creating a restore file using **shorewall save**. This error message is issued when **shorewall save** is not able to find that file.

ERROR: The program specified in IPTABLES does not exist or is not executable

The IPTABLES option in `/etc/shorewall/shorewall.conf` specifies a file that is not executable.

ERROR: Can't find iptables executable

There is no executable file named "iptables" in any directory in $PATH.

ERROR: The program specified in SHOREWALL_SHELL does not exist or is not executable

The SHOREWALL_SHELL option in `/etc/shorewall/shorewall.conf` names does not name an executable file.

ERROR: /var/lib/shorewall/<file> exists and is not a saved Shorewall configuration

The restore file (<file>) specified or implied in a **shorewall save** command already exists but is not executable (and hence cannot be a value restore file). Either remove/rename the file or specify a different file name.

Messages Produced by /usr/share/shorewall/firewall

The program `/usr/share/shorewall/firewall` is responsible for parsing the Shorewall configuration files and for creating and changing the Netfilter configuration. Some of the error messages generated by this program are listed below.

ERROR: Invalid nested zone syntax: :<parent-zone>

The zone name in the ZONE column of `/etc/shorewall/zones` may not start with a colon (":").

ERROR: Sub-zones of the firewall zone are not allowed

The firewall zone may not be defined to have zones nested within it.

ERROR: Parent zone not defined: <parent-zone>

When defining nested zones in `/etc/shorewall/zones`, the parent zone must be defined before any zones nested inside of it.

ERROR: Zone name longer than 5 characters: <zone>

Zone names are restricted to 5 characters or less in length.

ERROR: Illegal zone name "<zone>" in zones file

The zone name quoted in the error message begins with a digit -- zone names must begin with an alphabetic character.

ERROR: Reserved zone name "<zone>" in zones file

The names "none" and "all" are reserved and may not be used as zone names in `/etc/shorewall/zones`.

ERROR: Zone <zone> is defined more than once

There are two records in `/etc/shorewall/zones` that define the named zone.

ERROR: Your kernel and/or iptables does not support policy match

You have defined a zone of type **ipsec** in `/etc/shorewall/zones` or have specified the ipsec option in an `/etc/shorewall/hosts` record but your kernel and/or iptables don't include policy match support -- see this article for details.

ERROR: The firewall zone may not be nested

You have defined a zone of type **firewall** to be nested inside another zone. Shorewall does not support such nesting.

ERROR: OPTIONS not allowed on the firewall zone

The zone of type **firewall** may not have any options specified in the OPTIONS, IN OPTIONS or OUT OPTIONS columns of `/etc/shorewall/zones`.

ERROR: Only one firewall zone may be defined

You may have only one record in `/etc/shorewall/zones` that has type **firewall**.

ERROR: No ipv4 or ipsec Zones Defined

You must define at least one ipv4 or ipsec zone in `/etc/shorewall/zones`.

ERROR: No Firewall Zone Defined

You must define one (and only one) zone if type **firewall** in `/etc/shorewall/zones`.

ERROR: Invalid Mark or Mask value: <number>

Shorewall-assigned packet and connection marks are limited to the range 1-255.

ERROR: Invalid zone definition for zone <zone>

The zone named in the message is defined to be associated with an interface in `/etc/shorewall/interfaces` yet it also has an entry for that same interface in `/etc/shorewall/hosts`.

ERROR: Invalid zone (<zone>) in record "<record>"

The zone named in the ZONE column of the listed record from `/etc/shorewall/interfaces` or `/etc/shorewall/hosts` is not defined in `/etc/shorewall/zones`.

ERROR: The routeback option may not be specified on a multi-zone interface

The ZONE column of a record in `/etc/shorewall/interfaces` was empty ("-"). Such interfaces may not specify the **routeback** option.

ERROR: The "detectnets" option may not be used with a wild-card interface

The interface name in the INTERFACE column is a wild-card (ends with "+"). Such interfaces may not specify the **detectnets** option.

ERROR: Duplicate Interface <interface>

The named interface has two entries in `/etc/shorewall/interfaces`.

ERROR: Invalid Interface Name: <interface>

The interface name contains a colon (":") or is "+". If the name includes a ":", you probably need to read this article.

ERROR: The 'norfc1918' option may not be specified on an interface with an RFC 1918 address. Interface: <interface>

The <interface> named in the message is configured with an IP address that is reserved by RFC 1918 -- that address is incompatible with the **norfc1918** interface option.

ERROR: Unknown interface (<interface>) in record "<record>"

The <*interface*> name listed in the <*record*> from `/etc/shorewall/hosts` was not defined in `/etc/shorewall/interfaces`.

ERROR: Invalid HOST(S) column contents: <hosts>

The contests of the HOST(S) column in a record from `/etc/shorewall/hosts` does not follow the proper syntax for that column in that it doesn't contain at least one colon (":"). See the /etc/shorewall/hosts documentation.

ERROR: Bridged interfaces may not be defined in /etc/shorewall/interfaces: <interface>[:<address>]

The named interface appears in /etc/shorewall/hosts and appears as a bridge port (after a colon) but is also defined in `/etc/shorewall/interfaces`.

ERROR: Undefined zone <zone>

The named zone appears in the /etc/shorewall/policy file but not in the /etc/shorewall/zones file.

ERROR: <policy record>: NONE policy not allowed to/from the <firewall-zone-name> zone

Shorewall does not support a policy of NONE when the source or destination zone is the firewall itself.

ERROR: <policy record>: NONE policy not allowed with "all"

Shorewall does not support a policy of NONE when the source or destination zone is "all".

ERROR: Duplicate policy: <source zone> <destination zone> <policy>

There is an earlier record in the file with the same <source zone> and <destination zone>

ERROR: Can't determine the IP address of <interface>

You have specified DETECT_DNAT_ADDRS=Yes in /etc/shorewall/shorewall.conf and Shorewall is unablee to determine the IP address of the named <*interface*>. Be sure that the interface is started before starting Shorewall or set DETECT_DNAT_ADDRS=No.

ERROR: Invalid gateway zone (<zone>) -- Tunnel "<record>

The listed <*zone*> name appears in the GATEWAY ZONE column of the listed <*record*> from `/etc/shorewall/tunnels` but is not defined in `/etc/shorewall/zones`.

ERROR: No hosts on <interface> have the maclist option specified

The named <*interface*> appears in a record in `/etc/shorewall/maclist` yet that interface's record in `/etc/shorewall/interfaces` does not specify the **maclist** option and no record in `/etc/shorewall/hosts` that names that interface includes the **maclist** option.

ERROR: Interface <interface> must be up before Shorewall can start

You have specified the **maclist** option for this interface but the command **ip list show <interface>** fails.

ERROR: Unknown interface <interface>

The interface appears in a configuration file but is not defined in `/etc/shorewall/interfaces`.

ERROR: BRIDGING=Yes requires Physdev Match support in your Kernel and iptables

You have set BRIDGING=Yes in `/etc/shorewall/shorewall.conf` but it appears that your kernel and/or iptables do not have physdev match support.

ERROR: Invalid Action Name: <action>

The <action> contains one of the following characters: ".", "-", or "%". Those characters are not allowed in an action name.

ERROR: Invalid Macro Parameter in rule "<rule>"

The value being passed to a parameterized macro is not ACCEPT, DROP, REJECT, LOG, QUEUE or CONTINUE.

ERROR: Missing Action File: action.<action name>

The specified <action name> has an entry in `/usr/share/shorewall/actions.std` or in `/etc/shorewall/actions` but the corresponding action file does not exist on the CONFIG_PATH.

ERROR: Unknown interface <interface> in rule: "<rule>"

You have BRIDGING=No in `/etc/shorewall/shorewall.conf` and the <*interface*> given in a rule does not match an entry in `/etc/shorewall/interfaces`.

ERROR: SNAT may no longer be specified in a DNAT rule; use /etc/shorewall/masq instead

In earlier Shorewall versions, the ORIGINAL DEST column allowed following the original destination IP address with ":" and an address to use as the source of the forwarded connection request. Now that /etc/shorewall/masq supports qualification of SNAT rules by protocol and port, this feature is no longer required and has been deimplemented.

ERROR: "Invalid Source in rule "<rule>"

The SOURCE column has the firewall zone name immediately followed by "!". This syntax is use to exclude a subzone and Shorewall currently doesn't support subzones of the firewall zone.

ERROR: Rule "<rule>" - Destination may not be specified by MAC Address

Netfilter (and hence Shorewall) does not allow qualification of a rule by destination source IP address.

ERROR: Destination interface not allowed with <action>

The named <*action*> will be ACCEPT+ or NONAT. These actions are inforced in part in the

PREROUTING nat chain where the destination interface is not yet known (because the packet has not yet been routed). As a result, the DESTINATION column may not contain an interface name.

ERROR: Only DNAT and REDIRECT rules may specify destination mapping; rule "<rule>"

The *<rule>* specifies a server address that is different from the ORIGINAL DEST address and/or it specifies a server port that is different from the destination port but the ACTION is neither DNAT[-] nor REJECT[-].

ERROR: Empty source zone or qualifier: rule "<rule>"

The SOURCE column is of one of the forms *<zone>*:, :*<qualifier>* or :.

ERROR: Exclude list only allowed with DNAT or REDIRECT

In DNAT[-] and REDIRECT[-] rules, you can have a SOURCE of the form *<zone>*:*<net1>*!*<net2>*. This means *<net1>* in the *<zone>* zone **except for** *<net2>*. This syntax is not available with other ACTIONs.

ERROR: Invalid use of a user-qualification: rule "<rule>"

The USER/GROUP column may only have and entry if the SOURCE is the firewall zone.

ERROR: Empty destination zone or qualifier: rule "<rule>"

The DEST column is of one of the forms *<zone>*:, :*<qualifier>* or :.

ERROR: Undefined Client Zone in rule "<rule>"

The zone given in the SOURCE column was not defined in `/etc/shorewall/zones`.

ERROR: Undefined Server Zone in rule "<rule>"

The zone given in the DEST column was not defined in `/etc/shorewall/zones`.

ERROR: Rules may not override a NONE policy: rule "<rule>"

If the policy from zone z1 to zone z2 is NONE that means that Shorewall sets up no infrastructure to handle traffic from z1 to z2. Consequently, you cannot have any rules that control traffic from z1 to z2.

ERROR: Invalid Action in rule "<rule>"

The ACTION column contains an action that is not one of the built-in actions and it is not defined in `/etc/shorewall/actions` or in `/usr/share/shorewall/actions.std`.

ERROR: Unable to determine the routes through interface <interface>

You have specified *<interface>* in the SUBNET column of `/etc/shorewall/masq` which means that Shorewall is supposed to determine the network(s) routed through that interface. To do that, Shorewall issues the command **ip addr ls dev <interface>** and that command failed. This usually means that you are trying to start Shorewall before the *<interface>* is brought up.

ERROR: No appropriate chain for zone <z1> to zone <z2>

There is no policy defined in `/etc/shorewall/policy` for connections from zone *<z1>* to zone *<z2>*.

Warnings

This sections describes some of the more common warnings generated by Shorewall.

Warning: default route ignored on interface <interface>

This means that the interface named in the SUBNET column of `/etc/shorewall/masq` has the default route. This almost always means that you have the contents of the INTERFACE and SUBNET columns reversed.

Warning: Zone <zone> is empty

This warning alerts you to the fact tha <zone> is defined in `/etc/shorewall/zones` but has no corresponding entries in `/etc/shorewall/interfaces` or in `/etc/shorewall/hosts`.

WARNING: Shorewall startup is disabled. To enable startup, set STARTUP_ENABLED=Yes in /etc/shorewall/shorewall.conf

If you need help understanding that warning message then you probably need to take up another hobby or line of work.

Iptables Error Messages

By far the most asked about iptables error messages are:

iptables: No chain/target/match by that name

This almost always means that you are trying to use a Shorewall feature that your iptables and/or kernel do not support. Beginning with version 2.2.0, Shorewall follows this message with a copy of the iptables command that is failing. Most commonly, the problem is that one of the match types (keyword following "-m" in the command) isn't supported by your iptables/kernel. The output of "shorewall show capabilities" shows you what your iptables/kernel support:

```
gateway:~# shorewall show capabilities
Shorewall has detected the following iptables/netfilter capabilities:
   NAT: Available
   Packet Mangling: Available
   Multi-port Match: Available
   Extended Multi-port Match: Available
   Connection Tracking Match: Available
   Packet Type Match: Available
   Policy Match: Available
   Physdev Match: Available
   IP range Match: Available
   Recent Match: Available
   Owner Match: Available
   Ipset Match: Available
   ROUTE Target: Not available
   Extended MARK Target: Available
   CONNMARK Target: Available
   Connmark Match: Available
   Raw Table: Available
```

```
gateway:~#
```

iptables: invalid argument

Answer: 99.999% of the time, this error is caused by a mismatch between your iptables and kernel.

1. Your iptables must be compiled against a kernel source tree that is Netfilter-compatible with the kernel that you are running.
2. If you rebuild iptables using the defaults and install it, it will be installed in /usr/local/sbin/iptables. As shown above, you have the IPTABLES variable in shorewall.conf set to "/sbin/iptables".

Fallback and Uninstall

Tom Eastep

Copyright © 2001-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-09-16

Table of Contents

Falling Back to the Previous Version of Shorewall using the Fallback Script
Falling Back to the Previous Version of Shorewall using rpm
Uninstalling Shorewall

Falling Back to the Previous Version of Shorewall using the Fallback Script

If you install Shorewall and discover that it doesn't work for you, you can fall back to your previously installed version. To do that:

- cd to the distribution directory for the version of Seattle Firewall that you are currently running (NOT the version that you want to fall back to).
- Type "./fallback.sh"

Caution

The fallback script will replace /etc/shorewall/policy, /etc/shorewall/rules, /etc/shorewall/interfaces, /etc/shorewall/nat, /etc/shorewall/proxyarp and /etc/shorewall/masq with the version of these files from before the current version was installed. Any changes to any of these files will be lost.

Falling Back to the Previous Version of Shorewall using rpm

If your previous version of Shorewall was installed using RPM, you may fall back to that version by typing “rpm -Uvh --force <old rpm>” at a root shell prompt (Example: “rpm -Uvh --force /downloads/shorewall-3.1.1-0.noarch.rpm” would fall back to the 3.1.1-0 version of Shorewall).

Uninstalling Shorewall

If you no longer wish to use Shorewall, you may remove it by:

- cd to the distribution directory for the version of Shorewall that you have installed.
- type “./uninstall.sh”

If you installed using an rpm, at a root shell prompt type “rpm -e shorewall”.

Routing on One Interface

Tom Eastep

Copyright © 2003-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-10-01

Table of Contents

Introduction
Router in the Local Zone
- Can You Use the Standard Configuration?
- Will One Zone be Enough?
- I Need Separate Zones
  - Nested Zones
  - Parallel Zones

Some Hosts have Special Firewalling Requirements
One-armed Router

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Introduction

While most configurations can be handled with each of the firewall's network interfaces assigned to a single zone, there are cases where you will want to divide the hosts accessed through an interface between two or more zones.

- The interface has multiple addresses on multiple subnetworks. This case is covered in the Aliased Interface documentation.
- You are using some form of NAT and want to access a server by its external IP address from the same LAN segment. This is covered in FAQs 2 and 2a.
- There are routers accessible through the interface and you want to treat the networks accessed through that router as a separate zone.
- Some of the hosts accessed through an interface have significantly different firewalling requirements from the others so you want to assign them to a different zone.

The key points to keep in mind when setting up multiple zones per interface are:

- Shorewall generates rules for zones in the order that the zone declarations appear in /etc/shorewall/zones.
- The order of entries in /etc/shorewall/hosts is immaterial as far as the generated ruleset is concerned.

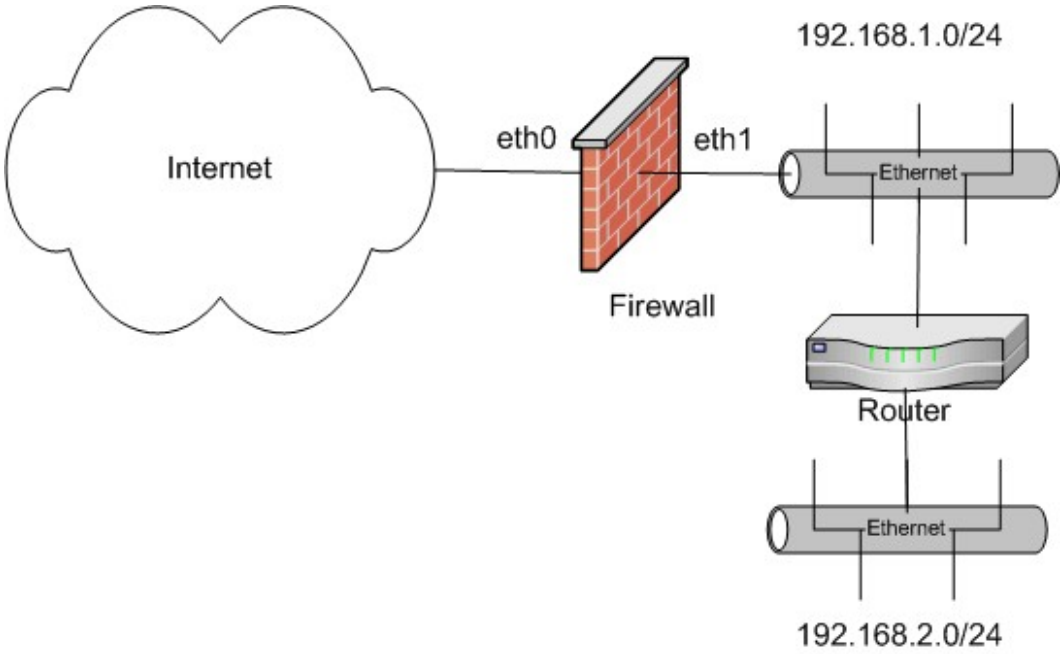
These examples use the local zone but the same technique works for any zone. Remember that Shorewall doesn't have any conceptual knowledge of "Internet", "Local", or "DMZ" so all zones except the firewall itself ($FW) are the same as far as Shorewall is concerned. Also, the examples use private (RFC 1918) addresses but public IP addresses can be used in exactly the same way.

Router in the Local Zone

Here is an example of a router in the local zone.

Note

the **box called "Router" could be a VPN server** or other such device; from the point of view of this discussion, it makes no difference.



Can You Use the Standard Configuration?

In many cases, the standard two-interface Shorewall setup will work fine in this configuration. It will work if:

- The firewall requirements to/from the internet are the same for 192.168.1.0/24 and 192.168.2.0/24.
- The hosts in 192.168.1.0/24 know that the route to 192.168.2.0/24 is through the **router**.

All you have to do on the firewall is add a route to 192.168.2.0/24 through the **router** and restart Shorewall.

Will One Zone be Enough?

If the firewalling requirements for the two local networks is the same but the hosts in 192.168.1.0/24 don't know how to route to 192.168.2.0/24 then you need to configure the firewall slightly differently. This type of configuration is rather stupid from an IP networking point of view but it is sometimes necessary because you simply don't want to have to reconfigure all of the hosts in 192.168.1.0/24 to add a persistent route to 192.168.2.0/24. On the firewall:

1. Add a route to 192.168.2.0/24 through the **Router**.
2. Set the "routeback" option for eth1 (the local firewall interface) in /etc/shorewall/interfaces.
3. Restart Shorewall.

If this still doesn't work at all or if it works for connections in one direction but not for connections in the other direction then:

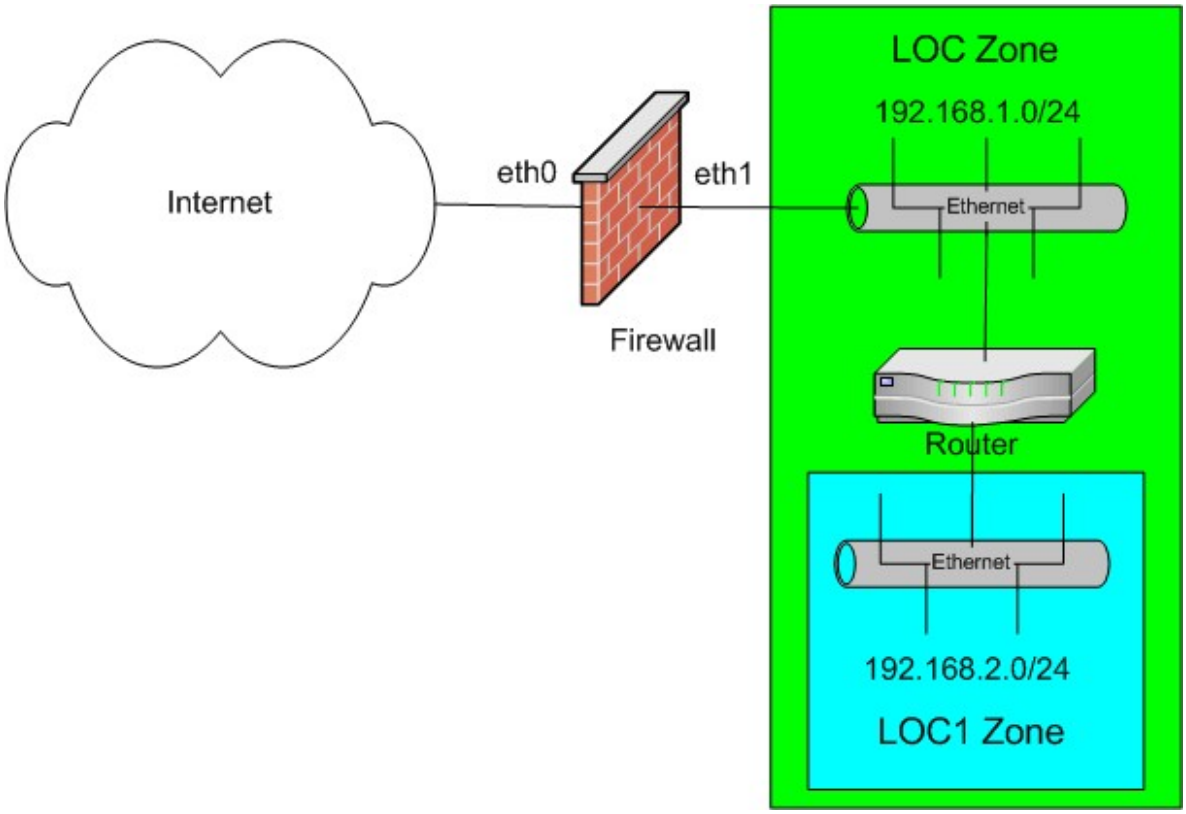
1. You must be running Shorewall version 2.0.16 or later; and
2. You need to set DROPINVALID=No in `/etc/shorewall/shorewall.conf`.

I Need Separate Zones

If you need to make 192.168.2.0/24 into it's own zone, you can do it one of two ways; Nested Zones or Parallel Zones. Again, it is likely that you will need to be running Shorewall 2.0.16 or later and that you will have to set DROPINVALID=No in `/etc/shorewall/shorewall.conf`.

Nested Zones

You can define one zone (called it "loc") as being all hosts connectied to eth1 and a second zone "loc1" (192.168.2.0/24) as a sub-zone.



The advantage of this approach is that the zone “loc1” can use CONTINUE policies such that if a connection request doesn't match a “loc1” rule, it will be matched against the “loc” rules. For example, if your loc1->net policy is CONTINUE then if a connection request from loc1 to the internet doesn't match any rules for loc1->net then it will be checked against the loc->net rules.

`/etc/shorewall/zones`

```
#ZONE       TYPE      OPTIONS
loc1        ipv4
loc         ipv4
```

Note

the sub-zone (loc1) is defined first!

`/etc/shorewall/interfaces`

```
#ZONE               INTERFACE           BROADCAST
loc                 eth1                192.168.1.255
```

`/etc/shorewall/hosts`

```
#ZONE               HOSTS
loc1                eth1:192.168.2.0/24
```

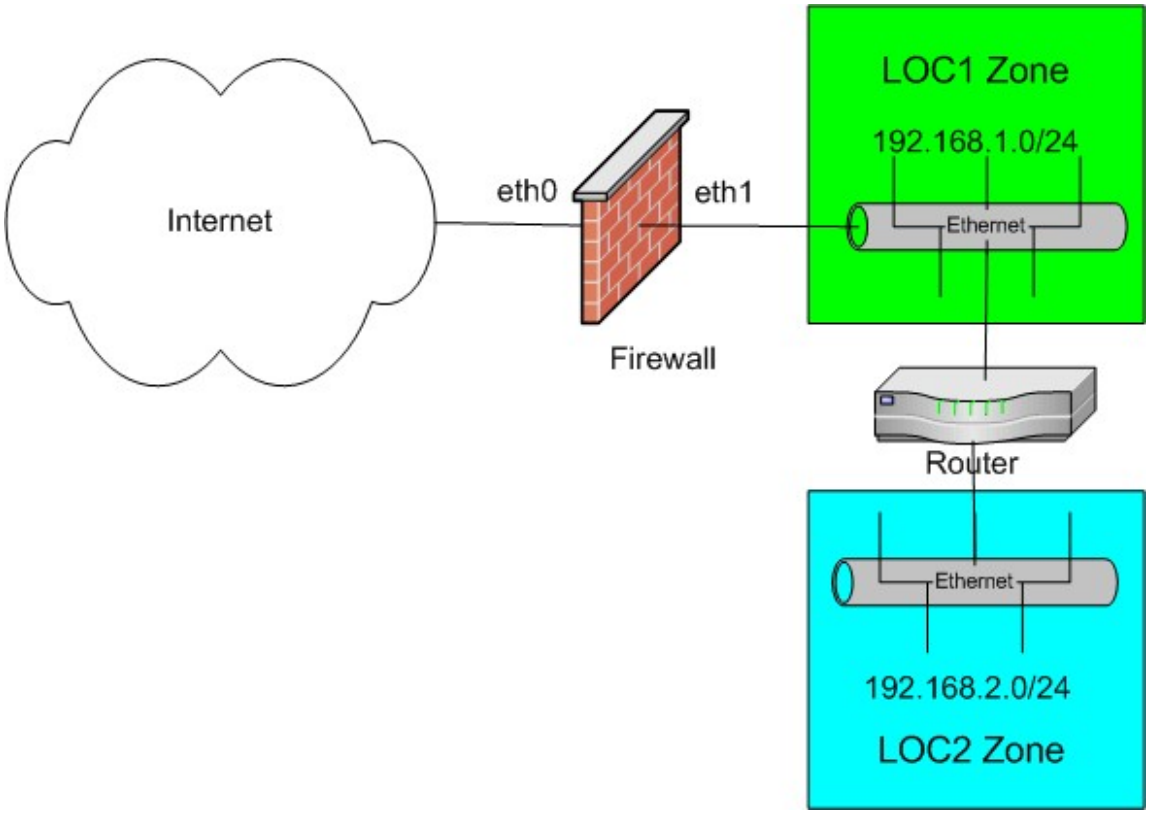
If you don't need Shorewall to set up infrastructure to route traffic between “loc” and “loc1”, add these two policies.

/etc/shorewall/policy

```
#SOURCE             DEST                POLICY
loc                 loc1                NONE
loc1                loc                 NONE
```

Parallel Zones

You define both zones in the /etc/shorewall/hosts file to create two disjoint zones.



/etc/shorewall/zones

```
#ZONE       TYPE      OPTIONS
loc1        ipv4
loc2        ipv4
```

Note

Here it doesn't matter which zone is defined first.

/etc/shorewall/interfaces

```
#ZONE               INTERFACE           BROADCAST
-                   eth1                192.168.1.255
```

/etc/shorewall/hosts

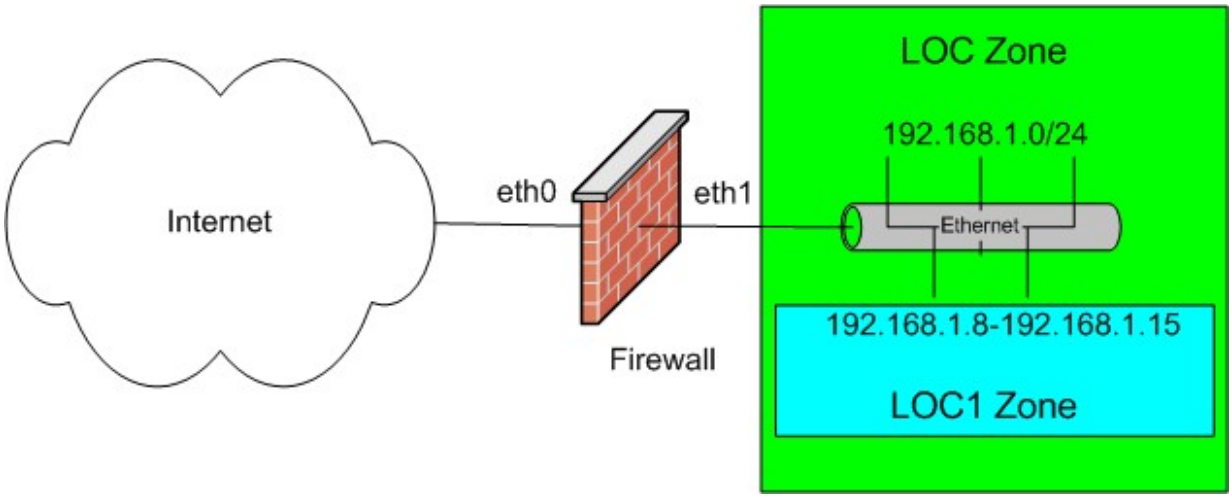
```
#ZONE               HOSTS
loc1                eth1:192.168.1.0/24
loc2                eth1:192.168.2.0/24
```

You don't need Shorewall to set up infrastructure to route traffic between “loc” and “loc1”, so add these two policies:

```
#SOURCE             DEST                POLICY
loc1                loc2                NONE
loc2                loc1                NONE
```

Some Hosts have Special Firewalling Requirements

There are cases where a subset of the addresses associated with an interface need special handling. Here's an example.



In this example, addresses 192.168.1.8 - 192.168.1.15 (192.168.1.8/29) are to be treated as their own zone (loc1).

`/etc/shorewall/zones`

```
#ZONE       TYPE      OPTIONS
loc1        ipv4
loc         ipv4
```

Note

the sub-zone (loc1) is defined first!

`/etc/shorewall/interfaces`

```
#ZONE               INTERFACE           BROADCAST
loc                 eth1                192.168.1.255
```

`/etc/shorewall/hosts`

```
#ZONE               HOSTS
loc1                eth1:192.168.1.8/29
```

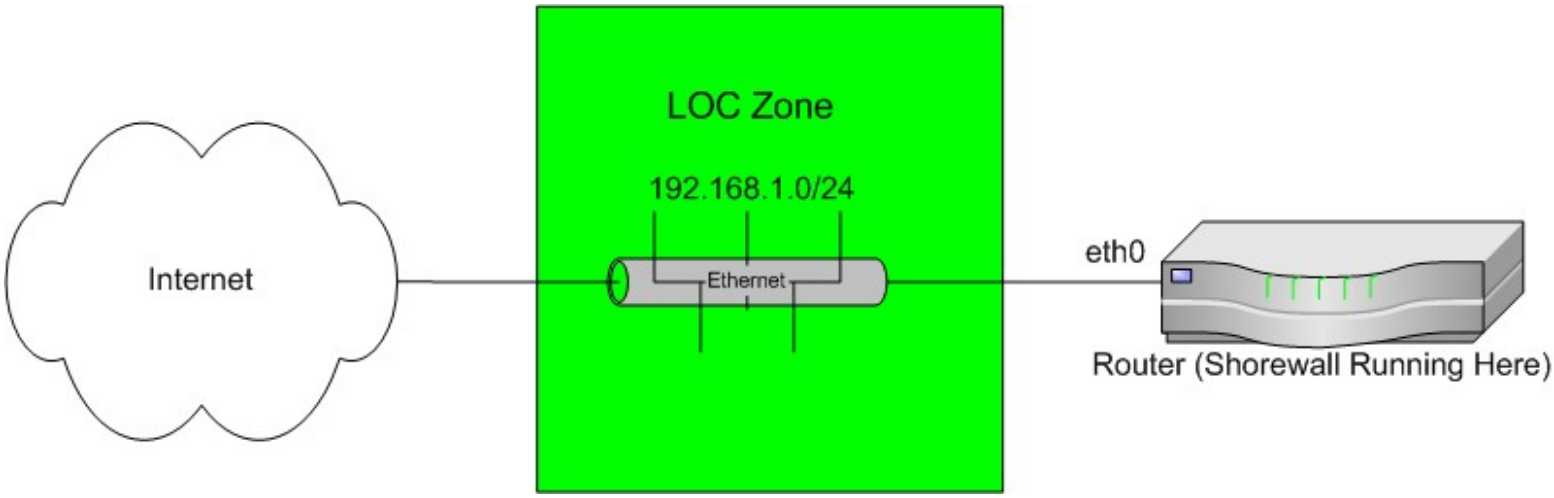
You probably don't want Shorewall to set up infrastructure to route traffic between “loc” and “loc1” so you should add these two policies.

`/etc/shorewall/policy`

```
#SOURCE             DEST                POLICY
loc                 loc1                NONE
loc1                loc                 NONE
```

One-armed Router

Nested zones may also be used to configure a “one-armed” router (I don't call it a “firewall” because it is very insecure. For example, if you connect to the internet via cable modem, your next door neighbor has full access to your local systems as does everyone else connected to the same cable modem head-end controller). Here eth0 is configured with both a public IP address and an RFC 1918 address (More on that topic may be found here). Hosts in the “loc” zone are configured with their default gateway set to the Shorewall router's RFC1918 address.



/etc/shorewall/zones

```
#ZONE       TYPE      OPTIONS
loc1        ipv4
net         ipv4
```

Note

the sub-zone (loc) is defined first!

/etc/shorewall/interfaces

```
#ZONE               INTERFACE           BROADCAST
net                 eth0                detect
```

/etc/shorewall/hosts

```
#ZONE               HOSTS                               OPTIONS
loc                 eth0:192.168.1.0/24                 maclist
```

/etc/shorewall/masq

```
#INTERFACE              SUBNET          ADDRESS
eth0:!192.168.1.0/24    192.168.1.0/24
```

Note that the maclist option is specified in `/etc/shorewall/interfaces`. This is to help protect your router from unauthorized access by your friends and neighbors. Start without maclist then add it and configure your `/etc/shorewall/maclist` file when everything else is working.

•

Shorewall Support Guide

Tom Eastep

Copyright © 2001-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Important

Problem reports that do not include the information requested in the Problem Reporting Guidelines below **will not be answered by the Shorewall author.**

2005-11-11

Table of Contents

Before Reporting a Problem or Asking a Question
Problem Reporting Guidelines
When using the mailing list, please post in plain text
Where to Send your Problem Report or to Ask for Help
Subscribing to the Users Mailing List
Subscribing to the Announce Mailing List
Subscribing to the Development Mailing List
Other Mailing Lists

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Before Reporting a Problem or Asking a Question

There are a number of sources of Shorewall information. Please try these before you post.

- The two currently-supported Shorewall major releases are 3.0 and 2.4.

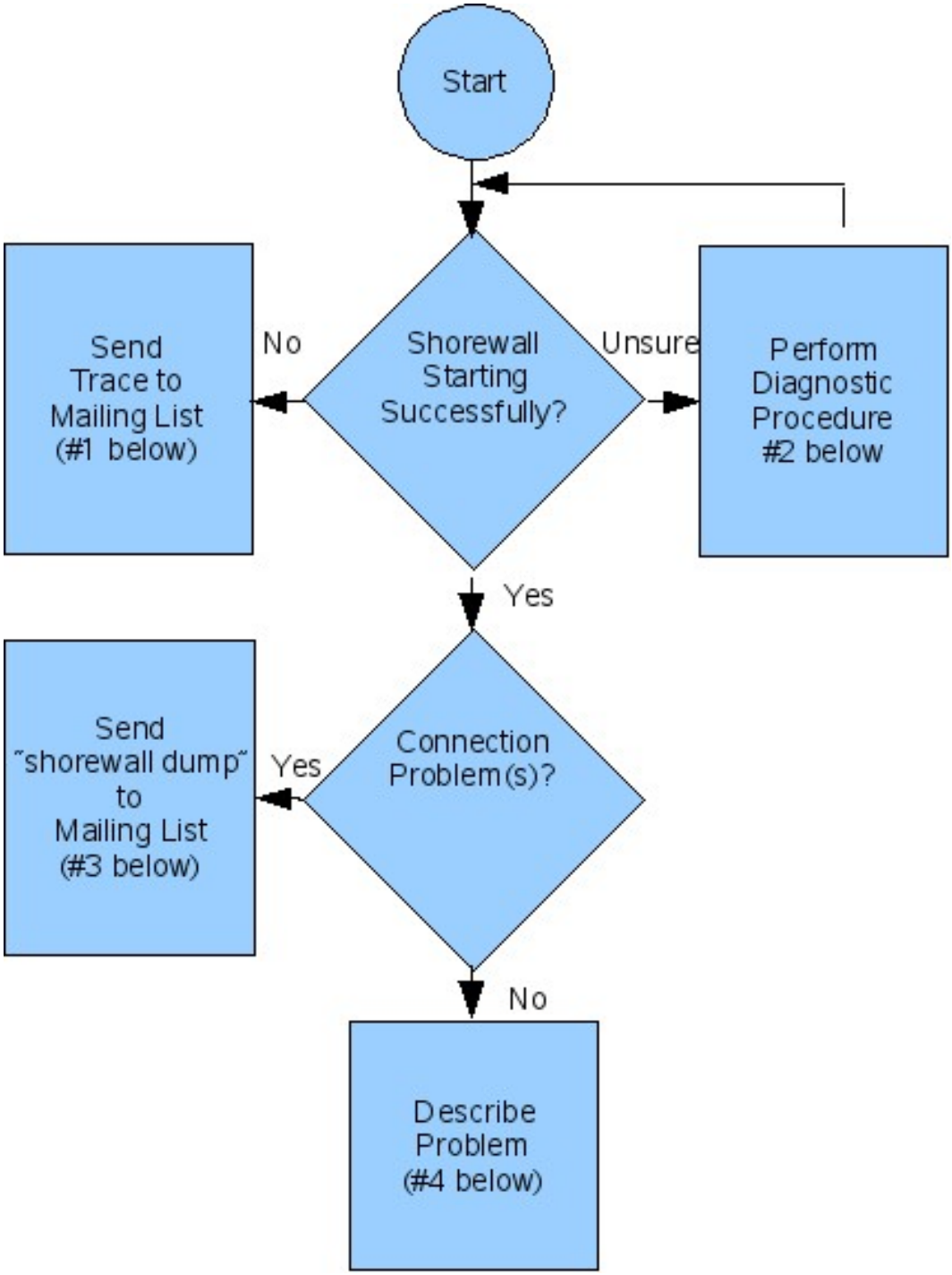
Note

Shorewall versions earlier than 2.4.0 are no longer supported; we will only answer your question if it deals with upgrading from these old releases to a current one.

- More than half of the questions posted on the support list have answers directly accessible from the Documentation Index
- The FAQ has solutions to more than 50 common problems.
- The Troubleshooting Information contains a number of tips to help you solve common problems.

Problem Reporting Guidelines

Please refer to the following flowchart to guide you through the problem reporting process.



1. If your problem is that an **error** occurs when you try to "**shorewall start**" or if Shorewall is otherwise failing to start properly, then please:

   ```
   /sbin/shorewall trace start 2> /tmp/trace
   ```

   Forward the `/tmp/trace` file as an attachment compressed with gzip or bzip2.

2. If you are unsure if Shorewall is starting successfully or not then first note that if Shorewall starts successfully, the last message it produces is "Shorewall Started":

   ```
   …
   Activating Rules...
   Shorewall Started
   gateway:~#
   ```

   If you are seeing this message then Shorewall is starting successfully.

   If you are still unsure if Shorewall is starting or not, enter the following command:

   ```
   /sbin/shorewall status
   ```

   If Shorewall has started successfully, you will see output similar to this:

   ```
   Shorewall-2.5.4 Status at gateway - Tue Aug 30 14:07:29 PDT 2005

   Shorewall is running
   State:Started (Tue Aug 30 07:18:07 PDT 2005)
   ```

   If Shorewall has not started properly, you will see output similar to this:

   ```
   Shorewall-2.5.4 Status at gateway - Tue Aug 30 14:08:11 PDT 2005

   Shorewall is stopped
   State:Stopped (Tue Aug 30 14:08:11 PDT 2005)
   ```

   The "State:" refers to the Shorewall State Diagram.

3. If Shorewall is starting successfully and your problem is that some set of **connections** to/from or through your firewall **isn't working** (examples: local systems can't access the internet, you can't send email through the firewall, you can't surf the web from the firewall, etc.) then please perform the following six steps:
   a. If Shorewall isn't started then **/sbin/shorewall start**. Otherwise **/sbin/shorewall reset**.
   b. Try making the connection that is failing.
   c. **/sbin/shorewall dump** > **/tmp/status.txt**
   d. Post the `/tmp/status.txt` file as an attachment compressed with gzip or bzip2.

   e. Describe where you are trying to make the connection from (IP address) and what host (IP address) you are trying to connect to.
   f. **Please do not edit the diagnostic information** in an attempt to conceal your IP address, netmask, nameserver addresses, domain name, etc. These **aren't secrets**, and concealing them often misleads us and may prevent your problem from being looked at all together.
4. Otherwise please include the following information:
   - the exact version of Shorewall you are running.

     ```
     /sbin/shorewall version
     ```
   - the complete exact output of

     ```
     ip addr show
     ```
   - the complete exact output of

     ```
     ip route show
     ```

- Please remember we only know what is posted in your message. Do not leave out any information that appears to be correct, or was mentioned in a previous post. There have been countless posts by people who were sure that some part of their configuration was correct when it actually contained a small error. We tend to be skeptics where detail is lacking.
- Please keep in mind that you're asking for **free** technical support. Any help we offer is an act of generosity, not an obligation. Try to make it easy for us to help you. Follow good, courteous practices in writing and formatting your e-mail. Provide details that we need if you expect good answers. Exact quoting of error messages, log entries, command output, and other output is better than a paraphrase or summary.
- Please give details about what doesn't work. Reports that say “I followed the directions and it didn't work” may elicit sympathy but probably little in the way of help. Again -- if ping from A to B fails, say so (and see below for information about reporting “ping” problems). If Computer B doesn't show up in “Network Neighborhood” then say so. If access by IP address works but by DNS names it doesn't then say so.
- Please don't describe your environment and then ask us to send you custom configuration files. We're here to answer your questions but we can't do your job for you.
- Please do NOT include the output of **iptables -L** — the output of **shorewall show** or **shorewall status** is much more useful.
- As a general matter, **please do not edit the diagnostic information** in an attempt to conceal your IP address, netmask, nameserver addresses, domain name, etc. These **aren't secrets**, and concealing them often misleads us (and 80% of the time, a hacker could derive them anyway from information contained in the SMTP headers of your post).
- Do you see any “Shorewall” messages (“**/sbin/shorewall show log**”) when you exercise the function that is giving you problems? If so, include the message(s) in your post along with a copy of your /etc/shorewall/interfaces file (and /etc/shorewall/hosts file if you have entries in that file).
- Please include any of the Shorewall configuration files (especially the /etc/shorewall/hosts file if you have modified that file) that you think are relevant. If you include /etc/shorewall/rules, please include /etc/shorewall/policy as well (rules are meaningless unless one also knows the policies).
- **The list server limits the size of posts to the lists, so don't post graphics of your network layout, etc. to the Mailing List -- your post will be rejected**.
- The author gratefully acknowleges that the above list was heavily plagiarized from the excellent LEAF document by *Ray Olszewski* found here.

When using the mailing list, please post in plain text

A growing number of MTAs serving list subscribers are rejecting all HTML traffic. At least one MTA has gone so far as to blacklist shorewall.net “for continuous abuse” because it has been my policy to allow HTML in list posts!!

I think that blocking all HTML is a Draconian way to control spam and that the ultimate losers here are not the spammers but the list subscribers whose MTAs are bouncing all shorewall.net mail. As one list subscriber wrote to me privately “These e-mail admin's need to get a (expletive deleted) life instead of trying to rid the planet of HTML based e-mail”. Nevertheless, to allow subscribers to receive list posts as must as possible, I have now configured the list server at shorewall.net to convert all HTML to plain text. Sometimes the conversion process fails in which case, the post sent to the list is empty. Even when conversion succeeds, the converted post is difficult to read so all of us will appreciate it if you just post in plain text to begin with.

