Web application tools

In this section, we will discuss several tools that can be used to test web applications.

Golismero

Golismero is an open source framework for web testing. It is written in the Python language. The interesting features of Golismero are listed as follows:

- It collects and unifies the results from well-known tools such as `sqlmap`, `xsser`, `openvas`, `dnsrecon`, and `theharvester`
- It integrates with CWE, CVE, and OWASP

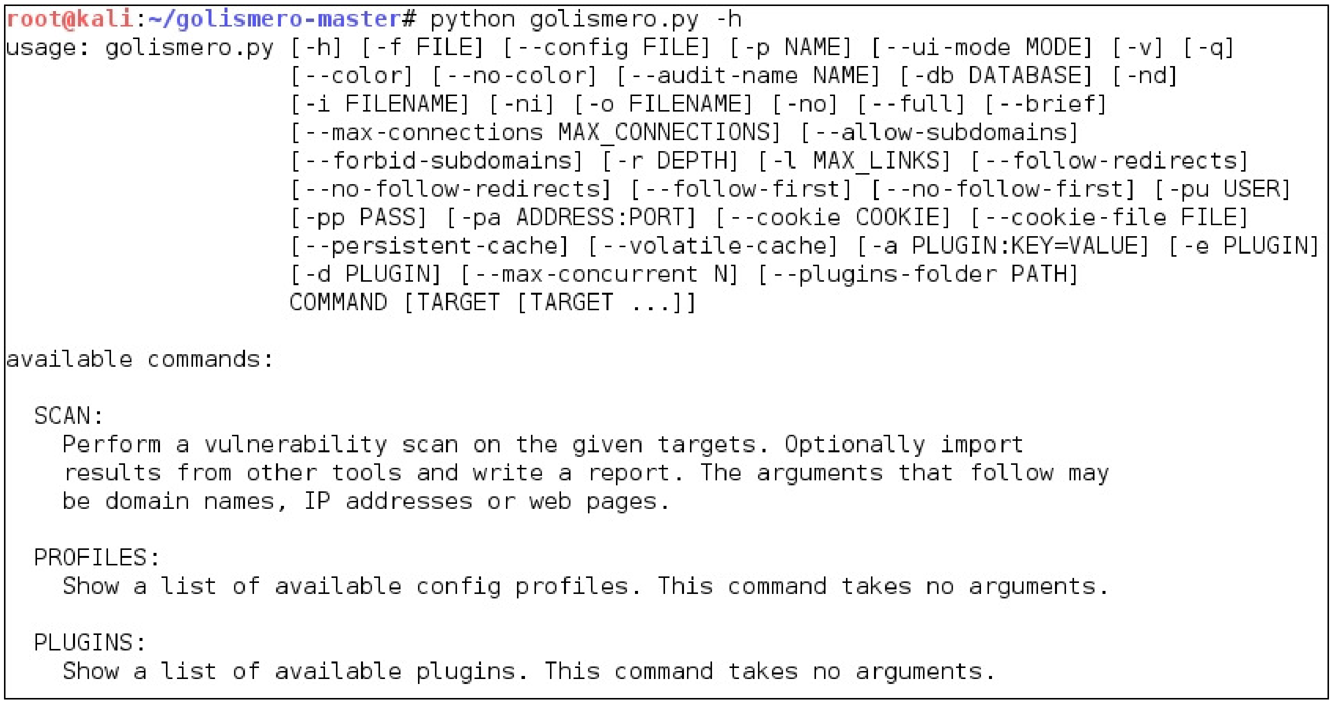
Golismero, which is included with Kali Linux, is an old version and doesn't have features for testing the security of web applications.

You can download the latest version at https://github.com/golismero/golismero/archive/master.zip.

