Webshag 1.00

User Manual

Table of Contents

0x0000 Description

Webshag is a multi-threaded, multi-platform web server audit tool. Written in Python, it gathers commonly useful functionalities for web server auditing like website crawling, URL scanning or file fuzzing.

Webshag can be used to scan a web server in HTTP or HTTPS, through a proxy and using HTTP authentication (Basic and Digest). In addition to that it proposes innovative IDS evasion functionalities aimed at making correlation between request more complicated (e.g. use a different random per request HTTP proxy server).

It also provides innovative functionalities like the capability of retrieving the list of domain names hosted on a target machine and file fuzzing using dynamically generated filenames (in addition to common list-based fuzzing).

Webshag URL scanner and file fuzzer are aimed at reducing the number of false positives and thus producing cleaner result sets. For this purpose, webshag implements a web page fingerprinting mechanism resistant to content changes. This fingerprinting mechanism is then used in a false positive removal algorithm specially aimed at dealing with "soft 404" server responses.

Webshag provides a full featured and intuitive graphical user interface as well as a text-based command line interface and is available for Linux and Windows platforms.

0x0001 Install

/* Requirements */

To be fully functional, webshag needs the following elements to be previously installed:

- ✗ Python[1] or ActivePython[2] virtual machine
- ✗ wxPython[3] GUI toolkit
- ✗ Nmap[4] port scanner (for port scanning module only)
- ✗ A valid *Live Search* AppID[5] (for domain information module only)

Note that if webshag is installed using the Windows installer Python and wxPython are not required.

/* Linux */

Download the application archive from **http://www.scrt.ch/pages/outils.html** into the desired location (e.g. ~/webshag_1.00). Move to the corresponding directory and extract the contents of the archive:

```
cd ~/webshag_1.00
tar zxvf ws100_linux.tar.gz
```

Execute the configuration script:

```
chmod +x config_linux.py
./config_linux.py
```

/* Windows */

Download the Windows installer from **http://www.scrt.ch/pages/outils.html** and execute it.

After installing the application, remember to setup your *Live Search AppID* in menu Tools > Config. If *Nmap* has not been automatically detected, also setup the correct path to *Nmap*.

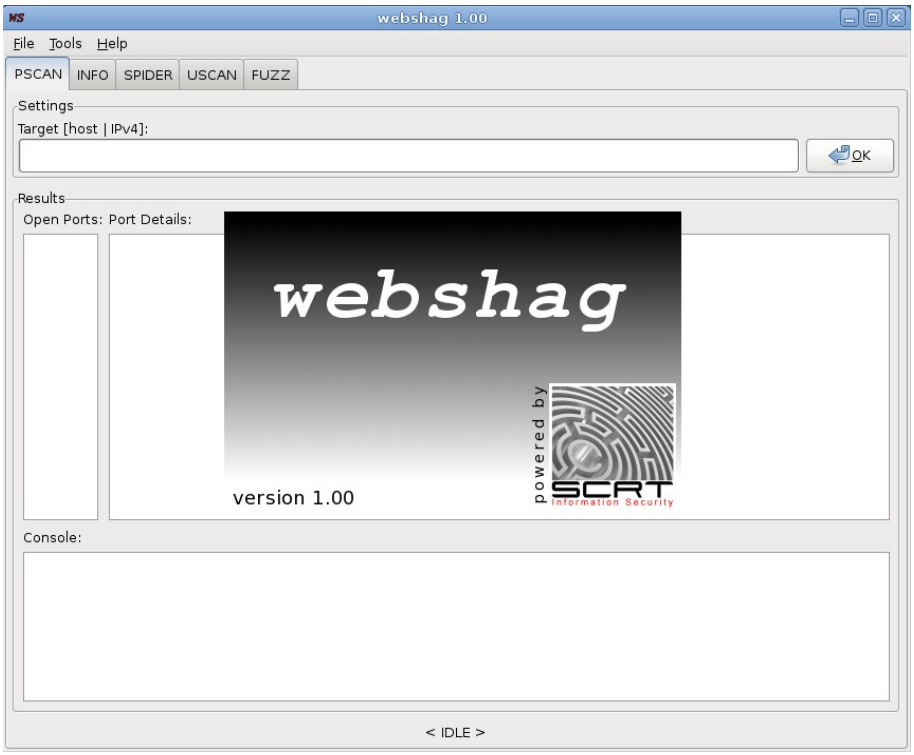
1 http://www.python.org
2 http://www.activestate.com/Products/activepython/
3 http://www.wxpython.org/
4 http://nmap.org/
5 http://search.live.com/developer