Where to Send your Problem Report or to Ask for Help

If you run the current development release and your question involves a feature that is only available in the development release (see the Shorewall Release Model page) -- please post your question or problem to the Shorewall Development Mailing List. **IMPORTANT**: You must subscribe to the list before you will be able to post to it (see link below).

If you run Shorewall under MandrakeSoft Multi Network Firewall (MNF) and you have not purchased an MNF license from MandrakeSoft then you can post non MNF-specific Shorewall questions to the Shorewall users mailing list. Do not expect to get free MNF support on the list.

Otherwise, please post your question or problem to the Shorewall users mailing list. **IMPORTANT**: You must subscribe to the list before you will be able to post to it (see link below).

Please **read the list usage instructions** (found on the information page for each list) before posting.

For **quick questions**, there is also a #shorewall channel at irc.freenode.net.

Subscribing to the Users Mailing List

To Subscribe to the users mailing list go to https://lists.sourceforge.net/mailman/listinfo/shorewall-users.

Subscribing to the Announce Mailing List

To Subscribe to the announce mailing list (low-traffic,read only) go to:

https://lists.sourceforge.net/lists/listinfo/shorewall-announce

Subscribing to the Development Mailing List

To Subscribe to the development mailing list go to https://lists.sourceforge.net/mailman/listinfo/shorewall-devel.

Other Mailing Lists

For information on other Shorewall mailing lists, go to http://sourceforge.net/mail/?group_id=22587 .

Shorewall Release Model

Tom Eastep

Copyright © 2004, 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-09-02

Table of Contents

Shorewall Releases
Old Release Model

Shorewall Releases

1. Releases have a three-level identification *x.y.z* (e.g., 2.0.3).
2. The first two levels (*x.y*) designate the *Major Release Number* (e.g., 2.0).
3. The third level (*z*) designates the *Minor Release Number*.
4. Even numbered major releases (e.g., 1.4, 2.0, 2.2, ...) are *Stable Releases*. No major new features are added to stable releases and new minor releases of a stable release will only contain bug fixes and simple low-risk enhancements. Installing a new minor release for the major release that you are currently running involves no migration issues unless you want to take advantage of an enhancement (for example, if you are running 1.4.10 and I release 1.4.11, your current configuration is 100% compatible with the new release).
5. Support is available through the Mailing List for the two most recent Stable Releases.
6. Odd numbered major releases (e.g., 2.1, 2.3, ...) are *Development Releases*. Development releases are where new functionality is introduced. Documentation for new features will be available but it may not be up to the standards of the stable release documentation. Sites running Development Releases should be prepared to play an active role in testing new features. Bug fixes and problem resolution for the development release take a back seat to support of the stable releases. Problem reports for the current development release should be sent to the Shorewall Development Mailing List.
7. When the level of functionality of the current development release is judged adaquate, the *Beta period* for a new Stable release will begin. Beta releases have identifications of the form *x.y.0-BetaN* where *x.y* is the number of the next Stable Release and *N*=1,2,3... . Betas are expected to

occur rougly once per year. Beta releases may contain new functionality not present in the previous beta release (e.g., 2.2.0-Beta4 may contain functionality not present in 2.2.0-Beta3). When I'm confident that the current Beta release is stable, I will release the first *Release Candidate*. Release candidates have identifications of the form *x.y.0-RCn* where *x.y* is the number of the next Stable Release and *n*=1,2,3... . Release candidates contain no new functionailty -- they only contain bug fixes. When the stability of the current release candidate is judged to be sufficient then that release candidate will be released as the new stable release (e.g., 2.2.0). At that time, the new stable release and the prior stable release are those that are supported.

8. What does it mean for a major release to be *supported*? It means that I will answer questions about the release and that if a bug is found, I will fix the bug and include the fix in the next minor release.
9. Between minor releases, bug fixes will continue to be made available through the Errata page for each major release.

The currently-supported major releases are 2.4.x and 3.x.

Old Release Model

This release model described above was adopted on 2004-07-03 and modified 2004-07-21. Prior to 2004-07-03, a different release model was followed. Highlights of that model were:

1. Releases were numbered in a manner similar to the current release model.
2. Major new functionality was added in minor releases of the current major release. There was no concept of Stable vs Development major releases.
3. Bug fix only releases were always against the last minor release of a major release and had identifications of the form *x.y.zX* (e.g., 2.0.3c) where *X*=1,b,c,... . Consequently, if a user required a bug fix but was not running the last minor release of the associated major release then it might be necessary to accept major new functionailty along with the bug fix.

Shorewall and ipp2p

Tom Eastep

Copyright © 2004, 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-10-04

Table of Contents

Introduction
Scope
Example:

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Introduction

Shorewall verions 2.2.0 and later include support for the ipp2p match facility. This is a departure from my usual policy in that the ipp2p match facility is included in Patch-O-Matic-NG and is unlikely to ever be included in the kernel.org source tree. Questions about how to install the patch or how to build your kernel and/or iptables should not be posted on the Shorewall mailing lists but should rather be referred to the Netfilter Mailing List.

Scope

In the following files, the "PROTO" or "PROTOCOL" column may contain "ipp2p":

/etc/shorewall/tcrules

/etc/shorewall/accounting

/etc/shorewall/rules (Recommend that you place the rules in the ESTABLISHED section of that file).

When the PROTO or PROTOCOL column contains "ipp2p" then the DEST PORT(S) or PORT(S) column may contain a recognized ipp2p option; for a list of the options and their meaning, at a root prompt type:

```
iptables -m ipp2p --help
```

You must not include the leading "--" on the option; Shorewall will supply those characters for you. If you do not include an option then "ipp2p" is assumed (Shorewall will generate "-m ipp2p --ipp2p").

Example:

Example 2 in the ipp2p documentation recommends the following iptables rules:

```
01# iptables -t mangle -A PREROUTING -p tcp -j CONNMARK --restore-mark
02# iptables -t mangle -A PREROUTING -p tcp -m mark ! --mark 0 -j ACCEPT
03# iptables -t mangle -A PREROUTING -p tcp -m ipp2p --ipp2p -j MARK --set-mark 1
04# iptables -t mangle -A PREROUTING -p tcp -m mark --mark 1 -j CONNMARK --save-mark

05# iptables -t mangle -A POSTROUTING -o eth0 -m mark --mark 1 -j CLASSIFY --set-
class 1:12
06# iptables -t mangle -A POSTROUTING -o eth1 -m mark --mark 1 -j CLASSIFY --set-
class 2:12
```

Let's examine the above rules more carefully.

The individual packets of a P2P data stream do not all carry tell-tale signs that are identifiable as being a particular P2P application. So simply asking the ipp2p match code to mark each individual packet isn't enough because only those packets that carry these tell-tale signs will be marked. Fortunately, Netfilter provides a different type of mark -- the *Connection Mark* which is associated with the entry in the conntrack table rather that with the individual packet. You can see connection mark values with the **shorewall show connections** command:

```
gateway:/etc/test# shorewall show connections
Shorewall-2.5.6 Connections at gateway - Tue Oct  4 15:45:11 PDT 2005

tcp      6 269712 ESTABLISHED src=192.168.3.8 dst=206.124.146.177 sport=50584
dport=993 packets=4899
         bytes=302282 src=206.124.146.177 dst=192.168.3.8 sport=993 dport=50584
packets=7760 bytes=10032928 [ASSURED] mark=0 use=1
...
```

Connection marks are persistent -- that is, once a connection mark is set it retains its value until the connection is terminated.

Netfilter provides features to:

1. Mark individual packets with a numeric value.
2. Save the current packet mark value in the connection mark.
3. Restore the value in the connection mark to the current packet.

The strategy employed in the above rules is to mark the connection of each P2P session with a mark value of 1. That way, each packet that is part of the session can be marked using the 'Restore' function and can be classified accordingly.

1. Rule 01# restores the connection mark into the current packet.
2. Rule 02# tests that restored mark and if it is not equal to zero, the packet is ACCEPTed (no further processing).
3. Rule 03# asks the ipp2p match module to examine the packet and if it is identifiable as part of a P2P session, mark the packet with value 1.
4. Rule 04# saves the current packet mark in the conntrack table if the current mark value is 1 (in other words, if it was marked by rule 03#).
5. Rule 05# classifies the packet to traffic shaping class 1:12 if it is going out of eth0 and has mark value 1[1].
6. Rule 06# classifies the packet to traffic shaping class 2:12 if it is going out of eth1 and has mark value 1.

These are implemented in the /etc/shorewall/tcrules file as follows:

```
#MARK                SOURCE        DEST        PROTO        PORT(S)        CLIENT
USER    TEST
#                                                                          PORT(S)
RESTORE:P            -             -           tcp
```

```
CONTINUE:P            -              -            tcp          -               -
-       !0
1:P                   -              -            ipp2p        ipp2p
SAVE:P                -              -            tcp          -               -
-       1
1:12                  -              eth0         -            -               -
-       1
2:12                  -              eth1         -            -               -
-       1
```

These rules do exactly the same thing as their counterparts described above.

---

[1] There are two ways that Netfilter/iptables can classify traffic. It can be classified directly (which is what this example does) by specifying a *classid* of the form <number>:<number> in the MARK column. That is the preferred method. A classid is specified when a traffic shaping class is defined. tc4shorewall assigns a classid of 1:<100 + mark value>. They may also be classified using an *fwmark classifier* which causes the traffic shaping code to classify the traffic based in the packet mark value. That is done by the traffic shaping solution using the **tc filter add** command. The built-in tc4shorewall shaper uses this command so if you are using the built-in traffic shaping solution, you may use either method.

Kazaa Filtering

Tom Eastep

Copyright © 2003-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-09-12

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Beginning with Shorewall version 1.4.8, Shorewall can interface to ftwall. **ftwall** is part of the p2pwall project and is a user-space filter for applications based on the "Fast Track" peer to peer protocol. Applications using this protocol include Kazaa, KazaaLite, iMash and Grokster.

To filter traffic from your "loc" zone with ftwall, you insert the following rules in the ESTABLISHED section of /etc/shorewall/rules file after any DROP or REJECT rules whose source is the "loc" zone.

```
#ACTION SOURCE     DEST       PROTO
QUEUE   loc        net        tcp
QUEUE   loc        net        udp
QUEUE   loc        $FW        udp
```

Now simply configure ftwall as described in the ftwall documentation and restart Shorewall.

Tip

There are ftwall init scripts for use with SuSE™ and Debian™ Linux at http://shorewall.net/pub/shorewall/contrib/ftwall.

Shorewall verions 2.2.0 and later also include support for the ipp2p match facility which can be use to control P2P traffic. See the Shorewall IPP2P documentation for details.

Netfilter Overview

Tom Eastep

Copyright © 2003, 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-01-23

Table of Contents

Netfilter Overview

Netfilter Overview

Netfilter consists of three tables: **Filter**, **Nat** and **Mangle**. Each table has a number of build-in chains: **PREROUTING**, **INPUT**, **FORWARD**, **OUTPUT** and **POSTROUTING**.

Rules in the various tables are used as follows:

Filter

Packet filtering (rejecting, dropping or accepting packets)

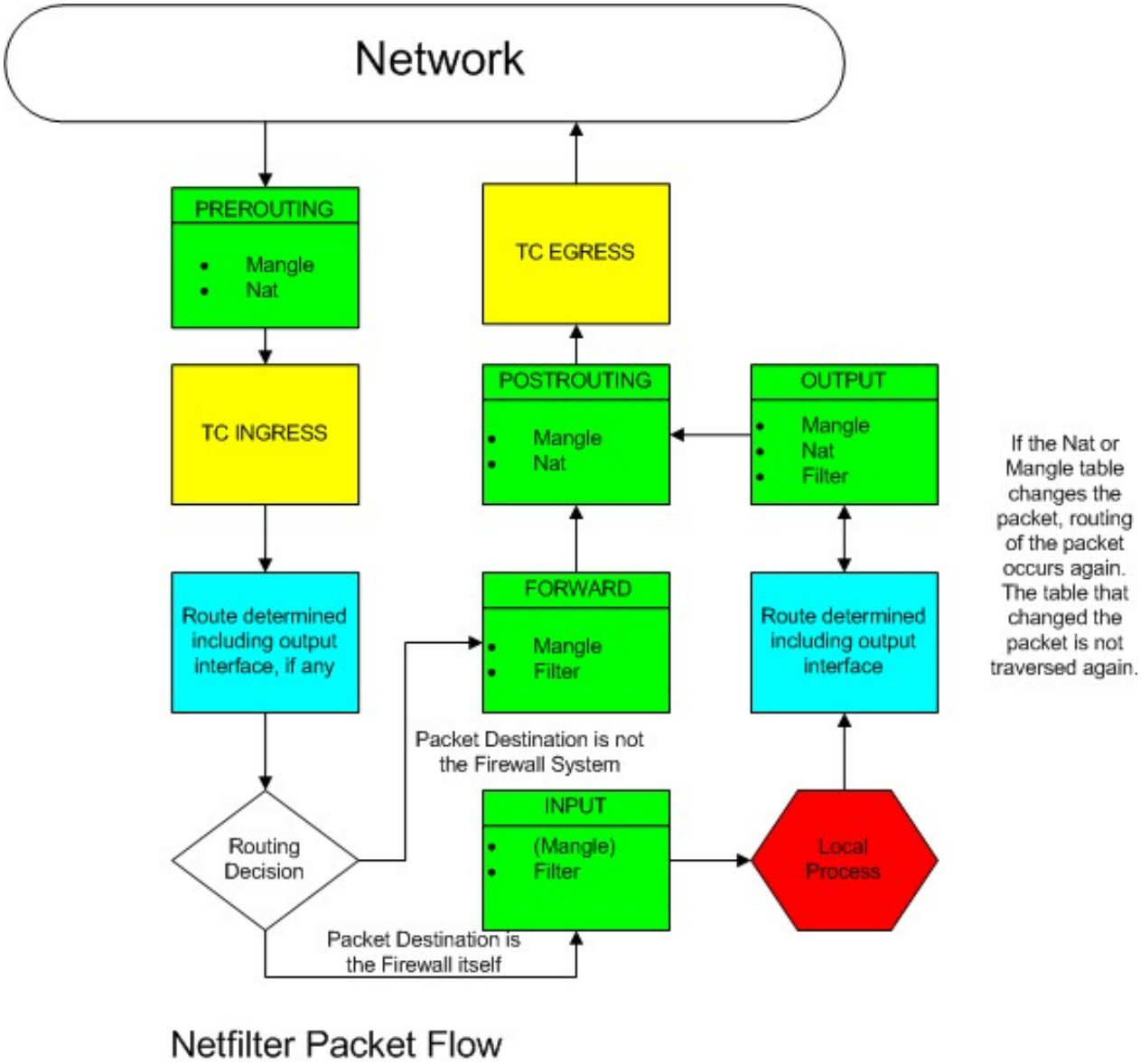
Nat

Network Address Translation including DNAT, SNAT and Masquerading

Mangle

General packet header modification such as setting the TOS value or marking packets for policy routing and traffic shaping.

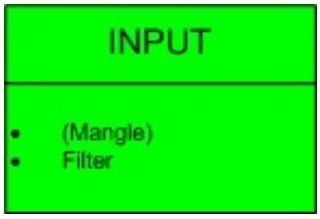
The following diagram shows how packets traverse the various builtin chains within Netfilter. Note that not all table/chain combinations are used.



"Local Process" means a process running on the Shorewall system itself.

A more elaborate version of this flow is available here and this one contrasts the Netfilter flow with that of ipchains.

In the above diagram are boxes similar to this:



The above box gives the name of the built-in chain (**INPUT**) along with the names of the tables (**Mangle** and **Filter**) that the chain exists in and in the order that the chains are traversed. The above sample indicates that packets go first through the **INPUT** chain of the **Mangle** table then through the **INPUT** chain of the **Filter** table. When a chain is enclosed in parentheses, Shorewall does not use the named chain (**INPUT**) in that table (**Mangle**).

Important

Keep in mind that chains in the **Nat** table are **only traversed for new connection requests** (including those related to existing connections) while the chains in the other tables are traversed on every packet.

The above diagram should help you understand the output of “shorewall status”. You may also wish to refer to this article that describes the flow of packets through a Shorewall-generated firewall.

Here are some excerpts from “shorewall status” on a server with one interface (eth0):

```
[root@lists html]# shorewall status

Shorewall-1.4.7 Status at lists.shorewall.net - Mon Oct 13 12:51:13 PDT 2003

Counters reset Sat Oct 11 08:12:57 PDT 2003
```

The first table shown is the **Filter** table.

```
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
 679K  182M ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
 785K   93M accounting  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 DROP      !icmp --  *      *       0.0.0.0/0            0.0.0.0/0
state INVALID
```

The following rule indicates that all traffic destined for the firewall that comes into the firewall on eth0 is passed to a chain called “eth0_in”. That chain will be shown further down.

```
 785K   93M eth0_in    all  --  eth0   *       0.0.0.0/0            0.0.0.0/0
    0     0 common     all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 LOG        all  --  *      *       0.0.0.0/0            0.0.0.0/0
LOG flags 0 level 6 prefix `Shorewall:INPUT:REJECT:'
    0     0 reject     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 accounting  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 DROP      !icmp --  *      *       0.0.0.0/0            0.0.0.0/0
state INVALID
    0     0 eth0_fwd   all  --  eth0   *       0.0.0.0/0            0.0.0.0/0
    0     0 common     all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 LOG        all  --  *      *       0.0.0.0/0            0.0.0.0/0
LOG flags 0 level 6 prefix `Shorewall:FORWARD:REJECT:'
    0     0 reject     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy DROP 1 packets, 60 bytes)
 pkts bytes target     prot opt in     out     source               destination
 679K  182M ACCEPT     all  --  *      lo      0.0.0.0/0            0.0.0.0/0
 922K  618M accounting  all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 DROP      !icmp --  *      *       0.0.0.0/0            0.0.0.0/0
state INVALID
 922K  618M fw2net     all  --  *      eth0    0.0.0.0/0            0.0.0.0/0
    0     0 common     all  --  *      *       0.0.0.0/0            0.0.0.0/0
    0     0 LOG        all  --  *      *       0.0.0.0/0            0.0.0.0/0
LOG flags 0 level 6 prefix `Shorewall:OUTPUT:REJECT:'
    0     0 reject     all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

Here is the eth0_in chain:

```
Chain eth0_in (1 references)
 pkts bytes target     prot opt in     out     source               destination
```

```
 785K   93M dynamic    all  --  *      *       0.0.0.0/0            0.0.0.0/0
 785K   93M net2fw     all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

The “dynamic” chain above is where dynamic blacklisting is done.

Next comes the **Nat** table:

```
NAT Table

Chain PREROUTING (policy ACCEPT 182K packets, 12M bytes)
 pkts bytes target     prot opt in     out     source               destination
20005 1314K net_dnat   all  --  eth0   *       0.0.0.0/0            0.0.0.0/0

Chain POSTROUTING (policy ACCEPT 678K packets, 44M bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 678K packets, 44M bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain net_dnat (1 references)
 pkts bytes target     prot opt in     out     source               destination
  638 32968 REDIRECT   tcp  --  *      *       0.0.0.0/0           !206.124.146.177
tcp dpt:80 redir ports 3128
```

And finally, the **Mangle** table:

```
Mangle Table

Chain PREROUTING (policy ACCEPT 14M packets, 2403M bytes)
 pkts bytes target     prot opt in     out     source               destination
1464K  275M pretos     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain INPUT (policy ACCEPT 14M packets, 2403M bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 15M packets, 7188M bytes)
 pkts bytes target     prot opt in     out     source               destination
1601K  800M outtos     all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain POSTROUTING (policy ACCEPT 15M packets, 7188M bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain outtos (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 TOS        tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
tcp dpt:22 TOS set 0x10
 315K  311M TOS        tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
tcp spt:22 TOS set 0x10
    0     0 TOS        tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
tcp dpt:21 TOS set 0x10
  683 59143 TOS        tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
tcp spt:21 TOS set 0x10
 3667 5357K TOS        tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
tcp spt:20 TOS set 0x08
    0     0 TOS        tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
```

```
tcp dpt:20 TOS set 0x08

Chain pretos (1 references)
 pkts bytes target     prot opt in     out     source               destination
 271K   15M TOS        tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
tcp dpt:22 TOS set 0x10
    0     0 TOS        tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
tcp spt:22 TOS set 0x10
  730 41538 TOS        tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
tcp dpt:21 TOS set 0x10
    0     0 TOS        tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
tcp spt:21 TOS set 0x10
    0     0 TOS        tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
tcp spt:20 TOS set 0x08
 2065  111K TOS        tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
tcp dpt:20 TOS set 0x08
```

Packet Handling

Tom Eastep

Copyright © 2004, 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-10-01

Table of Contents

Introduction
Packets Entering the Firewall from Outside
All Packets
Packets Originating on the Firewall
Packets Leaving the Firewall

Introduction

This article will try to help you understand how packets pass through a firewall configured by Shorewall. You may find it useful to have a copy of the Netfilter Overview handy to refer to.

The discussion that follows assumes that you are running a current kernel (2.4.2n or 2.6.m) with the recommended options included. Otherwise processing may be somewhat different from described below depending on the features supported by your kernel.

Where a packet is covered by steps in more than one of the following sections, processing occurs in the order in which the sections appear.

Packets Entering the Firewall from Outside

Certain processing occurs on packets entering the firewall from the outside that don't occur for packets that originate on the firewall itself.

- The TOS field in the packet is conditionally altered based on the contents of your

`/etc/shorewall/tos` file. This occurs in the **pretos** chain of the *mangle* table.

- Packets are marked based on the contents of your `/etc/shorewall/tcrules` file and the setting of MARK_IN_FORWARD_CHAIN in `/etc/shorewall/shorewall.conf`. This occurs in the **tcpre** chain of the *mangle* table.
- The destination IP address and/or port number are rewritten according to DNAT[-] and REDIRECT[-] rules in `/etc/shorewall/rules`. For new connection requests, this occurs in a chain in the *nat* table called ***zone*_dnat** where *zone* is the zone where the request originated. For packets that are part of an already established connection, the destination rewriting takes place without any involvement of a netfilter rule.
- If the destination was not rewritten in the previous step then it may be rewritten based on entries in /etc/shorewall/nat. For new connection requests, this occurs in a *nat* table chain called ***interface*_in** where *interface* is the interface on which the packet entered the firewall. For packets that are part of an already established connection, the destination rewriting takes place without any involvement of a Netfilter rule.
- The packet passes through the accounting rules defined in `/etc/shorewall/accounting`.
- If Netfilter doesn't know what to do with the packet (connection state INVALID) and the protocol is not ICMP then the packet is silently dropped.
- The packet is processed according to your Blacklisting configuration (dynamic blacklist first). If BLACKLISTNEWONLY=Yes in `/etc/shorewall/shorewall.conf` then only new connection requests are processed. Processing occurs in the dynamic and blacklst
- If the interface on which the packet entered the firewall has the *nosmurfs* option specified in `/etc/shorewall/interfaces`, then if the packet is a new connection request is checked for being a smurf in the *filter* table's **smurfs** chain.
- If:
    - the packet will be processed by the firewall itself
    - the interface on which the packet arrived has the *dhcp* option in `/etc/shorewall/interfaces`.
    - packet's protocol is UDP with destination port 67 or 68.

  then the packet is ACCEPTed in the *filter* table's ***interface*_in** chain (for example, eth0_in).

- If the interface on which the packet entered the firewall has the *norfc1918* option specified in `/etc/shorewall/interfaces`, then the packet is processed against your rfc1918 file (normally `/usr/share/shorewall/rfc1918` but that file may be copied to `/etc/shorewall/rfc1918` and modified). This happens in the *filter* table's **norfc1918** chain.
- If the interface on which the packet entered the firewall has the *nobogons* option specified in `/etc/shorewall/interfaces`, then the packet is processed against your bogons file (normally `/usr/share/shorewall/bogons` but that file may be copied to `/etc/shorewall/bogons` and modified).
- If the interface on which the packet entered the firewall has the *tcpflags* option specified in `/etc/shorewall/interfaces` and the packet's protocol is TCP then the TCP flags are checked by the **tcpflags** chain (*filter* table).

All Packets

Regardless of whether the packet originated on the firewall or came from outside, certain processing steps are common.

- Packets are marked based on the contents of your `/etc/shorewall/tcrules` file and the setting of MARK_IN_FORWARD_CHAIN in `/etc/shorewall/shorewall.conf`. This occurs in the **tcfor** chain of the *mangle* table.

  The remaining processing in this list occurs in the *filter* table.
- If either the host sending the packet or the host to which the packet is addressed is not in any defined zone then the all->all policy is applied to the packet (including logging). This can occur in the INPUT, FORWARD or OUTPUT chains.
- If the packet is part of an established connection or is part of a related connection then no further processing takes place in the filter table (*zone12zone2* chain where *zone1* is the source zone and *zone2* is the destination zone).
- The packet is processed according to your `/etc/shorewall/rules` file. This happens in chains named ***zone12zone2*** chain where *zone1* is the source zone and *zone2* is the destination zone. Note that in the presence of nested or overlapping zones and CONTINUE policies, a packet may go through more than one of these chains.
- Note: If the packet gets to this step, it did not match any rule.

  If the applicable policy has a common action then that action is applied (chain has the same name as the action).
- If the applicable policy has logging specified, the packet is logged.
- The policy is applied (the packet is accepted, dropped or rejected).

Packets Originating on the Firewall

Packets that originate on the firewall itself undergo additional processing.

- The TOS field in the packet is conditionally altered based on the contents of your `/etc/shorewall/tos` file. This occurs in the **outtos** chain of the *mangle* table.
- Packets are marked based on the contents of your `/etc/shorewall/tcrules` file. This occurs in the **tcout** chain of the *mangle* table.

Packets Leaving the Firewall

Packets being sent to another host undergo additional processing.

Note

The source IP address only gets rewritten by the first matching rule below.

- The source IP address may be rewritten according to DNAT rules that specify SNAT. If this is a new connection request, then the rewriting occurs in a *nat* table chain called ***zone*_snat** where *zone* is the destination zone. For packets that are part of an already established connection, the destination rewriting takes place without any involvement of a Netfilter rule.
- The source IP address may be rewritten according to an entry in the `/etc/shorewall/nat` file. If this is a new connection request, then the rewriting occurs in a *nat* table chain called ***interface*_snat** where *interface* is the interface on which the packet will be sent. For packets that are part of an already established connection, the destination rewriting takes place without any involvement of a Netfilter rule.
- The source IP address may be rewritten according to an entry in the `/etc/shorewall/masq` file. If this is a new connection request, then the rewriting occurs in a *nat* table chain called ***interface*_masq** where *interface* is the interface on which the packet will be sent. For packets that are part of an already established connection, the destination rewriting takes place without any involvement of a Netfilter rule.

One-to-one NAT

Tom Eastep

Copyright © 2001-2004 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-10-04

Table of Contents

One-to-one NAT
ARP cache

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

One-to-one NAT

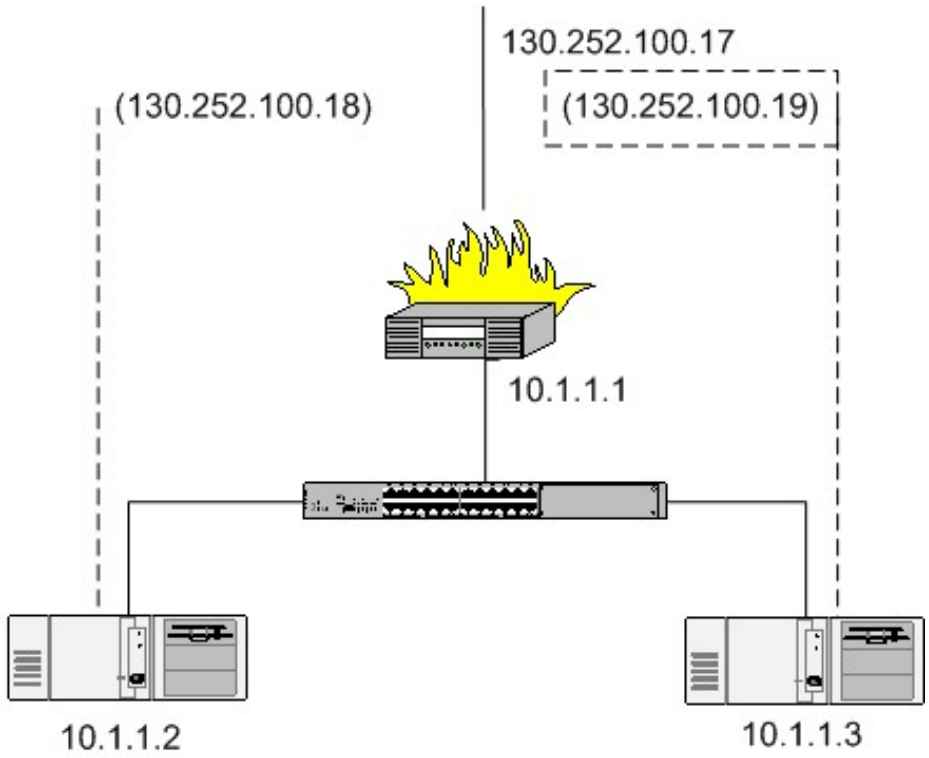
Important

If all you want to do is forward ports to servers behind your firewall, you do NOT want to use one-to-one NAT. Port forwarding can be accomplished with simple entries in the rules file.

One-to-one NAT is a way to make systems behind a firewall and configured with private IP addresses (those reserved for private use in RFC 1918) appear to have public IP addresses. Before you try to use this technique, I strongly recommend that you read the Shorewall Setup Guide.

The following figure represents a one-to-one NAT environment.

One-to-one NAT



One-to-one NAT can be used to make the systems with the 10.1.1.* addresses appear to be on the upper (130.252.100.*) subnet. If we assume that the interface to the upper subnet is eth0, then the following `/etc/shorewall/nat` file would make the lower left-hand system appear to have IP address 130.252.100.18 and the right-hand one to have IP address 130.252.100.19. It should be stressed that these entries in the `/etc/shorewall/nat` file do not automatically enable traffic between the external network and the internal host(s) — such traffic is still subject to your policies and rules.

`/etc/shorewall/nat`

```
#EXTERNAL        INTERFACE        INTERNAL      ALL INTERFACES     LOCAL
130.252.100.18   eth0             10.1.1.2      no                 no
130.252.100.19   eth0             10.1.1.3      no                 no
```

Be sure that the internal system(s) (10.1.1.2 and 10.1.1.3 in the above example) is (are) not included in any specification in `/etc/shorewall/masq` or `/etc/shorewall/proxyarp`.

Note

The “ALL INTERFACES” column is used to specify whether access to the external IP from all firewall interfaces should undergo NAT (Yes or yes) or if only access from the interface in the INTERFACE column should undergo NAT. If you leave this column empty, “No” is assumed . **Specifying “Yes” in this column will not by itself allow systems on the lower LAN to access each other using their public IP addresses.** For example, the lower left-hand system (10.1.1.2) cannot connect to 130.252.100.19 and expect to be connected to the lower right-hand system. See FAQ 2a.

Note

Shorewall will automatically add the external address to the specified interface unless you specify ADD_IP_ALIASES=“no” (or “No”) in `/etc/shorewall/shorewall.conf`; If you do not set ADD_IP_ALIASES or if you set it to “Yes” or “yes” then you must NOT configure your own alias(es).

Note

The contents of the “LOCAL” column determine whether packets originating on the firewall itself and destined for the EXTERNAL address are redirected to the internal ADDRESS. If this column contains “yes” or “Yes” (and the ALL INTERFACES COLUMN also contains “Yes” or “yes”) then such packets are redirected; otherwise, such packets are not redirected. This feature requires kernel 2.4.19 or later and iptables 1.2.6a or later and you must have enabled CONFIG_IP_NF_NAT_LOCAL in your kernel.

ARP cache

A word of warning is in order here. ISPs typically configure their routers with a long ARP cache timeout. If you move a system from parallel to your firewall to behind your firewall with one-to-one NAT, it will probably be HOURS before that system can communicate with the internet.

If you sniff traffic on the firewall's external interface, you can see incoming traffic for the internal system(s) but the traffic is never sent out the internal interface.

You can determine if your ISP's gateway ARP cache is stale using ping and tcpdump. Suppose that we suspect that the gateway router has a stale ARP cache entry for 130.252.100.19. On the firewall, run tcpdump as follows:

```
tcpdump -nei eth0 icmp
```

Now from 10.1.1.3, ping the ISP's gateway (which we will assume is 130.252.100.254):

```
ping 130.252.100.254
```

We can now observe the tcpdump output:

```
13:35:12.159321 0:4:e2:20:20:33 0:0:77:95:dd:19 ip 98: 130.252.100.19 >
130.252.100.254: icmp: echo request (DF)
13:35:12.207615 0:0:77:95:dd:19 0:c0:a8:50:b2:57 ip 98: 130.252.100.254 >
130.252.100.177 : icmp: echo reply
```

Notice that the source MAC address in the echo request is different from the destination MAC address in the echo reply!! In this case 0:4:e2:20:20:33 was the MAC of the firewall's eth0 NIC while 0:c0:a8:50:b2:57 was the MAC address of the system on the lower right. In other words, the gateway's ARP cache still associates 130.252.100.19 with the NIC in that system rather than with the firewall's eth0.

If you have this problem, there are a couple of things that you can try:

1. A reading of *TCP/IP Illustrated, Vol 1* by Stevens reveals[1] that a “gratuitous” ARP packet should cause the ISP's router to refresh their ARP cache (section 4.7). A gratuitous ARP is simply a host requesting the MAC address for its own IP; in addition to ensuring that the IP address isn't a duplicate...

   > if the host sending the gratuitous ARP has just changed its hardware address..., this packet causes any other host...that has an entry in its cache for the old hardware address to update its ARP cache entry accordingly.

   Which is, of course, exactly what you want to do when you switch a host from being exposed to the Internet to behind Shorewall using one-to-one NAT (or Proxy ARP for that matter). Happily enough, recent versions of Redhat's iputils package include “arping”, whose “-U” flag does just that:

   ```
   arping -U -I <net if> <newly proxied IP>
   arping -U -I eth0 66.58.99.83             # for example
   ```

   Stevens goes on to mention that not all systems respond correctly to gratuitous ARPs, but googling for “arping -U” seems to support the idea that it works most of the time.

To use arping with one-to-one NAT in the above example, you would have to:

```
shorewall clear
ip addr add 130.252.100.18 dev eth0     # You need to add the addresses only if
Shorewall clear
ip addr add 130.252.100.19 dev eth0     # deletes them
arping -U -c 10 -I eth0 130.252.100.18
arping -U -c 10 -I eth0 130.252.100.19
ip addr del 130.252.100.18 dev eth0     # You need to delete the addresses only if
you added
ip addr del 130.252.100.19 dev eth0     # them above
shorewall start
```

2. You can call your ISP and ask them to purge the stale ARP cache entry but many either can't or won't purge individual entries.

Warning

There are two distinct versions of **arping** available:

1. **arping** by Thomas Habets (Debian package *arping*).
2. **arping** as part of the iputils package by Alexey Kuznetsov (Debian package *iputils-arping*).

You want the second one by Alexey Kuznetsov.

---

[1] Courtesy of Bradey Honsinger

Shorewall Troubleshooting Guide

Tom Eastep

Copyright © 2001-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-09-11

Table of Contents

First Steps
- Check the FAQs.
- Check the Errata
- Try Searching the Shorewall Site and Mailing List Archives

"shorewall start" and "shorewall restart" Errors
Some Things to Keep in Mind
Your Network Environment
New Device Doesn't Work?
Connection Problems
Ping Problems
Other Gotchas
Still Having Problems?
1. Revision History

First Steps

Some problems are easily solved by checking one of the resources described in the following sections.

Check the FAQs.

Check the FAQs for solutions to over 50 common problems.

Check the Errata

Check the Shorewall Errata to be sure that there isn't an update that you are missing for your version of the firewall.

Try Searching the Shorewall Site and Mailing List Archives

The Site and Mailing List Archives search facility can locate documents and posts about similar problems.

“shorewall start” and “shorewall restart” Errors

If you receive an error message when starting or restarting the firewall and you can't determine the cause, then do the following:

- Make a note of the error message that you see.
- **shorewall debug start 2> /tmp/trace**
- Look at the `/tmp/trace` file and see if that helps you determine what the problem is. Be sure you find the place in the log where the error message you saw is generated -- If you are using Shorewall 1.4.0 or later, you should find the message near the end of the log.
- If you still can't determine what's wrong then see the support page.

Example 1. Startup Error

During startup, a user sees the following:

```
Adding Common Rules
iptables: No chain/target/match by that name
Terminated
```

A search through the trace for “No chain/target/match by that name” turned up the following:

```
+ echo 'Adding Common Rules'
+ add_common_rules
+ run_iptables -A reject -p tcp -j REJECT --reject-with tcp-reset
++ echo -A reject -p tcp -j REJECT --reject-with tcp-reset
++ sed 's/!/! /g'
+ iptables -A reject -p tcp -j REJECT --reject-with tcp-reset
iptables: No chain/target/match by that name
```

The command that failed was: “**iptables -A reject -p tcp -j REJECT --reject-with tcp-reset**”. In this case, the user had compiled his own kernel and had forgotten to include REJECT target support (see kernel.htm)

Some Things to Keep in Mind

- **You cannot test your firewall from the inside**. Just because you send requests to your firewall external IP address does not mean that the request will be associated with the external interface or the “net” zone. Any traffic that you generate from the local network will be associated with your local interface and will be treated as loc->fw traffic.
- **IP addresses are properties of systems, not of interfaces**. It is a mistake to believe that your firewall is able to forward packets just because you can ping the IP address of all of the firewall's interfaces from the local network. The only conclusion you can draw from such pinging success is that the link between the local system and the firewall works and that you probably have the local system's default gateway set correctly.
- **All IP addresses configured on firewall interfaces are in the $FW (fw) zone**. If 192.168.1.254 is the IP address of your internal interface then you can write “**$FW:192.168.1.254**” in a rule but you may not write “**loc:192.168.1.254**”. Similarly, it is nonsensical to add 192.168.1.254 to the **loc** zone using an entry in `/etc/shorewall/hosts`.
- **Reply packets do NOT automatically follow the reverse path of the one taken by the original request**. All packets are routed according to the routing table of the host at each step of the way. This issue commonly comes up when people install a Shorewall firewall parallel to an existing gateway and try to use DNAT through Shorewall without changing the default gateway of the system receiving the forwarded requests. Requests come in through the Shorewall firewall where the destination IP address gets rewritten but replies go out unmodified through the old gateway.
- **Shorewall itself has no notion of inside or outside**. These concepts are embodied in how Shorewall is configured.

Your Network Environment

Many times when people have problems with Shorewall, the problem is actually an ill-conceived network setup. Here are several popular snafus:

- Port Forwarding where client and server are in the same subnet. See FAQ 2.
- Changing the IP address of a local system to be in the external subnet, thinking that Shorewall will suddenly believe that the system is in the “net” zone.
- Multiple interfaces connected to the same HUB or Switch. Given the way that the Linux kernel respond to ARP “who-has” requests, this type of setup **does NOT work the way that you expect it to**. You can test using this kind of configuration if you specify the **arp_filter** option in `/etc/shorewall/interfaces` for all interfaces connected to the common hub/switch. **Using such a setup with a production firewall is strongly recommended against**.

New Device Doesn't Work?

If you have just added a new device such as VOIP and it doesn't work, be sure that you have assigned it an IP address in your local network and that its default gateway has been set to the IP address of your internal interface. For many of these devices, the simplest solution is to run a DHCP server; running it on your firewall is fine — be sure to set the **dhcp** option on your internal interface in /etc/shorewall/interfaces.

Connection Problems

If the appropriate policy for the connection that you are trying to make is ACCEPT, please DO NOT ADD ADDITIONAL ACCEPT RULES TRYING TO MAKE IT WORK. Such additional rules will NEVER make it work, they add clutter to your rule set and they represent a big security hole in the event that you forget to remove them later.

I also recommend against setting all of your policies to ACCEPT in an effort to make something work. That robs you of one of your best diagnostic tools - the “Shorewall” messages that Netfilter will generate when you try to connect in a way that isn't permitted by your rule set.

Check your log (“**/sbin/shorewall show log**”). If you don't see Shorewall messages, then your problem is probably NOT a Shorewall problem. If you DO see packet messages, it may be an indication that you are missing one or more rules -- see FAQ 17.

While you are troubleshooting, it is a good idea to clear two variables in `/etc/shorewall/shorewall.conf`:

```
LOGRATE=
LOGBURST=""
```

This way, you will see all of the log messages being generated (be sure to restart shorewall after clearing these variables).

Example 2. Log Message

```
Jun 27 15:37:56 gateway kernel: Shorewall:all2all:REJECT:IN=eth2
                                OUT=eth1 SRC=192.168.2.2
                                DST=192.168.1.3 LEN=67 TOS=0x00
                                PREC=0x00 TTL=63 ID=5805 DF
                                PROTO=UDP SPT=1803 DPT=53 LEN=47
```

Let's look at the important parts of this message:

- all2all:REJECT - This packet was REJECTed out of the all2all chain -- the packet was rejected under the “all”->“all” REJECT policy (see FAQ 17).
- IN=eth2 - the packet entered the firewall via eth2

- OUT=eth1 - if accepted, the packet would be sent on eth1
- SRC=192.168.2.2 - the packet was sent by 192.168.2.2
- DST=192.168.1.3 - the packet is destined for 192.168.1.3
- PROTO=UDP - UDP Protocol
- DPT=53 - DNS

In this case, 192.168.2.2 was in the “dmz” zone and 192.168.1.3 is in the “loc” zone. I was missing the rule:

```
#ACTION    SOURCE            DEST                    PROTO    DEST
#                                                            PORT(S)
ACCEPT     dmz               loc                     udp      53
```

Ping Problems

Either can't ping when you think you should be able to or are able to ping when you think that you shouldn't be allowed? Shorewall's “Ping” Management is [described here](). Here are a couple of tips:

- Remember that Shorewall doesn't automatically allow ICMP type 8 (“ping”) requests to be sent between zones. If you want pings to be allowed between zones, you need a rule of the form:

  ```
  #ACTION   SOURCE          DEST                    PROTO    DEST
  #                                                          PORT(S)
  Ping/ACCEPT <source zone>   <destination zone>
  ```

  The ramifications of this can be subtle. For example, if you have the following in `/etc/shorewall/nat`:

  ```
  #EXTERNAL    INTERFACE   INTERNAL
  10.1.1.2     eth0        130.252.100.18
  ```

  and you ping 130.252.100.18, unless you have allowed icmp type 8 between the zone containing the system you are pinging from and the zone containing 10.1.1.2, the ping requests will be dropped.
- Ping requests are subject to logging under your policies. So ping floods can cause an equally big flood of log messages. To eliminate these, as the last rule in your /etc/shorewall/rules file add:

  ```
  #ACTION   SOURCE          DEST                    PROTO    DEST
  #                                                          PORT(S)
  Ping/DROP net              all
  ```

Other Gotchas

- Seeing rejected/dropped packets logged out of the INPUT or FORWARD chains? This means that:
  1. your zone definitions are screwed up and the host that is sending the packets or the destination host isn't in any zone (using an `/etc/shorewall/hosts` file are you?); or
  2. the source and destination hosts are both connected to the same interface and you don't have a policy or rule for the source zone to or from the destination zone or you haven't set the **routeback** option for the interface in `/etc/shorewall/interfaces`.
- If you specify “routefilter” for an interface, that interface must be up prior to starting the firewall.
- Is your routing correct? For example, internal systems usually need to be configured with their default gateway set to the IP address of their nearest firewall interface. One often overlooked aspect of routing is that in order for two hosts to communicate, the routing between them must be set up **in both directions**. So when setting up routing between **A** and **B**, be sure to verify that the route from **B** back to **A** is defined.
- Some versions of LRP (EigerStein2Beta for example) have a shell with broken variable expansion. You can get a corrected shell from the Shorewall Errata download site.
- Do you have your kernel properly configured? Click here to see my kernel configuration.
- Shorewall requires the “ip” program. That program is generally included in the “iproute” package which should be included with your distribution (though many distributions don't install iproute by default). You may also download the latest source tarball from http://developer.osdl.org/dev/iproute2/download/ .
- Problems with NAT? Be sure that you let Shorewall add all external addresses to be use with NAT unless you have set ADD_IP_ALIASES =No in `/etc/shorewall/shorewall.conf`.