Then, extract the zip file. As a start, you can type the following command to display the Golismero help page:

```
python golismero.py -h
```

The Golismero help page looks like the following screenshot:

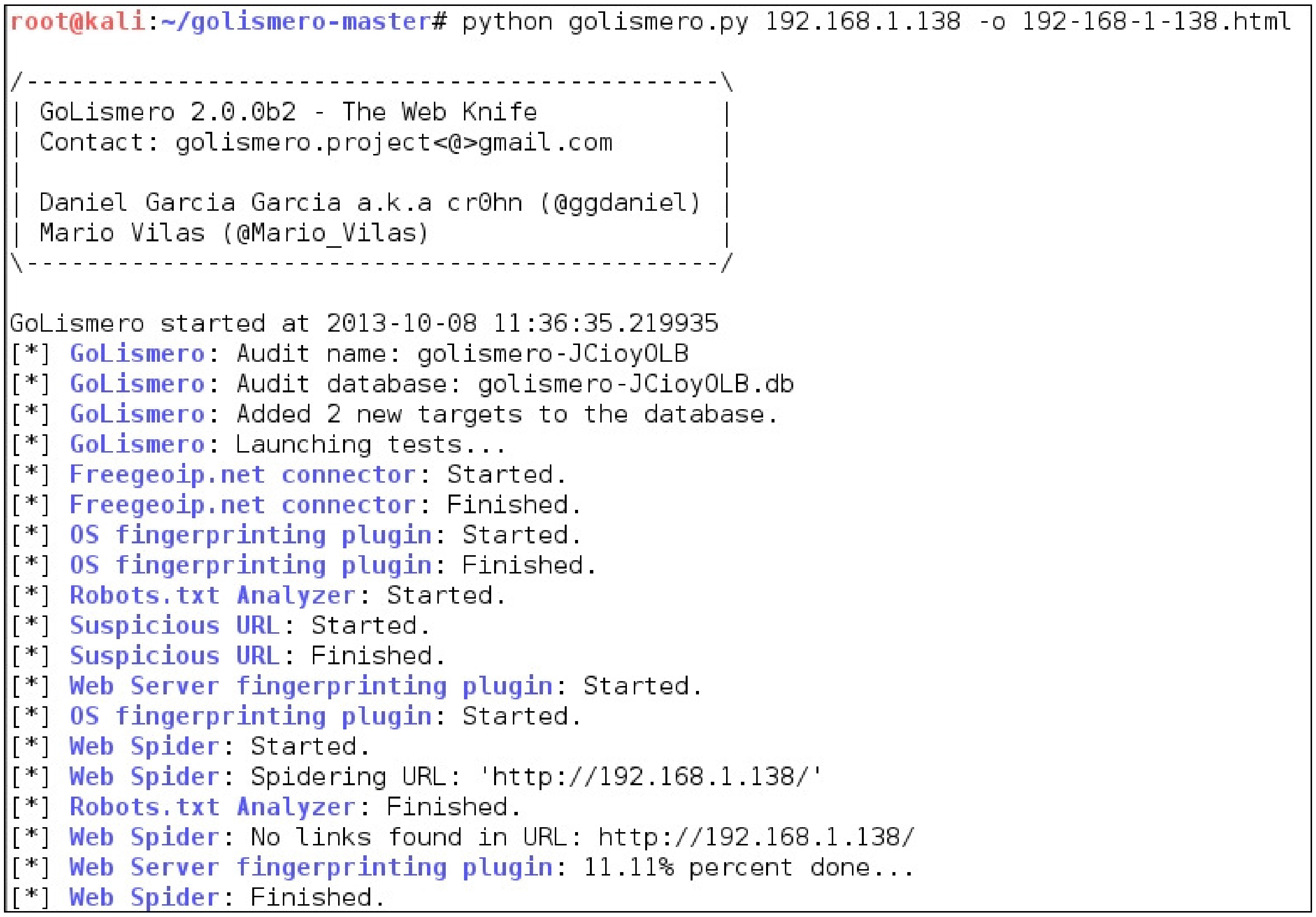


```
root@kali:~/golismero-master# python golismero.py -h
usage: golismero.py [-h] [-f FILE] [--config FILE] [-p NAME] [--ui-mode MODE] [-v] [-q]
                    [--color] [--no-color] [--audit-name NAME] [-db DATABASE] [-nd]
                    [-i FILENAME] [-ni] [-o FILENAME] [-no] [--full] [--brief]
                    [--max-connections MAX_CONNECTIONS] [--allow-subdomains]
                    [--forbid-subdomains] [-r DEPTH] [-l MAX_LINKS] [--follow-redirects]
                    [--no-follow-redirects] [--follow-first] [--no-follow-first] [-pu USER]
                    [-pp PASS] [-pa ADDRESS:PORT] [--cookie COOKIE] [--cookie-file FILE]
                    [--persistent-cache] [--volatile-cache] [-a PLUGIN:KEY=VALUE] [-e PLUGIN]
                    [-d PLUGIN] [--max-concurrent N] [--plugins-folder PATH]
                    COMMAND [TARGET [TARGET ...]]

available commands:

  SCAN:
    Perform a vulnerability scan on the given targets. Optionally import
    results from other tools and write a report. The arguments that follow may
    be domain names, IP addresses or web pages.

  PROFILES:
    Show a list of available config profiles. This command takes no arguments.

  PLUGINS:
    Show a list of available plugins. This command takes no arguments.
```