0x0010 Getting Started

/* Graphical User Interface */

Webshag is primarily meant to be run in a graphical environment. It is thus provided with a full featured graphical user interface. As this is the default mode, to start the application in GUI mode, simply run the main script without any parameter. The version installed by Windows installer is meant to be run in GUI mode **only**. To run webshag on Windows systems, simply do as usual... :-)

```
./webshag
```



To switch between the audit module, use the tabs located below the menu bar. Note that execution of modules is multi-threaded, it is thus possible to run several modules at the same time. The status of each module is indicated on bottom: <IDLE> or <RUNNING>.

/* Command Line Interface */

Despite being primarily meant to be run in GUI mode, webshag also provides a full featured command line interface (except the version installed by Windows installer). It can thus be run in CLI mode on machines without graphical environment, on remote machines (e.g. over ssh) or simply if you prefer to use a command line tool. To use webshag in CLI mode, use the -C option:

```
./webshag -C [options] target(s)
```

The complete set of options (some of which are specific to a given module) are detailed throughout this documentation, however a reference can be obtained directly by using CLI help.

```
./webshag.py -h
```

Note that if you omit the -C option, webshag will start in GUI mode and ignore any other option or parameter.

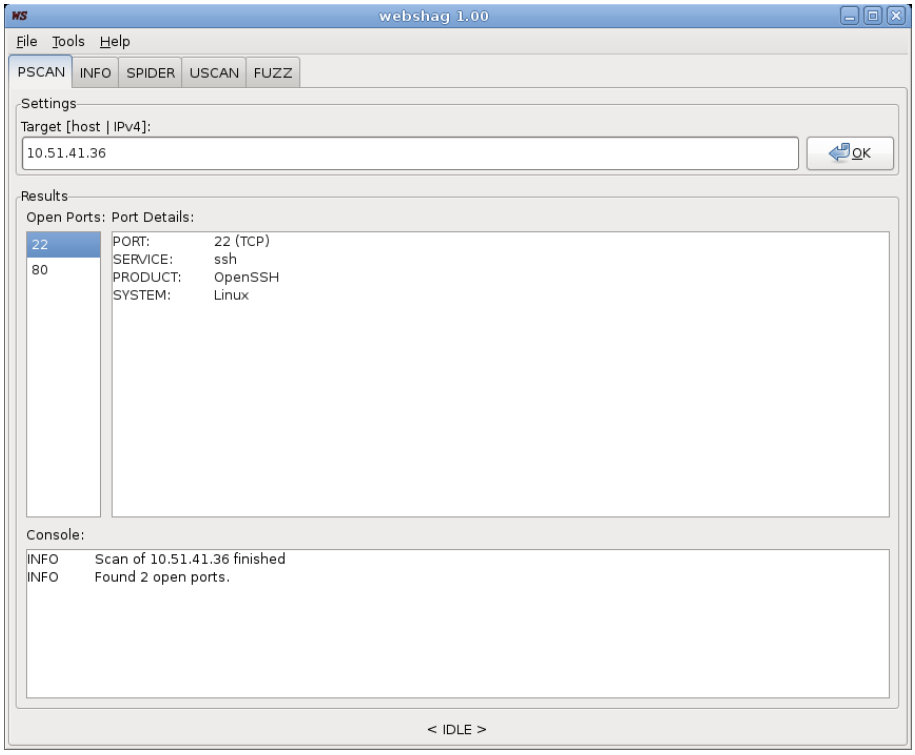
0x0011 PSCAN Module

/* Basics */

The PSCAN (port scanning) module of webshag allows discovering open HTTP ports on the target machine. To achieve this it relies on Nmap[6] port scanner. If Nmap is not installed and functional this module won't work.

/* Graphical User Interface */

To run the port scanning module in GUI mode, select the PSCAN tab, enter the target (hostname or an IPv4 address) and click the OK button. Note that all the open ports are displayed (not only those detected as HTTP).