Still Having Problems?

See the Shorewall Support Page.

1. Revision History

Revision History		
Revision 2.0	2005-09-11	CR
Updated for Shorewall 3.0		
Revision 1.9	2004-08-25	TE
Advice for the networking-challenged.		
Revision 1.8	2004-04-03	TE

Point out that firewall addresses are in the $FW zone.

Revision 1.7	2004-02-02	TE

Add hint about testing from inside the firewall.

Revision 1.6	2004-01-06	TE

Add pointer to Site and Mailing List Archives Searches.

Revision 1.5	2004-01-01	TE

Added information about eliminating ping-generated log messages.

Revision 1.4	2003-12-22	TE

Initial Docbook Conversion

GRE and IPIP Tunnels

Tom Eastep

Copyright © 2001, 2002, 2003, 2004, 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-09-30

Table of Contents

Bridging two Masqueraded Networks

Warning

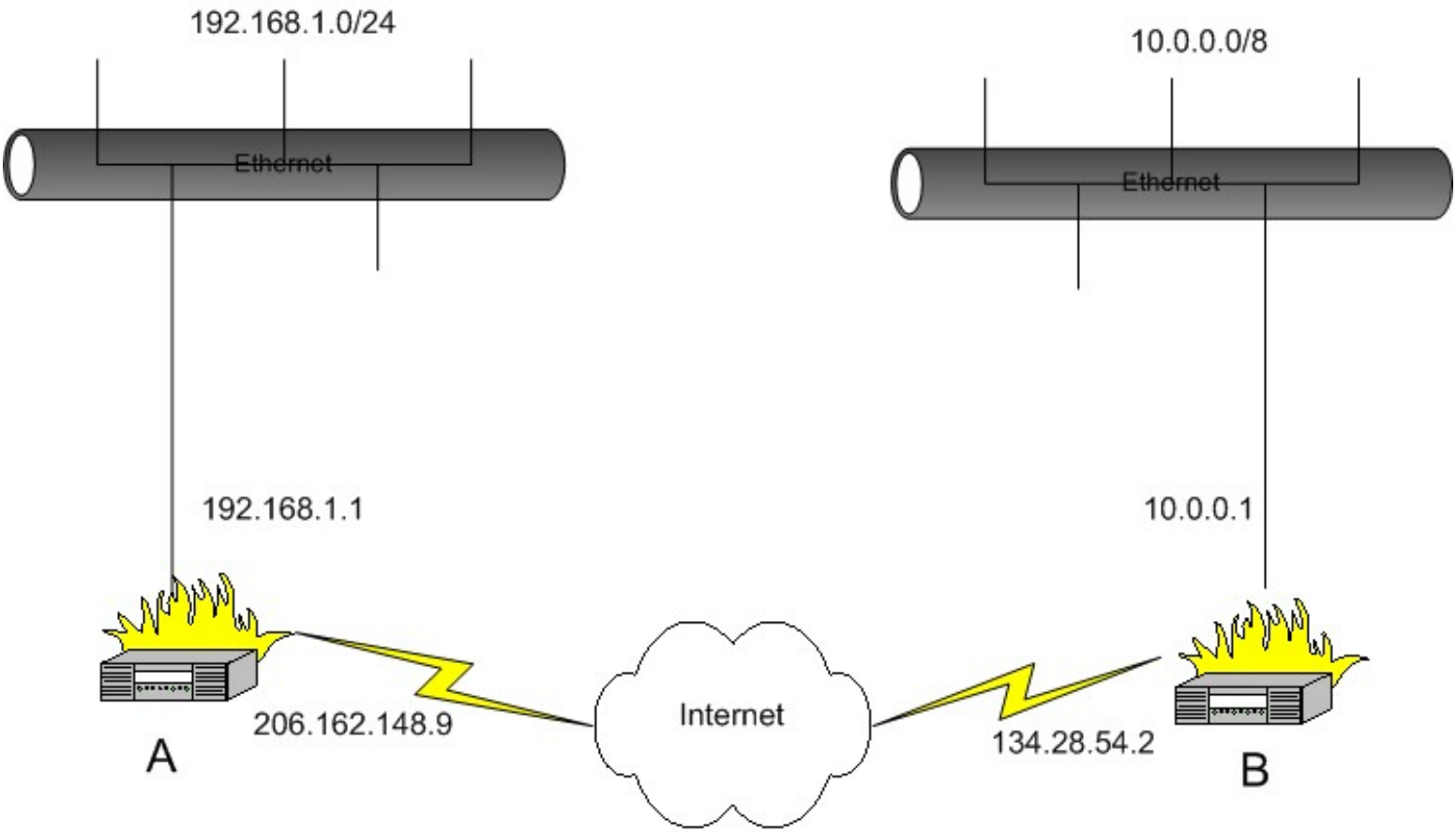
GRE and IPIP Tunnels are insecure when used over the internet; use them at your own risk

GRE and IPIP tunneling with Shorewall can be used to bridge two masqueraded networks.

The simple scripts described in the *Linux Advanced Routing and Shaping HOWTO* work fine with Shorewall. Shorewall also includes a tunnel script for automating tunnel configuration. If you have installed the RPM, the tunnel script may be found in the Shorewall documentation directory (usually /usr/share/doc/shorewall-<version>/).

Bridging two Masqueraded Networks

Suppose that we have the following situation:



We want systems in the 192.168.1.0/24 subnetwork to be able to communicate with the systems in the 10.0.0.0/8 network. This is accomplished through use of the /etc/shorewall/tunnels file, the /etc/shorewall/policy file and the /etc/shorewall/tunnel script that is included with Shorewall.

The “tunnel” script is not installed in /etc/shorewall by default -- If you install using the tarball, the script is included in the tarball; if you install using the RPM, the file is in your Shorewall documentation directory (normally /usr/share/doc/shorewall-<version>).

In the /etc/shorewall/tunnel script, set the “tunnel_type” parameter to the type of tunnel that you want to create.

Example 1. /etc/shorewall/tunnel

```
tunnel_type=gre
```

Warning

If you use the PPTP connection tracking modules from Netfilter Patch-O-Matic (ip_conntrack_proto_gre ip_conntrack_pptp, ip_nat_proto_gre and ip_nat_pptp) then you cannot use GRE tunnels.

On each firewall, you will need to declare a zone to represent the remote subnet. We'll assume that this zone is called “vpn” and declare it in /etc/shorewall/zones on both systems as follows.

```
#ZONE        TYPE          OPTIONS
vpn          ipv4
```

On system A, the 10.0.0.0/8 will comprise the **vpn** zone. In /etc/shorewall/interfaces:

```
#ZONE        INTERFACE     BROADCAST        OPTIONS
vpn          tosysb        10.255.255.255
```

In /etc/shorewall/tunnels on system A, we need the following:

```
#TYPE          ZONE           GATEWAY           GATEWAY ZONE
ipip           net            134.28.54.2
```

This entry in /etc/shorewall/tunnels, opens the firewall so that the IP encapsulation protocol (4) will be accepted to/from the remote gateway.

In the tunnel script on system A:

Example 2. tunnel script on system A

```
tunnel=tosysb
myrealip=206.161.148.9 (for GRE tunnel only)
myip=192.168.1.1
hisip=10.0.0.1
gateway=134.28.54.2
subnet=10.0.0.0/8
```

Similarly, On system B the 192.168.1.0/24 subnet will comprise the **vpn** zone. In /etc/shorewall/interfaces:

```
#ZONE         INTERFACE           BROADCAST
vpn           tosysa              192.168.1.255
```

In /etc/shorewall/tunnels on system B, we have:

```
#TYPE         ZONE            GATEWAY            GATEWAY ZONE
ipip          net             206.191.148.9
```

And in the tunnel script on system B:

Example 3. tunnel script on system B

```
tunnel=tosysa
myrealip=134.28.54.2 (for GRE tunnel only)
myip=10.0.0.1
hisip=192.168.1.1
gateway=206.191.148.9
subnet=192.168.1.0/24
```

You can rename the modified tunnel scripts if you like; be sure that they are secured so that root can execute them.

You will need to allow traffic between the “vpn” zone and the “loc” zone on both systems -- if you simply want to admit all traffic in both directions, you can use the policy file:

```
#SOURCE           DEST           POLICY          LOG LEVEL
loc               vpn            ACCEPT
vpn               loc            ACCEPT
```

On both systems, restart Shorewall and run the modified tunnel script with the “start” argument on each system. The systems in the two masqueraded subnetworks can now talk to each other

6to4 Tunnels

Eric de Thouars

Tom Eastep

Copyright © 2003-2004 Eric de Thoars and Tom Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2004-01-05

Table of Contents

Connecting two IPv6 Networks

Warning

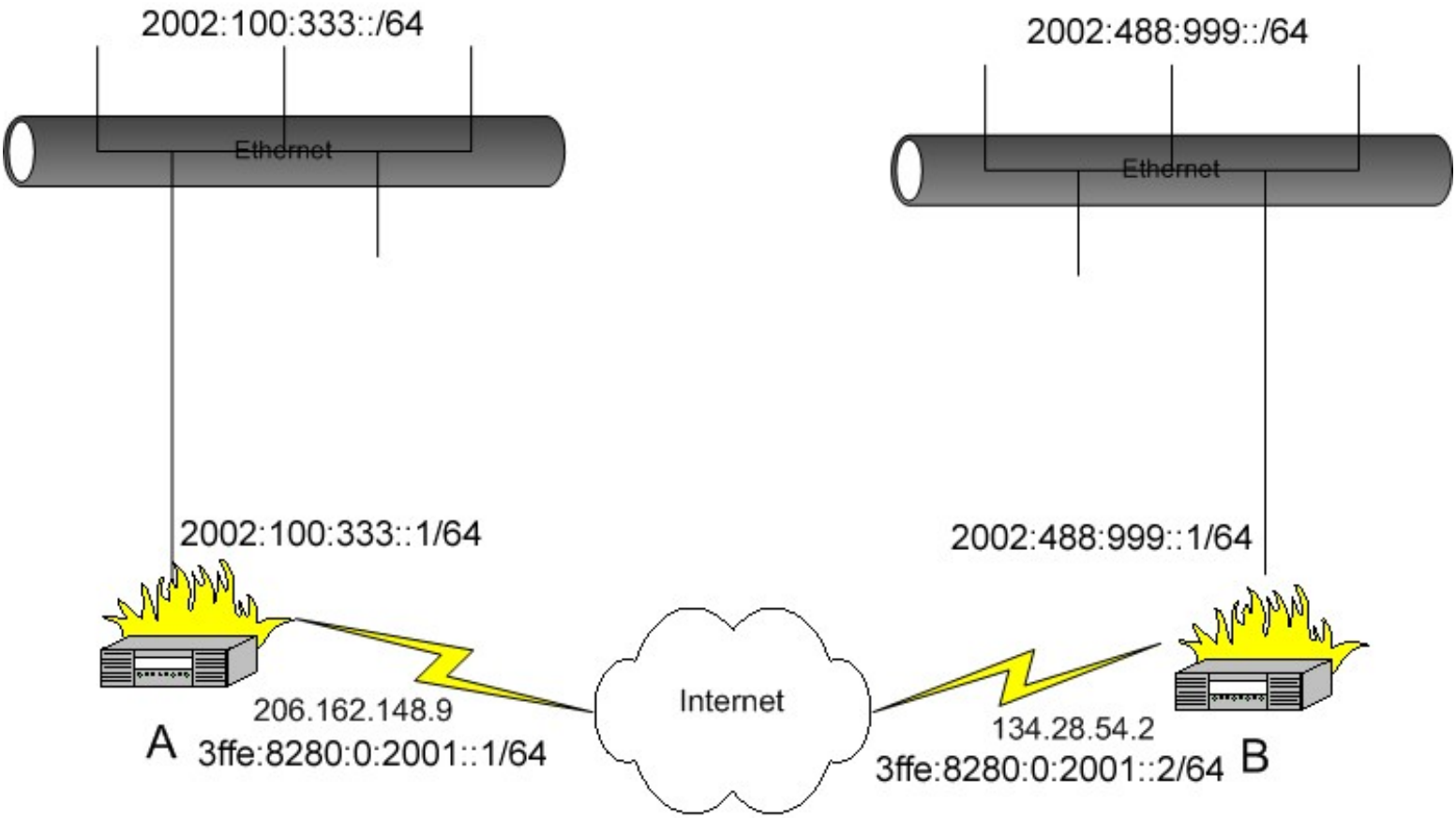
The 6to4 tunnel feature of Shorewall only facilitates IPv6 over IPv4 tunneling. It does not provide any IPv6 security measures.

6to4 tunneling with Shorewall can be used to connect your IPv6 network to another IPv6 network over an IPv4 infrastructure.

More information on Linux and IPv6 can be found in the Linux IPv6 HOWTO. Details on how to setup a 6to4 tunnels are described in the section Setup of 6to4 tunnels.

Connecting two IPv6 Networks

Suppose that we have the following situation:



We want systems in the 2002:100:333::/64 subnetwork to be able to communicate with the systems in the 2002:488:999::/64 network. This is accomplished through use of the `/etc/shorewall/tunnels` file and the “ip” utility for network interface and routing configuration.

Unlike GRE and IPIP tunneling, the `/etc/shorewall/policy`, `/etc/shorewall/interfaces` and `/etc/shorewall/zones` files are not used. There is no need to declare a zone to represent the remote IPv6 network. This remote network is not visible on IPv4 interfaces and to iptables. All that is visible on the IPv4 level is an IPv4 stream which contains IPv6 traffic. Separate IPv6 interfaces and ip6tables rules need to be defined to handle this traffic.

In `/etc/shorewall/tunnels` on system A, we need the following:

```
#TYPE      ZONE      GATEWAY         GATEWAY ZONE
6to4       net       134.28.54.2
```

This entry in `/etc/shorewall/tunnels`, opens the firewall so that the IPv6 encapsulation protocol (41) will be accepted to/from the remote gateway.

Use the following commands to setup system A:

```
>ip tunnel add tun6to4 mode sit ttl 254 remote 134.28.54.2
>ip link set dev tun6to4 up
>ip addr add 3ffe:8280:0:2001::1/64 dev tun6to4
>ip route add 2002:488:999::/64 via 3ffe:8280:0:2001::2
```

Similarly, in `/etc/shorewall/tunnels` on system B we have:

```
#TYPE      ZONE      GATEWAY         GATEWAY ZONE
6to4       net       206.191.148.9
```

And use the following commands to setup system B:

```
>ip tunnel add tun6to4 mode sit ttl 254 remote 206.191.148.9
>ip link set dev tun6to4 up
>ip addr add 3ffe:8280:0:2001::2/64 dev tun6to4
>ip route add 2002:100:333::/64 via 3ffe:8280:0:2001::1
```

On both systems, restart Shorewall and issue the configuration commands as listed above. The systems in both IPv6 subnetworks can now talk to each other using IPv6.

Generic Tunnels

Tom Eastep

Copyright © 2001, 2002, 2003, 2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2003-09-30

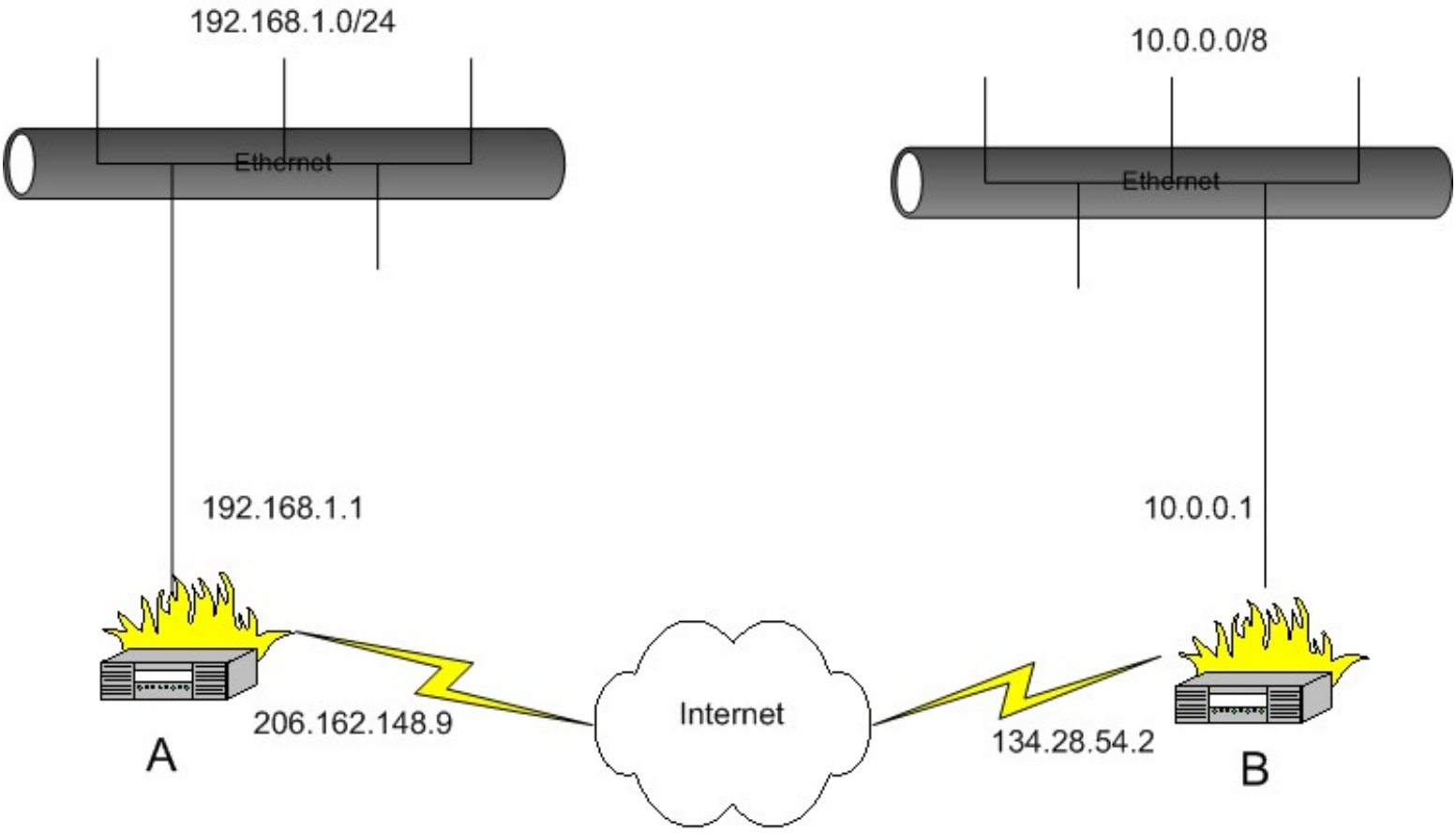
Table of Contents

Bridging two Masqueraded Networks

Shorewall includes built-in support for a wide range of VPN solutions. If you have need for a tunnel type that does not have explicit support, you can generally describe the tunneling software using "generic tunnels".

Bridging two Masqueraded Networks

Suppose that we have the following situation:



We want systems in the 192.168.1.0/24 subnetwork to be able to communicate with the systems in the 10.0.0.0/8 network. This is accomplished through use of the /etc/shorwall/tunnels file, the /etc/shorewall/policy file and the /etc/shorewall/tunnel script that is included with Shorewall.

Suppose that you have tunneling software that uses two different protocols:

a. TCP port 1071
b. GRE (Protocol 47)
c. The tunnel interface on system A is “tun0” and the tunnel interface on system B is also “tun0”.

On each firewall, you will need to declare a zone to represent the remote subnet. We'll assume that this zone is called “vpn” and declare it in /etc/shorewall/zones on both systems as follows.

```
#ZONE        TYPE          OPTIONS
vpn          ipv4
```

On system A, the 10.0.0.0/8 will comprise the **vpn** zone. In /etc/shorewall/interfaces:

```
#ZONE      INTERFACE       BROADCAST        OPTIONS
vpn        tun0            10.255.255.255
```

In /etc/shorewall/tunnels on system A, we need the following:

```
#TYPE            ZONE           GATEWAY         GATEWAY ZONE
generic:tcp:1071 net            134.28.54.2
generic:47       net            134.28.54.2
```

These entries in /etc/shorewall/tunnels, opens the firewall so that TCP port 1071 and the Generalized Routing Encapsulation Protocol (47) will be accepted to/from the remote gateway.

```
#ZONE        INTERFACE        BROADCAST         OPTIONS
vpn          tun0             192.168.1.255
```

In /etc/shorewall/tunnels on system B, we have:

```
#TYPE            ZONE           GATEWAY         GATEWAY ZONE
generic:tcp:1071 net            206.191.148.9
generic:47       net            206.191.148.9
```

You will need to allow traffic between the “vpn” zone and the “loc” zone on both systems -- if you simply want to admit all traffic in both directions, you can use the policy file:

```
#SOURCE      DEST        POLICY        LOG LEVEL
loc          vpn         ACCEPT
vpn          loc         ACCEPT
```

On both systems, restart Shorewall and start your VPN software on each system. The systems in the two masqueraded subnetworks can now talk to each other

Whitelisting Under Shorewall

Tom Eastep

Copyright © 2002-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-09-30

White lists are most often used to give special privileges to a set of hosts within an organization. Let us suppose that we have the following environment:

- A firewall with three interfaces -- one to the Internet, one to a local network and one to a DMZ.
- The local network uses SNAT to the internet and is comprised of the Class B network `10.10.0.0/16` (Note: While this example uses an RFC 1918 local network, the technique described here in no way depends on that or on SNAT. It may be used with Proxy ARP, Subnet Routing, Static NAT, etc.).
- The network operations staff have workstations with IP addresses in the Class C network `10.10.10.0/24`.
- We want the network operations staff to have full access to all other hosts.
- We want the network operations staff to bypass the transparent HTTP proxy running on our firewall.

The basic approach will be that we will place the operations staff's class C in its own zone called ops. Here are the appropriate configuration files:

Zone File

```
#ZONE      TYPE          OPTIONS
fw         firewall
net        ipv4
ops        ipv4
loc        ipv4
dmz        ipv4
```

The `ops` zone has been added to the standard 3-zone zones file -- since `ops` is a sub-zone of `loc`, we list it *BEFORE* `loc`.

Interfaces File

```
#ZONE      INTERFACE        BROACAST        OPTIONS
net        eth0             <whatever>      ...
dmz        eth1             <whatever>      ...
-          eth2             10.10.255.255
```

Because `eth2` interfaces to two zones (`ops` and `loc`), we don't specify a zone for it here.

Hosts File

```
#ZONE       HOST(S)                 OPTIONS
ops         eth2:10.10.10.0/24
loc         eth2:0.0.0.0/0
```

Here we define the `ops` and `loc` zones. When Shorewall is stopped, only the hosts in the `ops` zone will be allowed to access the firewall and the DMZ. I use `0.0.0.0/0` to define the `loc` zone rather than `10.10.0.0/16` so that the limited broadcast address (`255.255.255.255`) falls into that zone. If I used `10.10.0.0/16` then I would have to have a separate entry for that special address.

Policy File

```
#SOURCE          DEST         POLICY         LOG LEVEL
ops              all          ACCEPT
all              ops          CONTINUE
loc              net          ACCEPT
net              all          DROP           info
all              all          REJECT         info
```

Two entries for `ops` (in bold) have been added to the standard 3-zone policy file.

Rules File

```
#ACTION   SOURCE      DEST        PROTO        DEST PORT(S)     SOURCE PORTS(S)
ORIGINAL DEST
REDIRECT  loc!ops     3128        tcp          http
```

This is the rule that transparently redirects web traffic to the transparent proxy running on the firewall. The **SOURCE** column explicitly excludes the `ops` zone from the rule.

Routestopped File

```
#INTERFACE          HOST(S)           OPTIONS
eth1
eth2                10.10.10.0/24
```

Standalone Firewall

Tom Eastep

Copyright © 2002-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-11-02

Table of Contents

Introduction
- Requirements
- Before you start
- Conventions

PPTP/ADSL
Shorewall Concepts
External Interface
IP Addresses
Enabling other Connections
Starting and Stopping Your Firewall
Additional Recommended Reading
1. Revision History

Introduction

Setting up Shorewall on a standalone Linux system is very easy if you understand the basics and follow the documentation.

This guide doesn't attempt to acquaint you with all of the features of Shorewall. It rather focuses on what is required to configure Shorewall in one of its most common configurations:

- Linux system
- Single external IP address
- Connection through Cable Modem, DSL, ISDN, Frame Relay, dial-up... or connected to a LAN and you simply wish to protect your Linux system from other systems on that LAN.

Requirements

Shorewall requires that you have the iproute/iproute2 package installed (on RedHat, the package is called *iproute*). You can tell if this package is installed by the presence of an **ip** program on your firewall system. As root, you can use the “which” command to check for this program:

```
[root@gateway root]# which ip
/sbin/ip
[root@gateway root]#
```

Before you start

I recommend that you read through the guide first to familiarize yourself with what's involved then go back through it again making your configuration changes.

Caution

If you edit your configuration files on a Windows system, you must save them as Unix files if your editor supports that option or you must run them through dos2unix before trying to use them. Similarly, if you copy a configuration file from your Windows hard drive to a floppy disk, you must run dos2unix against the copy before using it with Shorewall.

Windows Version of dos2unix

Linux Version of dos2unix

Conventions

Points at which configuration changes are recommended are flagged with ➡.

PPTP/ADSL

➡

If you have an ADSL Modem and you use PPTP to communicate with a server in that modem, you must make the changes recommended here in addition to those described in the steps below. ADSL with PPTP is most commonly found in Europe, notably in Austria.

Shorewall Concepts

➡

The configuration files for Shorewall are contained in the directory `/etc/shorewall` -- for simple setups, you only need to deal with a few of these as described in this guide. After you have installed

Shorewall, you can find the Samples as follows:

1. If you installed using an RPM, the samples will be in the Samples/one-interface/ subdirectory of the Shorewall documentation directory. If you don't know where the Shorewall documentation directory is, you can find the samples using this command:

   ```
   ~# rpm -ql shorewall | fgrep one-interface
   /usr/share/doc/packages/shorewall/Samples/one-interface
   /usr/share/doc/packages/shorewall/Samples/one-interface/interfaces
   /usr/share/doc/packages/shorewall/Samples/one-interface/policy
   /usr/share/doc/packages/shorewall/Samples/one-interface/rules
   /usr/share/doc/packages/shorewall/Samples/one-interface/zones
   ~#
   ```
2. If you installed using the tarball, the samples are in the Samples/one-interface directory in the tarball.
3. If you installed using the .deb, the samples are in /usr/share/doc/shorewall/examples/one-interface.

Warning

Note to Debian Users

If you install using the .deb, you will find that your `/etc/shorewall` directory is empty. This is intentional. The released configuration file skeletons may be found on your system in the directory `/usr/share/doc/shorewall/default-config`. Simply copy the files you need from that directory to `/etc/shorewall` and modify the copies.

Note that you must copy `/usr/share/doc/shorewall/default-config/shorewall.conf` and /usr/share/doc/shorewall/default-config/modules to `/etc/shorewall` even if you do not modify those files.

As each file is introduced, I suggest that you look through the actual file on your system -- each file contains detailed configuration instructions and default entries.

Shorewall views the network where it is running as being composed of a set of *zones*. In the one-interface sample configuration, only two zones are defined:

```
#ZONE   TYPE    OPTIONS                 IN                      OUT
#                                       OPTIONS                 OPTIONS
fw      firewall
net     ipv4
```

Shorewall zones are defined in `/etc/shorewall/zones`.

Note that Shorewall recognizes the firewall system as its own zone. The name of the firewall zone (**fw** in the above example) is stored in the shell variable *$FW* which may be used throughout the rest of the

Shorewall configuration to refer to the firewall itself.

Rules about what traffic to allow and what traffic to deny are expressed in terms of zones.

- You express your default policy for connections from one zone to another zone in the `/etc/shorewall/policy` file.
- You define exceptions to those default policies in the `/etc/shorewall/rules` file.

For each connection request entering the firewall, the request is first checked against the `/etc/shorewall/rules` file. If no rule in that file matches the connection request then the first policy in `/etc/shorewall/policy` that matches the request is applied. If there is a comon action defined for the policy in `/etc/shorewall/actions` or `/usr/share/shorewall/actions.std` then that action is peformed before the action is applied.

The `/etc/shorewall/policy` file included with the one-interface sample has the following policies:

```
#SOURCE ZONE   DESTINATION ZONE   POLICY   LOG LEVEL   LIMIT:BURST
$FW            net                ACCEPT
net            all                DROP     info
all            all                REJECT   info
```

The above policy will:

1. allow all connection requests from the firewall to the internet
2. drop (ignore) all connection requests from the internet to your firewall
3. reject all other connection requests (Shorewall requires this catchall policy).

At this point, edit your `/etc/shorewall/policy` and make any changes that you wish.

External Interface

The firewall has a single network interface. Where Internet connectivity is through a cable or DSL "Modem", the *External Interface* will be the ethernet adapter (**eth0**) that is connected to that "Modem" unless you connect via *Point-to-Point Protocol over Ethernet* (PPPoE) or *Point-to-Point Tunneling Protocol* (PPTP) in which case the External Interface will be a **ppp0**. If you connect via a regular modem, your External Interface will also be **ppp0**. If you connect using ISDN, your external interface will be **ippp0**.

The Shorewall one-interface sample configuration assumes that the external interface is **eth0**. If your configuration is different, you will have to modify the sample /etc/shorewall/interfaces file accordingly. While you are there, you may wish to review the list of options that are specified for the interface. Some hints:

Tip

If your external interface is **ppp0** or **ippp0**, you can replace the “detect” in the second column with “-”.

Tip

If your external interface is **ppp0** or **ippp0** or if you have a static IP address, you can remove “dhcp” from the option list.

IP Addresses

Before going further, we should say a few words about IP Addresses. Normally, your ISP will assign you a single IP address. That address can be assigned statically, by the Dynamic Host Configuration Protocol (DHCP), through the establishment of your dial-up connection, or during establishment of your other type of PPP connection (PPPoA, PPPoE, etc.).

RFC 1918 reserves several *Private* IP address ranges for use in private networks:

```
10.0.0.0    - 10.255.255.255
172.16.0.0  - 172.31.255.255
192.168.0.0 - 192.168.255.255
```

These addresses are sometimes referred to as *non-routable* because the Internet backbone routers will not forward a packet whose destination address is reserved by RFC 1918. In some cases though, ISPs are assigning these addresses then using *Network Address Translation* to rewrite packet headers when forwarding to/from the internet.

Before starting Shorewall, **you should look at the IP address of your external interface and if it is one of the above ranges, you should remove the “norfc1918” option from the entry in `/etc/shorewall/interfaces`.**

Enabling other Connections

Shorewall includes a collection of macros that can be used to quickly allow or deny services. You can find a list of the macros included in your version of Shorewall using the command **ls `/usr/share/shorewall/macro.*`**.

If you wish to enable connections from the internet to your firewall and you find an appropriate macro in `/etc/shorewall/macro.*`, the general format of a rule in `/etc/shorewall/rules` is:

```
#ACTION          SOURCE     DESTINATION      PROTO        DEST PORT(S)
```

```
<macro>/ACCEPT  net        $FW
```

Example 1. You want to run a Web Server and a IMAP Server on your firewall system:

```
#ACTION      SOURCE    DESTINATION     PROTO        DEST PORT(S)
Web/ACCEPT   net       $FW
IMAP/ACCEPT  net       $FW
```

You may also choose to code your rules directly without using the pre-defined macros. This will be necessary in the event that there is not a pre-defined macro that meets your requirements. In that case the general format of a rule in `/etc/shorewall/rules` is:

```
#ACTION    SOURCE    DESTINATION     PROTO        DEST PORT(S)
ACCEPT     net       $FW             <protocol>   <port>
```

Example 2. You want to run a Web Server and a IMAP Server on your firewall system:

```
#ACTION    SOURCE    DESTINATION     PROTO        DEST PORT(S)
ACCEPT     net       $FW             tcp           80
ACCEPT     net       $FW             tcp           143
```

If you don't know what port and protocol a particular application uses, see here.

Important

I don't recommend enabling telnet to/from the internet because it uses clear text (even for login!). If you want shell access to your firewall from the internet, use SSH:

```
#ACTION      SOURCE    DESTINATION     PROTO        DEST PORT(S)
SSH/ACCEPT   net       $FW
```

At this point, edit `/etc/shorewall/rules` to add other connections as desired.

Starting and Stopping Your Firewall

The installation procedure configures your system to start Shorewall at system boot but beginning with Shorewall version 1.3.9 startup is disabled so that your system won't try to start Shorewall before configuration is complete. Once you have completed configuration of your firewall, you can enable Shorewall startup by removing the file `/etc/shorewall/startup_disabled`.

Important

Users of the .deb package must edit `/etc/default/shorewall` and set "startup=1".

Important

You must enable startup by editing /etc/shorewall/shorewall.conf and setting STARTUP_ENABLED=Yes.

The firewall is started using the "**shorewall start**" command and stopped using "**shorewall stop**". When the firewall is stopped, routing is enabled on those hosts that have an entry in `/etc/shorewall/routestopped`. A running firewall may be restarted using the "**shorewall restart**" command. If you want to totally remove any trace of Shorewall from your Netfilter configuration, use "**shorewall clear**".

Warning

If you are connected to your firewall from the internet, do not issue a "**shorewall stop**" command unless you have added an entry for the IP address that you are connected from to `/etc/shorewall/routestopped`. Also, I don't recommend using "**shorewall restart**"; it is better to create an *alternate configuration* and test it using the "**shorewall try**" command.

Additional Recommended Reading

I highly recommend that you review the Common Configuration File Features page -- it contains helpful tips about Shorewall features than make administering your firewall easier.

1. Revision History

Revision History		
Revision 2.0	2005-09-12	TE
More 3.0 Updates		
Revision 1.9	2005-09-02	CR
Update for Shorewall 3.0		
Revision 1.8	2005-07-12	TE
Change reference to rfc1918 to bogons.		
Revision 1.7	2004-02-16	TE
Move /etc/shorewall/rfc1918 to /usr/share/shorewall.		
Revision 1.6	2004-02-05	TE
Update for Shorewall 2.0		
Revision 1.5	2004-01-05	TE

Standards Changes

Revision 1.4	2003-12-30	TE

Add tip about /etc/shorewall/rfc1918 updates.

Revision 1.3	2003-11-15	TE

Initial Docbook Conversion

Standalone Firewall

Tom Eastep

Patrice Vetsel

Fabien Demassieux

Copyright © 2002-2005 Thomas M. Eastep, Patrice Vetsel, Fabien Demasieux

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-01-17

Table of Contents

Introduction
- Pré-requis
- Avant de commencer
- Conventions

PPTP/ADSL
Les Concepts de Shorewall
Interface Externe
Adresse IP
Permettre d'autres connexions
Starting and Stopping Your Firewall
Autres Lectures Recommandées
1. Revision History

Note

Notes du traducteur : Le guide initial a été traduit par VETSEL Patrice que je remercie. J'en ai assuré la révision pour l'adapter à la version 2 de Shorewall. J'espère vous faciliter l'accès et la prise en main d'un firewall performant, efficace, adaptable et facile d'utilisation. Donc félicitations pour la qualité du travail et la disponibilité offerte par Thomas M. Eastep. Si vous trouvez des erreurs ou des améliorations à apporter vous

pouvez me contacter Fabien Demassieux

Introduction

Configurer Shorewall sur un système isolé Linux est très simple si vous comprenez les bases et suivez la documentation.

Ce guide ne veut pas vous apprendre tous les rouages de Shorewall. Il se focalise sur ce qui est nécessaire pour configurer Shorewall, dans son utilisation la plus courante :

- Un système Linux
- Une seule adresse IP externe
- Une connexion passant par un modem câble, ADSL, ISDN, Frame Relay, rtc...

Pré-requis

Shorewall a besoin que le package **iproute**/**iproute2** soit installé (avec la distribution RedHat™, le package s'appelle **iproute**). Vous pouvez vérifier si le package est installé par la présence du programme **ip** sur votre firewall. En tant que `root`, vous pouvez utiliser la commande **which** pour cela:

```
[root@gateway root]# which ip
/sbin/ip
[root@gateway root]#
```

Avant de commencer

Je recommande en premier la lecture complète du guide afin de se familiariser avec les tenants et aboutissants puis de revenir sur les modifications de votre configuration adapté à votre système.

Caution

Si vous éditez vos fichiers de configuration sur un système Windows™, vous devez les sauver comme des fichiers Unix™ si votre éditeur supporte cette option sinon vous devez les convertir avec **dos2unix** avant d'essayer de les utiliser. De la même manière, si vous copiez un fichier de configuration depuis votre disque dur Windows™ vers une disquette, vous devez lancer **dos2unix** sur la copie avant de l'utiliser avec Shorewall.

- Windows™ Version of **dos2unix**
- Linux Version of **dos2unix**

Conventions

Les points ou les modifications s'imposent sont indiqués par ➡.

PPTP/ADSL

➡

Si vous êtes équipé d'un modem ADSL et utilisez PPTP pour communiquer avec un serveur à travers ce modem, vous devez faire le changement suivant en plus de ceux ci-dessous. ADSL avec PPTP est commun en Europe, ainsi qu'en Australie.

Les Concepts de Shorewall

➡

Les fichiers de configuration pour Shorewall sont situés dans le répertoire /etc/shorewall -- pour de simples paramétrages, vous n'avez à faire qu'avec quelques un d'entre eux comme décris dans ce guide.

> ## Tip
>
> Après avoir installé Shorewall, téléchargez l'exemple one-interface, décompressez le (**tar `-zxvf one-interface.tgz`**) et copiez les fichiers dans `/etc/shorewall` **(ces fichiers remplaceront les initiaux)**.

Parallèlement à la présentation, je vous suggère de jeter un oeil à ceux physiquement présents sur votre système -- chacun des fichiers contient des instructions de configuration détaillées et des entrées par défaut.

Shorewall voit le réseau où il fonctionne, comme un ensemble de zones.Dans les fichiers de configuration fournis pour une unique interface, une seule zone est définie :

Name	Description
net	The Internet

Les zones de Shorewall sont définies dans `/etc/shorewall/zones`.

Shorewall reconnaît aussi le système de firewall comme sa propre zone - par défaut, le firewall est connu comme **`fw`**.

Les règles concernant le trafic à autoriser ou à interdire sont exprimées en utilisant les termes de zones.

- Vous exprimez votre politique par défaut pour les connexions d'une zone vers une autre zone dans le fichier `/etc/shorewall/policy`.
- Vous définissez les exceptions à ces politiques pas défaut dans le fichier `/etc/shorewall/rules`.

Pour chaque connexion demandant à entrer dans le firewall, la requête est en premier lieu comparée par rapport au fichier `/etc/shorewall/rules`. Si aucune règle dans ce fichier ne correspond à la demande de connexion alors la première politique dans le fichier `/etc/shorewall/policy` qui y correspond sera appliquée. Si cette politique est `REJECT` ou `DROP` la requête est dans un premier temps comparée par rapport aux règles contenues dans le fichier `/etc/shorewall/common`, si ce fichier existe; sinon les régles dans le fichier `/etc/shorewall/common.def` sont vérifiées.

Le fichier /etc/shorewall/policy inclus dans l'archive d'exemple (one-interface) contient les politiques suivantes:

```
#SOURCE ZONE   DESTINATION ZONE   POLICY   LOG LEVEL   LIMIT:BURST
fw             net                ACCEPT
net            all                DROP     info
all            all                REJECT   info
```

Ces politiques vont :

1. Permettre toutes demandes de connexion depuis le firewall vers l'Internet
2. Drop (ignorer) toutes les demandes de connexion depuis l'Internet vers votre firewall
3. Reject (rejeter) toutes les autres requêtes de connexion (Shorewall à besoin de cette politique).

➡ A ce point, éditez votre /etc/shorewall/policy et faites y les changements que vous désirez.

Interface Externe

Le firewall possède une seule interface réseau. Lorsque la connexion Internet passe par un modem câble ou par un “Routeur” ADSL(pas un simple modem), l'*Interface Externe* sera l'adaptateur ethernet qui y est connecté à ce “Modem” (e.g., `eth0`) à moins de se que vous vous connectiez par *Point-to-Point Protocol* over Ethernet (PPPoE) ou *Point-to-Point Tunneling Protocol* (PPTP) dans ce cas l'interface externe sera (e.g., `ppp0`). Si vous vous connectez par un simple modem (RTC), votre interface externe sera aussi `ppp0`. Si vous vous connectez en utilisant l'ISDN, votre interface externe sera `ipppp0`.

➡

Si votre interface vers l'extérieur est **ppp0** ou **ippp0** alors vous mettrez `CLAMPMSS=yes` dans le fichier `/etc/shorewall/shorewall.conf`.

Le fichier de configuration d'exemple pour une interface suppose que votre interface externe est eth0. Si votre configuration est différente, vous devrez modifier le fichier`/etc/shorewall/interfaces` en conséquence. Tant que vous y êtes, vous pourriez parcourir la liste des options qui sont spécifiées pour les interfaces. Quelques trucs:

Tip

Si votre interface vers l'extérieur est `ppp0` ou `ipppp0`, vous pouvez remplacer le detect dans la seconde colonne par un “-” (sans les quotes).

Tip

Si votre interface vers l'extérieur est `ppp0` or `ipppp0` u si vous avez une adresse IP statique, vous pouvez enlever `dhcp` dans la liste des options .

Tip

Si vous spécifiez *norfc1918* pour votre interface externe, vous pouvez vérifier périodiquement le Shorewall Errata pour mettre à jour le fichier `/usr/share/shorewall/rfc1918`. Sinon, vous pouvez copier `le fichier /usr/share/shorewall/rfc1918` vers `/etc/shorewall/rfc1918` et adapter votre fichier `/etc/shorewall/rfc1918` comme je le fais.

Adresse IP

Avant d'aller plus loin, nous devons dire quelques mots au sujet des adresses Internet Protocol (IP). Normalement, votre fournisseur Internet ISP vous assignera une seule adresse IP. Cette adresse peut être assignée par le Dynamic Host Configuration Protocol (DHCP) ou lors de l'établissement de votre connexion lorsque vous vous connectez (modem standard) ou établissez votre connexion PPP. Dans de rares cas , votre provider peut vous assigner une adresse statique IP ; cela signifie que vous devez configurer l'interface externe de votre firewall afin d'utiliser cette adresse de manière permanente. La **RFC 1918** réserve plusieures plages d'adresses privées *Private* IP à cet fin:

Table 1. Example sub-network

Range:	`10.10.10.0 - 10.10.10.255`
Subnet Address:	`10.10.10.0`
Broadcast Address:	`10.10.10.255`
CIDR Notation:	`10.10.10.0/24`

Ces adresses sont parfois nommées comme *non-routable* car les routers centraux d'Internet ne

renvoient pas un paquet dont la destination est reservée par la RFC 1918. Dans certain cas cependant, les FAI (fournisseurs d'accès Internet) assignent ces adresses et utilisent ensuite NAT *Network Address Translation* pour réécrire les en-têtes de paquets renvoyés vers/depuis Internet.

Avant de lancer Shorewall, regarder l'adresse IP de votre interface externe, et si elle est dans les plages précédentes, vous devez enlever l'option 'norfc1918' dans la ligne concernant l'interface externe dans le fichier `/etc/shorewall/interfaces`.

Permettre d'autres connexions

Shorewall version 2.0.0 et postérieure inclus une collection d'actions qui peuvent être utilisées pour rapidemement autoriser ou refuser des services. Pour voir les actions comprises avec votre version de Shorewall, regardez dans le fichier `/etc/shorewall/actions.std`. Le nom de celles qui acceptent des connexions débutent par “Allow”.