If you want to scan a website, you can issue the following command:

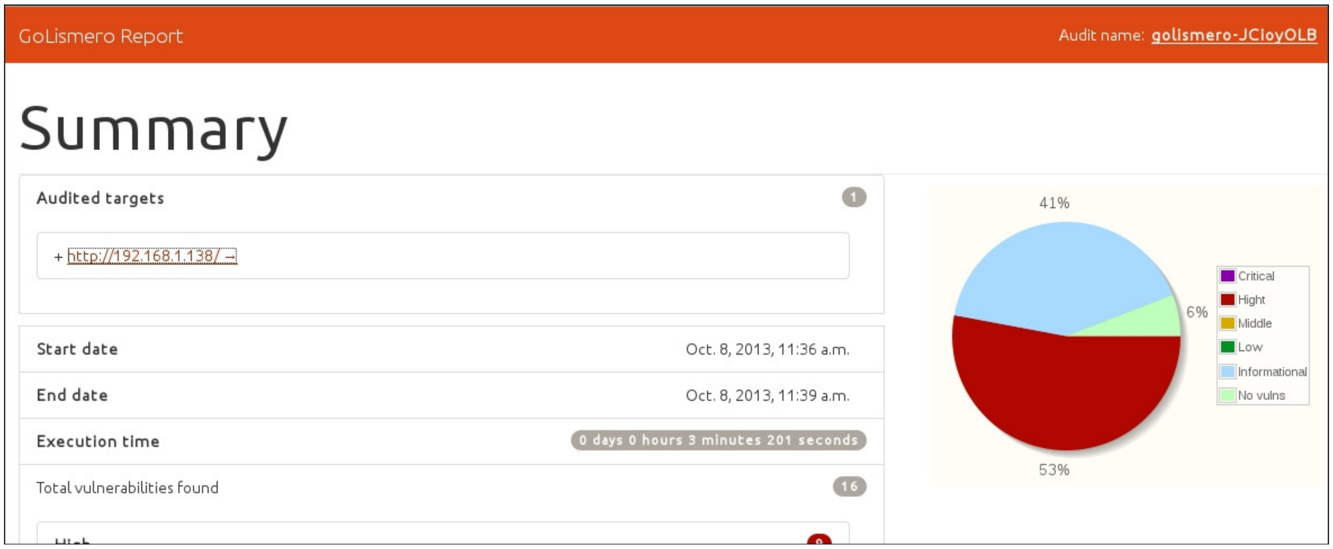
```
python golismero.py 192.168.1.138 -o 192-168-1-138.html
```