/* Command Line Interface */

To run the port scanning module in CLI mode run webshag with **-C** option and set **-m** option to **pscan**.

```
./webshag.py -C -m pscan 10.51.41.36
```

6 http://nmap.org/

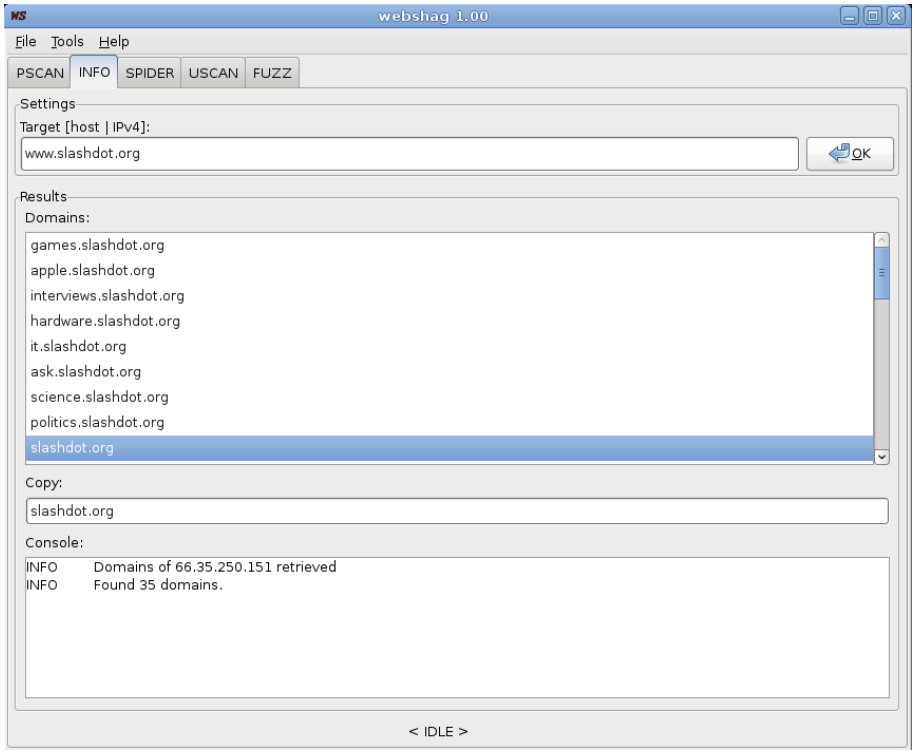
0x0100 INFO Module

/* Basics */

The domain information module of webshag allows retrieving the referenced domain names (vhosts) hosted on the target machine. To achieve this, it uses the web service provided by the *Microsoft Live Search*[7]. To use this module you thus need to have a valid *AppID*[8] and enter it in the webshag configuration file. Refer to the Configuration chapter for more details.

/* Graphical User Interface */

To run the domain information module in GUI mode, select INFO tab, simply enter the target (hostname or IPv4 address) and click OK.



The *Copy* field below the results simply repeats the selected result to allow user to copy/paste long domain names.

/* Command Line Interface */

To run the port scanning module in CLI mode run webshag with **-C** option and set **-m** option to **info**.

```
./webshag.py -C -m info www.slashdot.org
```