Si vous souhaitez autoriser d'autre connexions depuis internet vers votre firewall, le format général utilisant l'action type “Allow” est:

```
#ACTION    SOURCE     DESTINATION      PROTO        DEST PORT(S)
<action>   net        fw
```

Example 1. Vous voulez un serveur Web et POP3 accessible de l'extérieur sur votre firewall:

```
#ACTION    SOURCE     DESTINATION      PROTO        DEST PORT(S)
AllowWeb   net        fw
AllowPOP3  net        fw
```

Au cas ou Shorewall n'inclue pas d'actions définies qui vous conviennent, vous pouvez les définir vous même ou coder directement les régles dans `/etc/shorewall/rules` selon le format suivant:

```
#ACTION    SOURCE     DESTINATION      PROTO        DEST PORT(S)
ACCEPT     net        fw               <protocol>   <port>
```

Example 2. Vous voulez un serveur Web et POP3 accessible de l'extérieur sur votre firewall:

```
#ACTION    SOURCE     DESTINATION      PROTO        DEST PORT(S)
ACCEPT     net        fw               tcp           80
ACCEPT     net        fw               tcp           110
```

Si vous ne savez pas quel port(s) et protocole(s) requièrent une application particulière, vous pouvez regarder ici.

Important

Je ne recommande pas d'autoriser **telnet** vers/de l'Internet parce qu'il utilise du texte en clair (même pour le login!). Si vous voulez un accès shell à votre firewall, utilisez SSH:

```
#ACTION     SOURCE     DEST                PROTO      DEST PORT(S)
AllowSSH    net        fw
```

Maintenant, éditez votre fichier de configuration `/etc/shorewall/rules` pour ajouter, modifier ou supprimer les autres connexions voulues.

Starting and Stopping Your Firewall

La procédure d'installation configure votre système pour lancer Shorewall au boot du système, mais au début avec la version 1.3.9 de Shorewall le lancement est désactivé, n'essayer pas de lancer Shorewall avant que la configuration soit finie. Une fois que vous en aurez fini avec la configuration du firewall, vous pouvez permettre le lancement de Shorewall en supprimant le fichier `/etc/shorewall/startup_disabled`.

Important

Les utilisateurs des paquets .deb doivent éditer `/etc/default/shorewall` et mettre `startup=1`.

Le firewall est activé en utilisant la commande "**shorewall start**" et arrêté avec "**shorewall stop**". Lorsque le firewall est stoppé, le routage est autorisé sur les hôtes qui possèdent une entrée dans `/etc/shorewall/routestopped`. Un firewall qui tourne peut être relancé en utilisant la commande "**shorewall restart**" command. Si vous voulez enlever toutes traces de Shorewall sur votre configuration de Netfilter, utilisez "**shorewall clear**".

Warning

Si vous êtes connecté à votre firewall depuis Internet, n'essayez pas une commande "**shorewall stop**" tant que vous n'avez pas ajouté une entrée pour votre adresse IP (celle à partir de laquelle vous êtes connectée) dans `/etc/shorewall/routestopped`. De

la même manière, je ne vous recommande pas d'utiliser “**shorewall restart**”; il est plus intéressant de créer une configuration alternative et de la tester en utilisant la commande “**shorewall try**”.

Autres Lectures Recommandées

Je vous recommande vivement de lire la page des Fonctionnalités Générales des Fichiers de Configuration -- elle contient des trucs sur les possibilités de Shorewall pour rendre aisé l'administration de votre firewall Shorewall.

1. Revision History

Revision History		
Revision 1.7	2004-02-16	TE
Move /etc/shorewall/rfc1918 to /usr/share/shorewall.		
Revision 1.6	2004-02-05	TE
Update for Shorewall 2.0		
Revision 1.5	2004-01-05	TE
Standards Changes		
Revision 1.4	2003-12-30	TE
Add tip about /etc/shorewall/rfc1918 updates.		
Revision 1.3	2003-11-15	TE
Initial Docbook Conversion		

Firewall standard à deux interfaces

Tom Eastep

Patrice Vetsel

Fabien Demassieux

Copyright © 2002, 2003, 2004, 2005 Thomas M. Eastep, Patrice Vetsel, Fabien Demassieux

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled " GNU Free Documentation License".

2005-01-17

Table of Contents

Introduction
- Pré-requis
- Conventions

PPTP/ADSL
Les Concepts de Shorewall
Interfaces Réseau
Adresses IP
IP Masquerading (SNAT)
Port Forwarding (DNAT)
Domain Name Server (DNS)
Autres Connexions
Quelques Points à Garder en Mémoire
Démarrer et Arrêter Votre Firewall
Autres Lectures Recommandées
Ajouter un Segment Sans-fil à votre Firewall à deux interfaces

Note

Notes du traducteur : Le guide initial a été traduit par VETSEL Patrice que je remercie. J'en ai assuré la révision pour l'adapter à la version 2 de Shorewall. J'espère vous faciliter l'accès et la prise en main d'un firewall performant, efficace, adaptable et facile d'utilisation. Donc félicitations pour la qualité du travail et la disponibilité offerte par Thomas M. Eastep. Si vous trouvez des erreurs ou des améliorations à apporter vous pouvez me contacter Fabien Demassieux

Introduction

Mettre en place un système Linux en tant que firewall pour un petit réseau est une chose assez simple, si vous comprenez les bases et suivez la documentation.

Ce guide ne prétend pas vous apprendre tous les rouages de Shorewall. Il se focalise sur ce qui est nécessaire pour configurer Shorewall, dans son utilisation la plus courante:

- Un système Linux utilisé en tant que firewall/routeur pour un petit réseau local.
- **Une seule adresse IP publique.**

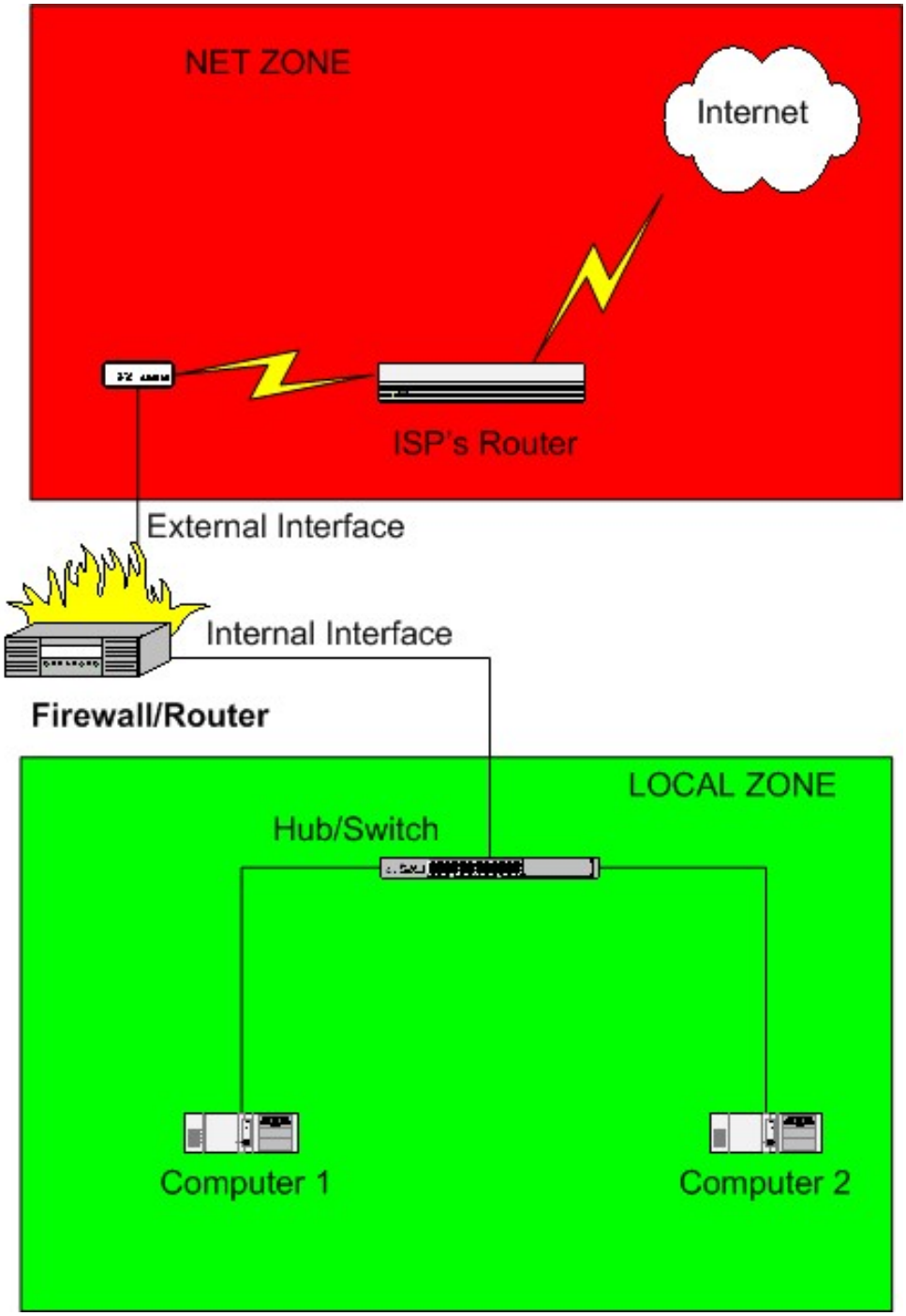
Note

Si vous avez plus d'une adresse IP, ce n'est pas le guide qui vous convient -- regrdez plutôt du coté du Guide de Configuration Shorewall.

- Une connexion Internet par le biais d'un modem câble, ADSL, ISDN, "Frame Relay", RTC ...

Voici un schéma d'une installation typique:

Figure 1. Configuration standard d'un firewall avec deux interfaces



Shorewall and Mandrake™ 9.0+

Si vous utilisez Mandrake™ 9.0 ou version postérieure, vous pouvez facilement utiliser l'utilitaire Mandrake™ “Partage de Connexion Internet”. Dans le *Centre de Contrôle Mandrake*, selectionner “Réseau & Internet” puis “Partage de Connexion”.

Cependant, la configuration de Shorewall générée par le *Partage de Connexion Internet Mandrake* est étrange et peut rendre confus l'utilisation de la suite de cette documentation (elle paramètre deux zones; `loc` and `masq` ou `loc` est vide; Cela est en conflit avec la documentation basée sur une unique zone `loc`). Nous recommandons qu'une fois configuré ce partage, de désinstaller le paquet RPM de Shorewall Mandrake™ et d'installer celui de la page de download avant de suivre l'utilisation de ce Guide.

Note

Le problème précédent est résolu à partir de la version 10.0 et supérieure de Mandrake.

Caution

Si vous éditez vos fichiers de configuration sur un système Windows™, vous devez les sauver comme des fichiers Unix™ si votre éditeur supporte cette option sinon vous devez les convertir avec **dos2unix** avant d'essayer de les utiliser. De la même manière, si vous copiez un fichier de configuration depuis votre disque dur Windows™ vers une disquette, vous devez lancer **dos2unix** sur la copie avant de l'utiliser avec Shorewall.

- Windows™ Version of **dos2unix**
- Linux Version of **dos2unix**

Pré-requis

Shorewall a besoin que le package **iproute**/**iproute2** soit installé (avec la distribution RedHat™, le package s'appelle **iproute**). Vous pouvez vérifier si le package est installé par la présence du programme **ip** sur votre firewall. En tant que `root`, vous pouvez utiliser la commande **which** pour cela:

```
[root@gateway root]# which ip
/sbin/ip
[root@gateway root]#
```

Je recommande en premier la lecture complète du guide afin de se familiariser avec les tenants et aboutissants puis de revenir sur les modifications de votre configuration adapté à votre système.

Conventions

Les points ou les modifications s'imposent sont indiqués par ➡.

Les notes de configuration qui sont propres à LEAF/Bering sont marqués avec .

PPTP/ADSL

➡

Si vous êtes équipé d'un modem ADSL et utilisez PPTP pour communiquer avec un serveur à travers ce modem, vous devez

faire le changement suivant en plus de ceux ci-dessous. ADSL avec PPTP est commun en Europe, ainsi qu'en Australie.

Les Concepts de Shorewall

Les fichiers de configuration pour Shorewall sont situés dans le répertoire /etc/shorewall -- pour de simples paramétrages, vous n'avez à faire qu'avec quelques un d'entre eux comme décris dans ce guide.

Tip

Après avoir installé Shorewall, téléchargez l'exemple two-interface, décompressez le (**`tar -zxvf two-interfaces.tgz`**) et copiez les fichiers dans `/etc/shorewall` **(ces fichiers remplaceront les initiaux)**.

Parallèlement à la présentation, je vous suggère de jeter un oeil à ceux physiquement présents sur votre système -- chacun des fichiers contient des instructions de configuration détaillées et des entrées par défaut.

Shorewall voit le réseau où il fonctionne, comme un ensemble de zones. Dans une configuration avec deux interfaces, les noms des zones suivantes sont utilisés:

Name	Description
`net`	The Internet
`loc`	Your Local Network

Les zones de Shorewall sont définies dans le fichier `/etc/shorewall/zones`.

Shorewall reconnaît aussi le système de firewall comme sa propre zone - par défaut, le firewall est connu comme **`fw`**.

Les règles à propos du trafic à autoriser et à interdire sont exprimées en terme de zones.

- Vous exprimez votre politique par défaut pour les connexions d'une zone vers une autre zone dans le fichier `/etc/shorewall/policy`.
- Vous définissez les exceptions à ces politiques pas défaut dans le fichier `/etc/shorewall/rules`.

Pour chaque connexion demandant à entrer dans le firewall, la requête est en premier lieu comparée par rapport au fichier `/etc/shorewall/rules`. Si aucune règle dans ce fichier ne correspond à la demande de connexion alors la première politique dans le fichier `/etc/shorewall/policy` qui y correspond sera appliquée. Si cette politique est `REJECT` ou `DROP` la requête est dans un premier temps comparée par rapport aux règles contenues dans le fichier `/etc/shorewall/common`, si ce fichier existe; sinon les régles dans le fichier `/etc/shorewall/common.def` sont vérifiées.

Le fichier /etc/shorewall/policy inclus dans l'archive d'exemple (two-interface) contient les politiques suivantes:

```
#SOURCE     DEST        POLICY      LOG LEVEL    LIMIT:BURST
loc         net         ACCEPT
net         all         DROP        info
all         all         REJECT      info
```

Dans le fichier d'exemple (two-interface), la ligne suivante est incluse mais elle est commentée. Si vous voulez que votre firewall puisse avoir un accès complet aux serveurs sur Internet, décommentez la ligne.

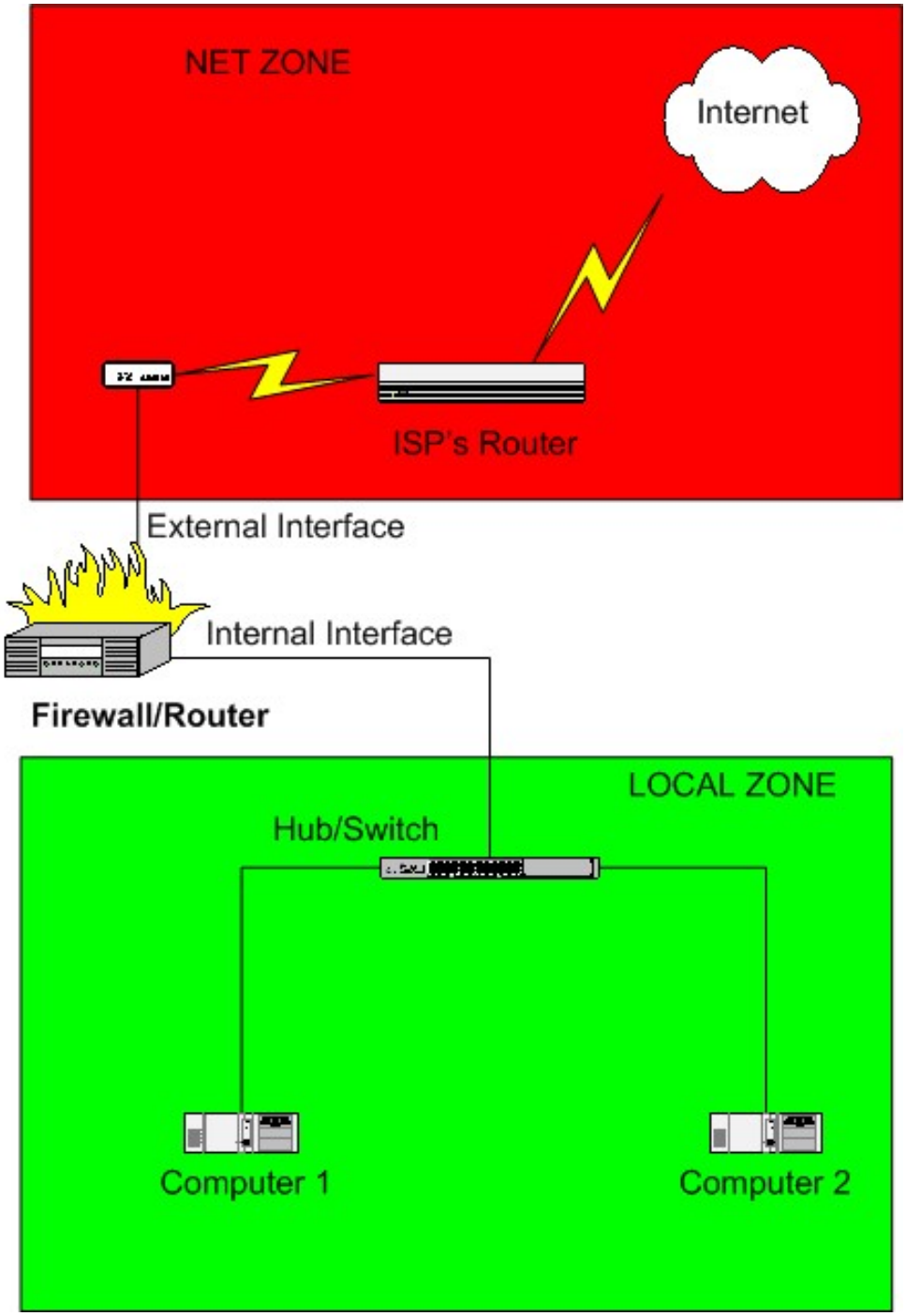
```
#SOURCE     DEST        POLICY      LOG LEVEL    LIMIT:BURST
fw          net         ACCEPT
```

Les politiques précédentes vont:

- Permettre toutes demandes de connexion depuis votre réseau local vers Internet
- Drop (ignorer) toutes les demandes de connexion depuis l'Internet vers votre firewall ou votre réseau local
- Accept (accepter) facultativement toutes les demandes de connexion de votre firewall vers l'Internet (si vous avez décommenté la politique additionnelle)
- Reject (rejeter) toutes les autres requêtes de connexion.

A ce point, éditez votre fichier `/etc/shorewall/policy` et appliquer les changements que vous désirez.

Interfaces Réseau



Le firewall a deux interfaces réseau. Lorsque la connexion Internet passe par un modem câble ou par un “Routeur” ADSL (pas un simple modem), l'*Interface Externe* sera l'adaptateur ethernet qui y est connecté à ce “Modem” (e.g., `eth0`) à moins de se que vous vous connectiez par *Point-to-Point Protocol* over Ethernet (PPPoE) ou *Point-to-Point Tunneling Protocol* (PPTP) dans ce cas l'interface externe sera (e.g., `ppp0`). Si vous vous connectez par un simple modem (RTC), votre interface externe sera aussi `ppp0`. Si vous vous connectez en utilisant l'ISDN, votre interface externe sera `ippp0`.

Si votre interface vers l'extérieur est **ppp0** ou **ippp0** alors vous mettrez `CLAMPMSS=yes` dans le fichier `/etc/shorewall/shorewall.conf`.

Votre *Interface Interne* (interface vers votre réseau local -> LAN) sera un adaptateur Ethernet (`eth1` or `eth0`) et sera connectée à un hub ou switch (câble droit). Vos autres ordinateurs seront connectés à ce même hub/switch (note: Si vous avez un unique ordinateur, vous pouvez connecter le firewall directement en utilisant un câble croisé).

Warning

Ne connectez pas l'interface interne et externe sur le même hub ou switch, sauf pour tester avec une version postérieure à Shorewall 1.4.7. Quand vous utilisez ces versions récentes, vous pouvez tester ce type de configuration si vous spécifiez l'option arp_filter dans le fichier `/etc/shorewall/interfaces` pour toutes les interfaces connectées au hub/switch commun. Utiliser une telle configuration avec un firewall en production est fortement déconseillé.

Le fichier de configuration d'exemple pour deux interfaces suppose que votre interface externe est `eth0` et que l'interface interne est `eth1`. Si votre configuration est différente, vous devrez modifier le fichier `/etc/shorewall/interfaces` en conséquence. Tant que vous y êtes, vous pourriez parcourir la liste des options qui sont spécifiées pour les interfaces. Quelques trucs:

Tip

Si votre interface vers l'extérieur est `ppp0` ou `ippp0`, vous pouvez remplacer le detect dans la seconde colonne par un “-” (sans les quotes).

Tip

Si votre interface vers l'extérieur est `ppp0` or `ippp0` u si vous avez une adresse IP statique, vous pouvez enlever `dhcp` dans la liste des options .

Tip

Si votre interface est un bridge utilisant l'utilitaire **brctl** alors vous devez ajouter l'option `routeback` à la liste des options.

Tip

Si vous spécifiez *norfc1918* pour votre interface externe, vous pouvez vérifier périodiquement le Shorewall Errata pour mettre à jour le fichier `/usr/share/shorewall/rfc1918`. Sinon, vous pouvez copier le fichier `/usr/share/shorewall/rfc1918` vers `/etc/shorewall/rfc1918` et adapter votre fichier `/etc/shorewall/rfc1918` comme je le fais.

Adresses IP

Avant d'aller plus loin, nous devons dire quelques mots au sujet des adresses Internet Protocol (IP). Normalement, votre fournisseur Internet FAI vous assignera une seule adresse IP. Cette adresse peut être assignée par le Dynamic Host Configuration Protocol (DHCP) ou lors de l'établissement de votre connexion lorsque vous vous connectez (modem standard) ou établissez votre connexion PPP. Dans de rares cas , votre provider peut vous assigner une adresse statique IP ; cela signifie que vous devez configurer l'interface externe de votre firewall afin d'utiliser cette adresse de manière permanente. Votre adresse externe assignée, elle va être partagée par tous vos systèmes lors de l'accès à Internet. Vous devrez assigner vos propres adresses dans votre réseau local (votre interface interne sur le firewall ainsi que les autres ordinateurs). La **RFC 1918** réserve plusieurs plages d'adresses privées *Private* IP à cet fin:

```
10.0.0.0    - 10.255.255.255
172.16.0.0  - 172.31.255.255
192.168.0.0 - 192.168.255.255
```

Avant de lancer Shorewall, regarder l'adresse IP de votre interface externe, et si elle est dans les plages précédentes, vous devez enlever l'option 'norfc1918' dans la ligne concernant l'interface externe dans le fichier `/etc/shorewall/interfaces`.

Vous devrez assigner vos adresses depuis le même sous-réseau (sub-network-subnet). Pour ce faire, nous pouvons considérer un sous-réseau dans une plage d'adresses `x.y.z.0 - x.y.z.255`. Chaque sous-réseau aura un masque (Subnet Mask) `255.255.255.0`. L'adresse `x.y.z.0` est réservée comme l'adresse de sous-réseau *Subnet Address* et `x.y.z.255` est réservée en tant qu'adresse de broadcast *Subnet Broadcast Address*. Dans Shorewall, un sous-réseau est décrit en utilisant Classless InterDomain Routing (CIDR) notation Il consiste en l'adresse du sous-réseau suivie par `/24`. Le “24” se réfère au nombre consécutif de bits marquant “1” dans la partie gauche du masque de sous-réseau.

Table 1. Un exemple de sous-réseau (sub-network) :

Range:	`10.10.10.0 - 10.10.10.255`
Subnet Address:	`10.10.10.0`
Broadcast Address:	`10.10.10.255`
CIDR Notation:	`10.10.10.0/24`

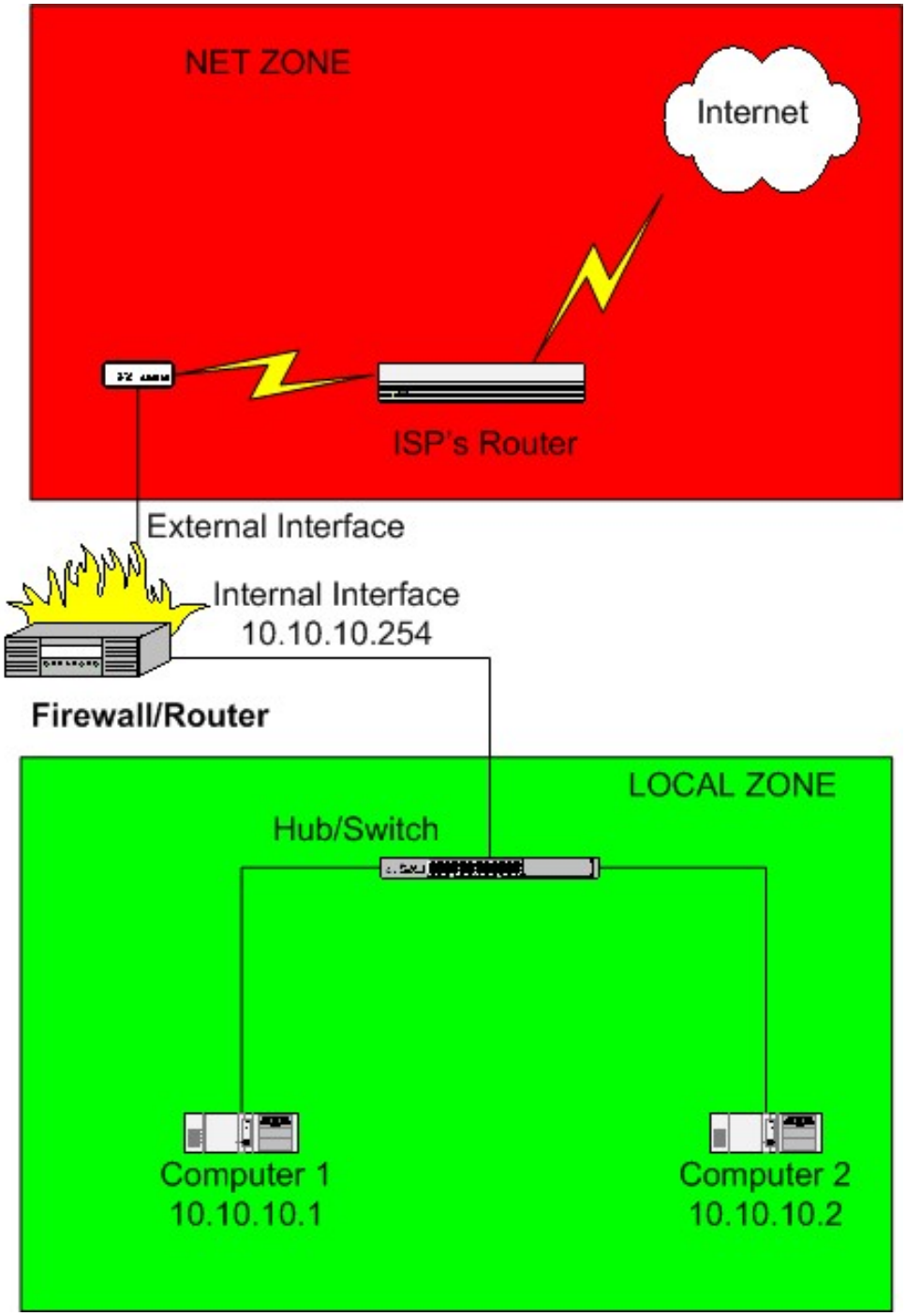
Il est de mise d'assigner l'interface interne à la première adresse utilisable du sous-réseau (`10.10.10.1` dans l'exemple précédent) ou la dernière adresse utilisable (`10.10.10.254`).

L'un des buts d'un sous-réseau est de permettre à tous les ordinateurs dans le sous-réseau de savoir avec quels autres ordinateurs ils peuvent communiquer directement. Pour communiquer avec des systèmes en dehors du sous-réseau, les ordinateurs envoient des paquets à travers le gateway (routeur).

Vos ordinateurs en local (ordinateur 1 et ordinateur 2 dans le diagramme) doivent être configurés avec leur passerelle par défaut (default gateway) pointant sur l'adresse IP de l'interface interne du firewall.

La présentation précédente ne fait que d'effleurer la question des sous réseaux et du routage. Si vous êtes intéressé pour apprendre plus sur l'adressage IP et le routage, je recommande “IP Fundamentals: What Everyone Needs to Know about Addressing & Routing”, Thomas A. Maufer, Prentice-Hall, 1999, ISBN 0-13-975483-0 (link).

Le reste de ce guide assumera que vous avez configuré votre réseau comme montré ci-dessous :



La passerelle par défaut pour les ordinateurs 1 et 2 devrait être `10.10.10.254`.

Warning

Votre FAI (fournisseur d'accès) pourrait assigner une adresse **RFC 1918** à votre interface externe. Si cette adresse est le sous-réseau `10.10.10.0/24` alors vous aurez besoin d'un sous-réseau DIFFERENT RFC 1918 pour votre réseau local.

IP Masquerading (SNAT)

Les adresses réservées par la RFC 1918 sont parfois désignées comme non-routables car les routeurs Internet (backbone) ne font pas circuler les paquets qui ont une adresse de destination appartenant à la RFC-1918. Lorsqu'un de vos systèmes en local (supposons l'ordinateur1) demande une connexion à un serveur par Internet, le firewall doit appliquer un *Network Address Translation* (NAT). Le firewall réécrit l'adresse source dans le paquet, et l'a remplacé par l'adresse de l'interface externe du firewall; en d'autres mots, le firewall fait croire que c'est lui même qui initie la connexion. Ceci est nécessaire afin que l'hôte de destination soit capable de renvoyer les paquets au firewall (souvenez vous que les paquets qui ont pour adresse

de destination, une adresse réservée par la RFC 1918 ne pourront pas être routés à travers Internet, donc l'hôte Internet ne pourra adresser sa réponse à l'ordinateur 1). Lorsque le firewall reçoit le paquet de réponse, il remet l'adresse de destination à `10.10.10.1` et fait passer le paquet vers l'ordinateur 1.

Sur les systèmes Linux, ce procédé est souvent appelé *IP Masquerading* mais vous verrez aussi le terme de *Source Network Address Translation* (SNAT). Shorewall suit la convention utilisée avec Netfilter:

- *Masquerade* désigne le cas ou vous laissez votre firewall détecter automatiquement l'adresse de l'interface externe.
- *SNAT* désigne le cas où vous spécifiez explicitement l'adresse source des paquets sortant de votre réseau local.

Sous Shorewall, autant le *Masquerading* et le *SNAT* sont configurés avec des entrées dans le fichier `/etc/shorewall/masq`. Vous utiliserez normalement le Masquerading si votre adresse IP externe est dynamique, et SNAT si l'adresse IP est statique.

Si votre interface externe du firewall est `eth0`, vous n'avez pas besoin de modifier le fichier fourni avec l'exemple. Dans le cas contraire, éditez `/etc/shorewall/masq` et changer la première colonne par le nom de votre interface externe, et la seconde colonne par le nom de votre interface interne.

Si votre adresse externe IP est statique, vous pouvez la mettre dans la troisième colonne dans `/etc/shorewall/masq` si vous le désirez, de toutes façons votre firewall fonctionnera bien si vous laissez cette colonne vide. Le fait de mettre votre adresse IP statique dans la troisième colonne permet un traitement des paquets sortant un peu plus efficace.

Si vous utilisez les paquets Debian, vérifiez que votre fichier de configuration `shorewall.conf` contient bien les valeurs suivantes, si elles n'y sont pas faite les changements nécessaires:

- `NAT_ENABLED=Yes` (Shorewall versions earlier than 1.4.6)
- `IP_FORWARDING=On`

Port Forwarding (DNAT)

Un de nos buts est de , peut être, faire tourner un ou plusieurs serveurs sur nos ordinateurs locaux. Parce que ces ordinateurs on une adresse RFC-1918, il n' est pas possible pour les clients sur Internet de se connecter directement à eux. Il est nécessaire à ces clients d'adresser leurs demandes de connexion au firewall qui réécrit l'adresse de destination de votre serveur, et fait passer le paquet à celui-ci. Lorsque votre serveur répond, le firewall applique automatiquement un SNAT pour réécrire l'adresse source dans la réponse.

Ce procédé est appelé *Port Forwarding* or *Destination Network Address Translation* (DNAT). Vous configurez le port forwarding en utilisant les règles DNAT dans le fichier `/etc/shorewall/rules`.

La forme générale d'une simple règle de port forwarding dans `/etc/shorewall/rules` est:

```
#ACTION   SOURCE    DEST                                          PROTO      DEST
PORT(S)
DNAT      net       loc:<server local ip address>[:<server port>] <protocol> <port>
```

Example 1. Web Server

Vous faites tourner un serveur Web sur l'ordinateur 2 et vous voulez faire passer les requêtes TCP sur le port 80 à ce système

:

```
#ACTION    SOURCE     DEST              PROTO      DEST PORT(S)
DNAT       net        loc:10.10.10.2    tcp        80
```

Example 2. FTP Server

Vous faites tourner un serveur FTPsur l'ordinateur 1 et vous voulez rediriger les requêtes TCP entrantes sur le port 21 à ce système:

```
#ACTION    SOURCE     DEST              PROTO      DEST PORT(S)
DNAT       net        loc:10.10.10.1    tcp        21
```

Concernant FTP, vous aurez aussi besoin d'avoir le support FTP et le NAT dans votre kernel. Pour les fournisseurs de kernels, cela veut dire que les modules `ip_conntrack_ftp` et `ip_nat_ftp` doivent être disponibles. Shorewall chargera automatiquement ces modules si ils sont disponibles à leur place habituelle `/lib/modules/<kernel version>/kernel/net/ipv4/netfilter`.

Deux points importants à garder en mémoire :

- Vous devez tester la règle précédente depuis un client à l'extérieur de votre réseau local (c.a.d., ne pas tester depuis un navigateur tournant sur l'ordinateur 1 ou 2 ou sur le firewall). Si vous voulez avoir la possibilité d'accéder à votre serveur web et/ou FTP de l'intérieur de votre firewall en utilisant l'adresse de l'interface externe IP, regardez Shorewall FAQ #2.
- Quelques fournisseurs Internet (Provider/ISP) bloquent les requêtes de connexion entrantes sur le port 80. Si vous avez des problèmes pour vous connecter à votre serveur web, essayez la règle suivante et connectez vous sur le port 5000 (c.a.d., connectez vous à `http://w.x.y.z:5000 ou w.x.y.z` est votre IP externe).

  ```
  #ACTION    SOURCE     DEST                PROTO      DEST PORT(S)
  DNAT       net        loc:10.10.10.2:80   tcp        5000
  ```

A ce point, modifiez `/etc/shorewall/rules` pour ajouter les règles DNAT dont vous avez besoin.

Domain Name Server (DNS)

Normalement, quand vous vous connectez à votre fournisseur (FAI/ISP), une partie consiste à obtenir votre adresse IP, votre *Domain Name Service* (DNS) pour le firewall est configuré automatiquement (c.a.d.,le fichier `/etc/resolv.conf` sera mis à jour). Il arrive que votre provider vous donne une paire d'adresse IP pour les serveurs DNS afin que vous configuriez manuellement votre serveur de nom primaire et secondaire. La manière dont le DNS est configuré sur votre firewall est de votre responsabilité. Vous pouvez procéder d'une de ses deux façons :

- Vous pouvez configurer votre système interne pour utiliser les noms de serveurs de votre provider. Si votre fournisseur vous donne les adresses de leurs serveurs ou si ces adresses sont disponibles sur leur site web, vous pouvez configurer votre système interne afin de les utiliser. Si cette information n' est pas disponible, regardez dans `/etc/resolv.conf` sur votre firewall -- les noms des serveurs sont donnés dans l'enregistrement "nameserver" dans ce fichier.
- Vous pouvez configurer un cache dns *Caching Name Server* sur votre firewall. Red Hat™ a un RPM pour serveur dns de cache (le RPM à besoin aussi du paquetage **bind** RPM) et pour les utilisateurs de Bering, il y a **dnscache.lrp**. Si vous adoptez cette approche, vous configurez votre système interne pour utiliser le firewall lui même comme étant le seul serveur de nom primaire. Vous pouvez utiliser l'adresse IP interne du firewall (`10.10.10.254` dans l'exemple précédent) pour l'adresse de serveur de nom. Pour permettre à vos systèmes locaux de discuter avec votre

serveur cache de nom, vous devez ouvrir le port 53 (à la fois UDP and TCP) sur le firewall vers le réseau local; vous ferez ceci en ajoutant les règles suivantes dans `/etc/shorewall/rules`.

```
#ACTION     SOURCE     DEST                PROTO      DEST PORT(S)
AllowDNS    loc        fw
```

Autres Connexions

Les fichiers exemples inclus dans l'archive (two-interface) contiennent les règles suivantes :

```
#ACTION     SOURCE     DEST                PROTO      DEST PORT(S)
AllowDNS    fw         net
```

Ces règles autorisent l'accès DNS à partir de votre firewall et peuvent être enlevées si vous avez décommenté la ligne dans `/etc/shorewall/policy` autorisant toutes les connexions depuis le firewall vers Internet.

Dans la régle ci-dessus, “AllowDNS” est un exemple d'action prédéfinie *defined action*. Shorewall inclus un nombre d'actions prédéfinies et vous pouvez ajouter les vôtres. Pour voir les actions comprises avec votre version de Shorewall, regardez dans le fichier `/etc/shorewall/actions.std`. Le nom de celles qui acceptent des connexions débutent par “Allow”.

Vous n'êtes pas obligés d'utiliser des actions prédéfinies quand vous ajoutez des régles dans le fichier `/etc/shorewall/rules`; les régles générées par Netfilter sont plus performantes sans actions prédéfinies. La régle vue ci-dessus peut aussi être codé comme cela:

```
#ACTION     SOURCE     DEST                PROTO      DEST PORT(S)
ACCEPT      fw         net                 udp        53
ACCEPT      fw         net                 tcp        53
```

Au cas ou Shorewall n'inclue pas d'actions définies qui vous conviennent, vous pouvez les définir vous même ou coder directement les régles.

L'exemple inclus aussi:

```
#ACTION     SOURCE     DEST                PROTO      DEST PORT(S)
AllowSSH    loc        fw
```

Cette régle autorise un serveur SSH sur votre firewall et la connexion à celui-ci depuis votre réseau local.

Si vous souhaitez autoriser d'autre connexions de votre firewall vers d'autres systèmes, la sysntaxe générale utilisant l'action type “Allow” est:

```
#ACTION     SOURCE     DEST                PROTO       DEST PORT(S)
<action>    fw         <destination zone>
```

La syntaxe générale lorsqu'on utilise pas des actions prédéfinies est:

```
#ACTION     SOURCE     DEST                PROTO       DEST PORT(S)
ACCEPT      fw         <destination zone> <protocol> <port>
```

Example 3. Serveur Web sur le Firewall

Vous voulez ouvrir un serveur Web Server sur votre firewall au réseau local et externe:

```
#ACTION     SOURCE     DEST                PROTO      DEST PORT(S)
AllowWeb    net        fw
AllowWeb    loc        fw
```

Ces deux régles viennent évidemment s'ajouter à celles listées sous "Vous pouvez configurer un cache dns sur votre firewall".

Si vous ne savez pas quel port(s) et protocole(s) requièrent une application particulière, vous pouvez regarder ici.

Important

Je ne recommande pas d'autoriser **telnet** vers/de l'Internet parce qu'il utilise du texte en clair (même pour le login!). Si vous voulez un accès shell à votre firewall, utilisez SSH:

```
#ACTION     SOURCE     DEST                PROTO      DEST PORT(S)
AllowSSH    net        fw
```

Les utilisateurs de Bering pourront ajouter les deux régles suivantes pour être compatible avec la configuration du firewall Jacques's Shorewall.

```
#ACTION     SOURCE     DEST     PROTO      DEST PORT(S)
ACCEPT      loc        fw       udp        53          #Allow DNS Cache to work
ACCEPT      loc        fw       tcp        80          #Allow Weblet to work
```

Maintenant, éditez votre fichier de configuration `/etc/shorewall/rules` pour ajouter, modifier ou supprimer les autres connexions voulues.

Quelques Points à Garder en Mémoire

- **Vous ne pouvez tester votre firewall de l'intérieur de votre réseau**. Car les requêtes que vous envoyez à votre adresse IP ne veux pas dire qu'elle seront associées à votre interface externe ou la zone "net". Tout trafic généré par le réseau local sera traité par loc->fw.
- **Les adresses IP sont des propriétés des systèmes, pas des interfaces**. C'est une erreur de croire que votre firewall est capable de renvoyer des paquets simplement parce que vous pouvez faire un ping sur l'adresse IP de toutes les interfaces du firewall depuis le réseau local. La seul conclusion est de conclure que le lien entre le réseau local et le firewall est établi et que vous avez probablement la bonne adresse de la passerelle sur votre système.
- **Toutes les adresses IP configurées sur le firewall sont dans la zone $FW (fw)**. Si 192.168.1.254 est l'adresse IP de votre interface interne, alors vous pouvez écrire "**$FW:192.168.1.254**" dans une régle mais vous ne devez pas écrire "**loc:192.168.1.254**". C'est aussi un non-sens d'ajouter 192.168.1.254 à la zone **loc** en utilisant une entrée dans `/etc/shorewall/hosts`.
- **Les paquets de retour (Reply) ne suivent PAS automatiquement le chemin inverse de la requête d'origine**. Tous les paquets sont routés en se référant à la table de routage respective de chaque hôte à chaque étape du trajet. C'est commun chez ceux qui installent le firewall Shorewall en parallèle à une passerelle existante et essayent d'utiliser DNAT dans Shorewall sans changer la passerelle par défaut sur les systèmes recevant le retour des requêtes. Les requêtes dont, à travers le firewall Shorewall, l'adresse de destination IP est réécrite mais la réponse va directement vers l'ancienne passerelle.

- **Shorewall lui-même n'a aucune notion du dedans et du dehors**. Ces concepts dépendent de la façon dont Shorewall est configuré.

Démarrer et Arrêter Votre Firewall

La procédure d'installation configure votre système pour lancer Shorewall au boot du système, mais au début avec la version 1.3.9 de Shorewall le lancement est désactivé, n'essayer pas de lancer Shorewall avant que la configuration soit finie. Une fois que vous en aurez fini avec la configuration du firewall, vous pouvez permettre le lancement de Shorewall en supprimant le fichier `/etc/shorewall/startup_disabled`.

Important

Les utilisateurs des paquets .deb doivent éditer `/etc/default/shorewall` et mettre `startup=1`.

Le firewall est activé en utilisant la commande "**shorewall start**" et arrêté avec "**shorewall stop**". Lorsque le firewall est stoppé, le routage est autorisé sur les hôtes qui possèdent une entrée dans `/etc/shorewall/routestopped`. Un firewall qui tourne peut être relancé en utilisant la commande "**shorewall restart**" command. Si vous voulez enlever toutes traces de Shorewall sur votre configuration de Netfilter, utilisez "**shorewall clear**".

Les exemples (two-interface) supposent que vous voulez permettre le routage depuis ou vers `eth1` (le réseau local) lorsque Shorewall est stoppé. Si votre réseau local n' est pas connecté à `eth1` ou si vous voulez permettre l'accès depuis ou vers d'autres hôtes, changez `/etc/shorewall/routestopped` en conséquence.

Warning

Si vous êtes connecté à votre firewall depuis Internet, n'essayez pas une commande "**shorewall stop**" tant que vous n'avez pas ajouté une entrée pour votre adresse IP (celle à partir de laquelle vous êtes connectée) dans `/etc/shorewall/routestopped`. De la même manière, je ne vous recommande pas d'utiliser "**shorewall restart**"; il est plus intéressant de créer une configuration alternative et de la tester en utilisant la commande "**shorewall try**".

Autres Lectures Recommandées

Je vous recommande vivement de lire la page des Fonctionnalités Générales des Fichiers de Configuration -- elle contient des trucs sur les possibilités de Shorewall pour rendre aisé l'administration de votre firewall Shorewall.