The command will display the following screenshot:



```
root@kali:~/golismero-master# python golismero.py 192.168.1.138 -o 192-168-1-138.html

/----------------------------------------------\
| GoLismero 2.0.0b2 - The Web Knife            |
| Contact: golismero.project<@>gmail.com       |
|                                              |
| Daniel Garcia Garcia a.k.a cr0hn (@ggdaniel) |
| Mario Vilas (@Mario_Vilas)                   |
\----------------------------------------------/

GoLismero started at 2013-10-08 11:36:35.219935
[*] GoLismero: Audit name: golismero-JCioyOLB
[*] GoLismero: Audit database: golismero-JCioyOLB.db
[*] GoLismero: Added 2 new targets to the database.
[*] GoLismero: Launching tests...
[*] Freegeoip.net connector: Started.
[*] Freegeoip.net connector: Finished.
[*] OS fingerprinting plugin: Started.
[*] OS fingerprinting plugin: Finished.
[*] Robots.txt Analyzer: Started.
[*] Suspicious URL: Started.
[*] Suspicious URL: Finished.
[*] Web Server fingerprinting plugin: Started.
[*] OS fingerprinting plugin: Started.
[*] Web Spider: Started.
[*] Web Spider: Spidering URL: 'http://192.168.1.138/'
[*] Robots.txt Analyzer: Finished.
[*] Web Spider: No links found in URL: http://192.168.1.138/
[*] Web Server fingerprinting plugin: 11.11% percent done...
[*] Web Spider: Finished.
```

The following screenshot is the report from Golismero:



Summary

Audited targets	1
+ http://192.168.1.138/	
Start date	Oct. 8, 2013, 11:36 a.m.
End date	Oct. 8, 2013, 11:39 a.m.
Execution time	0 days 0 hours 3 minutes 201 seconds
Total vulnerabilities found	16

Arachni

Arachni (http://www.arachni-scanner.com/) is a modular, high-performance, Ruby-based framework to help us evaluate the web applications' security.

Arachni has several features (http://www.arachni-scanner.com/about/features/) that include the following:

- Support for SSL
- Automatic logout detection and re-login during the audit
- High-performance HTTP requests
- Parallel scans
- Platform fingerprinting to make efficient use of available bandwidth

- Audit for vulnerabilities such as a SQL Injection, CSRF, code injection, LDAP injection, path traversal, file inclusion, and XSS

However, Arachni also has the following limitations (http://www.arachni-scanner.com/about/limitations/):

- It has no support for DOM, JavaScript, AJAX, and HTML5
- It may generate false positive results

By default, Kali Linux comes with Arachni Version 0.4.4.

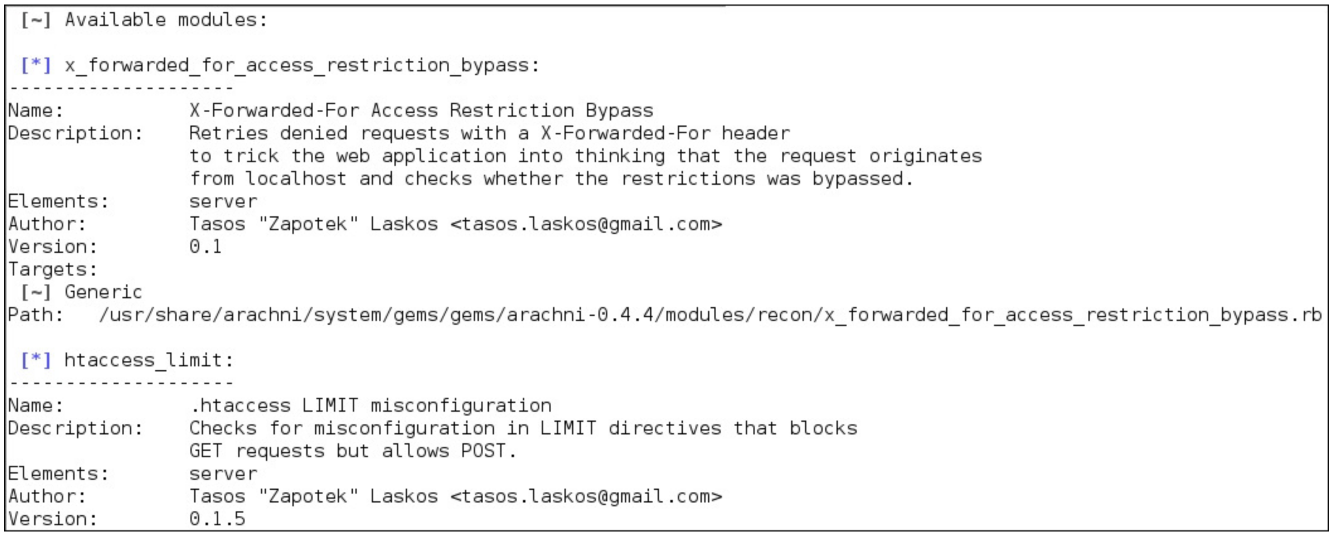
If you want to find out the commands supported by Arachni, you can type the following command to display the help page:

```
arachni -h
```