Ajouter un Segment Sans-fil à votre Firewall à deux interfaces

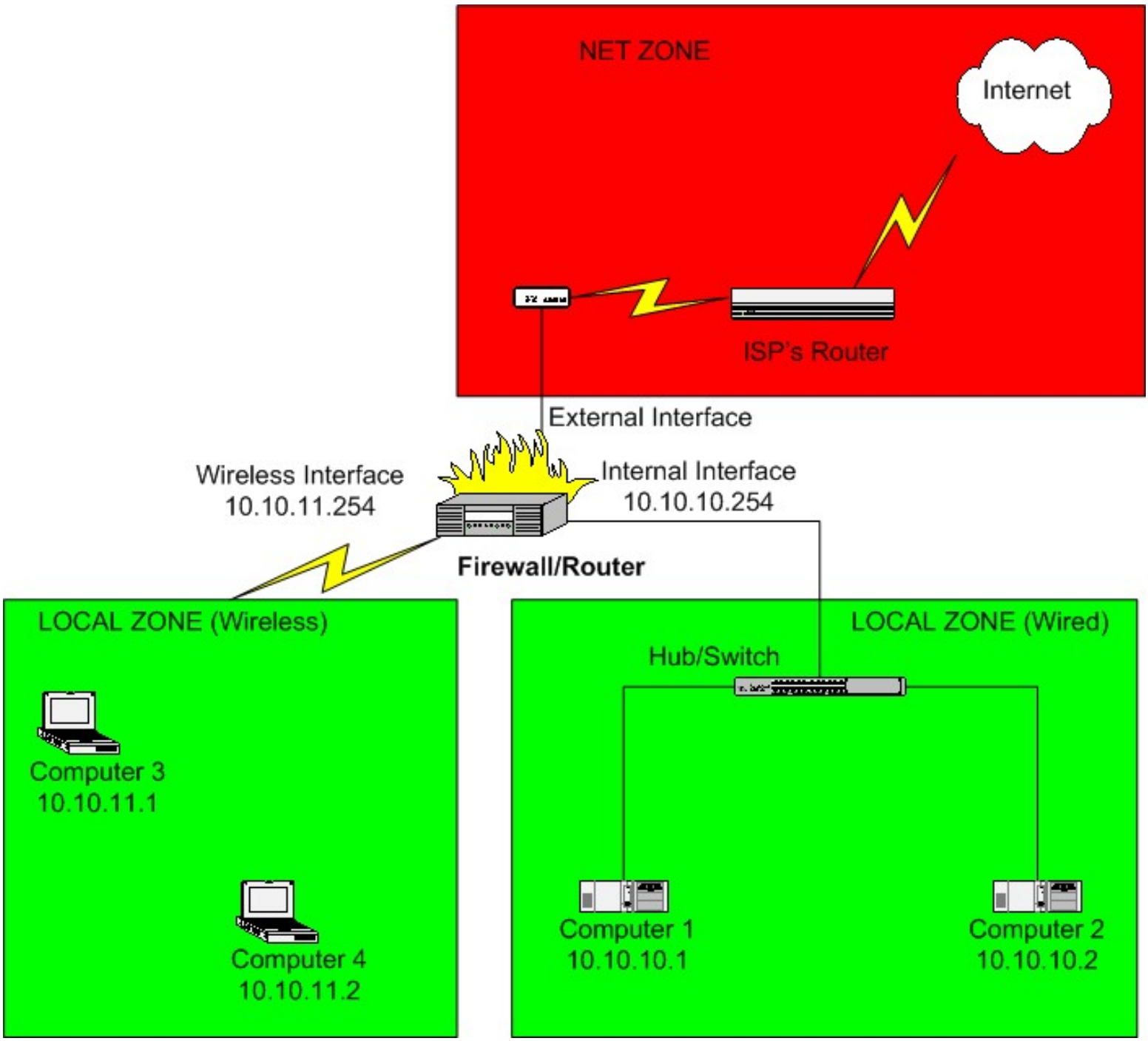
Maintenant que vous avez une configuration deux interfaces qui marche, l'étape suivante logique est d'ajouter un Réseau Sans-fil. La première étape est d'ajouter une carte à votre firewall, soit une carte Sans-fil ou une carte ethernet relié à un Point d'accès Sans-fil.

Caution

Quant vous ajoutez une carte réseau, il se peut qu'elle ne soit pas détecté comme celle suivant la plus haute interface. Par exemple, si vous avez deux cartes interfaces sur votre système (`eth0` and `eth1`) et que vous

ajoutez une troisième qui utilise le même drivers qu'une des deux autres, cette troisième carte ne sera pas obligatoirement détecté en tant que `eth2`; elle peut très bien être détecté en tant que `eth0` ou `eth1`! Vous pouvez faire avec ou intervertir les cartes dans les slots jusqu'à obtenir valeur `eth2`.

Votre nouveau réseau ressemblera à la figure ci-dessous.



La première chose à noter est que les ordinateurs sur votre réseau sans-fil seront sur un sous-réseau différent de celui de votre réseau local LAN. Dans l'exemple précédent, nous avons choisi de lui attribuer le réseau 10.10.11.0/24. Les ordinateurs 3 et 4 seront configurés avec une passerelle par défaut dont l'adresse IP sera 10.10.11.254.

Ensuite, nous avons choisi d'inclure le réseau sans-fil à la zone local. Depuis que Shorewall autorise du trafic intra-zone par défaut, le trafic pourra circuler librement entre le réseau local et sans-fil.

Il n'y a que deux changements à effectuer à la configuration de Shorewall:

- Une entrée doit être ajouté au fichier d'interfaces `/etc/shorewall/interfaces` pour l'interface du réseau sans-fil. Si l'interface du réseau sans-fil est `wlan0`, l'entrée correspondante pourrait être:

```
#ZONE      INTERFACE        BROADCAST           OPTIONS
loc        wlan0            detect              maclist
```

  Comme montré dans l'entrée ci-dessus, je recommande d'utiliser l'option maclist pour le segment sans-fil. En ajoutant les entrées pour les ordinateurs 3 et 4 dans le fichier `/etc/shorewall/maclist`, vous pouvez vous assurer que vos voisins n'utiliseront pas votre connexion internet. Commencez sans cette option; quant tout fonctionnera, alors ajouter l'option et configurez votre fichier `/etc/shorewall/maclist`.
- Vous avez besoin d'ajouter une entrée au fichier `/etc/shorewall/masq` afin de masquer le trafic de votre réseau sans-fil vers Internet. Si votre interface Internet est `eth0` et votre interface sans-fil est `wlan0`, l'entrée sera:

```
#INTERFACE            SUBNET              ADDRESS
eth0                  wlan0
```

Autre chose. Pour que le réseau Microsoft™ fonctionne entre réseau filaire et sans-fil, vous avez besoin soit d'un serveur WINS ou un PDC. J'utilise personnellement Samba configuré en serveur WINS qui tourne sur mon firewall. Utiliser un serveur WINS sur le firewall nécessite de configurer les régles nécessaires listées dans le document Shorewall/Samba.

Shorewall Setup Guide

Tom Eastep

Fabien Demassieux

Copyright © 2001-2005 Thomas M. Eastep, Fabien Demassieux

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-01-22

Table of Contents

Introduction
- Pré-requis
- Avant de commencer

Les Concepts de Shorewall

Interfaces Réseau

Adressage, Sous-réseaux et Routage
- Adressage IP
- Sous-réseaux
- Routage
- Protocole de Résolution d'Adresse (ARP)
- RFC 1918

Configurer votre Réseau
- Routage
- Non-routé
  - SNAT
  - DNAT
  - Proxy ARP
  - One-to-one NAT
- Règles
- D'autres petites choses

DNS

Quelques Points à Garder en Mémoire

Démarrer et Arrêter Votre Firewall

Note

Notes du traducteur : J'espère vous faciliter l'accès et la prise en main d'un firewall performant, efficace, adaptable et facile d'utilisation. Donc félicitations pour la qualité du travail et la disponibilité offerte par Thomas M. Eastep. Si vous trouvez des erreurs ou des améliorations à apporter vous pouvez me contacter Fabien Demassieux

Introduction

Ce guide est destiné aux utilisateurs qui configurent Shorewall dans un environnement ou un ensemble d'adresses IP publiques doivent être prises en compte ou à ceux qui souhaitent en savoir plus à propos de Shorewall que ce que contient le guide pour une utilisation avec une adresse ID unique. Parce que le champ d'utilisation est si important, le guide vous donnera les indications

générales à suivre et vous renseignera sur d'autres ressources si nécessaire.

Caution

Si vous utilisez LEAF Bering, votre configuration Shorewall n'est PAS ce que je publie -- Je suggère de prendre en considération l'installation de Shorewall LPR disponible sur le site de shorewall.net avant de poursuivre.

Pré-requis

Shorewall a besoin que le package **iproute**/**iproute2** soit installé (avec la distribution RedHat™, le package s'appelle **iproute**). Vous pouvez vérifier si le package est installé par la présence du programme **ip** sur votre firewall. En tant que `root`, vous pouvez utiliser la commande **which** pour cela:

```
[root@gateway root]# which ip
/sbin/ip
[root@gateway root]#
```

Avant de commencer

Je recommande en premier la lecture complète du guide afin de se familiariser avec les tenants et aboutissants puis de revenir sur les modifications de votre configuration adapté à votre système.

Caution

Si vous éditez vos fichiers de configuration sur un système Windows™, vous devez les sauver comme des fichiers Unix™ si votre éditeur supporte cette option sinon vous devez les convertir avec **dos2unix** avant d'essayer de les utiliser. De la même manière, si vous copiez un fichier de configuration depuis votre disque dur Windows™ vers une disquette, vous devez lancer **dos2unix** sur la copie avant de l'utiliser avec Shorewall.

- Windows™ Version of **dos2unix**
- Linux Version of **dos2unix**

Les Concepts de Shorewall

Les fichiers de configuration de Shorewall se trouvent dans le répertoire `/etc/shorewall` -- pour la plus par des paramétrages, vous avez juste besoin de quelques-uns d'entre eux comme cela est décrit dans le manuel. Des squelettes de fichiers sont créés durant la procédure d'installation de Shorewall.

Comme chaque fichier est abordé, je vous suggère de regarder celui de votre système -- chaque fichier contient des instructions détaillées de configuration et d'autres des entrées par défaut.

Shorewall voit le réseau où il fonctionne, comme un ensemble de zones. Dans la configuration par défaut, les noms des zones suivantes sont utilisés:

Table 1. Zones

Name	Description
net	L'internet
loc	Votre Réseau local
dmz	Zone Démilitarisée

Les Zones sont définies dans le fichier `/etc/shorewall/zones`.

Shorewall reconnaît aussi le système firewall comme sa propre zone - par défaut, le firewall lui-même est connu sous le nom **fw**, mais cela peut être modifié dans le fichier `/etc/shorewall/shorewall.conf`. Dans ce guide, le nom par défaut (**fw**) sera utilisé.

Mise à par **fw**, Shorewall n'attache aucune importance au nom des zones. Les Zones sont entièrement ce que VOUS en faites. Cela veut dire que vous ne devez pas vous attendre à ce que Shorewall fasse quelque chose de spécial “car il s'agit de la zone Internet” ou “car c'est la zone DMZ”.

Éditez le fichier /etc/shorewall/zones file et faites tous changements qui s'imposent.

Les Règles qui concernent le trafic à autoriser ou à refuser sous exprimés en terme de Zones.

- Vous désignez les politiques par défaut entre une zone et une autre dans le fichier `/etc/shorewall/policy`.
- Vous définissez les exceptions à ces politiques par défaut dans le fichier `/etc/shorewall/rules`.

Shorewall est construit sur les possibilités du noyau (kernel) Netfilter. Netfilter implémente une fonction de tracking qui autorise ce qui est souvent désigné comme une inspection déclarée de paquets. Les propriétés de déclaration permettent au firewall d'être définie en terme de connexions plutôt qu'en terme de paquet. Avec Shorewall, vous:

1. Identifiez la zone source.
2. Identifiez la zone destination.
3. Si la politique de la zone client vers la zone destination est ce que vous souhaitez pour cette paire client/serveur, vous n'avez besoin de rien de plus.
4. Si la politique n'est pas ce que vous souhaitez, alors vous devez ajouter une règle. Cette règle est exprimé en terme de zone client et de zone serveur.

Si les connexions d'un certain type sont autorisés de la zone A au firewall et sont aussi autorisés du firewall à la zone B cela **NE VEUT PAS dire que ces connections sont autorisés de la zone A à la zone B**. Cela veut plutôt dire que vous avez un proxy qui tourne sur le firewall qui accepte les connections de la zone A et qui ensuite établit ces propres connections du firewall à la zone B.

Pour chaque requête de connexion sur le firewall, la requête est d'abord évalué à travers le fichier `/etc/shorewall/rules`. Si aucune règle dans ce fichier ne correspond, la connexion interroge ensuite la première politique dans `/etc/shorewall/policy` qui correspond à la requête et l'applique. Si cette politique est REJECT ou DROP, la requête est a nouveau évaluée à travers les règles du fichier `/etc/shorewall/common.def`.

Le fichier de défaut `/etc/shorewall/policy` a les politiques suivantes:

```
#SOURCE ZONE     DESTINATION ZONE    POLICY     LOG     LIMIT:BURST
#                                               LEVEL
fw               net                 ACCEPT
net              all                 DROP       info
all              all                 REJECT     info
```

La politique précédente:

1. Permet toutes les connexions de votre réseau local vers Internet
2. Drop (ignore) toutes les connexions d'Internet vers le firewall ou votre réseau local et génère un message au niveau info (ici se trouve la description des niveaux de log).
3. Reject (rejette) toutes les autres connexions et génère un message au niveau info. Quant la requête est rejeté, le firewall retourne un RST (si le protocole est TCP) ou un ICMP port-unreachable paquet pour les autres protocoles.

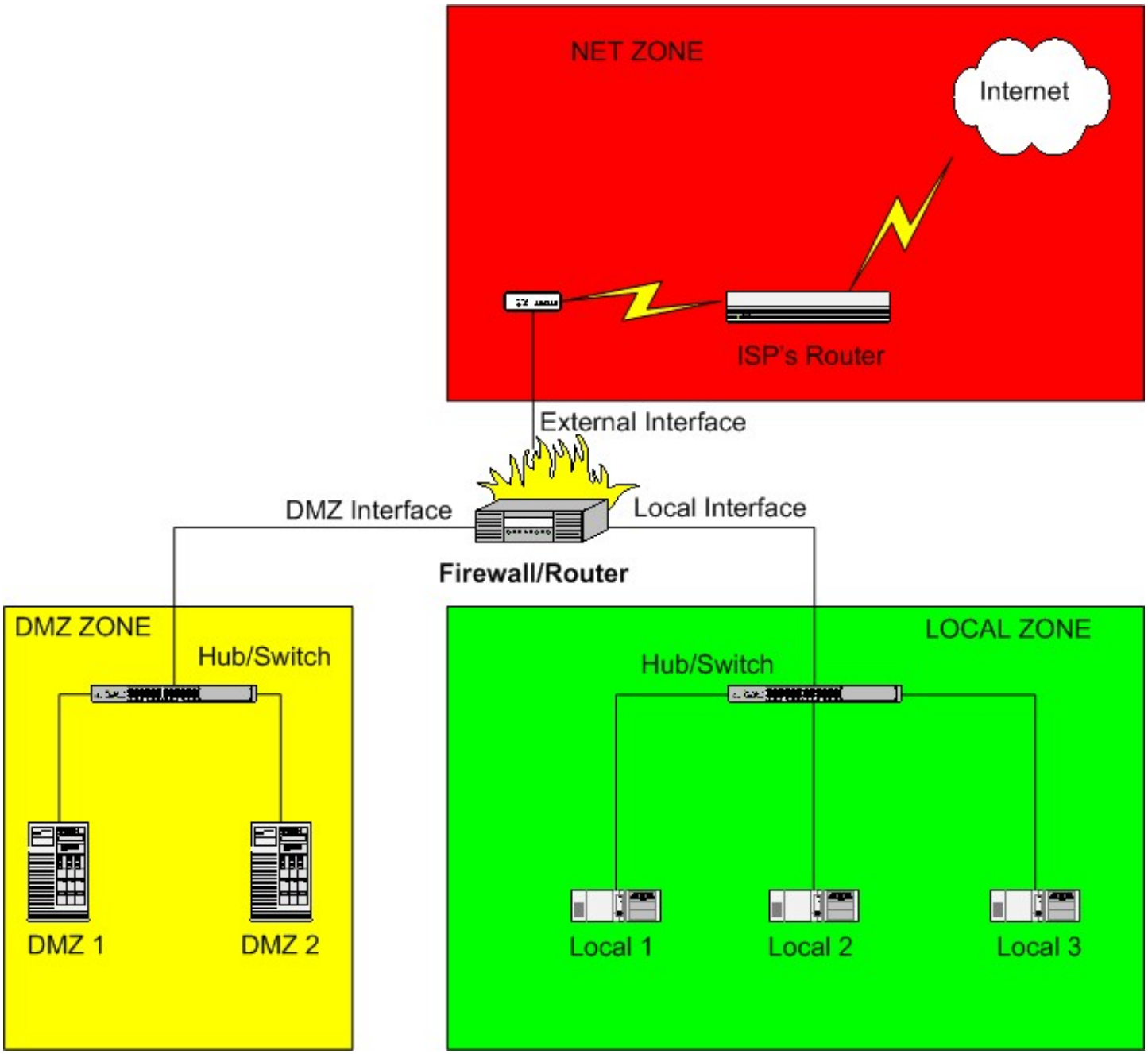
Maintenant, éditez votre `/etc/shorewall/policy` et apportez tous les changements que vous souhaitez.

Interfaces Réseau

Pour le reste de ce guide, nous utiliserons le schéma ci-dessous. Bien qu'il ne puisse correspondre à votre propre réseau, il peut être utilisé pour illustrer les aspects importants de la configuration de Shorewall.

Sur ce schéma:

- La zone DMZ est composée des systèmes DMZ 1 et DMZ 2. Une DMZ est utilisée pour isoler vos serveurs accessibles depuis Internet de vos systèmes locaux. Ainsi si un de ces serveurs est compromis, vous avez encore votre firewall entre le système compromis et vos systèmes locaux.
- La zone Local est composée des systèmes Local 1, Local 2 et Local 3.
- Tous les systèmes du FAI vers l'extérieur et qui englobe la Zone Internet.



La façon la plus simple pour définir les zones est d'associer le nom de la zone (définie précédemment dans /etc/shorewall/zones) avec une interface réseau. C'est fait dans le fichier /etc/shorewall/interfaces. Le firewall illustré ci-dessus à trois interfaces. Si la connexion se fait à travers un câble ou un DSL “Modem”, l'*Interface Externe* sera l'adaptateur qui est branché au “Modem” (e.g., `eth0`) tant que vous ne vous n'utilisez pas le Point-to-Point Protocol over Ethernet (PPPoE) ou le Point-to-Point Tunneling Protocol(PPTP) dans ce cas l'Interface Externe sera de type ppp (e.g., `ppp0`). Si vous utilisez un modem classique, votre Interface externe sera également `ppp0`. Si vous utilisez ISDN, votre Interface Externe sera `ipppp0`.

Si votre Interface Externe est `ppp0` ou `ippp0` alors vous pouvez fixer CLAMPMSS=yes dans `/etc/shorewall/shorewall.conf`.

Votre *Interface Locale* sera un adaptateur Ethernet (`eth0`, `eth1` or `eth2`) et doit être connecté à un hub ou un switch. Vos ordinateurs locaux doivent être connectés au même switch (note: Si vous avez une machine unique, vous pouvez connecter le firewall directement à l'ordinateur en utilisant un câble croisé).

Votre *Interface DMZ* sera aussi être un adaptateur Ethernet (`eth0`, `eth1` ou `eth2`) et doit être connecté à un hub ou un switch. Vos ordinateurs DMZ doivent être connectés au même switch (note: Si vous avez une machine DMZ unique, vous pouvez connecter le firewall directement à l'ordinateur en utilisant un câble croisé).

Caution

Ne connectez pas l'interface interne et externe sur le même hub ou switch, sauf pour tester avec une version postérieure à Shorewall 1.4.7. Quand vous utilisez ces versions récentes, vous pouvez tester ce type de configuration si vous spécifiez l'option *arp_filter* dans le fichier `/etc/shorewall/interfaces` pour toutes les interfaces connectées au hub/switch commun. Utiliser une telle configuration avec un firewall en production est fortement déconseillé.

Pour le besoin de ce Guide, nous décidons que:

- L'interface externe est `eth0`.
- L'interface locale est `eth1`.
- L'interface DMZ est `eth2`.

La configuration par défaut de Shorewall ne définit pas le contenu de chaque zone. Pour définir la précédente configuration en utilisant le fichier /etc/shorewall/interfaces, ce fichier doit contenir:

```
#ZONE    INTERFACE      BROADCAST     OPTIONS
net      eth0           detect        rfc1918
loc      eth1           detect
dmz      eth2           detect
```

Éditer le fichier `/etc/shorewall/interfaces` et définissez les interfaces du réseau sur votre firewall et associez chaque interface avec une zone. Si vous avez une zone qui est interfacée avec plus d'une interface, incluez simplement une entrée pour chaque interface et répéter le nom de zone autant de fois que nécessaire.

Example 1. Multiple Interfaces associé une Zone

```
#ZONE    INTERFACE      BROADCAST     OPTIONS
net      eth0           detect        rfc1918
loc      eth1           detect
loc      eth2           detect
```

Vous pouvez définir des zones plus compliquées en utilisant le fichier `/etc/shorewall/hosts` mais dans la plus part des cas, ce n'est pas nécessaire.

Adressage, Sous-réseaux et Routage

Normalement, votre FAI vous assigne des adresses Publiques. Vous pouvez configurer l'interface externe du firewall en utilisant l'une de ces adresses permanentes et vous pouvez décider comment utiliser le reste de vos adresses.

Si vous êtes déjà familier avec l'adressage IP et le routage, vous pouvez aller à la prochaine section.

La présentation précédente ne fait que d'effleurer la question des sous réseaux et du routage. Si vous êtes intéressé pour apprendre plus sur l'adressage IP et le routage, je recommande “IP Fundamentals: What Everyone Needs to Know about Addressing & Routing”, Thomas A. Maufer, Prentice-Hall, 1999, ISBN 0-13-975483-0 (link).

Adressage IP

L'adressage IP version 4 (IPv4) est codé sur 32-bit. La notation w.x.y.z se réfère à une adresse dont le byte d'ordre supérieur est “w”, le suivant à pour valeur “x”, etc. Si nous prenons l'adresse 192.0.2.14 et l'exprimons en hexadécimal, nous obtenons:

```
C0.00.02.0E
```

ou l'exprimons comme un entier de 32-bit

```
C000020E
```

Sous-réseaux

Vous entendrez toujours les termes “Class A network”, “Class B network” et “Class C network”. Au début de l'existence de l'IP, les réseaux ne comportaient que trois tailles (il y avait aussi le réseau de Class D mais il était utilisé différemment):

Class A - netmask 255.0.0.0, size = 2 ** 24
Class B - netmask 255.255.0.0, size = 2 ** 16
Class C - netmask 255.255.255.0, size = 256

La taille d'un réseau était uniquement déterminée par la valeur du byte de l'ordre supérieur, ainsi vous pouviez regarder une adresse IP et déterminer immédiatement le masque réseau. Le masque réseau est un nombre qui se termine logiquement avec une adresse qui isole le *numéro de réseau*; le reste de l'adresse est le *numéro d'hôte*. Par exemple, dans la Classe C l'adresse 192.0.2.14, le numéro hexadécimal du réseau est C00002 et le numéro hexadécimal d'hôte est 0E.

Comme l'Internet se développait, il semblait clair que la classification en adressage 32-bit allait devenir très limité (rapidement, les grandes sociétés et les universités s'étaient assigné leur propre réseau de classe A!). Après quelques faux départs, la technique courante du sous-adressage de ces réseaux en plus petits sous-réseaux évolua; cette technique est consignée par le *Classless InterDomain Routing* (CIDR). Aujourd'hui, tous les systèmes avec lesquels vous travaillerez comprennent probablement la notation CIDR. Le réseau basé sur les Classes est du domaine du passé.

Un *sous-reseau* (aussi appelé *subnet* ou *subnetwork*) est un ensemble d'adresses IP tel que:

1. Le nombre d'adresses dans le jeu est un multiple de 2; et
2. La première adresse dans le jeu est un multiple de la taille du jeu.
3. La première adresse du sous-réseau est réservée et se réfère à *l'adresse du sous-réseau*.
4. La dernière adresse du sous-réseau est réservée comme *adresse broadcast du sous-réseau*.

Comme vous pouvez le constater par cette définition, dans chaque sous-réseau de taille n il y a (n - 2) adresses utilisables (adresses qui peuvent être assignés à une hôte). La première et la dernière adresse du sous-réseau sont utilisées respectivement pour identifier l'adresse sous-réseau et l'adresse broadcast du sous-réseau. En conséquence, de petits sous-réseaux sont plus gourmands en adresses IP que de plus étendus.

Comme n est une puissance de deux, nous pouvons aisément calculer le *Logarithme Naturel* (log2) de n. Pour les plus communs des sous-réseaux, la taille et leur logarithme naturel sont donnés par la table suivante:

Table 2. Logarithme Naturel

n	log2 n	(32 - log2 n)

8	3	29
16	4	28
32	5	27
64	6	26
128	7	25
256	8	24
512	9	23
1024	10	22
2048	11	21
4096	12	20
8192	13	19
16384	14	18
32768	15	17
65536	16	16

Vous pourrez voir que la table ci-dessus contient aussi une colonne (32 - log2 **n**). Ce nombre est la *Variable de Longueur du Masque de Sous-réseau* (VLSM Variable Length Subnet Mask) pour un réseau de taille n. De la table ci-dessus, nous pouvons dériver celle-ci, ce qui est plus facile à utiliser.

Table 3. VLSM

Subnet Size	VLSM	Subnet Mask
8	/29	255.255.255.248
16	/28	255.255.255.240
32	/27	255.255.255.224
64	/26	255.255.255.192
128	/25	255.255.255.128
256	/24	255.255.255.0
512	/23	255.255.254.0
1024	/22	255.255.252.0
2048	/21	255.255.248.0
4096	/20	255.255.240.0
8192	/19	255.255.224.0
16384	/18	255.255.192.0
32768	/17	255.255.128.0
65536	/16	255.255.0.0
2 ** 24	/8	255.0.0.0

Notez que le VLSM est écrit avec un slash (“/”) -- vous pouvez souvent entendre un sous-réseau de taille 64 qui fait référence à un sous-réseau “slash 26” et un de taille 8 faisant référence à un “slash 29”.

Le masque de sous-réseau (aussi référencé par son *netmask*) est simplement un nombre de 32-bit avec le premier bit “VLSM” à un et les autres à zéro. Par exemple, pour un sous-réseau de taille 64, le masque de sous-réseau débute par 26 bits à un:

```
11111111111111111111111111000000 = FFFFFFC0 = FF.FF.FF.C0 = 255.255.255.192
```

Le masque de sous-réseau a la propriété suivante: si vous terminez logiquement le masque de sous-réseau avec une adresse dans le sous-réseau, le résultat est l'adresse du sous-réseau. Attention, si vous terminez logiquement le masque de sous-réseau avec une

adresse en dehors du sous-réseau, le résultat n'est PAS l'adresse du sous-réseau.

Comme nous l'avons vu précédemment, la propriété du masque de sous-réseau est très importante dans le routage.

Pour un sous-réseau dont l'adresse est **a.b.c.d** et dont la VLSM est **/v**, nous notons le sous-réseau “**a.b.c.d/v**” en utilisant la *notation CIDR*.

Table 4. Un exemple de sous-réseau (sub-network) :

Subnet:	10.10.10.0 - 10.10.10.127
Subnet Size:	128
Subnet Address:	10.10.10.0
Broadcast Address:	10.10.10.127
CIDR Notation:	10.10.10.0/25

Il y a deux sous-réseaux dérivés qui doivent être mentionnés; A savoir, le sous-réseau avec un membre et le sous-réseau avec 2 ** 32 membres.

Table 5. /32 and /0

Subnet Size	VLSM Length	Subnet Mask	CIDR Notation
1	32	255.255.255.255	a.b.c.d/32
32	0	0.0.0.0	0.0.0.0/0

Ainsi, chaque adresse **a.b.c.d** peut aussi être écrite **a.b.c.d/32** et l'ensemble des adresses possibles est écrit **0.0.0.0/0**.

Plus loin dans ce manuel, vous verrez la notation **a.b.c.d/v** utilisé pour décrire la configuration IP d'une interface réseau (l'utilitaire 'ip' utilise aussi cette syntaxe). cela veut simplement dire que l'interface est configuré avec une adresse ip **a.b.c.d** et avec le masque de réseau qui correspond à la variable VLSM /**v**.

Example 2. 192.0.2.65/29

L'interface est configuré avec l'adresse IP 192.0.2.65 et le netmask 255.255.255.248.

Depuis Shorewall 1.4.6, /sbin/shorewall supporte une command ipcalc qui calcule automatiquement l'information sur le [sous]réseau.

Example 3. En utilisant la commande ipcalc.

```
shorewall ipcalc 10.10.10.0/25
   CIDR=10.10.10.0/25
   NETMASK=255.255.255.128
   NETWORK=10.10.10.0
   BROADCAST=10.10.10.127
```

Example 4. En utilisant la commande ipcalc.

```
shorewall ipcalc 10.10.10.0 255.255.255.128
   CIDR=10.10.10.0/25
   NETMASK=255.255.255.128
   NETWORK=10.10.10.0
   BROADCAST=10.10.10.127
```

Routage

L'un des buts des sous-réseaux est la base du routage. Ci-dessous se trouve la table de routage de mon firewall (compressé pour du PDF):

```
[root@gateway root]# netstat -nr
Kernel IP routing table
Destination     Gateway         Genmask         Flgs MSS Win irtt Iface
192.168.9.1     0.0.0.0         255.255.255.255 UH   40  0      0 texas
206.124.146.177 0.0.0.0         255.255.255.255 UH   40  0      0 eth1
206.124.146.180 0.0.0.0         255.255.255.255 UH   40  0      0 eth3
192.168.3.0     0.0.0.0         255.255.255.0   U    40  0      0 eth3
192.168.2.0     0.0.0.0         255.255.255.0   U    40  0      0 eth1
192.168.1.0     0.0.0.0         255.255.255.0   U    40  0      0 eth2
206.124.146.0   0.0.0.0         255.255.255.0   U    40  0      0 eth0
192.168.9.0     192.0.2.223     255.255.255.0   UG   40  0      0 texas
127.0.0.0       0.0.0.0         255.0.0.0       U    40  0      0 lo
0.0.0.0         206.124.146.254 0.0.0.0         UG   40  0      0 eth0
[root@gateway root]#
```

Le périphérique *texas* est le tunnel GRE vers un site peer à Dallas, la zone Texas.

Les trois premières routes sont des *host routes* puisqu'elles indiquent comment aller vers un hôte unique. Dans la sortie de “netstat” cela peut-être vu par le “Genmask” (Masque sous-réseau) de 255.255.255.255 et le “H” dans la colonne “Flags” . Les autres sont des routes *“net” routes* car elles indiquent au noyau comment router des paquets à un sous-réseau. La dernière route est *la route par défaut* correspondant à la passerelle (gateway) mentionnée aussi appelé *passerelle par défaut (default gateway).*

Quant le noyau essaye d'envoyer un paquet à une adresse IP **A**, il commence au début de la table de routage et:

- **A** est logiquement terminé avec la valeur du “Genmask” dans l'entrée de la table.
- Le résultat est comparé avec la valeur de la “Destination” dans l'entrée de la table.
- Si le résultat et la valeur de la “Destination” sont identiques, alors:
  - Si la colonne “Gateway” n'est pas nulle, le paquet est envoyé au gateway à travers l'interface nommée dans la colonne “Iface”.
  - Sinon, le paquet est directement envoyé à **A** à travers l'interface nommée dans la colonne “iface”.
- Autrement, les étapes précédentes sont répétées sur l'entrée suivante de la table.

Puisque la route par défaut correspond à toutes les adresses IP (**A** donne 0.0.0.0 = 0.0.0.0), les paquets qui ne correspondent à aucune des autres entrées de la table de routage sont envoyés au gateway par défaut qui généralement est un routeur vers le FAI.

Voici un exemple. Supposez que vous souhaitez router un paquet à 192.168.1.5. Cette adresse ne correspond à aucune route d'hôte dans la table mais si nous terminons logiquement cette adresse avec 255.255.255.0, le résultat est 192.168.1.0 qui correspond à la l'entrée dans la table:

```
192.168.1.0   0.0.0.0       255.255.255.0   U     40  0         0 eth2
```

Donc le paquet vers 192.168.1.5 est directement envoyé à travers eth2.

Un des points qui doit être souligné -- tous les paquets sont envoyés en utilisant la table de routage et les réponses ne sont pas exclues de ce principe. Il semble y avoir une idée fausse chez ceux qui croient que les paquets réponses sont comme les saumons et contiennent un code génétique qui leur permet de suivre la route emprunté par les paquets envoyés. Ce n'est pas le cas; La réponse peut prendre un chemin totalement différent de celui de la requête du client -- les routes requête/réponse sont totalement indépendantes.

Protocole de Résolution d'Adresse (ARP)

Quant on envoie des paquets à travers Ethernet, les adresses IP ne sont pas utilisées. Bien que l'adressage Ethernet soit basé sur les adresses *Media Access Control* (MAC). Chaque périphérique Ethernet à sa propre adresse MAC qui est contenu dans une PROM

lors de la fabrication. Vous pouvez obtenir l'adresse MAC grâce à l'utilitaire “ip”:

```
[root@gateway root]# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc htb qlen 100
link/ether 02:00:08:e3:fa:55 brd ff:ff:ff:ff:ff:ff
inet 206.124.146.176/24 brd 206.124.146.255 scope global eth0
inet 206.124.146.178/24 brd 206.124.146.255 scope global secondary eth0
inet 206.124.146.179/24 brd 206.124.146.255 scope global secondary eth0
[root@gateway root]#
```

Comme vous pouvez le constater ci-dessus, l'adresse MAC codé sur 6 bytes (48 bits). L'adresse MAC est généralement aussi imprimée sur la carte elle-même.

Comme IP utilise les adresses IP et Ethernet les adresses MAC, un mécanisme est nécessaire pour transcrire une adresse IP en adresse MAC; C'est ce dont est chargé le protocole de résolution d'adresse *Address Resolution Protocol* (ARP). Voici ARP en action:

```
[root@gateway root]# tcpdump -nei eth2 arp
tcpdump: listening on eth2
09:56:49.766757 2:0:8:e3:4c:48 0:6:25:aa:8a:f0 arp 42:
                arp who-has 192.168.1.19 tell 192.168.1.254
09:56:49.769372 0:6:25:aa:8a:f0 2:0:8:e3:4c:48 arp 60:
                arp reply 192.168.1.19 is-at 0:6:25:aa:8a:f0

2 packets received by filter
0 packets dropped by kernel
[root@gateway root]#
```

Dans cet échange , 192.168.1.254 (MAC 2:0:8:e3:4c:48) veut connaître l'adresse MAC du périphérique avec l'adresse IP 192.168.1.19. Le système ayant cette adresse IP répond que l'adresse MAC du périphérique avec l'adresse IP 192.168.1.19 est 0:6:25:aa:8a:f0.

Afin de rendre disponible les informations d'échange ARP chaque fois qu'un paquet est envoyé, le système maintient un *cache ARP* of IP<->MAC correspondances. Vous pouvez voir le cache ARP sur votre système (également sur les systèmes Windows™) en utilisant la commande “arp”:

```
[root@gateway root]# arp -na
? (206.124.146.177) at 00:A0:C9:15:39:78 [ether] on eth1
? (192.168.1.3) at 00:A0:CC:63:66:89 [ether] on eth2
? (192.168.1.5) at 00:A0:CC:DB:31:C4 [ether] on eth2
? (206.124.146.254) at 00:03:6C:8A:18:38 [ether] on eth0
? (192.168.1.19) at 00:06:25:AA:8A:F0 [ether] on eth2
```

Les détails de réponse sont le résultat de l'utilisation de l'option “n” (Windows™ “arp” n'accepte pas cette option) qui force le programme “arp” à la translation de résolution de noms IP->DNS. Si je n'utilise pas cette option, le point d'interrogation sera remplacé par le noms correspondant à chaque adresse IP. Notez que la dernière information dans la table d'enregistrement est celle que nous voyons en utilisant précédemment tcpdump.

RFC 1918

Les adresses IP sont allouées par l'autorité Internet Assigned Number Authority (IANA) qui délégue des allocations géographiques basées sur le Regional Internet Registries (RIR). Par exemple, les allocations pour les Etats-Unis sont déléguées à American Registry for Internet Numbers (ARIN). Ces RIR peuvent déléguer à des bureaux nationaux. La plus part d'entre nous ne traite pas avec autorités mais obtienne plutôt leur adresse IP par leur FAI.

Dans la réalité, généralement on ne peut se permettre autant d'adresses IP Publiques que de périphériques à assigner si bien que nous utiliseront des adresses IP Privées. RFC 1918 réserve plusieurs plages d'adresse IP à cet usage:

```
10.0.0.0    - 10.255.255.255
172.16.0.0  - 172.31.255.255
192.168.0.0 - 192.168.255.255
```

Les adresses réservées par la RFC 1918 sont parfois appelées non-routable car le routeur passerelle Internet ne renvoi pas les paquets qui ont une adresse de destination RFC-1918. Cela est compréhensible car tout le monde peut choisir ces adresses pour un usage privé.

Quant on choisit des adresses de ces plages, il y a deux choses à garder en mémoire:

- Comme l'espace des adresses IPv4 s'épuise, de plus en plus d'organisation (comprenant les FAI) commencent à utiliser les adresses RFC 1918 dans leur infrastructures.
- Vous ne voulez pas utiliser des adresses IP qui sont utilisés par votre FAI ou une autre organisation avec laquelle vous souhaiter établir une liaison VPN

C'est pourquoi c'est une bonne idée de vérifier avec votre FAI s'il n'utilise pas (ou ne prévoie pas d'utiliser) des adresses privées avant de décider les adresses que vous allez utiliser.

Note

Dans ce document, les adresses IP externes “réels” du type 192.0.2.x. 192.0.2.0/24 sont réservées par RFC 3330 pour l'utilisation d'adresses IP publiques. Ces adresses ne doivent pas être confondues avec les adresses 192.168.0.0/16; comme décrit ci-dessus, celles-ci sont réservées par RFC 1918 pour une utilisation privée.

Configurer votre Réseau

Le choix de configuration de votre réseau dépend d'abord du nombre d'adresses Public IP dont vous avez besoin, c'est à dire du nombre d'entités adressables que vous avez sur votre réseau. En fonction du nombre d'adresses que vous avez, votre FAI peut servir ce jeu d'adresses de deux manières:

- **Routed** - Le trafic vers chacune de vos adresses sera routé à travers une unique adresse passerelle. Cela sera généralement fait si votre FAI vous assigne un sous-réseau complet (/29 ou plus). Dans ce cas, vous assignerez l'adresse passerelle comme adresse IP de l'interface externe de votre firewall/router.
- **Non-routed** - Votre FAI vous donnera directement le trafic de chaque adresse directement.

Dans les paragraphes qui suivent, nous étudierons chaque cas séparément.

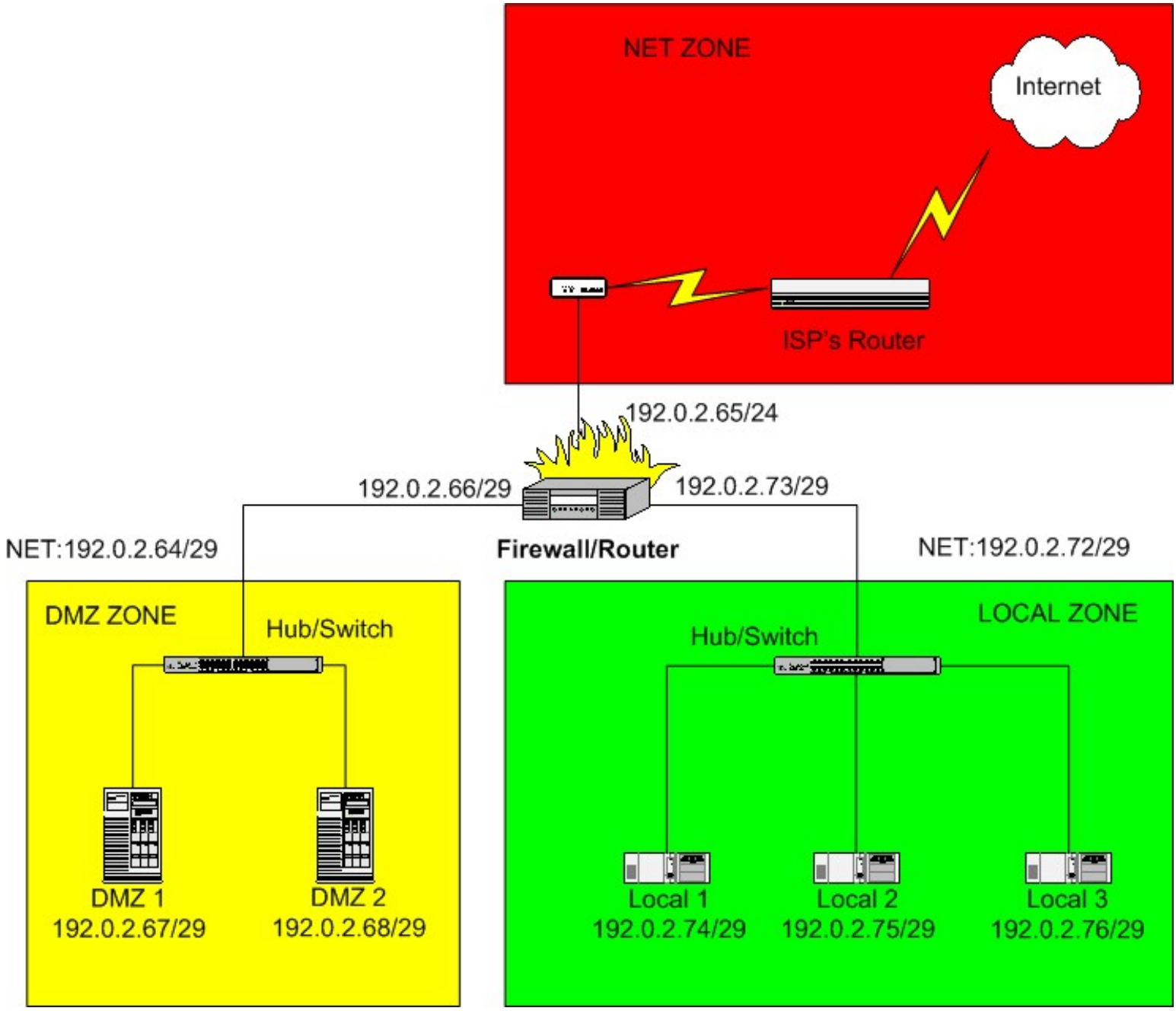
Avant de commencer, il y a une chose que vous devez vérifier:

Si vous utilisez le package Debian, vérifier svp votre fichier shorewall.conf afin de contrôler les paramètres suivants; si ce n'est pas juste, appliquer les changements nécessaires:

- NAT_ENABLED=Yes (Shorewall versions antérieures à 1.4.6)
- IP_FORWARDING=On

Routage

Supposons que votre fournisseur d'accès FAI vous a assigné le sous-réseau 192.0.2.64/28 routé à travers 192.0.2.65. Cela veut dire que vous avez les adresses IP 192.0.2.64 - 192.0.2.79 et que l'adresse externe de votre firewall est 192.0.2.65. Votre FAI vous a aussi dit que vous pouvez utiliser le masque de réseau 255.255.255.0 (ainsi votre /28 est une partie de /24). Avec ces adresses IP, vous pouvez scinder votre réseau /28 en deux /29 et configurer votre réseau comme l'indique le diagramme suivant.



Ici, la zone démilitarisé DMZ comprend le sous-réseau 192.0.2.64/29 et le réseau Local 192.0.2.72/29. La passerelle par défaut pour les hôtes dans la DMZ pourra être configuré à 192.0.2.66 et la passerelle par défaut pour les hôtes du réseau local pourra être 192.0.2.73.

Notez que cet arrangement est plus gourmand en adresses publiques puisqu'il utilise 192.0.2.64 et 192.0.2.72 pour les adresses du sous-réseau, 192.0.2.71 et 192.0.2.79 pour les adresses broadcast du réseau, de même que 192.0.2.66 et 168.0.2.73 pour les adresses internes que le firewall/routeur. Néanmoins, cela montre comment nous pouvons faire avec un réseau /24 plutôt qu'un /28, l'utilisation de 6 adresses IP parmi les 256 peut être justifié par la simplicité du paramétrage.