If you want to see the available modules, you can use the `--lsmod` option:

```
arachni --lsmod
```

The following screenshot is a sample of the modules that are available in Arachni:



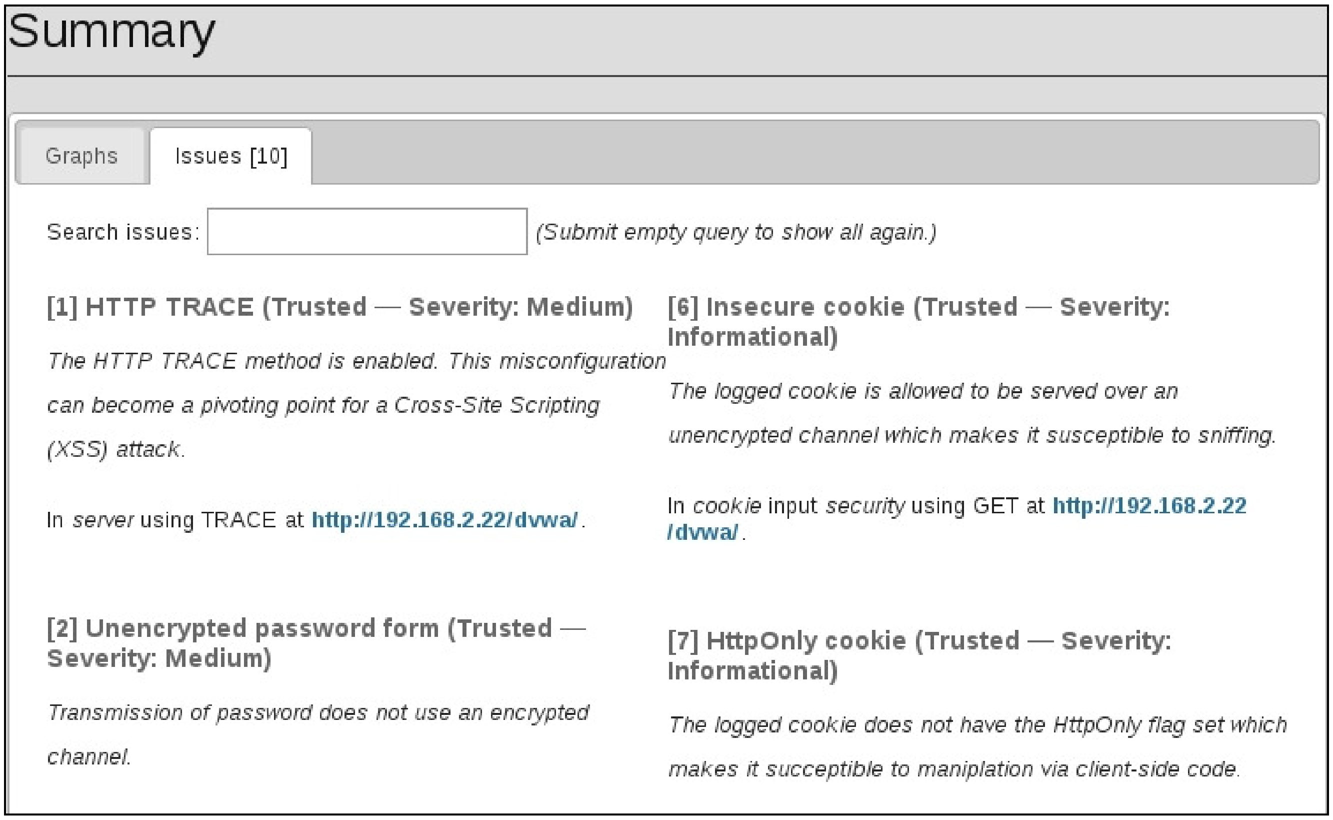
```
 [~] Available modules:

 [*] x_forwarded_for_access_restriction_bypass:
--------------------
Name:           X-Forwarded-For Access Restriction Bypass
Description:    Retries denied requests with a X-Forwarded-For header
                to trick the web application into thinking that the request originates
                from localhost and checks whether the restrictions was bypassed.
Elements:       server
Author:         Tasos "Zapotek" Laskos <tasos.laskos@gmail.com>
Version:        0.1
Targets:
 [~] Generic
Path:   /usr/share/arachni/system/gems/gems/arachni-0.4.4/modules/recon/x_forwarded_for_access_restriction_bypass.rb

 [*] htaccess_limit:
--------------------
Name:           .htaccess LIMIT misconfiguration
Description:    Checks for misconfiguration in LIMIT directives that blocks
                GET requests but allows POST.
Elements:       server
Author:         Tasos "Zapotek" Laskos <tasos.laskos@gmail.com>
Version:        0.1.5
```