Le lecteur astucieux aura remarqué que l'interface externe du firewall/Routeur est actuellement incluse dans le sous-réseau DMZ (192.0.2.64/29). Que se passe-t-il si DMZ 1 (192.0.2.67) essaye de communiquer avec 192.0.2.65? La table de routage sur DMZ 1 peut ressembler à cela:

```
Kernel IP routing table
Destination     Gateway     Genmask         Flags MSS Window irtt Iface
192.0.2.64      0.0.0.0     255.255.255.248 U     40  0         0 eth0
0.0.0.0         192.0.2.66  0.0.0.0         UG    40  0         0 eth0
```

Cela indique que DMZ 1 enverra une requête ARP "qui a 192.0.2.65" et aucune interface sur le segment Ethernet DMZ à cette adresse IP. Assez bizarrement, le firewall répondra à la requête avec l'adresse MAC de sa propre DMZ Interface!! DMZ 1 peut alors

envoyer des trames Ethernet adressées à cette adresse MAC et les trames seront reçues (correctement) par le firewall/routeur.

C'est plutôt une possibilité inattendue d'ARP sur la partie du Noyau Linux qui pousse cet avertissement très tôt dans ce manuel à propos de la connexion de plusieurs interfaces firewall/routeur au même hub ou switch. Quant une requête ARP destinée à une des adresses firewall/routeur est envoyée par un autre système connecté au hub/switch, toutes les interfaces du firewall qui se connectent au hub/switch peuvent répondre! C'est alors une course à la réponse qui "est-là" qui atteindra en premier l'émetteur.

Non-routé

Avec la situation précédente mais non-routé, vous pouvez configurer votre réseau exactement comme décrit ci-dessus avec une condition supplémentaire; spécifiez simplement l'option “proxyarp” sur les trois interfaces du firewall dans le fichier /etc/shorewall/interfaces file.

La plus part d'entre nous n'ont pas le luxe d'avoir assez d'adresses publiques IP pour configurer notre réseau comme montré dans le précédent exemple (même si la configuration est routée).

Pour le besoin de cette section, admettons que notre FAI nous a assigné les adresses IP 192.0.2.176-180 et nous a dit d'utiliser le masque de réseau 255.255.255.0 et la passerelle par défaut 192.0.2.254.

Clairement, ce jeu d'adresses ne comprend pas de sous-réseau et n'a pas suffisamment d'adresses pour toutes les interfaces de notre réseau. Il y a quatre possibilités qui peuvent être utilisées pour régler ce problème.

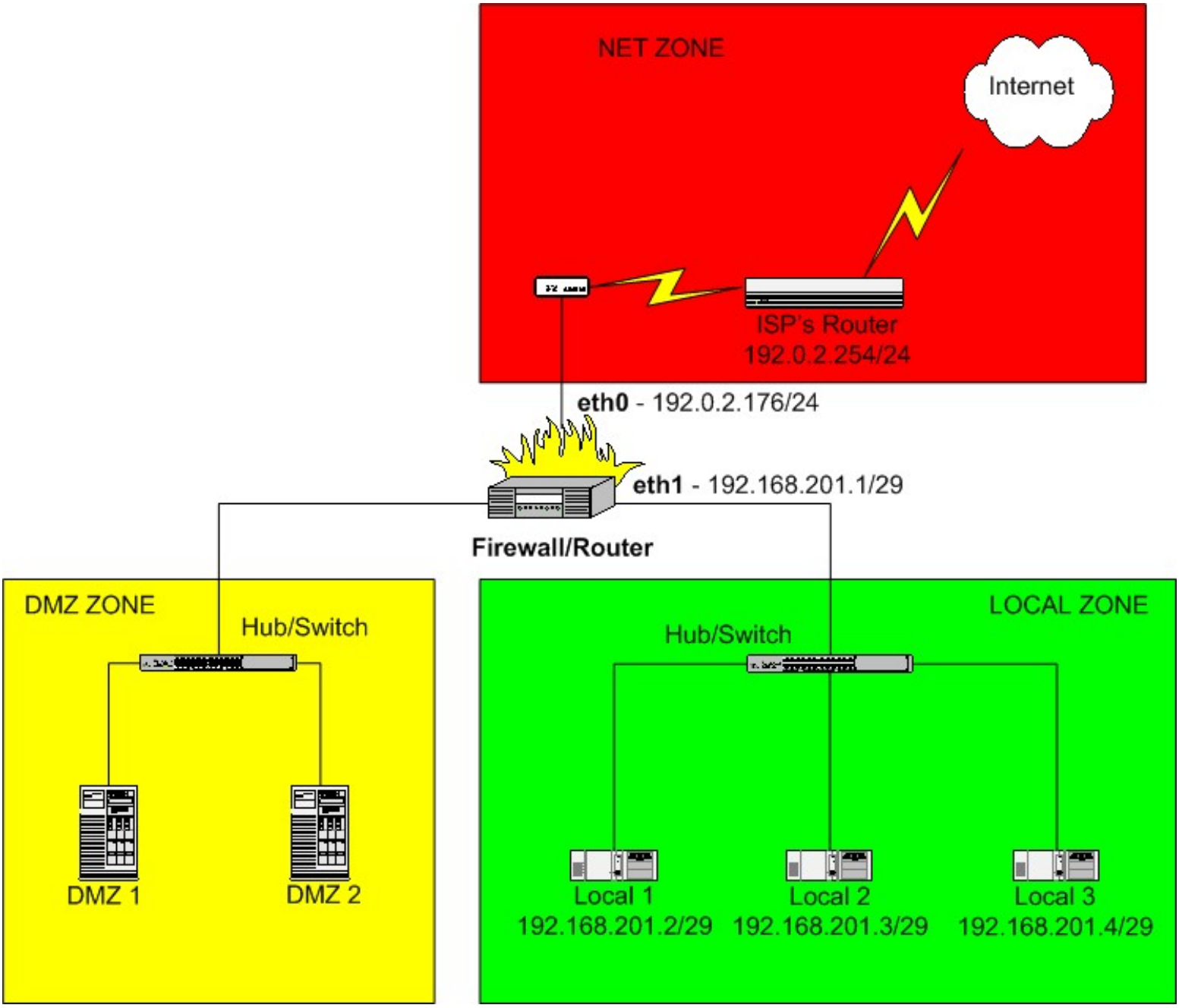
- *Source Network Address Translation* (SNAT).
- *Destination Network Address Translation* (DNAT) aussi nommé *Port Forwarding*.
- *Proxy ARP*.
- *Network Address Translation* (NAT) aussi appelé *One-to-one NA*T.

Souvent une combinaison de ces techniques est utilisée. Chacune d'entre elle sera détaillée dans la section suivante.

SNAT

Avec SNAT, un segment interne LAN est configuré en utilisant les adresses RFC 1918. Quant un hôte **A** sur ce segment interne initialise une connexion vers un hôte **B** sur Internet, le firewall/routeur réécrit les entêtes IP dans la requête pour utiliser une de vos adresses publiques IP en tant qu'adresse source. Quant **B** répond et que la réponse est reçu par le firewall, le firewall change l'adresse destination par celle RFC 1918 de **A** et renvoi la réponse à **A.**

Supposons que vous décidiez d'utiliser SNAT sur votre zone locale et utilisiez l'adresse publique 192.0.2.176 à la fois comme adresse externe du firewall et l'adresse source des requêtes Internet envoyées depuis cette zone.



La zone locale a été assigné au sous-réseau 192.168.201.0/29 (netmask 255.255.255.248).

Le système dans la zone locale pourra être configuré avec la passerelle par défaut 192.168.201.1 (L'adresse IP de l'interface local du firewall).

SNAT est configuré dans Shorewall avec le fichier `/etc/shorewall/masq`.

```
#INTERFACE      SUBNET              ADDRESS
eth0            192.168.201.0/29    192.0.2.176
```

Cet exemple utilise la technique normale pour assigner la même adresse publique IP pour l'interface externe du firewall et pour SNAT. Si vous souhaitez utiliser une adresse IP différente, vous pouvez soit utiliser les outils de configuration réseau de votre distribution pour ajouter cette adresse IP ou vous pouvez mettre la variable ADD_SNAT_ALIASES=Yes dans /etc/shorewall/shorewall.conf si bien que Shorewall ajoutera l'adresse pour vous.

DNAT

Quant SNAT est utilisé, il est impossible pour les hôtes sur Internet d'initialiser une connexion avec un des systèmes puisque ces systèmes n'ont pas d'adresses publiques IP. DNAT fournit une méthode pour autoriser des connexions sélectionnés depuis Internet.

Supposons que votre fille souhaite héberger un serveur Web sur son système "Local 3". Vous pouvez autoriser les connexions d'Internet à son serveur en ajoutant l'entrée suivante dans le fichier `/etc/shorewall/rules`:

```
#ACTION   SOURCE   DEST                 PROTO   DEST     SOURCE     ORIGINAL
#                                               PORT(S)  PORT(S)    DEST
DNAT      net      loc:192.168.201.4    tcp     www
```

Si une des amies de votre fille avec une adresse **A** veut accéder au serveur de votre fille, elle peut se connecter à l'adresse http://192.0.2.176 (l'adresse IP externe de votre firewall) et le firewall réécrira l'adresse IP à 192.168.201.4 (le système de votre fille) et enverra la requête. Quant le serveur de votre fille répond, le firewall réécrira la source de réponse avec 192.0.2.176 et retournera la réponse à **A**.

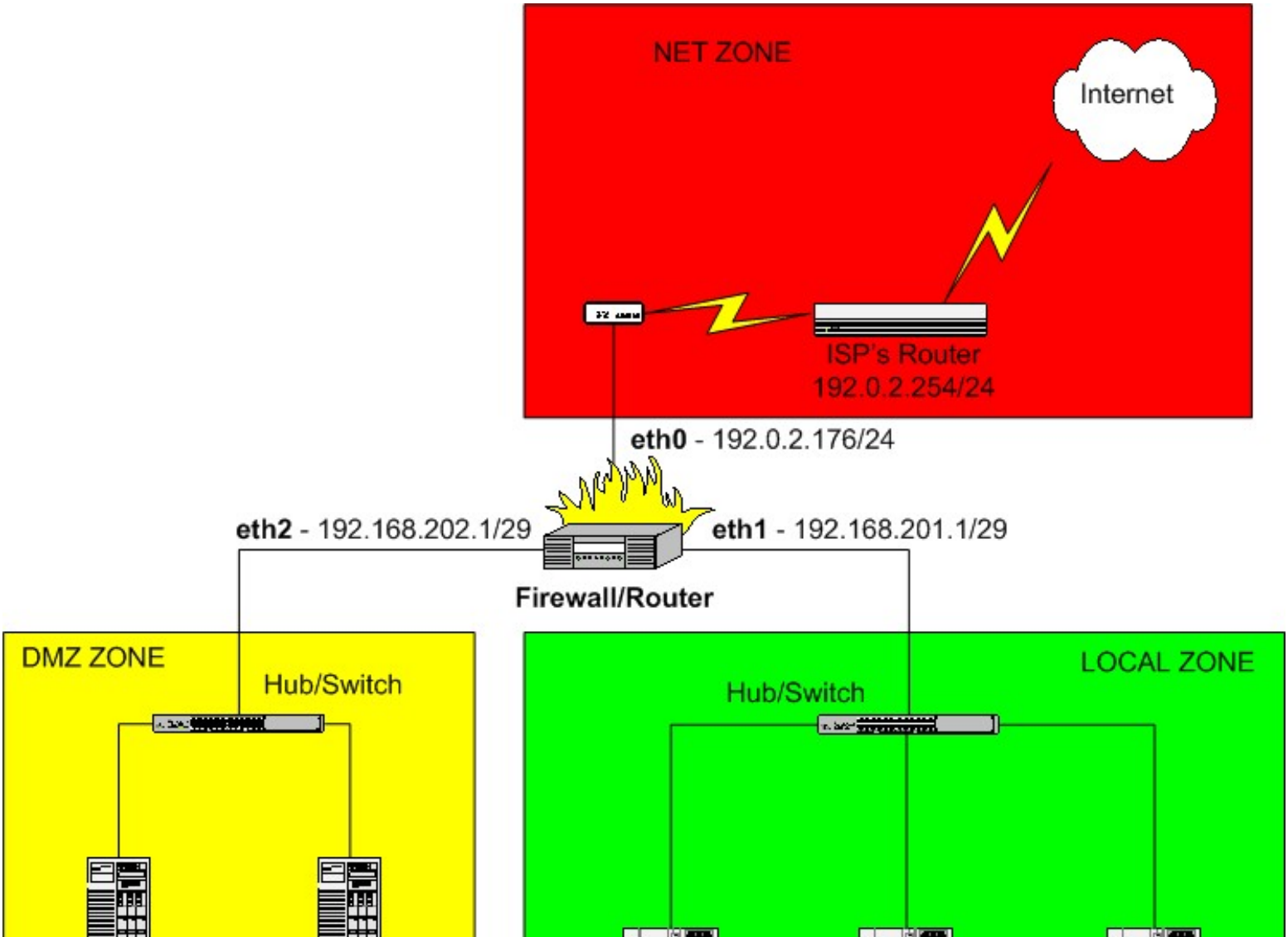
Cet exemple l'adresse externe IP du firewall pour DNAT. Vous pouvez utiliser une autre de vos adresses IP publiques, mais Shorewall n'ajoutera pas pour vous cette adresse à l'interface externe du firewall.

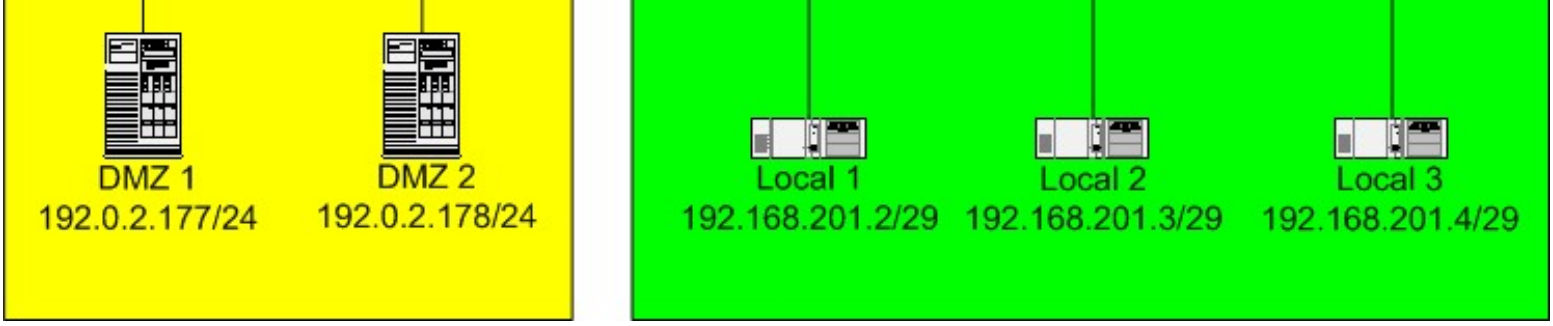
Proxy ARP

Le principe du proxy ARP est:

- Un hôte **H** derrière votre firewall est assigné à une de vos adresses publiques (**A**), a le même masque de réseau (**M**) que l'interface externe du firewall.
- Le firewall répond à ARP "qui a" demandé **A**.
- Quant **H** délivre une requête ARP "qui a" pour une adresse du sous-réseau définit par **A** et **M**, le firewall répondra (avec l'adresse MAC si le firewall s'interface à **H**).

Supposons que nous décidons d'utiliser Proxy ARP sur DMZ de notre exemple réseau.





Ici, nous avons assigné les adresses IP 192.0.2.177 au système DMZ 1 et 192.0.2.178 à DMZ 2. Notez que nous avons juste assigné une adresse arbitraire RFC 1918 et un masque de sous-réseau à l'interface DMZ de notre firewall. Cette adresse et le masque ne sont pas pertinentes - vérifiez juste que celle-ci n'écrase pas un autre sous-réseau déjà défini.

La configuration de Proxy ARP est faite dans le fichier `/etc/shorewall/proxyarp`.

```
#ADDRESS      INTERFACE     EXTERNAL      HAVE ROUTE        PERSISTANT
192.0.2.177   eth2          eth0          No
192.0.2.178   eth2          eth0          No
```

Parce que la variable HAVE ROUTE contient No, Shorewall ajoutera les routes d'hôte à travers eth2 à 192.0.2.177 et 192.0.2.178. Les interfaces ethernet de DMZ 1 et DMZ 2 pourront être configurées pour avoir les adresses IP apparentes mais devront avoir la même passerelle par défaut que le firewall lui-même -- nommé 192.0.2.254. En d'autres termes, elles pourront être configurées juste comme elles devraient être si elles étaient parallèles au firewall plutôt que derrière lui.

Caution

Ne pas ajouter le(s) adresse(s) ARP (192.0.2.177 et 192.0.2.178 dans l'exemple ci-dessus) à l'interface externe (eth0 dans cet exemple) du firewall.

Un mot de mise en garde à sa place ici. Les FAI configure(nt) typiquement leur routeur avec un timeout de cache ARP élevé. Si vous déplacer un système parallèle à votre firewall derrière le Proxy ARP du firewall, cela peut mettre des HEURES avant que le système puisse communiquer avec Internet. Il y a deux choses que vous pouvez essayer de faire:

1. (Courtoisement de Bradey Honsinger) Une lecture de Stevens' TCP/IP Illustrated, Vol 1 révèle qu'un paquet ARP “gratuitous” peut entraîner le routeur de votre FAI à rafraîchir son cache(section 4.7). Une "gratuitous" ARP est simplement une requête d'un hôte demandant l'adresse MAC de sa propre adresse IP; éventuellement pour vérifier que l'adresse IP n'est pas dupliquée,...

   Si l'hôte envoyant la commande “gratuitous” ARP vient juste de changer son adresse IP..., ce paquet entraîne tous les autres hôtes...qui ont une entrée dans son cache pour l'ancienne adresse matériel de mettre à jour également ses caches ARP.

   Ce qui est exactement, bien sûr, ce que vous souhaitez faire lorsque vous basculez un hôte vulnérable à Internet derrière Shorewall utilisant proxy ARP (ou one-to-one NAT). Heureusement, des packages récents (Redhat™) iputils incluent "arping", avec l'option "-U" qui fait cela:

   ```
   arping -U -I <net if> <newly proxied IP>

   arping -U -I eth0 66.58.99.83 # for example
   ```

   Stevens continue en mentionnant que tous les systèmes répondent correctement au gratuitous ARPs,et “googling” pour “arping -U” semble aller dans ce sens.
2. Vous pouvez appeler votre FAI et dire de purger l'ancienne entrée du cache ARP mais la plupart ne veulent ou ne peuvent le faire.

Vous pouvez vérifier si le cache ARP de votre FAI est ancien en utilisant ping et tcpdump. Supposez que vous pensez que la passerelle routeur a une ancienne entrée ARP pour 192.0.2.177. Sur le firewall, lancez tcpdump de cette façon:

```
tcpdump -nei eth0 icmp
```

Maintenant depuis 192.0.2.177, utilisez ping vers la passerelle du FAI (que nous supposons être 192.0.2.254):

```
ping 192.0.2.254
```

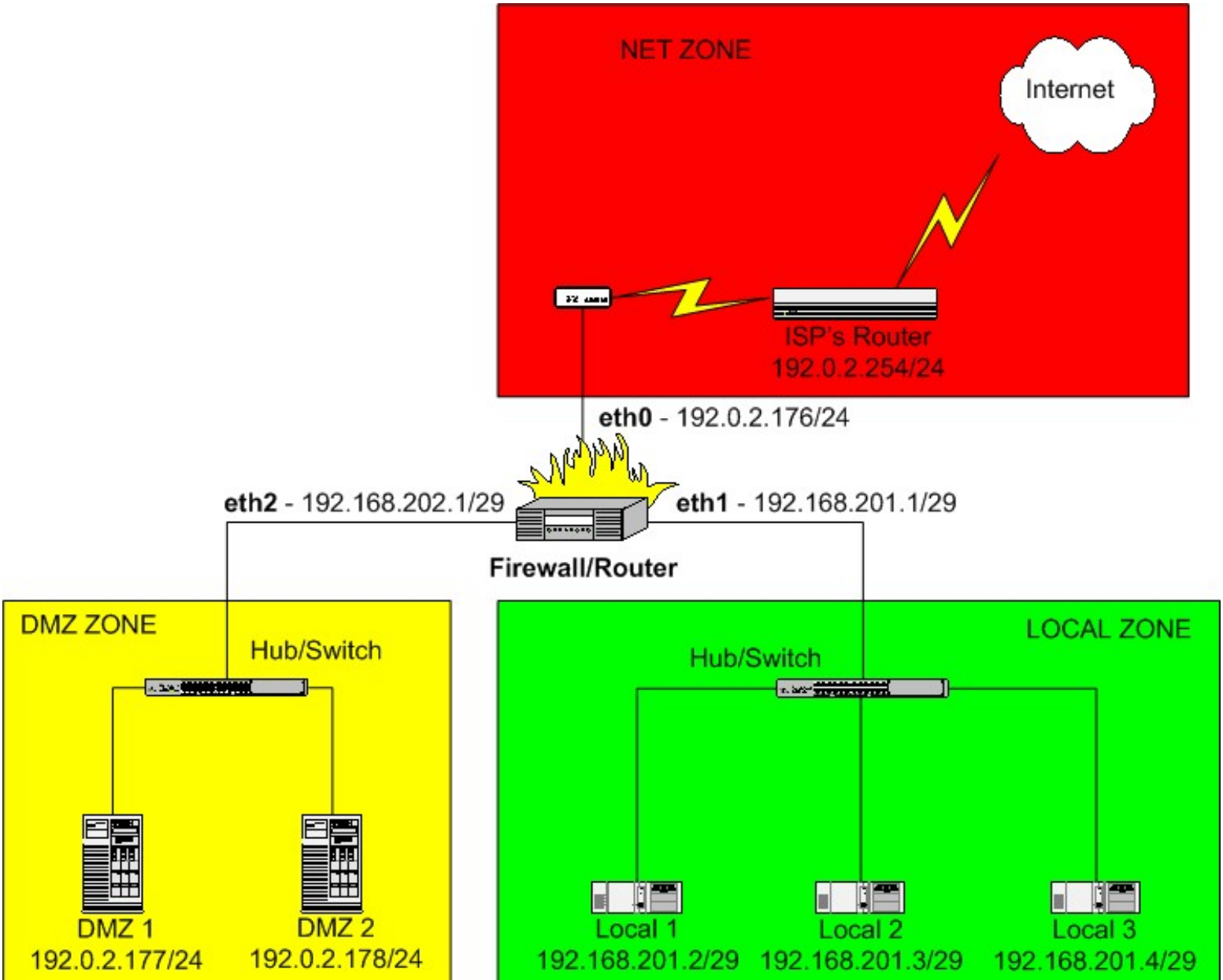
Nous pouvons maintenant observer le résultat de tcpdump:

```
13:35:12.159321 0:4:e2:20:20:33 0:0:77:95:dd:19 ip 98:
                192.0.2.177 > 192.0.2.254: icmp: echo request (DF)
13:35:12.207615 0:0:77:95:dd:19 0:c0:a8:50:b2:57 ip 98:
                192.0.2.254 > 192.0.2.177 : icmp: echo reply
```

Notez que l'adresse source MAC dans la requête echo est différente de l'adresse de destination dans la réponse echo!! Dans le cas ou 0:4:e2:20:20:33 était l'adresse MAC de l'interface NIC eth0 du firewall tandis que 0:c0:a8:50:b2:57 était l'adresse MAC de DMZ 1. En d'autre termes, le cache ARP de la passerelle associe encore 192.0.2.177 avec la NIC de DMZ 1 plutôt qu'avec `eth0` du firewall.

One-to-one NAT

Avec one-to-one NAT, vous assignez les adresses systèmes RFC 1918 puis établissez une à une l'assignation entre ces adresses et les adresses publiques. Pour les occurrences des connexions sortantes SNAT (Source Network Address Translation) et pour les occurrences des connexions entrantes DNAT (Destination Network Address Translation). Voyons avec l'exemple précédent du serveur web de votre fille tournant sur le système Local 3.



Rappel du paramétrage, le réseau local utilise SNAT et partage l'IP externe du firewall (192.0.2.176) pour les connexions sortantes. Cela est obtenu avec l'entrée suivante dans le fichier `/etc/shorewall/masq`:

```
#INTERFACE      SUBNET                ADDRESS
eth0            192.168.201.0/29      192.0.2.176
```

Supposons maintenant que vous avez décidé d'allouer à votre fille sa propre adresse IP (192.0.2.179) pour l'ensemble des connexions entrantes et sortantes. Vous devrez faire cela en ajoutant une entrée dans le fichier `/etc/shorewall/nat`.

```
#EXTERNAL    INTERFACE    INTERNAL        ALL INTERFACES    LOCAL
192.0.2.179  eth0         192.168.201.4   No                No
```

Avec cette entrée active, votre fille a sa propre adresse IP et les deux autres systèmes locaux partagent l'adresse IP du firewall.

Une fois que la relation entre 192.0.2.179 et192.168.201.4 est établie par l'entrée ci-dessus, ce n'est pas nécessaire d'utiliser une règle DNAT pour le serveur Web de votre fille -- vous devez simplement utiliser une règle ACCEPT:

```
#ACTION   SOURCE   DEST                 PROTO   DEST     SOURCE     ORIGINAL
#                                               PORT(S)  PORT(S)    DEST
ACCEPT    net      loc:192.168.201.4    tcp     www
```

Un mot de mise en garde à sa place ici. Les FAI configure(nt) typiquement leur routeur avec un timeout de cache ARP élevé. Si vous déplacer un système parallèle à votre firewall derrière le One-on-one NAT du firewall, cela peut mettre des HEURES avant que le système puisse communiquer avec Internet. Il y a deux choses que vous pouvez essayer de faire:

1. (Courtoisement de Bradey Honsinger) Une lecture de Stevens' TCP/IP Illustrated, Vol 1 révèle qu'un paquet ARP “gratuitous” peut entraîner le routeur de votre FAI à rafraîchir son cache(section 4.7). Une "gratuitous" ARP est simplement une requête d'un hôte demandant l'adresse MAC de sa propre adresse IP; éventuellement pour vérifier que l'adresse IP n'est pas dupliquée,...

   Si l'hôte envoyant la commande “gratuitous” ARP vient juste de changer son adresse IP..., ce paquet entraîne tous les autres hôtes...qui ont une entrée dans son cache pour l'ancienne adresse matériel de mettre à jour également ses caches ARP.

   Ce qui est exactement, bien sûr, ce que vous souhaitez faire lorsque vous basculez un hôte vulnérable à Internet derrière Shorewall utilisant proxy ARP (ou one-to-one NAT). Heureusement, des packages récents (Redhat™) iputils incluent "arping", avec l'option "-U" qui fait cela:

   ```
   arping -U -I <net if> <newly proxied IP>

   arping -U -I eth0 66.58.99.83 # for example
   ```

   Stevens continue en mentionnant que tous les systèmes répondent correctement au gratuitous ARPs,et “googling” pour “arping -U” semble aller dans ce sens.
2. Vous pouvez appeler votre FAI et dire de purger l'ancienne entrée du cache ARP mais la plupart ne veulent ou ne peuvent le faire.

Vous pouvez vérifier si le cache ARP de votre FAI est ancien en utilisant ping et tcpdump. Supposez que vous pensez que la passerelle routeur a une ancienne entrée ARP pour 192.0.2.177. Sur le firewall, lancez tcpdump de cette façon::

```
tcpdump -nei eth0 icmp
```

Maintenant depuis 192.0.2.177, utilisez ping vers la passerelle du FAI (que nous supposons être 192.0.2.254):

```
ping 192.0.2.254
```

Nous pouvons maintenant observer le résultat de tcpdump:

```
13:35:12.159321 0:4:e2:20:20:33 0:0:77:95:dd:19 ip 98:
                192.0.2.177 > 192.0.2.254: icmp: echo request (DF)
13:35:12.207615 0:0:77:95:dd:19 0:c0:a8:50:b2:57 ip 98:
                192.0.2.254 > 192.0.2.177 : icmp: echo reply
```

Notez que l'adresse source MAC dans la requête echo est différente de l'adresse de destination dans la réponse echo!! Dans le cas ou 0:4:e2:20:20:33 était l'adresse MAC de l'interface NIC eth0 du firewall tandis que 0:c0:a8:50:b2:57 était l'adresse MAC de DMZ 1. En d'autre termes, le cache ARP de la passerelle associe encore 192.0.2.177 avec la NIC de DMZ 1 plutôt qu'avec `eth0` du firewall.

Règles

Avec les politiques par défaut, vos systèmes locaux (Local 1-3) peuvent accéder à tous les serveurs sur Internet et la DMZ ne peut accéder à aucun autre hôte (incluant le firewall). Avec les exceptions des règles règles NAT qui entraînent la translation d'adresses et permet aux requêtes de connexion translatées de passer à travers le firewall, la façon d'autoriser des requêtes à travers le firewall est d'utiliser des règles ACCEPT.

Note

Puisque les colonnes SOURCE PORT et ORIG. DEST. ne sont pas utilisées dans cette section, elle ne sont pas affichées.

Vous souhaiter certainement autoriser ping entre vos zones:

```
#ACTION  SOURCE  DEST                PROTO  DEST
#                                           PORT(S)
ACCEPT   net     dmz                 icmp   echo-request
ACCEPT   net     loc                 icmp   echo-request
ACCEPT   dmz     loc                 icmp   echo-request
ACCEPT   loc     dmz                 icmp   echo-request
```

En supposant que vous avez des serveurs mail et pop3 actifs sur DMZ 2 et un serveur Web sur DMZ 1. Les règles dont vous avez besoin sont:

```
#ACTION  SOURCE          DEST                PROTO  DEST    COMMENTS
#                                                   PORT(S)
ACCEPT   net             dmz:192.0.2.178     tcp    smtp    #Mail from
                                                            #Internet
ACCEPT   net             dmz:192.0.2.178     tcp    pop3    #Pop3 from
                                                            #Internet
ACCEPT   loc             dmz:192.0.2.178     tcp    smtp    #Mail from local
                                                            #Network
ACCEPT   loc             dmz:192.0.2.178     tcp    pop3    #Pop3 from local
                                                            #Network
ACCEPT   fw              dmz:192.0.2.178     tcp    smtp    #Mail from the
                                                            #Firewall
```

```
ACCEPT   dmz:192.0.2.178 net                tcp    smtp    #Mail to the
                                                           #Internet
ACCEPT   net             dmz:192.0.2.177    tcp    http    #WWW from
                                                           #Internet
ACCEPT   net             dmz:192.0.2.177    tcp    https   #Secure WWW
                                                           #from Internet
ACCEPT   loc             dmz:192.0.2.177    tcp    https   #Secure WWW
                                                           #from local
                                                           #Network
```

Si vous utilisez un serveur DNS publique sur 192.0.2.177, vous devez ajouter les règles suivantes:

```
#ACTION  SOURCE          DEST               PROTO  DEST    COMMENTS
#                                                  PORT(S)
ACCEPT   net             dmz:192.0.2.177    udp    domain  #UDP DNS from
                                                           #Internet
ACCEPT   net             dmz:192.0.2.177    tcp    domain  #TCP DNS from
                                                           #Internet
ACCEPT   loc             dmz:192.0.2.177    udp    domain  #UDP DNS from
                                                           #Local Network
ACCEPT   loc             dmz:192.0.2.177    tcp    domain  #TCP DNS from
                                                           #Local Network
ACCEPT   fw              dmz:192.0.2.177    udp    domain  #UDP DNS from
                                                           #the Firewall
ACCEPT   fw              dmz:192.0.2.177    tcp    domain  #TCP DNS from
                                                           #the Firewall
ACCEPT   dmz:192.0.2.177 net                udp    domain  #UDP DNS to
                                                           #the Internet
ACCEPT   dmz:192.0.2.177 net                tcp    domain  #TCPP DNS to
                                                           #the Internet
```

Vous souhaitez probablement communiquer entre votre firewall et les systèmes DMZ depuis le réseau local -- Je recommande SSH qui, grâce à son utilitaire scp peut aussi faire de la diffusion et de la mise à jour de logiciels.

```
#ACTION  SOURCE          DEST               PROTO  DEST    COMMENTS
#                                           PORT(S)
ACCEPT   loc             dmz                tcp    ssh     #SSH to the DMZ
ACCEPT   net             fw                 tcp    ssh     #SSH to the
                                                           #Firewall
```

D'autres petites choses

La discussion précédente reflète ma préférence personnelle pour l'utilisation de Proxy ARP associé à mes serveurs de la DMZ et SNAT/NAT pour mes systèmes locaux. Je préfère utiliser NAT seulement dans le cas ou un système qui fait partie d'un sous-réseau RFC 1918 à besoin d'avoir sa propre adresse IP.

Si vous ne l'avez pas fait, ce peut-être une bonne idée de parcourir le fichier `/etc/shorewall/shorewall.conf` afin de voir si autre chose pourrait être intéressant. Vous pouvez aussi regarder aux autres fichiers de configuration que vous n'avez pas touché pour un aperçu des autres possibilités de Shorewall.

Dans le cas ou vous n'auriez pas validé les étapes, ci-dessous se trouve un jeu final des fichiers de configuration pour notre réseau exemple. Uniquement ceux modifiés de la configuration originale sont montrés.

`/etc/shorewall/interfaces` (Les "options" seront spécifiques aux sites).

```
#ZONE    INTERFACE      BROADCAST     OPTIONS
net      eth0           detect        rfc1918,routefilter
loc      eth1           detect
dmz      eth2           detect
```

La configuration décrite nécessite que votre réseau soit démarré avant que Shorewall puisse se lancer. Cela ouvre un lapse de temps durant lequel vous n'avez pas de protection firewall.

Si vous remplacez “detect” par les valeurs des adresses broadcoast dans les entrées suivantes, vous pouvez activer Shorewall avant les interfaces réseau.

```
#ZONE    INTERFACE      BROADCAST     OPTIONS
net      eth0           192.0.2.255   rfc1918
loc      eth1           192.168.201.7
dmz      eth2           192.168.202.7
```

`/etc/shorewall/masq` - Sous-réseau Local

```
#INTERFACE     SUBNET                ADDRESS
eth0           192.168.201.0/29      192.0.2.176
```

`/etc/shorewall/proxyarp` - DMZ

```
#ADDRESS     EXTERNAL     INTERFACE     HAVE ROUTE
192.0.2.177  eth2         eth0          No
192.0.2.178  eth2         eth0          No
```

`/etc/shorewall/nat`- Le système de ma fille

```
#EXTERNAL   INTERFACE   INTERNAL       ALL INTERFACES   LOCAL
192.0.2.179 eth0        192.168.201.4  No               No
```

`/etc/shorewall/rules`

```
#ACTION  SOURCE          DEST                PROTO  DEST    COMMENTS
#                                                   PORT(S)
ACCEPT   net             dmz                 icmp   echo-request
ACCEPT   net             loc                 icmp   echo-request
ACCEPT   dmz             loc                 icmp   echo-request
ACCEPT   loc             dmz                 icmp   echo-request
ACCEPT   net             loc:192.168.201.4   tcp    www     #Daughter's
                                                            #Server
ACCEPT   net             dmz:192.0.2.178     tcp    smtp    #Mail from
                                                            #Internet
ACCEPT   net             dmz:192.0.2.178     tcp    pop3    #Pop3 from
                                                            #Internet
ACCEPT   loc             dmz:192.0.2.178     tcp    smtp    #Mail from local
                                                            #Network
ACCEPT   loc             dmz:192.0.2.178     tcp    pop3    #Pop3 from local
                                                            #Network
ACCEPT   fw              dmz:192.0.2.178     tcp    smtp    #Mail from the
                                                            #Firewall
ACCEPT   dmz:192.0.2.178 net                 tcp    smtp    #Mail to the
                                                            #Internet
ACCEPT   net             dmz:192.0.2.177     tcp    http    #WWW from
                                                            #Internet
ACCEPT   net             dmz:192.0.2.177     tcp    https   #Secure WWW
```

```
                                                        #from Internet
ACCEPT   loc             dmz:192.0.2.177    tcp    https   #Secure WWW
                                                        #from local
                                                        #Network
ACCEPT   net             dmz:192.0.2.177    udp    domain  #UDP DNS from
                                                        #Internet
ACCEPT   net             dmz:192.0.2.177    tcp    domain  #TCP DNS from
                                                        #Internet
ACCEPT   loc             dmz:192.0.2.177    udp    domain  #UDP DNS from
                                                        #Local Network
ACCEPT   loc             dmz:192.0.2.177    tcp    domain  #TCP DNS from
                                                        #Local Network
ACCEPT   fw              dmz:192.0.2.177    udp    domain  #UDP DNS from
                                                        #the Firewall
ACCEPT   fw              dmz:192.0.2.177    tcp    domain  #TCP DNS from
                                                        #the Firewall
ACCEPT   dmz:192.0.2.177 net                udp    domain  #UDP DNS to
                                                        #the Internet
ACCEPT   dmz:192.0.2.177 net                tcp    domain  #TCPP DNS to
                                                        #the Internet
ACCEPT   loc             dmz                tcp    ssh     #SSH to the DMZ
ACCEPT   net             fw                 tcp    ssh     #SSH to the
                                                        #Firewall
```

DNS

En donnant une collection d'adresses RFC 1918 et publiques dans la configuration, cela justifie d'avoir des serveurs DNS interne et externe. Vous pouvez combiner les deux dans un unique serveur BIND 9 utilisant les vues (Views). Si vous n'êtes pas intéressé par les vues BIND 9, vous pouvez allez à la section suivante.

Supposons que votre domaine est foobar.net et vous voulez que les deux systèmes DMZ s'appellent www.foobar.net et mail.foobar.net, les trois systèmes locaux "winken.foobar.net, blinken.foobar.net et nod.foobar.net. Vous voulez que le firewall soit connu à l'extérieur sous le nom firewall.foobar.net, son interface vers le réseau local gateway.foobar.net et son interface vers la DMZ dmz.foobar.net. Mettons le serveur DNS sur 192.0.2.177 qui sera aussi connu sous le nom ns1.foobar.net.

`Le fichier /etc/named.conf` devrait ressembler à cela:

```
options {
        directory "/var/named";
        listen-on { 127.0.0.1 ; 192.0.2.177; };
};

logging {
        channel xfer-log {
                file "/var/log/named/bind-xfer.log";
                print-category yes;
                print-severity yes;
                print-time yes;
                severity info;
};

        category xfer-in { xfer-log; };
        category xfer-out { xfer-log; };
        category notify { xfer-log; };
};

#
# This is the view presented to our internal systems
```

```
#

view "internal" {
        #
        # These are the clients that see this view
        #
        match-clients { 192.168.201.0/29;
                        192.168.202.0/29;
                        127.0.0.0/8;
                        192.0.2.176/32;
                        192.0.2.178/32;
                        192.0.2.179/32;
                        192.0.2.180/32; };
        #
        # If this server can't complete the request, it should use
        # outside servers to do so
        #
        recursion yes;

        zone "." in {
                type hint;
                file "int/root.cache";
        };

        zone "foobar.net" in {
                type master;
                notify no;
                allow-update { none; };
                file "int/db.foobar";
        };

        zone "0.0.127.in-addr.arpa" in {
                type master;
                notify no;
                allow-update { none; };
                file "int/db.127.0.0";
        };

        zone "201.168.192.in-addr.arpa" in {
                type master;
                notify no;
                allow-update { none; };
                file "int/db.192.168.201";
        };

        zone "202.168.192.in-addr.arpa" in {
                type master;
                notify no;
                allow-update { none; };
                file "int/db.192.168.202";
        };

        zone "176.2.0.192.in-addr.arpa" in {
                type master;
                notify no;
                allow-update { none; };
                file "db.192.0.2.176";
        };

        zone "177.2.0.192.in-addr.arpa" in {
                type master;
                notify no;
```

```
                allow-update { none; };
                file "db.192.0.2.177";
        };

        zone "178.2.0.192.in-addr.arpa" in {
                type master;
                notify no;
                allow-update { none; };
                file "db.192.0.2.178";
        };

        zone "179.2.0.192.in-addr.arpa" in {
                type master;
                notify no;
                allow-update { none; };
                file "db.206.124.146.179";
        };

};
#
# This is the view that we present to the outside world
#
view "external" {
         match-clients { any; };
         #
         # If we can't answer the query, we tell the client so
         #
         recursion no;

         zone "foobar.net" in {
                 type master;
                 notify yes;
                 allow-update {none; };
                 allow-transfer { <secondary NS IP>; };
                 file "ext/db.foobar";
         };

         zone "176.2.0.192.in-addr.arpa" in {
                 type master;
                 notify yes;
                 allow-update { none; };
                 allow-transfer { <secondary NS IP>; };
                 file "db.192.0.2.176";
         };

         zone "177.2.0.192.in-addr.arpa" in {
                 type master;
                 notify yes;
                 allow-update { none; };
                 allow-transfer { <secondary NS IP>; };
                 file "db.192.0.2.177";
         };

         zone "178.2.0.192.in-addr.arpa" in {
                 type master;
                 notify yes;
                 allow-update { none; };
                 allow-transfer { <secondary NS IP>; };
                 file "db.192.0.2.178";
         };

         zone "179.2.0.192.in-addr.arpa" in {
```

```
                type master;
                notify yes;
                allow-update { none; };
                allow-transfer { <secondary NS IP>; };
                file "db.192.0.2.179";
        };
};
```

Voici les fichiers de `/var/named` (ceux qui ne sont pas présents font partis de votre distribution BIND).