As an example, we are going to scan a web application called DVWA (http://www.dvwa.co.uk/), located in server `192.168.2.22`; the result will be stored in an HTML file. Following is the command that you can use:

```
arachni http://192.168.2.22/dvwa/ --report=html:outfile=./192-168-2-22-
dvwa.html
```

The report file will be stored in the `/usr/share/arachni/bin/ directory` file.

The following screenshot shows the report content as displayed by a web browser:


Summary

Search issues: (Submit empty query to show all again.)

[1] HTTP TRACE (Trusted — Severity: Medium)

The HTTP TRACE method is enabled. This misconfiguration can become a pivoting point for a Cross-Site Scripting (XSS) attack.

In *server* using TRACE at http://192.168.2.22/dvwa/ .

[2] Unencrypted password form (Trusted — Severity: Medium)

Transmission of password does not use an encrypted channel.

[6] Insecure cookie (Trusted — Severity: Informational)

The logged cookie is allowed to be served over an unencrypted channel which makes it susceptible to sniffing.

In *cookie* input *security* using GET at http://192.168.2.22/dvwa/ .

[7] HttpOnly cookie (Trusted — Severity: Informational)

The logged cookie does not have the HttpOnly flag set which makes it succeptible to maniplation via client-side code.

BlindElephant

BlindElephant is a web application fingerprint tool that attempts to discover the version of a known web application by comparing the static files at known locations against precomputed hashes for versions of those files in all available releases.

The technique that is utilized here is fast, low-bandwidth, non-invasive, generic, and highly automated.