`db.192.0.2.176` - Zone inverse de l'interface externe du firewall

```
; ##############################################################
; Start of Authority (Inverse Address Arpa) for 192.0.2.176/32
; Filename: db.192.0.2.176
; ##############################################################
@ 604800 IN SOA ns1.foobar.net. netadmin.foobar.net. (
                        2001102303 ; serial
                        10800 ; refresh (3 hour)
                        3600 ; retry (1 hour)
                        604800 ; expire (7 days)
                        86400 ) ; minimum (1 day)
;
; ##############################################################
; Specify Name Servers for all Reverse Lookups (IN-ADDR.ARPA)
; ##############################################################
@        604800  IN NS   ns1.foobar.net.
@        604800  IN NS   <name of secondary ns>.
;
; ##############################################################
; Iverse Address Arpa Records (PTR's)
; ##############################################################
176.2.0.192.in-addr.arpa. 86400 IN PTR firewall.foobar.net.
```

`db.192.0.2.177` - Zone inverse pour le serveur www/DNS server

```
; ##############################################################
; Start of Authority (Inverse Address Arpa) for 192.0.2.177/32
; Filename: db.192.0.2.177
; ##############################################################
@ 604800 IN SOA ns1.foobar.net. netadmin.foobar.net. (
                        2001102303 ; serial
                        10800 ; refresh (3 hour)
                        3600 ; retry (1 hour)
                        604800 ; expire (7 days)
                        86400 ) ; minimum (1 day)
;
; ##############################################################
; Specify Name Servers for all Reverse Lookups (IN-ADDR.ARPA)
; ##############################################################
@        604800  IN NS   ns1.foobar.net.
@        604800  IN NS   <name of secondary ns>.
;
; ##############################################################
; Iverse Address Arpa Records (PTR's)
; ##############################################################
177.2.0.192.in-addr.arpa. 86400 IN PTR www.foobar.net.
```

`db.192.0.2.178` - Zone inverse du serveur mail

```
; ##############################################################
; Start of Authority (Inverse Address Arpa) for 192.0.2.178/32
; Filename: db.192.0.2.178
; ##############################################################
@ 604800 IN SOA ns1.foobar.net. netadmin.foobar.net. (
                      2001102303 ; serial
                      10800 ; refresh (3 hour)
                      3600 ; retry (1 hour)
                      604800 ; expire (7 days)
                      86400 ) ; minimum (1 day)
;
; ##############################################################
; Specify Name Servers for all Reverse Lookups (IN-ADDR.ARPA)
; ##############################################################
@        604800  IN NS   ns1.foobar.net.
@        604800  IN NS   <name of secondary ns>.
;
; ##############################################################
; Iverse Address Arpa Records (PTR's)
; ##############################################################
178.2.0.192.in-addr.arpa. 86400 IN PTR mail.foobar.net.
```

`db.192.0.2.179` - Zone inverse du serveur web publique de votre fille

```
; ##############################################################
; Start of Authority (Inverse Address Arpa) for 192.0.2.179/32
; Filename: db.192.0.2.179
; ##############################################################
@ 604800 IN SOA ns1.foobar.net. netadmin.foobar.net. (
                      2001102303 ; serial
                      10800 ; refresh (3 hour)
                      3600 ; retry (1 hour)
                      604800 ; expire (7 days)
                      86400 ) ; minimum (1 day)
;
; ##############################################################
; Specify Name Servers for all Reverse Lookups (IN-ADDR.ARPA)
; ##############################################################
@        604800  IN NS   ns1.foobar.net.
@        604800  IN NS   <name of secondary ns>.
;
; ##############################################################
; Iverse Address Arpa Records (PTR's)
; ##############################################################
179.2.0.192.in-addr.arpa. 86400 IN PTR nod.foobar.net.
```

`int/db.127.0.0` - Zone inverse pour localhost

```
; ##############################################################
; Start of Authority (Inverse Address Arpa) for 127.0.0.0/8
; Filename: db.127.0.0
; ##############################################################
@ 604800 IN SOA ns1.foobar.net. netadmin.foobar.net. (
                             2001092901 ; serial
                             10800 ; refresh (3 hour)
                             3600 ; retry (1 hour)
                             604800 ; expire (7 days)
```

```
                              86400 ) ; minimum (1 day)
; ##############################################################
; Specify Name Servers for all Reverse Lookups (IN-ADDR.ARPA)
; ##############################################################
@     604800          IN NS   ns1.foobar.net.

; ##############################################################
; Iverse Address Arpa Records (PTR's)
; ##############################################################
1       86400           IN PTR  localhost.foobar.net.
```

`int/db.192.168.201` - Zone inverse pour le réseau local. cela n'est montré qu'aux clients internes

```
; ##############################################################
; Start of Authority (Inverse Address Arpa) for 192.168.201.0/29
; Filename: db.192.168.201
; ##############################################################
@ 604800 IN SOA ns1.foobar.net netadmin.foobar.net. (
                                2002032501 ; serial
                                10800 ; refresh (3 hour)
                                3600 ; retry (1 hour)
                                604800 ; expire (7 days)
                                86400 ) ; minimum (1 day)

; ##############################################################
; Specify Name Servers for all Reverse Lookups (IN-ADDR.ARPA)
; ##############################################################
@       604800          IN NS   ns1.foobar.net.
; ##############################################################
; Iverse Address Arpa Records (PTR's)
; ##############################################################
1       86400           IN PTR  gateway.foobar.net.
2       86400           IN PTR  winken.foobar.net.
3       86400           IN PTR  blinken.foobar.net.
4       86400           IN PTR  nod.foobar.net.
```

`int/db.192.168.202` - Zone inverse de l'interface DMZ du firewall

```
; ##############################################################
; Start of Authority (Inverse Address Arpa) for 192.168.202.0/29
; Filename: db.192.168.202
; ##############################################################
@ 604800 IN SOA ns1.foobar.net netadmin.foobar.net. (
                                2002032501 ; serial
                                10800 ; refresh (3 hour)
                                3600 ; retry (1 hour)
                                604800 ; expire (7 days)
                                86400 ) ; minimum (1 day)

; ##############################################################
; Specify Name Servers for all Reverse Lookups (IN-ADDR.ARPA)
; ##############################################################
@             604800  IN NS   ns1.foobar.net.

; ##############################################################
; Iverse Address Arpa Records (PTR's)
; ##############################################################
1               86400 IN PTR  dmz.foobar.net.
```

`int/db.foobar` - Forward zone pour l'utilisation des clients internes.

```
;##############################################################
; Start of Authority for foobar.net.
; Filename: db.foobar
;##############################################################
@ 604800 IN SOA ns1.foobar.net. netadmin.foobar.net. (
                        2002071501 ; serial
                        10800 ; refresh (3 hour)
                        3600 ; retry (1 hour)
                        604800 ; expire (7 days)
                        86400 ); minimum (1 day)
;#############################################################
; foobar.net Nameserver Records (NS)
;#############################################################
@                 604800  IN NS   ns1.foobar.net.

;#############################################################
; Foobar.net Office Records (ADDRESS)
;#############################################################
localhost 86400   IN A    127.0.0.1

firewall       86400   IN A    192.0.2.176
www            86400   IN A    192.0.2.177
ns1            86400   IN A    192.0.2.177
www            86400   IN A    192.0.2.177

gateway              86400   IN A    192.168.201.1
winken               86400   IN A    192.168.201.2
blinken              86400   IN A    192.168.201.3
nod                  86400   IN A    192.168.201.4
```

`ext/db.foobar` - Forward zone pour les clients externes

```
;##############################################################
; Start of Authority for foobar.net.
; Filename: db.foobar
;##############################################################
@ 86400 IN SOA ns1.foobar.net. netadmin.foobar.net. (
                        2002052901 ; serial
                        10800 ; refresh (3 hour)
                        3600 ; retry (1 hour)
                        604800 ; expire (7 days)
                        86400 ); minimum (1 day)
;#############################################################
; Foobar.net Nameserver Records (NS)
;#############################################################
@                86400   IN NS   ns1.foobar.net.
@                86400   IN NS   <secondary NS>.
;#############################################################
; Foobar.net        Foobar Wa Office Records (ADDRESS)
;#############################################################
localhost         86400   IN A    127.0.0.1
;
; The firewall itself
;
firewall          86400   IN A    192.0.2.176
;
; The DMZ
;
```

```
ns1                 86400   IN A    192.0.2.177
www                 86400   IN A    192.0.2.177
mail                86400   IN A    192.0.2.178
;
; The Local Network
;
nod               86400   IN A    192.0.2.179

;###########################################################
; Current Aliases for foobar.net (CNAME)
;###########################################################

;###########################################################
; foobar.net MX Records (MAIL EXCHANGER)
;###########################################################
foobar.net.       86400   IN A    192.0.2.177
                  86400   IN MX 0 mail.foobar.net.
                  86400   IN MX 1 <backup MX>.
```

Quelques Points à Garder en Mémoire

- **Vous ne pouvez tester votre firewall de l'intérieur de votre réseau**. Car les requêtes que vous envoyez à votre adresse IP ne veux pas dire qu'elles seront associées à votre interface externe ou la zone “net”. Tout trafic généré par le réseau local sera traité par loc->fw.
- **Les adresses IP sont des propriétés des systèmes, pas des interfaces**. C'est une erreur de croire que votre firewall est capable de renvoyer des paquets simplement parce que vous pouvez faire un ping sur l'adresse IP de toutes les interfaces du firewall depuis le réseau local. La seul conclusion est de conclure que le lien entre le réseau local et le firewall est établi et que vous avez probablement la bonne adresse de la passerelle sur votre système.
- **Toutes les adresses IP configurées sur le firewall sont dans la zone $FW (fw)**. Si 192.168.1.254 est l'adresse IP de votre interface interne, alors vous pouvez écrire “**$FW:192.168.1.254**” dans une régle mais vous ne devez pas écrire “**loc:192.168.1.254**”. C'est aussi un non-sens d'ajouter 192.168.1.254 à la zone **loc** en utilisant une entrée dans `/etc/shorewall/hosts`.
- **Les paquets de retour (Reply) ne suivent PAS automatiquement le chemin inverse de la requête d'origine**. Tous les paquets sont routés en se référant à la table de routage respective de chaque hôte à chaque étape du trajet. C'est commun chez ceux qui installent le firewall Shorewall en parallèle à une passerelle existante et essayent d'utiliser DNAT dans Shorewall sans changer la passerelle par défaut sur les systèmes recevant le retour des requêtes. Les requêtes dont, à travers le firewall Shorewall, l'adresse de destination IP est réécrite mais la réponse va directement vers l'ancienne passerelle.
- **Shorewall lui-même n'a aucune notion du dedans et du dehors**. Ces concepts dépendent de la façon dont Shorewall est configuré.

Démarrer et Arrêter Votre Firewall

La procédure d'installation configure votre système pour lancer Shorewall au boot du système, mais au début avec la version 1.3.9 de Shorewall le lancement est désactivé, n'essayer pas de lancer Shorewall avant que la configuration soit finie. Une fois que vous en aurez fini avec la configuration du firewall, vous pouvez permettre le lancement de Shorewall en supprimant le fichier `/etc/shorewall/startup_disabled`.

Important

Les utilisateurs des paquets .deb doivent éditer `/etc/default/shorewall` et mettre `startup=1`.

Le firewall est activé en utilisant la commande “**shorewall start**” et arrêté avec “**shorewall stop**”. Lorsque le firewall est stoppé, le routage est autorisé sur les hôtes qui possèdent une entrée dans `/etc/shorewall/routestopped`. Un firewall qui tourne peut être relancé en utilisant la commande “**shorewall restart**” command. Si vous voulez enlever toutes traces de Shorewall sur votre configuration de Netfilter, utilisez “**shorewall clear**”.

Les exemples supposent que vous voulez permettre le routage depuis ou vers `eth1` (le réseau local) et `eth2` (DMZ) lorsque Shorewall est stoppé. Si ces deux interfaces ne sont pas connectées à votre réseau local et votre DMZ, ou si vous voulez permettre un ensemble d'hôtes différents, modifiez `/etc/shorewall/routestopped` en conséquence.

Warning

Si vous êtes connecté à votre firewall depuis Internet, n'essayez pas une commande “**shorewall stop**” tant que vous n'avez pas ajouté une entrée pour votre adresse IP (celle à partir de laquelle vous êtes connectée) dans `/etc/shorewall/routestopped`. De la même manière, je ne vous recommande pas d'utiliser “**shorewall restart**”; il est plus intéressant de créer une configuration alternative et de la tester en utilisant la commande “**shorewall try**”.

Three-Interface Firewall

Tom Eastep

Copyright © 2002-2005 Thomas M. Eastep

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-11-10

Table of Contents

Introduction
- Requirements
- Before you start
- Conventions

PPTP/ADSL
Shorewall Concepts
Network Interfaces
IP Addresses
IP Masquerading (SNAT)
Port Forwarding (DNAT)
Domain Name Server (DNS)
Other Connections
Some Things to Keep in Mind
Starting and Stopping Your Firewall
Additional Recommended Reading

Caution

This article applies to Shorewall 3.0 and later. If you are running a version of Shorewall earlier than Shorewall 3.0.0 then please see the documentation for that release.

Introduction

Setting up a Linux system as a firewall for a small network with DMZ is a fairly straight-forward task if you understand the basics and follow the documentation.

This guide doesn't attempt to acquaint you with all of the features of Shorewall. It rather focuses on what is required to configure Shorewall in one of its more popular configurations:

- Linux system used as a firewall/router for a small local network.
- Single public IP address.

  ### Note

  If you have more than one public IP address, this is not the guide you want -- see the Shorewall Setup Guide instead.

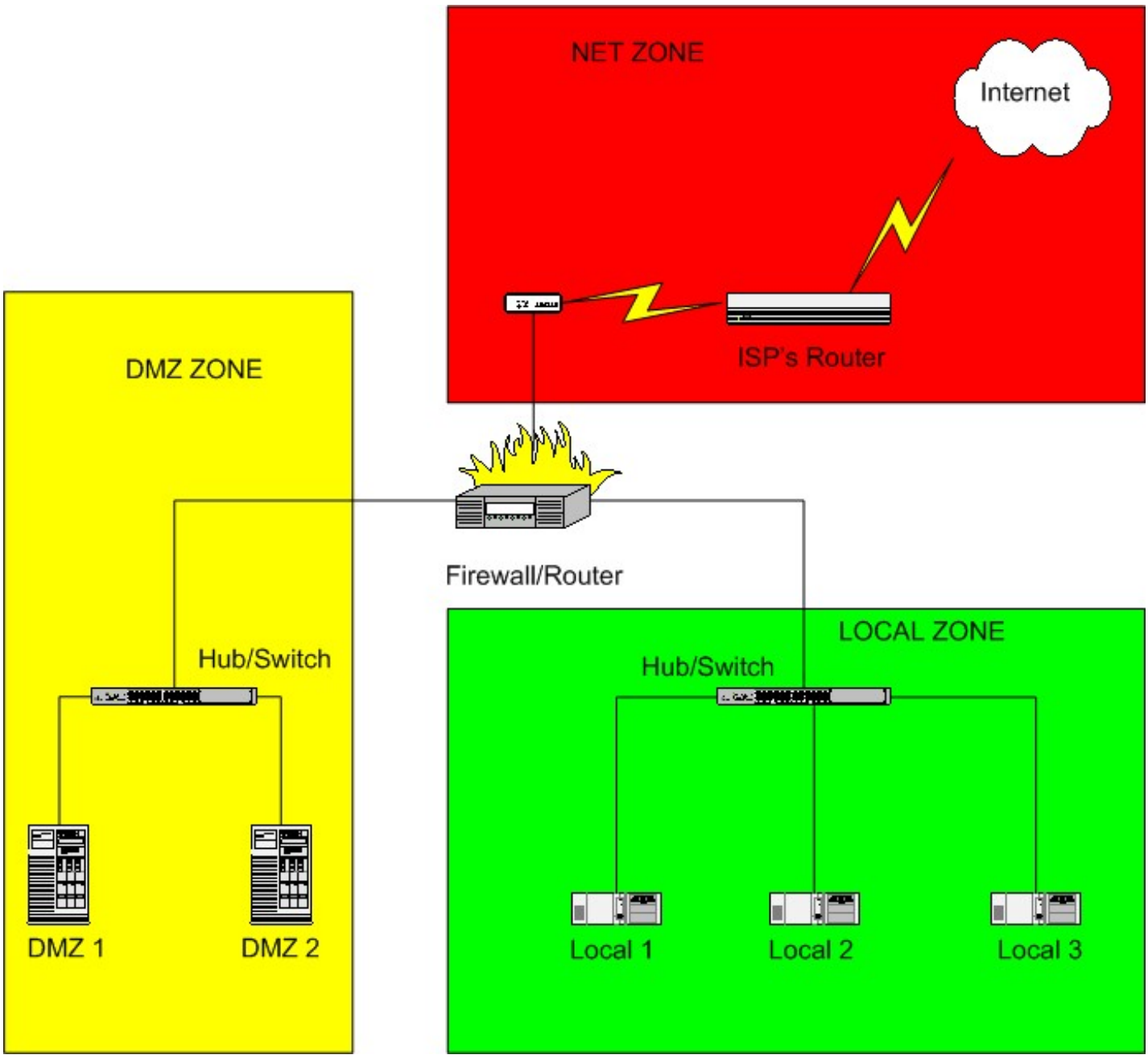
- DMZ connected to a separate ethernet interface. The purpose of a DMZ is to isolate those servers that are exposed to the

Internet from your local systems so that if one of those servers is compromised there is still a firewall between the hacked server and your local systems.

- Connection through DSL, Cable Modem, ISDN, Frame Relay, dial-up, ...

Here is a schematic of a typical installation.

Figure 1. schematic of a typical installation



Requirements

Shorewall requires that you have the **iproute**/**iproute2** package installed (on RedHat™, the package is called **iproute**). You can tell if this package is installed by the presence of an **ip** program on your firewall system. As `root`, you can use the **which** command to check for this program:

```
[root@gateway root]# which ip
/sbin/ip
[root@gateway root]#
```

Before you start

I recommend that you first read through the guide to familiarize yourself with what's involved then go back through it again making your configuration changes.

Caution

If you edit your configuration files on a Windows™ system, you must save them as Unix™ files if your editor supports that option or you must run them through **dos2unix** before trying to use them. Similarly, if you copy a configuration file from your Windows™ hard drive to a floppy disk, you must run **dos2unix** against the copy before using it with Shorewall.

- Windows Version of dos2unix
- Linux Version of dos2unix

Conventions

Points at which configuration changes are recommended are flagged with ➡.

Configuration notes that are unique to LEAF/Bering are marked with .

PPTP/ADSL

➡

If you have an ADSL Modem and you use PPTP to communicate with a server in that modem, you must make the changes recommended here in addition to those detailed below. ADSL with PPTP is most commonly found in Europe, notably in Austria.

Shorewall Concepts

The configuration files for Shorewall are contained in the directory `/etc/shorewall` -- for simple setups, you will only need to deal with a few of these as described in this guide.

Warning

Note to Debian Users

If you install using the .deb, you will find that your `/etc/shorewall` directory is empty. This is intentional. The released configuration file skeletons may be found on your system in the directory `/usr/share/doc/shorewall/default-config`. Simply copy the files you need from that directory to `/etc/shorewall` and modify the copies.

Note that you must copy `/usr/share/doc/shorewall/default-config/shorewall.conf` and /usr/share/doc/shorewall/default-config/modules to /etc/shorewall even if you do not modify those files.

➡

After you have installed Shorewall, locate the three-interface Sample configuration:

1. If you installed using an RPM, the samples will be in the Samples/three-interfaces/ subdirectory of the Shorewall documentation directory. If you don't know where the Shorewall documentation directory is, you can find the samples using this command:

```
~# rpm -ql shorewall | fgrep three-interfaces
/usr/share/doc/packages/shorewall/Samples/three-interfaces
/usr/share/doc/packages/shorewall/Samples/three-interfaces/interfaces
/usr/share/doc/packages/shorewall/Samples/three-interfaces/masq
```

```
/usr/share/doc/packages/shorewall/Samples/three-interfaces/policy
/usr/share/doc/packages/shorewall/Samples/three-interfaces/routestopped
/usr/share/doc/packages/shorewall/Samples/three-interfaces/rules
/usr/share/doc/packages/shorewall/Samples/three-interfaces/zones
~#
```

2. If you installed using the tarball, the samples are in the Samples/three-interfaces directory in the tarball.
3. If you installed using the .deb, the samples are in /usr/share/doc/shorewall/examples/three-interfaces.

As each file is introduced, I suggest that you look through the actual file on your system -- each file contains detailed configuration instructions and default entries.

Shorewall views the network where it is running as being composed of a set of zones. In the three-interface sample configuration, the following zone names are used:

```
#ZONE   TYPE   OPTIONS                 IN                      OUT
#                                      OPTIONS                 OPTIONS
fw      firewall
net     ipv4
loc     ipv4
dmz     ipv4
```

Zone names are defined in `/etc/shorewall/zones`.

Note that Shorewall recognizes the firewall system as its own zone. When the /etc/shorewall/zones file is processed, he name of the firewall zone is stored in the shell variable *$FW* which may be used throughout the Shorewall configuration to refer to the firewall zone.

Rules about what traffic to allow and what traffic to deny are expressed in terms of zones.

- You express your default policy for connections from one zone to another zone in the `/etc/shorewall/policy` file.
- You define exceptions to those default policies in the `/etc/shorewall/rules` file.

For each connection request entering the firewall, the request is first checked against the `/etc/shorewall/rules` file. If no rule in that file matches the connection request then the first policy in `/etc/shorewall/policy` that matches the request is applied. If there is a comon action defined for the policy in `/etc/shorewall/actions` or `/usr/share/shorewall/actions.std` then that action is peformed before the action is applied.

The `/etc/shorewall/policy` file included with the three-interface sample has the following policies:

```
#SOURCE    DEST        POLICY      LOG LEVEL    LIMIT:BURST
loc        net         ACCEPT
net        all         DROP        info
all        all         REJECT      info
```

Important

In the three-interface sample, the line below is included but commented out. If you want your firewall system to have full access to servers on the internet, uncomment that line.

```
#SOURCE    DEST        POLICY      LOG LEVEL    LIMIT:BURST
$FW        net         ACCEPT
```

The above policy will:

1. allow all connection requests from your local network to the internet
2. drop (ignore) all connection requests from the internet to your firewall or local network
3. optionally accept all connection requests from the firewall to the internet (if you uncomment the additional policy)

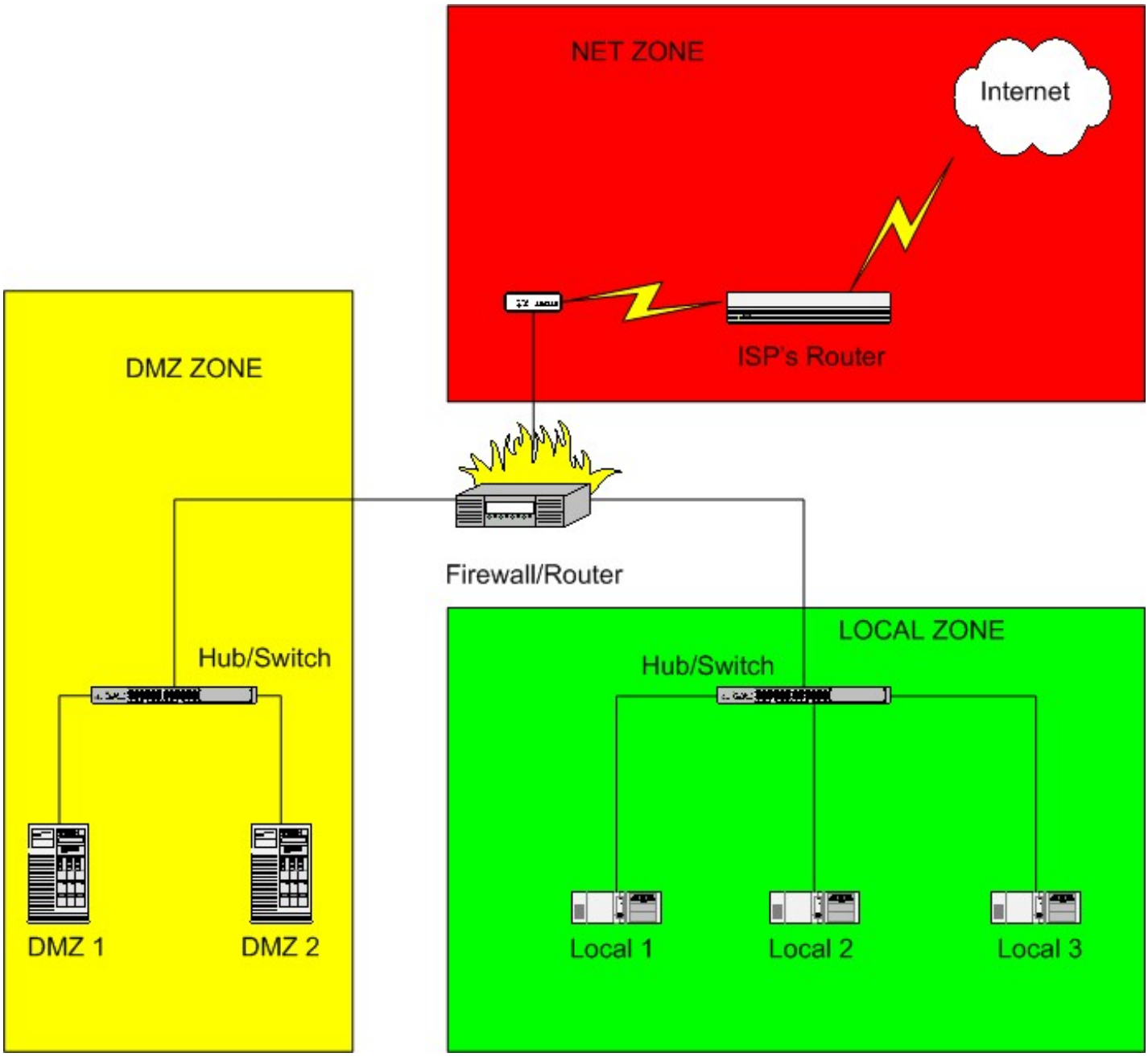
4. reject all other connection requests.

It is important to note that Shorewall policies (and rules) refer to **connections** and not packet flow. With the policies defined in the `/etc/shorewall/policy` file shown above, connections are allowed from the *loc* zone to the *net* zone even though connections are not allowed from the *loc* zone to the firewall itself.

At this point, edit your `/etc/shorewall/policy` file and make any changes that you wish.

Network Interfaces

Figure 2. DMZ



The firewall has three network interfaces. Where Internet connectivity is through a cable or DSL "Modem", the External Interface will be the ethernet adapter that is connected to that "Modem" (e.g., `eth0`) unless you connect via *Point-to-Point Protocol* over Ethernet (PPPoE) or *Point-to-Point Tunneling Protocol* (PPTP) in which case the External Interface will be a `ppp` interface (e.g., `ppp0`). If you connect via a regular modem, your External Interface will also be `ppp0`. If you connect using ISDN, you external interface will be `ipppp0`.

If your external interface is `ppp0` or `ipppp0` then you will want to set `CLAMPMSS=yes` in `/etc/shorewall/shorewall.conf`.

Your Local Interface will be an ethernet adapter (`eth0`, `eth1` or `eth2`) and will be connected to a hub or switch. Your local computers will be connected to the same switch (note: If you have only a single local system, you can connect the firewall directly to the computer using a cross-over cable).

Your DMZ Interface will also be an ethernet adapter (`eth0`, `eth1` or `eth2`) and will be connected to a hub or switch. Your DMZ computers will be connected to the same switch (note: If you have only a single DMZ system, you can connect the firewall directly to the computer using a cross-over cable).

Caution

Do NOT connect the internal and external interface to the same hub or switch except for testing. You can test using this kind of configuration if you specify the arp_filter option in `/etc/shorewall/interfaces` for all interfaces connected to the common hub/switch. **Using such a setup with a production firewall is strongly recommended against**.

The Shorewall three-interface sample configuration assumes that the external interface is `eth0`, the local interface is `eth1` and the DMZ interface is `eth2`. If your configuration is different, you will have to modify the sample `/etc/shorewall/interfaces` file accordingly. While you are there, you may wish to review the list of options that are specified for the interfaces. Some hints:

Tip

If your external interface is `ppp0` or `ipppp0`, you can replace the “detect” in the second column with “-” (without the quotes).

Tip

If your external interface is `ppp0` or `ipppp0` or if you have a static IP address, you can remove “dhcp” from the option list.

IP Addresses

Before going further, we should say a few words about Internet Protocol (IP) addresses. Normally, your ISP will assign you a single Public IP address. This address may be assigned via the Dynamic Host Configuration Protocol (DHCP) or as part of establishing your connection when you dial in (standard modem) or establish your PPP connection. In rare cases, your ISP may assign you a static IP address; that means that you configure your firewall's external interface to use that address permanently. Regardless of how the address is assigned, it will be shared by all of your systems when you access the Internet. You will have to assign your own addresses for your internal network (the local and DMZ Interfaces on your firewall plus your other computers). RFC 1918 reserves several Private IP address ranges for this purpose:

```
10.0.0.0    - 10.255.255.255
172.16.0.0  - 172.31.255.255
192.168.0.0 - 192.168.255.255
```

Before starting Shorewall, **you should look at the IP address of your external interface and if it is one of the above ranges, you should remove the `norfc1918` option from the external interface's entry in `/etc/shorewall/interfaces`.**

You will want to assign your local addresses from one sub-network or subnet and your DMZ addresses from another subnet. For our purposes, we can consider a subnet to consists of a range of addresses `x.y.z.0 - x.y.z.255`. Such a subnet will have a Subnet Mask of `255.255.255.0`. The address `x.y.z.0` is reserved as the Subnet Address and `x.y.z.255` is reserved as the Subnet Broadcast Address. In Shorewall, a subnet is described using Classless InterDomain Routing (CIDR) notation with consists of the

subnet address followed by `/24`. The `24` refers to the number of consecutive “1” bits from the left of the subnet mask.

Table 1. Example sub-network

Range:	`10.10.10.0 - 10.10.10.255`
Subnet Address:	`10.10.10.0`
Broadcast Address:	`10.10.10.255`
CIDR Notation:	`10.10.10.0/24`

It is conventional to assign the internal interface either the first usable address in the subnet (`10.10.10.1` in the above example) or the last usable address (`10.10.10.254`).

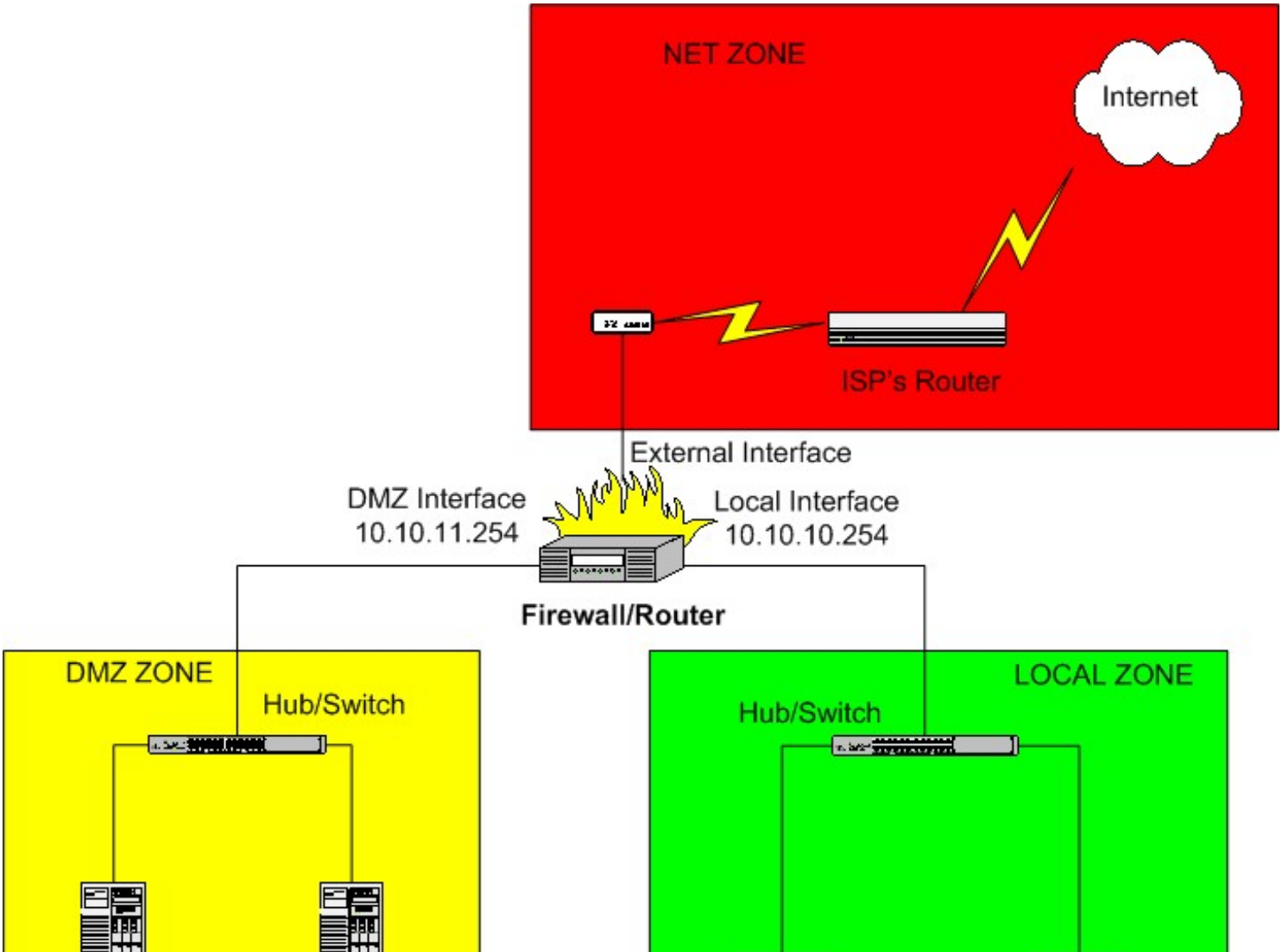
One of the purposes of subnetting is to allow all computers in the subnet to understand which other computers can be communicated with directly. To communicate with systems outside of the subnetwork, systems send packets through a gateway (router).

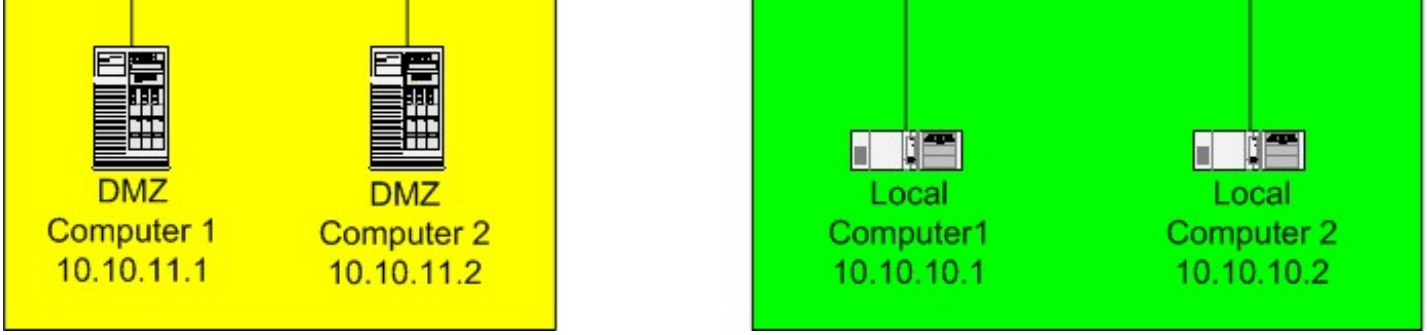
Your local computers (Local Computers 1 & 2) should be configured with their default gateway set to the IP address of the firewall's internal interface and your DMZ computers (DMZ Computers 1 & 2) should be configured with their default gateway set to the IP address of the firewall's DMZ interface.

The foregoing short discussion barely scratches the surface regarding subnetting and routing. If you are interested in learning more about IP addressing and routing, I highly recommend “IP Fundamentals: What Everyone Needs to Know about Addressing & Routing”, Thomas A. Maufer, Prentice-Hall, 1999, ISBN 0-13-975483-0.

The remainder of this quide will assume that you have configured your network as shown here:

Figure 3. DMZ





The default gateway for the DMZ computers would be `10.10.11.254` and the default gateway for the Local computers would be `10.10.10.254`.

Warning

Your ISP might assign your external interface an RFC 1918 address. If that address is in the `10.10.10.0/24` subnet then you will need to select a DIFFERENT RFC 1918 subnet for your local network and if it is in the `10.10.11.0/24` subnet then you will need to select a different RFC 1918 subnet for your DMZ.

IP Masquerading (SNAT)

The addresses reserved by RFC 1918 are sometimes referred to as non-routable because the Internet backbone routers don't forward packets which have an RFC-1918 destination address. When one of your local systems (let's assume local computer 1) sends a connection request to an internet host, the firewall must perform Network Address Translation (NAT). The firewall rewrites the source address in the packet to be the address of the firewall's external interface; in other words, the firewall makes it look as if the firewall itself is initiating the connection. This is necessary so that the destination host will be able to route return packets back to the firewall (remember that packets whose destination address is reserved by RFC 1918 can't be routed accross the internet). When the firewall receives a return packet, it rewrites the destination address back to 10.10.10.1 and forwards the packet on to local computer 1.

On Linux systems, the above process is often referred to as IP Masquerading and you will also see the term Source Network Address Translation (SNAT) used. Shorewall follows the convention used with Netfilter:

- *Masquerade* describes the case where you let your firewall system automatically detect the external interface address.
- *SNAT* refers to the case when you explicitly specify the source address that you want outbound packets from your local network to use.

In Shorewall, both Masquerading and SNAT are configured with entries in the `/etc/shorewall/masq` file.

If your external firewall interface is `eth0`, your local interface `eth1` and your DMZ interface is `eth2` then you do not need to modify the file provided with the sample. Otherwise, edit `/etc/shorewall/masq` and change it to match your configuration.

If, in spite of all advice to the contrary, you are using this guide and want to use one-to-one NAT or Proxy ARP for your DMZ, remove the entry for eth2 from `/etc/shorewall/masq`.

If your external IP is static, you can enter it in the third column in the `/etc/shorewall/masq` entry if you like although your firewall will work fine if you leave that column empty. Entering your static IP in column 3 makes processing outgoing packets a little more efficient.

If you are using the Debian package, please check your `shorewall.conf` file to ensure that the following is set correctly; if it is not, change it appropriately:

- `IP_FORWARDING=On`

Port Forwarding (DNAT)

One of your goals will be to run one or more servers on your DMZ computers. Because these computers have RFC-1918 addresses, it is not possible for clients on the Internet to connect directly to them. It is rather necessary for those clients to address their connection requests to your firewall who rewrites the destination address to the address of your server and forwards the packet to that server. When your server responds, the firewall automatically performs SNAT to rewrite the source address in the response.

The above process is called *Port Forwarding* or *Destination Network Address Translation* (DNAT). You configure port forwarding using DNAT rules in the `/etc/shorewall/rules` file.

The general form of a simple port forwarding rule in `/etc/shorewall/rules` is:

```
#ACTION   SOURCE    DEST                                          PROTO      DEST
PORT(S)
DNAT      net       dmz:<server local IP address>[:<server port>] <protocol> <port>
```

If you don't specify the *<server port>*, it is assumed to be the same as *<port>*.

Example 1. You run a Web Server on DMZ Computer 2 and you want to forward incoming TCP port 80 to that system

```
#ACTION     SOURCE    DEST                PROTO      DEST PORT(S)
Web/DNAT     net    dmz:10.10.11.2
Web/ACCEPT   loc    dmz:10.10.11.2
```

- Entry 1 forwards port 80 from the Internet.
- Entry 2 allows connections from the local network.

Several important points to keep in mind:

- When you are connecting to your server from your local systems, you must use the server's internal IP address (`10.10.11.2`).
- Many ISPs block incoming connection requests to port 80. If you have problems connecting to your web server, try the following rule and try connecting to port 5000 (e.g., connect to `http://w.x.y.z:5000 where w.x.y.z` is your external IP).

  ```
  #ACTION   SOURCE    DEST                PROTO      DEST PORT(S)    SOURCE
  #                                                                  PORT(S)
  DNAT      net       dmz:10.10.11.2:80   tcp        80              5000
  ```
- If you want to be able to access your server from the local network using your external address, then if you have a static external IP you can replace the loc->dmz rule above with:

  ```
  #ACTION   SOURCE    DEST            PROTO  DEST PORT(S)  SOURCE   ORIGINAL
  #                                                        PORT(S)  DEST
  DNAT      loc       dmz:10.10.11.2  tcp    80            -        <external IP>
  ```

  If you have a dynamic IP then you must ensure that your external interface is up before starting Shorewall and you must take steps as follows (assume that your external interface is `eth0`):
  1. Include the following in /etc/shorewall/params:

     ETH0_IP=$(find_interface_address eth0)
  2. Make your `loc->dmz` rule:

     ```
     #ACTION   SOURCE    DEST             PROTO   DEST PORT(S)  SOURCE   ORIGINAL
     #                                                          PORT(S)  DEST
     DNAT      loc       dmz:10.10.11.2   tcp     80            -        $ETH0_IP
     ```
- If you want to access your server from the DMZ using your external IP address, see FAQ 2a.

At this point, add the DNAT and ACCEPT rules for your servers.

Important

When testing DNAT rules like those shown above, you must test from a client OUTSIDE YOUR FIREWALL (in the 'net' zone). You cannot test these rules from inside the firewall!

For DNAT troubleshooting tips, see FAQs 1a and 1b.

Domain Name Server (DNS)

Normally, when you connect to your ISP, as part of getting an IP address your firewall's *Domain Name Service* (DNS) resolver will be automatically configured (e.g., the `/etc/resolv.conf` file will be written). Alternatively, your ISP may have given you the IP address of a pair of DNS name servers for you to manually configure as your primary and secondary name servers. It is your responsibility to configure the resolver in your internal systems. You can take one of two approaches:

- You can configure your internal systems to use your ISP's name servers. If your ISP gave you the addresses of their servers or if those addresses are available on their web site, you can configure your internal systems to use those addresses. If that information isn't available, look in `/etc/resolv.conf` on your firewall system -- the name servers are given in “nameserver” records in that file.
- You can configure a *Caching Name Server* on your firewall or in your DMZ. Red Hat™ has an RPM for a caching name server (which also requires the '**bind**' RPM) and for Bering users, there is `dnscache.lrp`. If you take this approach, you configure your internal systems to use the caching name server as their primary (and only) name server. You use the internal IP address of the firewall (`10.10.10.254` in the example above) for the name server address if you choose to run the name server on your firewall. To allow your local systems to talk to your caching name server, you must open port 53 (both UDP and TCP) from the local network to the server; you do that by adding the rules in `/etc/shorewall/rules`.

If you run the name server on the firewall:

```
#ACTION      SOURCE     DEST                    PROTO       DEST PORT(S)
DNS/ACCEPT   loc        $FW
DNS/ACCEPT   dmz        $FW
```

Run name server on DMZ computer 1:

```
#ACTION      SOURCE     DEST                    PROTO       DEST PORT(S)
DNS/ACCEPT   loc        dmz:10.10.11.1
DNS/ACCEPT   $FW        dmz:10.10.11.1
```

In the rules shown above, “DNS/ACCEPT” is an example of a *defined macro*. Shorewall includes a number of defined macros and you can add your own. To see the list of macros included with your version of Shorewall, run the command **ls `/usr/share/shorewall/macro.*`**.

You don't have to use defined macros when coding a rule in `/etc/shorewall/rules`. The first example above (name server on the firewall) could also have been coded as follows:

```
#ACTION    SOURCE     DEST                    PROTO       DEST PORT(S)
ACCEPT     loc        $FW                     tcp         53
ACCEPT     loc        $FW                     udp         53
ACCEPT     dmz        $FW                     tcp         53
ACCEPT     dmz        $FW                     udp         53
```

In cases where Shorewall doesn't include a defined macro to meet your needs, you can either define the macro yourself or you can simply code the appropriate rules directly. This page can be of help if you don't know the protocol and port involved.

Other Connections

The three-interface sample includes the following rule:

```
#ACTION     SOURCE    DEST                PROTO      DEST PORT(S)
DNS/ACCEPT  $FW       net
```

That rule allow DNS access from your firewall and may be removed if you commented out the line in `/etc/shorewall/policy` allowing all connections from the firewall to the Internet.

The sample also includes:

```
#ACTION     SOURCE    DEST                PROTO      DEST PORT(S)
SSH/ACCEPT  loc       $FW
SSH/ACCEPT  loc       dmz
```

Those rules allow you to run an SSH server on your firewall and in each of your DMZ systems and to connect to those servers from your local systems.

If you wish to enable other connections between your systems, the general format for using a defined macro is:

```
#ACTION          SOURCE        DEST                PROTO      DEST PORT(S)
<macro>/ACCEPT   <source zone> <destination zone>
```

The general format when not using a defined action is:

```
#ACTION   SOURCE        DEST                PROTO      DEST PORT(S)
ACCEPT    <source zone> <destination zone>  <protocol> <port>
```

Example 2. You want to run a publicly-available DNS server on your firewall system

Using defined macros:

```
#ACTION     SOURCE    DEST                PROTO      DEST PORT(S)
DNS/ACCEPT  net       $FW
```

Not using defined actions:

```
#ACTION   SOURCE    DEST                PROTO      DEST PORT(S)
ACCEPT    net       $FW                 tcp        53
ACCEPT    net       $FW                 udp        53
```

Those rules would of course be in addition to the rules listed above under "If you run the name server on your firewall".

If you don't know what port and protocol a particular application uses, look here.

Important

I don't recommend enabling telnet to/from the Internet because it uses clear text (even for login!). If you want shell access to your firewall from the Internet, use SSH:

```
#ACTION      SOURCE     DEST                PROTO      DEST PORT(S)
SSH/ACCEPT   net        $FW
```

Bering users will want to add the following two rules to be compatible with Jacques's Shorewall configuration:

```
#ACTION    SOURCE     DEST                PROTO      DEST PORT(S)
ACCEPT     loc        $FW                 udp        53
ACCEPT     net        $FW                 tcp        80
```

- Entry 1 allows the DNS Cache to be used.
- Entry 2 allows the “weblet” to work.

Now modify `/etc/shorewall/rules` to add or remove other connections as required.

Some Things to Keep in Mind

- **You cannot test your firewall from the inside**. Just because you send requests to your firewall external IP address does not mean that the request will be associated with the external interface or the “net” zone. Any traffic that you generate from the local network will be associated with your local interface and will be treated as loc->fw traffic.
- **IP addresses are properties of systems, not of interfaces**. It is a mistake to believe that your firewall is able to forward packets just because you can ping the IP address of all of the firewall's interfaces from the local network. The only conclusion you can draw from such pinging success is that the link between the local system and the firewall works and that you probably have the local system's default gateway set correctly.
- **All IP addresses configured on firewall interfaces are in the $FW (fw) zone**. If 192.168.1.254 is the IP address of your internal interface then you can write “**$FW:192.168.1.254**” in a rule but you may not write “**loc:192.168.1.254**”. Similarly, it is nonsensical to add 192.168.1.254 to the **loc** zone using an entry in `/etc/shorewall/hosts`.
- **Reply packets do NOT automatically follow the reverse path of the one taken by the original request**. All packets are routed according to the routing table of the host at each step of the way. This issue commonly comes up when people install a Shorewall firewall parallel to an existing gateway and try to use DNAT through Shorewall without changing the default gateway of the system receiving the forwarded requests. Requests come in through the Shorewall firewall where the destination IP address gets rewritten but replies go out unmodified through the old gateway.
- **Shorewall itself has no notion of inside or outside**. These concepts are embodied in how Shorewall is configured.

Starting and Stopping Your Firewall

The installation procedure configures your system to start Shorewall at system boot but startup is disabled so that your system won't try to start Shorewall before configuration is complete. Once you have completed configuration of your firewall, you can enable Shorewall startup by removing the file `/etc/shorewall/startup_disabled`.

Important

Users of the `.deb` package must edit `/etc/default/shorewall` and set `startup=1`.

Important

You should edit `/etc/shorewall/shorewall.conf` and set STARTUP_ENABLED=Yes.

The firewall is started using the **shorewall start** command and stopped using **shorewall stop**. When the firewall is stopped, routing is enabled on those hosts that have an entry in `/etc/shorewall/routestopped`. A running firewall may be restarted using the **shorewall restart** command. If you want to totally remove any trace of Shorewall from your Netfilter configuration, use **shorewall clear**.

The three-interface sample assumes that you want to enable routing to/from `eth1` (your local network) and `eth2` (DMZ) when Shorewall is stopped. If these two interfaces don't connect to your local network and DMZ or if you want to enable a different set of hosts, modify `/etc/shorewall/routestopped` accordingly.

Warning

If you are connected to your firewall from the Internet, do not issue a **shorewall stop** command unless you have added an entry for the IP address that you are connected from to `/etc/shorewall/routestopped`. Also, I don't recommend using **shorewall restart**; it is better to create an alternate configuration and test it using the **shorewall try** command.

Additional Recommended Reading

I highly recommend that you review the Common Configuration File Features page -- it contains helpful tips about Shorewall features than make administering your firewall easier.

Three-Interface Firewall

Tom Eastep

Patrice Vetsel

Fabien Demassieux

Copyright © 2002-2005 Thomas M. Eastep, Patrice Vetsel, Fabien Demassieux

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

2005-01-17

Table of Contents

Introduction
- Pré-requis
- Avant de commencer
- Conventions

PPTP/ADSL
Les Concepts de Shorewall
Les Interfaces Réseau
Adresses IP
IP Masquerading (SNAT)
Port Forwarding (DNAT)
Domain Name Server (DNS)
Autres Connexions
Quelques Points à Garder en Mémoire
Démarrer et Arrêter Votre Firewall
Autres Lectures Recommandées

Note

Notes du traducteur : Le guide initial a été traduit par VETSEL Patrice que je remercie. J'en ai assuré la révision pour l'adapter à la version 2 de Shorewall. J'espère vous faciliter l'accès et la prise en main d'un firewall performant, efficace, adaptable et facile d'utilisation. Donc félicitations pour la qualité du travail et la disponibilité offerte par Thomas M. Eastep. Si vous trouvez des erreurs ou des améliorations à apporter vous pouvez me contacter Fabien Demassieux

Introduction

Mettre en place un système Linux en tant que firewall pour un petit réseau contenant une DMZ est une chose assez simple, si vous comprenez les bases et suivez la documentation.

Ce guide ne veut pas vous apprendre tous les rouages de Shorewall. Il se focalise sur ce qui est nécessaire pour configurer Shorewall, dans son utilisation la plus courante :

- Un système Linux utilisé en tant que firewall/routeur pour un petit réseau local.
- Une seule adresse IP publique.

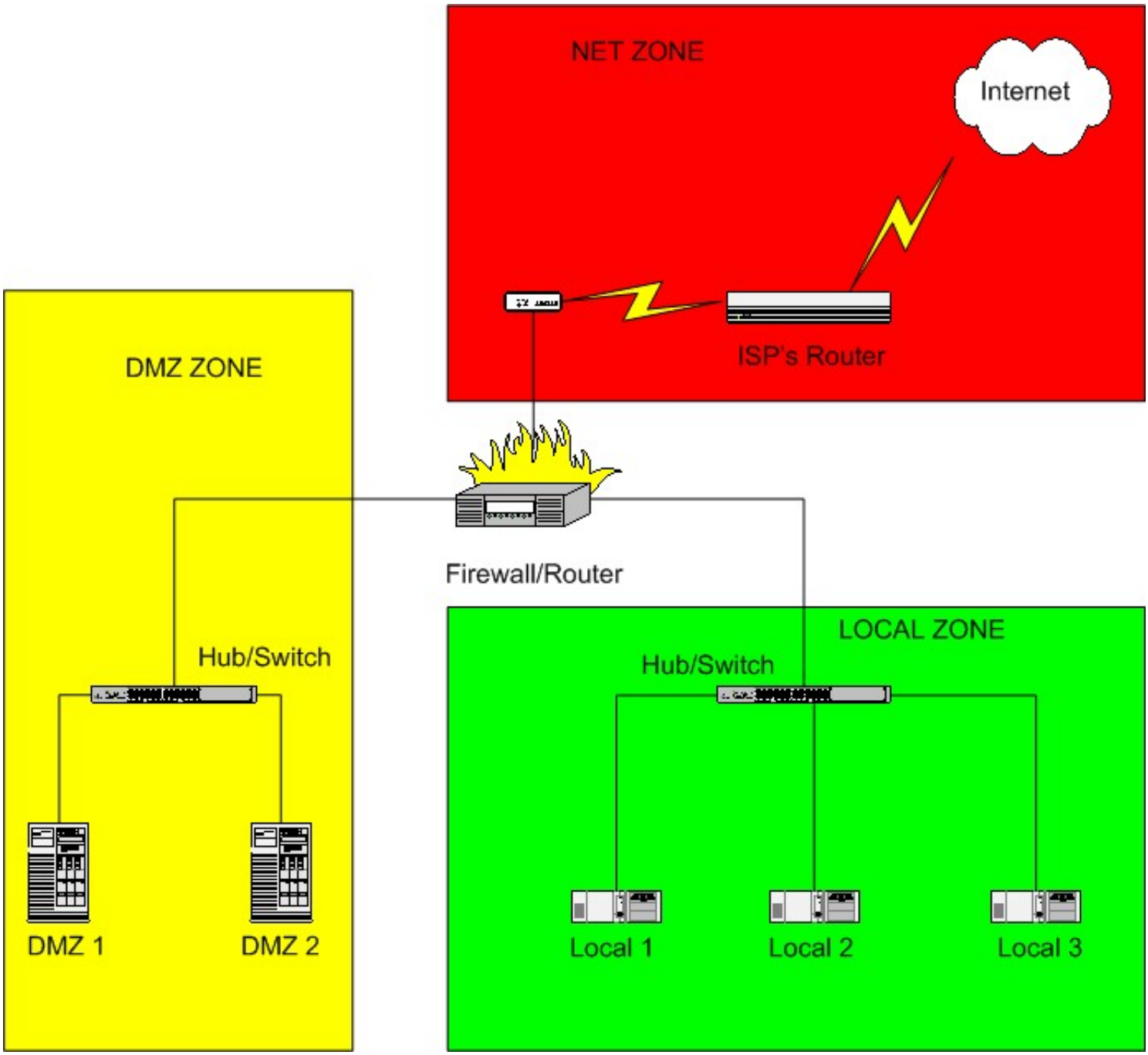
Note

Si vous avez plus d'une adresse IP, ce n'est pas le guide qui vous convient -- regardez plutôt du coté du Guide de Configuration Shorewall.

- Une DMZ (Zone démilitarisée) connectée sur une interface Ethernet séparée.
- Une connexion Internet par le biais d'un modem câble, ADSL, ISDN, "Frame Relay", RTC ...

Voici un schéma d'une installation typique.

Figure 1. Schéma d'une installation typique



Pré-requis

Shorewall a besoin que le package **iproute**/**iproute2** soit installé (avec la distribution RedHat™, le package s'appelle **iproute**).

Vous pouvez vérifier si le package est installé par la présence du programme **ip** sur votre firewall. En tant que `root`, vous pouvez utiliser la commande **which** pour cela:

```
[root@gateway root]# which ip
/sbin/ip
[root@gateway root]#
```

Avant de commencer

Je recommande en premier la lecture complète du guide afin de se familiariser avec les tenants et aboutissants puis de revenir sur les modifications de votre configuration adapté à votre système.

Caution

Si vous éditez vos fichiers de configuration sur un système Windows™, vous devez les sauver comme des fichiers Unix™ si votre éditeur supporte cette option sinon vous devez les convertir avec **dos2unix** avant d'essayer de les utiliser. De la même manière, si vous copiez un fichier de configuration depuis votre disque dur Windows™ vers une disquette, vous devez lancer **dos2unix** sur la copie avant de l'utiliser avec Shorewall.

- Windows™ Version of **dos2unix**
- Linux Version of **dos2unix**

Conventions

Les points ou les modifications s'imposent sont indiqués par ➡.

Les notes de configuration qui sont propres à LEAF/Bering sont marqués avec .

PPTP/ADSL

➡

Si vous êtes équipé d'un modem ADSL et utilisez PPTP pour communiquer avec un serveur à travers ce modem, vous devez faire le changement suivant en plus de ceux ci-dessous. ADSL avec PPTP est commun en Europe, ainsi qu'en Australie.

Les Concepts de Shorewall

➡

Les fichiers de configuration pour Shorewall sont situés dans le répertoire /etc/shorewall -- pour de simples paramétrages, vous n'avez à faire qu'avec quelques un d'entre eux comme décris dans ce guide.

Tip

Après avoir installé Shorewall, téléchargez l'exemple three-interface, décompressez le (**tar `-zxvf three-interfaces.tgz`**) et copiez les fichiers dans `/etc/shorewall` **(ces fichiers remplaceront les initiaux)**.

Parallèlement à la présentation, je vous suggère de jeter un oeil à ceux physiquement présents sur votre système -- chacun des fichiers contient des instructions de configuration détaillées et des entrées par défaut.

Shorewall voit le réseau où il fonctionne, comme un ensemble de zones. Dans une configuration avec trois interfaces, les noms

des zones suivantes sont utilisés:

Name	Description
net	The Internet
loc	Your Local Network
dmz	Demilitarized Zone

Les zones de Shorewall sont définies dans le fichier `/etc/shorewall/zones`.

Shorewall reconnaît aussi le système de firewall comme sa propre zone - par défaut, le firewall est connu comme **`fw`**.

Les règles à propos du trafic à autoriser et à interdire sont exprimées en terme de zones.

- Vous exprimez votre politique par défaut pour les connexions d'une zone vers une autre zone dans le fichier `/etc/shorewall/policy`.
- Vous définissez les exceptions à ces politiques pas défaut dans le fichier `/etc/shorewall/rules`.

Pour chaque connexion demandant à entrer dans le firewall, la requête est en premier lieu comparée par rapport au fichier `/etc/shorewall/rules`. Si aucune règle dans ce fichier ne correspond à la demande de connexion alors la première politique dans le fichier `/etc/shorewall/policy` qui y correspond sera appliquée. Si cette politique est `REJECT` ou `DROP` la requête est dans un premier temps comparée par rapport aux règles contenues dans le fichier `/etc/shorewall/common`, si ce fichier existe; sinon les régles dans le fichier `/etc/shorewall/common.def` sont vérifiées.

Le fichier /etc/shorewall/policy inclus dans l'archive d'exemple (three-interface) contient les politiques suivantes:

```
#SOURCE     DEST        POLICY      LOG LEVEL    LIMIT:BURST
loc         net         ACCEPT
net         all         DROP        info
all         all         REJECT      info
```

Important

Dans le fichier d'exemple (three-interface), la ligne suivante est incluse mais elle est commentée. Si vous voulez que votre firewall puisse avoir un accès complet aux serveurs sur Internet, décommentez la ligne.

```
#SOURCE     DEST        POLICY      LOG LEVEL    LIMIT:BURST
fw          net         ACCEPT
```

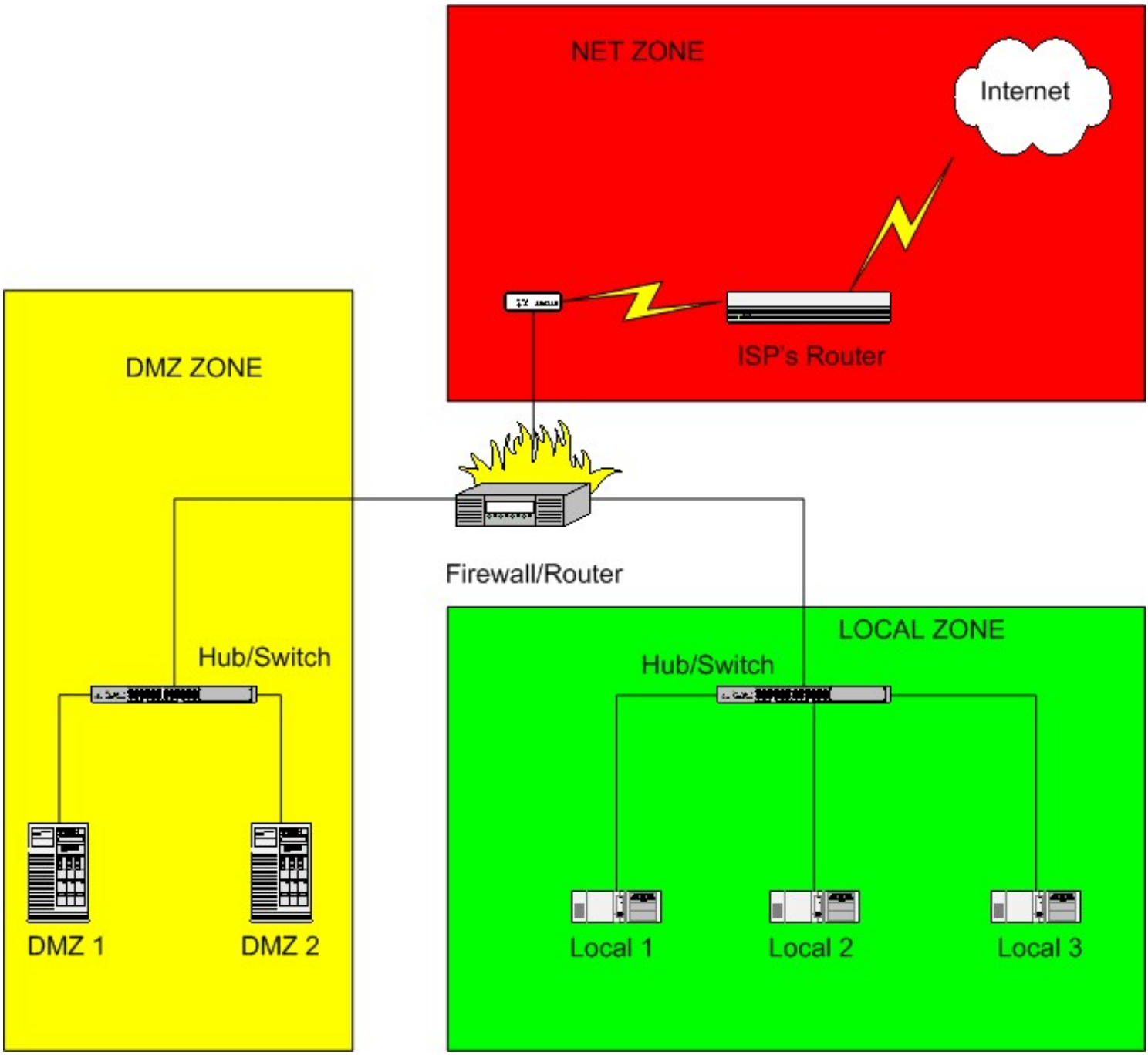
Les politiques précédentes vont:

- Permettre toutes demandes de connexion depuis votre réseau local vers Internet
- Drop (ignorer) toutes les demandes de connexion depuis l'Internet vers votre firewall ou votre réseau local
- Accept (accepter) facultativement toutes les demandes de connexion de votre firewall vers l'Internet (si vous avez décommenté la politique additionnelle)
- Reject (rejeter) toutes les autres requêtes de connexion.

Maintenant, editez votre propre fichier `/etc/shorewall/policy` et apportez les modifications et ajouter ce que vous voulez.

Les Interfaces Réseau

Figure 2. DMZ



Le firewall a trois interfaces de réseau. Lorsque la connexion Internet passe par le câble ou par un ROUTEUR (pas un simple modem) ADSL (non USB) “Modem”, l'interface vers l'extérieur (External Interface) sera l'adaptateur sur lequel est connecté le routeur “Modem” (e.g., `eth0`) à moins que vous ne vous connectiez par *Point-to-Point Protocol* over Ethernet (PPPoE) ou par *Point-to-Point Tunneling Protocol* (PPTP),dans ce cas l'interface extérieure sera une interface de type ppp (e.g., `ppp0`). Si vous vous connectez par un simple modem (RTC), votre interface extérieure sera aussi ppp0. Si votre connexion passe par Numéris (ISDN), votre interface extérieure sera `ipppp0`.

Si votre interface vers l'extérieur est `ppp0` ou `ipppp0` alors vous mettrez CLAMPMSS=yes dans le fichier `/etc/shorewall/shorewall.conf`.

Votre *Interface locale* sera un adaptateur Ethernet (`eth0`, `eth1` or `eth2`) et sera connecté à un hub ou un switch. Vos ordinateurs locaux seront connectés à ce même switch (note : si vous n'avez qu'un seul ordinateur en local, vous pouvez le connecter directement au firewall par un câble croisé).

Votre *interface DMZ* sera aussi un adaptateur Ethernet (`eth0`, `eth1` or `eth2`) et sera connecté à un hub ou un switch. Vos ordinateurs appartenant à la DMZ seront connectés à ce même switch (note : si vous n'avez qu'un seul ordinateur dans la DMZ, vous pouvez le connecter directement au firewall par un câble croisé).

Caution

Ne connectez pas l'interface interne et externe sur le même hub ou switch, sauf pour tester avec une version postérieure à Shorewall 1.4.7. Quand vous utilisez ces versions récentes, vous pouvez tester ce type de configuration si vous spécifiez l'option arp_filter dans le fichier `/etc/shorewall/interfaces` pour toutes les interfaces connectées au hub/switch commun. Utiliser une telle configuration avec un firewall en production est fortement déconseillé.

L'exemple de configuration de Shorewall pour trois interfaces suppose que l'interface externe est `eth0`, l'interface locale est `eth1` et que la DMZ est sur l'interface `eth2`. Si votre configuration diffère, vous devrez modifier le fichier d'exemple `/etc/shorewall/interfaces` en conséquence. Tant que vous y êtes, vous pourriez parcourir la liste des options qui sont spécifiées pour les interfaces. Quelques trucs :

Tip

Si votre interface vers l'extérieur est `ppp0` ou `ipppp0`, vous pouvez remplacer le `detect` dans la seconde colonne par un “-” (sans les quotes).

Tip

Si votre interface vers l'extérieur est `ppp0` or `ipppp0` u si vous avez une adresse IP statique, vous pouvez enlever `dhcp` dans la liste des options .

Tip

Si votre interface est un bridge utilisant l'utilitaire **brctl** alors vous devez ajouter l'option `routeback` à la liste des options.

Tip

Si vous spécifiez *norfc1918* pour votre interface externe, vous pouvez vérifier périodiquement le Shorewall Errata pour mettre à jour le fichier `/usr/share/shorewall/rfc1918`. Sinon, vous pouvez copier le fichier `/usr/share/shorewall/rfc1918` vers `/etc/shorewall/rfc1918` et adapter votre fichier `/etc/shorewall/rfc1918` comme je le fais.

Adresses IP

Avant d'aller plus loin, nous devons dire quelques mots au sujet des adresses Internet Protocol (IP). Normalement, votre fournisseur Internet ISP vous assignera une seule adresse IP. Cette adresse peut être assignée par le Dynamic Host Configuration Protocol (DHCP) ou lors de l'établissement de votre connexion lorsque vous vous connectez (modem standard) ou établissez votre connexion PPP. Dans de rares cas , votre provider peut vous assigner une adresse statique IP ; cela signifie que vous devez configurer l'interface externe de votre firewall afin d'utiliser cette adresse de manière permanente. Votre adresse externe assignée, elle va être partagée par tous vos systèmes lors de l'accès à Internet. Vous devrez assigner vos propres adresses dans votre réseau local (votre interface interne sur le firewall ainsi que les autres ordinateurs). La **RFC 1918** réserve plusieurs plages d'adresses privées *Private* IP à cet fin:

```
10.0.0.0     - 10.255.255.255
172.16.0.0   - 172.31.255.255
```

```
192.168.0.0 - 192.168.255.255
```

Avant de lancer Shorewall, regarder l'adresse IP de votre interface externe, et si elle est dans les plages précédentes, vous devez enlever l'option 'norfc1918' dans la ligne concernant l'interface externe dans le fichier `/etc/shorewall/interfaces`.

Vous devrez assigner vos adresses depuis le même sous-réseau (sub-network-subnet). Pour ce faire, nous pouvons considérer un sous-réseau dans une plage d'adresses `x.y.z.0 - x.y.z.255`. Chaque sous-réseau aura un masque (Subnet Mask) `255.255.255.0`. L'adresse `x.y.z.0` est réservée comme l'adresse de sous-réseau *Subnet Address* et `x.y.z.255` est réservée en tant qu'adresse de broadcast *Subnet Broadcast Address*. Dans Shorewall, un sous-réseau est décrit en utilisant Classless InterDomain Routing (CIDR) notation Il consiste en l'adresse du sous-réseau suivie par`/24`. Le “24” se réfère au nombre consécutif de bits marquant “1” dans la partie gauche du masque de sous-réseau.

Table 1. Un exemple de sous-réseau (sub-network) :

Range:	`10.10.10.0 - 10.10.10.255`
Subnet Address:	`10.10.10.0`
Broadcast Address:	`10.10.10.255`
CIDR Notation:	`10.10.10.0/24`

Il est de mise d'assigner l'interface interne à la première adresse utilisable du sous-réseau (`10.10.10.1` dans l'exemple précédent) ou la dernière adresse utilisable (`10.10.10.254`).

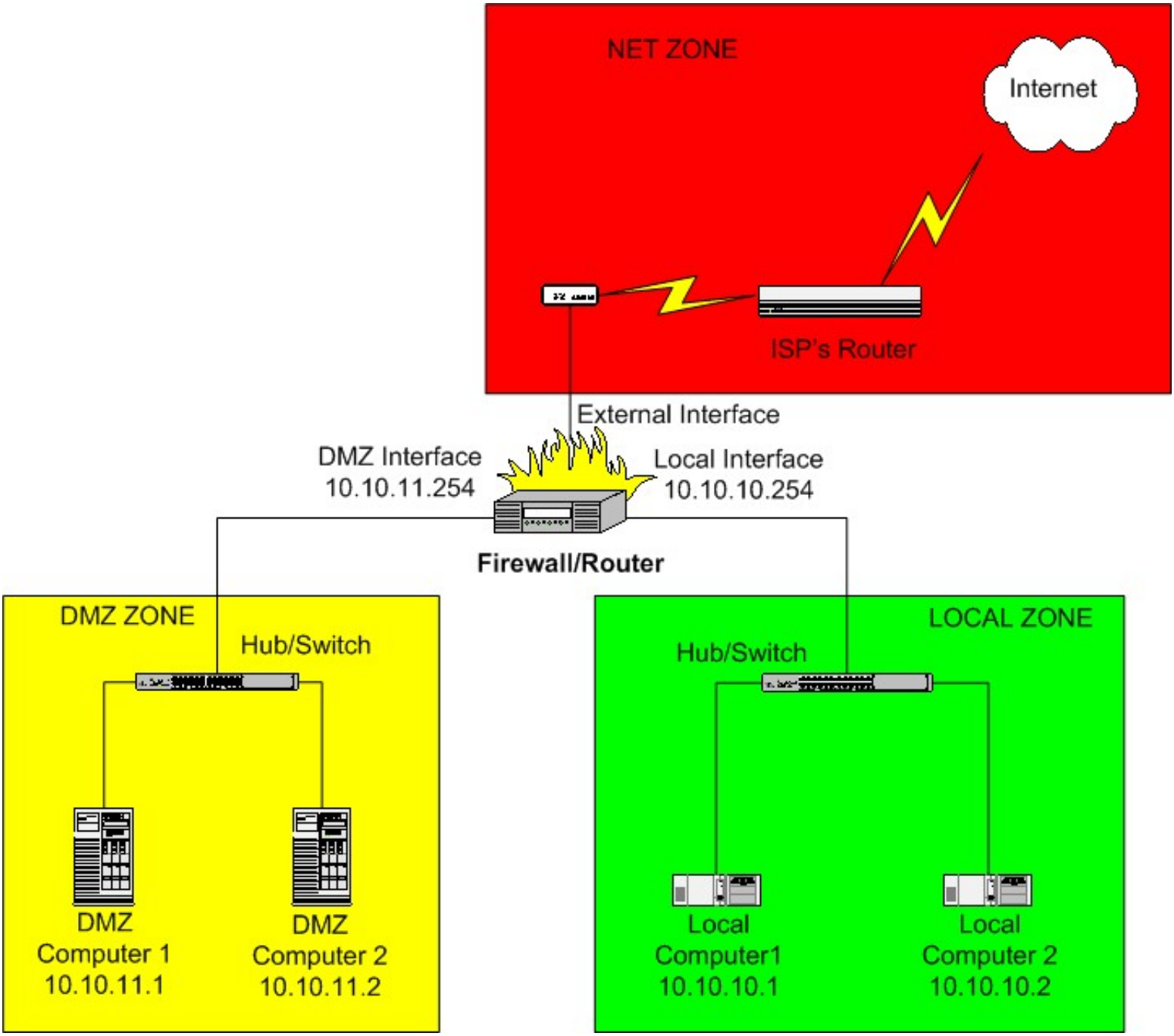
L'un des buts d'un sous-réseau est de permettre à tous les ordinateurs dans le sous-réseau de savoir avec quels autres ordinateurs ils peuvent communiquer directement. Pour communiquer avec des systèmes en dehors du sous-réseau, les ordinateurs envoient des paquets à travers le gateway (routeur).

Vos ordinateurs en local (ordinateur 1 et ordinateur 2 dans le diagramme) devraient être configurés avec leur passerelle par défaut (default gateway) pointant sur l'adresse IP de l'interface interne du firewall.

La présentation précédente ne fait que d'effleurer la question des sous réseaux et du routage. Si vous êtes intéressé pour apprendre plus sur l'adressage IP et le routage, je recommande “IP Fundamentals: What Everyone Needs to Know about Addressing & Routing”, Thomas A. Maufer, Prentice-Hall, 1999, ISBN 0-13-975483-0 (link).

Le reste de ce guide assumera que vous avez configuré votre réseau comme montré ci-dessous :

Figure 3. DMZ



La passerelle par défaut (default gateway) pour les ordinateurs de la DMZ sera `10.10.11.254` et le passerelle par défaut pour les ordinateurs en local sera `10.10.10.254`

Warning

Votre FAI (fournisseur d'accès) pourrait assigner une adresse **RFC 1918** à votre interface externe. Si cette adresse est le sous-réseau `10.10.10.0/24` alors vous aurez besoin d'un sous-réseau DIFFERENT RFC 1918 pour votre réseau local.

IP Masquerading (SNAT)

Les adresses réservées par la RFC 1918 sont parfois désignées comme non-routables car les routeurs Internet (backbone) ne font pas circuler les paquets qui ont une adresse de destination appartenant à la RFC-1918. Lorsqu'un de vos systèmes en local (supposons l'ordinateur1) demande une connexion à un serveur par Internet, le firewall doit appliquer un *Network Address Translation* (NAT). Le firewall réécrit l'adresse source dans le paquet, et l'a remplacé par l'adresse de l'interface externe du firewall; en d'autres mots, le firewall fait croire que c'est lui même qui initie la connexion. Ceci est nécessaire afin que l'hôte de destination soit capable de renvoyer les paquets au firewall (souvenez vous que les paquets qui ont pour adresse de destination, une adresse réservée par la RFC 1918 ne pourront pas être routés à travers Internet, donc l'hôte Internet ne pourra adresser sa

réponse à l'ordinateur 1). Lorsque le firewall reçoit le paquet de réponse, il remet l'adresse de destination à `10.10.10.1` et fait passer le paquet vers l'ordinateur 1.

Sur les systèmes Linux, ce procédé est souvent appelé *IP Masquerading* mais vous verrez aussi le terme de *Source Network Address Translation* (SNAT). Shorewall suit la convention utilisée avec Netfilter:

- *Masquerade* désigne le cas ou vous laissez votre firewall détecter automatiquement l'adresse de l'interface externe.
- *SNAT* désigne le cas où vous spécifiez explicitement l'adresse source des paquets sortant de votre réseau local.

Sous Shorewall, autant le *Masquerading* et le *SNAT* sont configuré avec des entrés dans le fichier `/etc/shorewall/masq`. Vous utiliserez normalement le Masquerading si votre adresse IP externe i est dynamique, et SNAT si l'adresse IP est statique.

Si votre interface externe est `eth0`, votre interface locale `eth1` et votre interface pour la DMZ `eth2` vous n'avez pas besoin de modifier le fichier fourni avec l'exemple. Dans le cas contraire, éditez `/etc/shorewall/masq` et changez le en conséquence.

Si, malgré les avertissements, vous utilisez ce guide pour un utilisation de one-to-one NAT ou de Proxy ARP pour votre DMZ, enlever l'entrée pour `eth2` de `/etc/shorewall/masq`.

Si votre IP externe est statique, vous pouvez la mettre dans la troisième colonne dans `/etc/shorewall/masq` si vous le désirez, de toutes façons votre firewall fonctionnera bien si vous laissez cette colonne vide. Le fait de mettre votre IP statique dans la troisième colonne permet un traitement des paquets sortant un peu plus efficace.

Si vous utilisez les paquets Debian, vérifiez que votre fichier de configuration `shorewall.conf` contient bien les valeurs suivantes, si elles n'y sont pas faite les changements nécessaires:

- `NAT_ENABLED=Yes` (Shorewall versions earlier than 1.4.6)
- `IP_FORWARDING=On`

Port Forwarding (DNAT)

Un de nos buts est de, peut être, faire tourner un ou plusieurs serveurs sur nos ordinateurs dans la DMZ. que ces ordinateurs on une adresse RFC-1918, il n'est pas possible pour les clients sur Internet de se connecter directement à eux. Il est nécessaire à ces clients d'adresser leurs demandes de connexion au firewall qui ré écrit l'adresse de destination de votre serveur, et fait passer le paquet à celui-ci. Lorsque votre serveur répond, le firewall applique automatiquement un SNAT pour réécrire l'adresse source dans la réponse.

Ce procédé est appelé *Port Forwarding* ou *Destination Network Address Translation* (DNAT). Vous configurez le port forwarding en utilisant les règles DNAT dans le fichier `/etc/shorewall/rules` file.

La forme générale d'une simple règle de port forwarding dans `/etc/shorewall/rules` est:

```
#ACTION   SOURCE    DEST                                        PROTO      DEST
PORT(S)
DNAT      net       dmz:<server local IP address>[:<server port>] <protocol> <port>
```

Si vous ne spécifiez pas le `<server port>`, il est supposé être le même que `<port>`.

Example 1. Vous faites tourner un serveur Web dans votre DMZ (2) et vous voulez faire passer les paquets entrant en TCP sur le port 80 à ce système

```
#ACTION    SOURCE     DEST                 PROTO       DEST PORT(S)
DNAT       net        dmz:10.10.11.2       tcp         80
ACCEPT     loc        dmz:10.10.11.2       tcp         80
```

- L'entrée 1 forward le port 80 depuis Internet.
- L'entrée 2 autorise les connexions du réseau local.

Deux points importants à garder en mémoire :

- Lorsque vous vous connectez à votre serveur à partir de votre réseau local, vous devez utiliser l'adresse IP interne du serveur (`10.10.11.2`).
- Quelques fournisseurs Internet (Provider/ISP) bloquent les requêtes de connexion entrantes sur le port 80. Si vous avez des problèmes pour vous connecter à votre serveur web, essayez la règle suivante et connectez vous sur le port 5000 (c.a.d., connectez vous à `http://w.x.y.z:5000 ou w.x.y.z` est votre IP externe).

  ```
  #ACTION    SOURCE     DEST                 PROTO       DEST PORT(S)     SOURCE
  #                                                                       PORT(S)
  DNAT       net        dmz:10.10.11.2:80    tcp         80               5000
  ```
- Si vous voulez avoir la possibilité de vous connecter à votre serveur depuis le réseau local en utilisant votre adresse externe, et si vous avez une adresse IP externe statique (fixe), vous pouvez remplacer la règle loc->dmz précédente par :

  ```
  #ACTION    SOURCE     DEST             PROTO   DEST PORT(S)   SOURCE    ORIGINAL
  #                                                              PORT(S)   DEST
  DNAT       loc        dmz:10.10.11.2   tcp     80             -         <external IP>
  ```

  Si vous avez une IP dynamique, alors vous devez vous assurer que votre interface externe est en route avant de lancer Shorewall et vous devez suivre les étapes suivantes (en supposant que votre interface externe est `eth0`):
  1. Insérez ce qui suit dans /etc/shorewall/params:

     ETH0_IP=$(find_interface_address eth0)
  2. Faites votre règle `loc->dmz` rule:

     ```
     #ACTION    SOURCE     DEST             PROTO   DEST PORT(S)   SOURCE    ORIGINAL
     #                                                              PORT(S)   DEST
     DNAT       loc        dmz:10.10.11.2   tcp     80             -         $ETH0_IP
     ```
- Si vous voulez accéder à votre serveur dans la DMZ en utilisant votre adresse IP externe, regardez FAQ 2a.

A ce point, ajoutez les règles DNAT et ACCEPT pour vos serveurs.

Domain Name Server (DNS)

Normalement, quand vous vous connectez à votre fournisseur (FAI/ISP), une partie consiste à obtenir votre adresse IP, votre *Domain Name Service* (DNS) pour le firewall est configuré automatiquement (c.a.d.,le fichier `/etc/resolv.conf` sera mis à jour). Il arrive que votre provider vous donne une paire d'adresse IP pour les serveurs DNS afin que vous configuriez manuellement votre serveur de nom primaire et secondaire. La manière dont le DNS est configuré sur votre firewall est de votre responsabilité. Vous pouvez procéder d'une de ses deux façons :

- Vous pouvez configurer votre système interne pour utiliser les noms de serveurs de votre provider. Si votre fournisseur vous donne les adresses de leurs serveurs ou si ces adresses sont disponibles sur leur site web, vous pouvez configurer votre système interne afin de les utiliser. Si cette information n' est pas disponible, regardez dans `/etc/resolv.conf` sur votre firewall -- les noms des serveurs sont donnés dans l'enregistrement "nameserver" dans ce fichier.

- Vous pouvez configurer un cache dns *Caching Name Server* sur votre firewall. Red Hat™ a un RPM pour serveur dns de cache (le RPM à besoin aussi du paquetage **bind** RPM) et pour les utilisateurs de Bering, il y a **dnscache.lrp**. Si vous adoptez cette approche, vous configurez votre système interne pour utiliser le firewall lui même comme étant le seul serveur de nom primaire. Vous pouvez utiliser l'adresse IP interne du firewall (`10.10.10.254` dans l'exemple précédent) pour l'adresse de serveur de nom. Pour permettre à vos systèmes locaux de discuter avec votre serveur cache de nom, vous devez ouvrir le port 53 (à la fois UDP and TCP) sur le firewall vers le réseau local; vous ferez ceci en ajoutant les règles suivantes dans `/etc/shorewall/rules`.

  Si vous faites tourner le serveur de nom sur le firewall:

```
#ACTION   SOURCE    DEST                PROTO      DEST PORT(S)
AllowDNS  loc       fw
AllowDNS  dmz       fw
```

Dans la régle ci-dessus, “AllowDNS” est un exemple d'action prédéfinie *defined action*. Shorewall inclus un nombre d'actions prédéfinies et vous pouvez ajouter les vôtres. Pour voir les actions comprises avec votre version de Shorewall, regardez dans le fichier `/etc/shorewall/actions.std`. Le nom de celles qui acceptent des connexions débutent par “Allow”.

Vous n'êtes pas obligé d'utiliser des actions prédéfinies quand vous ajoutez des régles dans le fichier `/etc/shorewall/rules`; les régles générées par Netfilter sont plus performantes sans actions prédéfinies. La régle vue ci-dessus peut aussi être codé comme cela:

```
#ACTION   SOURCE    DEST                PROTO      DEST PORT(S)
ACCEPT    loc       fw                  tcp        53
ACCEPT    loc       fw                  udp        53
ACCEPT    dmz       fw                  tcp        53
ACCEPT    dmz       fw                  udp        53
```

Au cas ou Shorewall n'inclue pas d'actions définies qui vous conviennent, vous pouvez les définir vous même ou coder directement les régles.

Autres Connexions

Les fichiers exemples inclus dans l'archive (three-interface) contiennent les règles suivantes :

```
#ACTION   SOURCE    DEST                PROTO      DEST PORT(S)
AllowDNS  fw        net
```

Ces règles autorisent l'accès DNS à partir de votre firewall et peuvent être enlevées si vous avez décommenté la ligne dans `/etc/shorewall/policy` autorisant toutes les connexions depuis le firewall vers Internet.

L'exemple inclus aussi:

```
#ACTION   SOURCE    DEST                PROTO      DEST PORT(S)
AllowSSH  loc       fw
AllowSSH  loc       dmz
```

Ces régles autorisent un serveur SSH sur votre firewall et chacun des systèmes de votre DMZ et y autoriser la connexion à ceux-ci depuis votre réseau local.

Si vous désirez permettre d'autres connexions entre vos systèmes, la syntaxe générale est:

```
#ACTION   SOURCE        DEST                PROTO      DEST PORT(S)
<action>  <source zone> <destination zone>
```

La syntaxe générale lorsqu'on utilise pas des actions prédéfinies est:

```
#ACTION   SOURCE        DEST                 PROTO      DEST PORT(S)
ACCEPT    <source zone> <destination zone>  <protocol> <port>
```

Example 2. Vous souhaitez rendre publiquement accessible votre serveur DNS sur le firewall

En utiliser une action prédéfinie:

```
#ACTION   SOURCE    DEST                 PROTO      DEST PORT(S)
AllowDNS  net       fw
```

Sans action prédéfinie:

```
#ACTION   SOURCE    DEST                 PROTO      DEST PORT(S)
ACCEPT    net       fw                   tcp        53
ACCEPT    net       fw                   udp        53
```

Ces deux régles viennent évidemment s'ajouter à celles listées sous “Vous pouvez configurer un cache dns sur votre firewall”.

Si vous ne savez pas quel port(s) et protocole(s) requièrent une application particulière, vous pouvez regarder ici.

Important

Je ne recommande pas d'autoriser **telnet** vers/de l'Internet parce qu'il utilise du texte en clair (même pour le login!). Si vous voulez un accès shell à votre firewall, utilisez SSH:

```
#ACTION    SOURCE    DEST                 PROTO      DEST PORT(S)
AllowSSH   net       fw
```

Les utilisateurs de Bering pourront ajouter les deux régles suivantes pour être compatible avec la configuration du firewall Jacques's Shorewall.

```
#ACTION   SOURCE    DEST                 PROTO      DEST PORT(S)
ACCEPT    loc       fw                   udp        53
ACCEPT    net       fw                   tcp        80
```

- L'entrée 1 autorise l'utilisation du Cache DNS.
- L'entrée 2 autorise le “weblet” à fonctionner.

Maintenant, éditez votre fichier de configuration `/etc/shorewall/rules` pour ajouter, modifier ou supprimer les autres connexions voulues.

Quelques Points à Garder en Mémoire

- **Vous ne pouvez tester votre firewall de l'intérieur de votre réseau**. Car les requêtes que vous envoyez à votre adresse IP ne veux pas dire qu'elle seront associées à votre interface externe ou la zone “net”. Tout trafic généré par le réseau local sera traité par loc->fw.

- **Les adresses IP sont des propriétés des systèmes, pas des interfaces**. C'est une erreur de croire que votre firewall est capable de renvoyer des paquets simplement parce que vous pouvez faire un ping sur l'adresse IP de toutes les interfaces du firewall depuis le réseau local. La seul conclusion est de conclure que le lien entre le réseau local et le firewall est établi et que vous avez probablement la bonne adresse de la passerelle sur votre système.
- **Toutes les adresses IP configurées sur le firewall sont dans la zone $FW (fw)**. Si 192.168.1.254 est l'adresse IP de votre interface interne, alors vous pouvez écrire "**$FW:192.168.1.254**" dans une régle mais vous ne devez pas écrire "**loc:192.168.1.254**". C'est aussi un non-sens d'ajouter 192.168.1.254 à la zone **loc** en utilisant une entrée dans `/etc/shorewall/hosts`.
- **Les paquets de retour (Reply) ne suivent PAS automatiquement le chemin inverse de la requête d'origine**. Tous les paquets sont routés en se référant à la table de routage respective de chaque hôte à chaque étape du trajet. C'est commun chez ceux qui installent le firewall Shorewall en parallèle à une passerelle existante et essayent d'utiliser DNAT dans Shorewall sans changer la passerelle par défaut sur les systèmes recevant le retour des requêtes. Les requêtes dont, à travers le firewall Shorewall, l'adresse de destination IP est réécrite mais la réponse va directement vers l'ancienne passerelle.
- **Shorewall lui-même n'a aucune notion du dedans et du dehors**. Ces concepts dépendent de la façon dont Shorewall est configuré.

Démarrer et Arrêter Votre Firewall

La procédure d'installation configure votre système pour lancer Shorewall au boot du système, mais au début avec la version 1.3.9 de Shorewall le lancement est désactivé, n'essayer pas de lancer Shorewall avant que la configuration soit finie. Une fois que vous en aurez fini avec la configuration du firewall, vous pouvez permettre le lancement de Shorewall en supprimant le fichier `/etc/shorewall/startup_disabled`.

Important

Les utilisateurs des paquets .deb doivent éditer `/etc/default/shorewall` et mettre `startup=1`.

Le firewall est activé en utilisant la commande "**shorewall start**" et arrêté avec "**shorewall stop**". Lorsque le firewall est stoppé, le routage est autorisé sur les hôtes qui possèdent une entrée dans `/etc/shorewall/routestopped`. Un firewall qui tourne peut être relancé en utilisant la commande "**shorewall restart**" command. Si vous voulez enlever toutes traces de Shorewall sur votre configuration de Netfilter, utilisez "**shorewall clear**".

Les exemples (three-interface) supposent que vous voulez permettre le routage depuis ou vers `eth1` (le réseau local) et `eth2` (DMZ) lorsque Shorewall est stoppé. Si ces deux interfaces ne sont pas connectées à votre réseau local et votre DMZ, ou si vous voulez permettre un ensemble d'hôtes différents, modifiez `/etc/shorewall/routestopped` en conséquence.

Warning

Si vous êtes connecté à votre firewall depuis Internet, n'essayez pas une commande "**shorewall stop**" tant que vous n'avez pas ajouté une entrée pour votre adresse IP (celle à partir de laquelle vous êtes connectée) dans `/etc/shorewall/routestopped`. De la même manière, je ne vous recommande pas d'utiliser "**shorewall restart**"; il est plus intéressant de créer une configuration alternative et de la tester en utilisant la commande "**shorewall try**".

Autres Lectures Recommandées

Je vous recommande vivement de lire la page des Fonctionnalités Générales des Fichiers de Configuration -- elle contient des trucs sur les possibilités de Shorewall pour rendre aisé l'administration de votre firewall Shorewall.