To display the BlindElephant help page, you can type the following command:

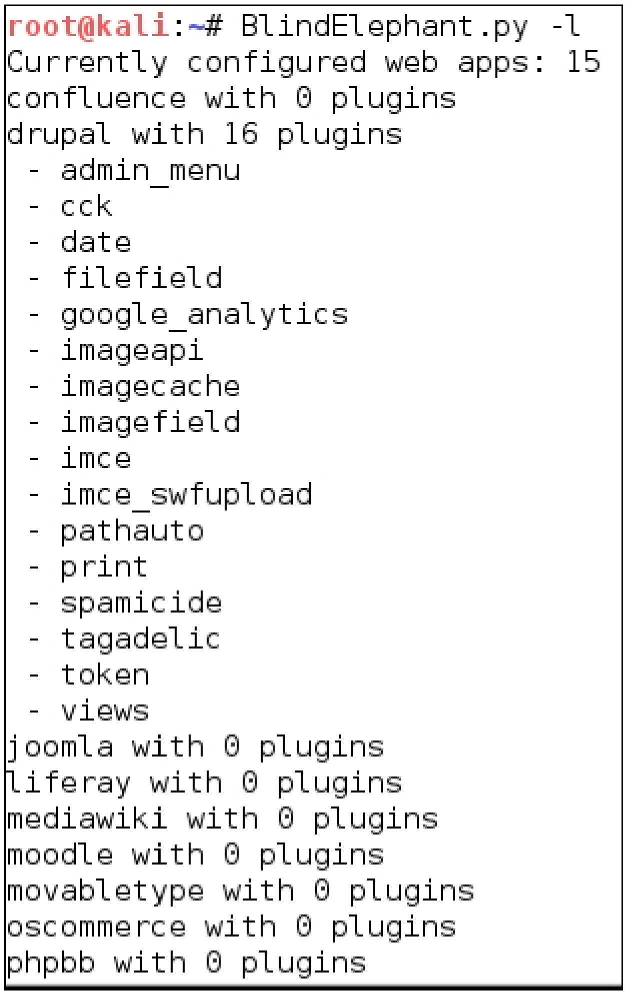
```
BlindElephant.py -h
```

This will display the help message on your screen.

If you want to know about the web applications and plugins supported by BlindElephant, you can type the following command:

```
BlindElephant.py –l
```

The following screenshot is the result:



```
root@kali:~# BlindElephant.py -l
Currently configured web apps: 15
confluence with 0 plugins
drupal with 16 plugins
 - admin_menu
 - cck
 - date
 - filefield
 - google_analytics
 - imageapi
 - imagecache
 - imagefield
 - imce
 - imce_swfupload
 - pathauto
 - print
 - spamicide
 - tagadelic
 - token
 - views
joomla with 0 plugins
liferay with 0 plugins
mediawiki with 0 plugins
moodle with 0 plugins
movabletype with 0 plugins
oscommerce with 0 plugins
phpbb with 0 plugins
```

For our example, we want to find out the WordPress version used by the target website. The following is the command to do that:

```
BlindElephant.py target wordpress
```

The following is the result of that command:

```
Hit http://target/readme.html
Possible versions based on result: 3.1.3, 3.1.3-IIS

Hit http://target/wp-includes/js/tinymce/tiny_mce.js
Possible versions based on result: 3.1.1, 3.1.1-IIS, 3.1.1-RC1, 3.1.1-
RC1-IIS, 3.1.2, 3.1.2-IIS, 3.1.3, 3.1.3-IIS, 3.1.4, 3.1.4-IIS

...

Possible versions based on result: 3.1, 3.1.1, 3.1.1-IIS, 3.1.1-RC1, 3.1.1-
RC1-IIS, 3.1.2, 3.1.2-IIS, 3.1.3, 3.1.3-IIS, 3.1.4, 3.1.4-IIS, 3.1-beta1,
3.1-beta1-IIS, 3.1-beta2, 3.1-beta2-IIS, 3.1-IIS, 3.1-RC1, 3.1-RC2,
3.1-RC2-IIS, 3.1-RC3, 3.1-RC3-IIS, 3.1-RC4, 3.1-RC4-IIS


Fingerprinting resulted in:
3.1.3
3.1.3-IIS
```

```
        Best Guess: 3.1.3
```

The target website uses WordPress Version 3.1.3 based on a BlindElephant guess. After knowing this information, we can find out the vulnerabilities that exist in that particular version.