7 http://www.live.com/
8 http://search.live.com/developer

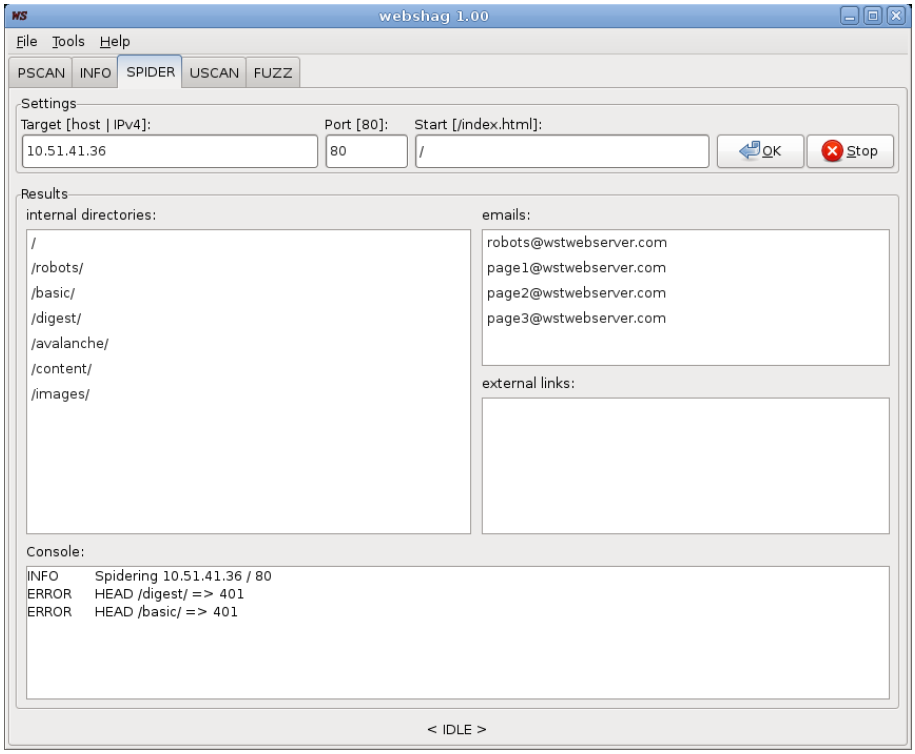
0x0101 SPIDER Module

/* Basics */

The web spider module allows to crawl a whole website while extracting all the encountered directory names, links to external websites and e-mail addresses. When using the spider module, you can specify the target (hostname of IPv4 address), the port and the crawling starting point (most of the time '/' will do the trick, however you can use any other path as starting point).

/* Graphical User Interface */

To run the spider module, select the SPIDER tab, fill in the target, the port and the starting point (root) and press the OK button. Note that in addition to visible links the spider explicitly (mis)uses existing *robots.txt* file. It extracts all the directories specified in *robots.txt* and tries to crawl them. This functionality can however be deactivated in configuration.



/* Command Line Interface */

To run this module in CLI mode, run webshag with **-C** option and set the **-m** option to **spider**.

```
./webshag.py -C -m spider 10.51.41.36
```

To set the port and initial point optional settings use the **-p** and **-i** options respectively.

```
./webshag.py -C -m spider -p 80 -i / 10.51.41.36
```

0x0110 USCAN Module

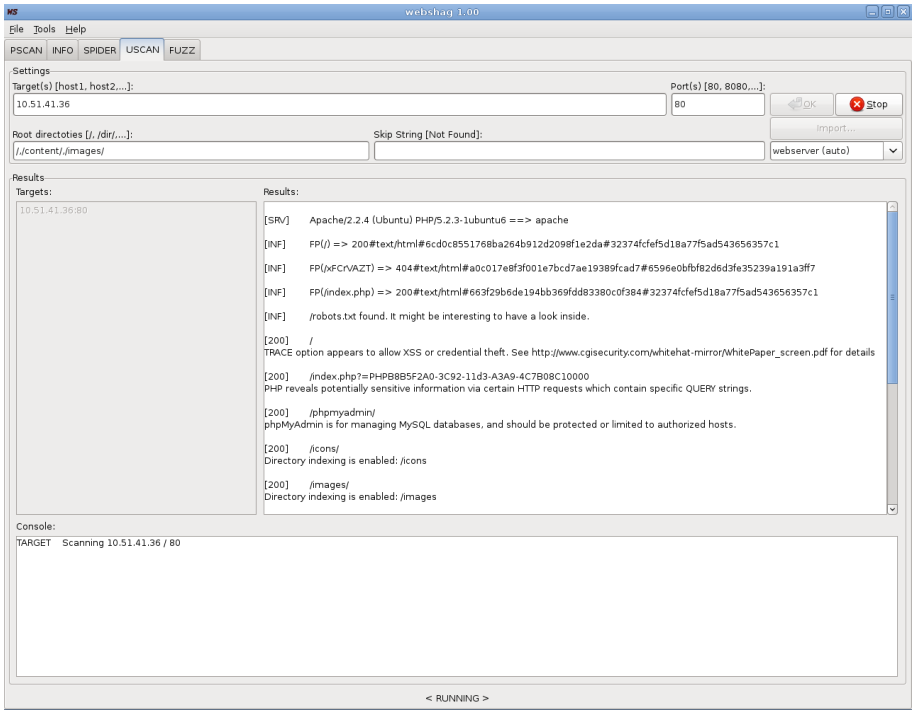
/* Basics */

The URL scanner module allows performing a vulnerability scan on the target web server/application. For this, it relies both on the *Nikto 2*[9] database and on a custom database (specially meant for "home" tests).

The URL scanner module of webshag uses an innovative false positive detection algorithm based on page fingerprinting. In more details, before scanning the target, it fingerprints a few interesting pages to analyze the typical responses of the server. It then uses these fingerprints to detect false positives and remove them "on the fly" from results.

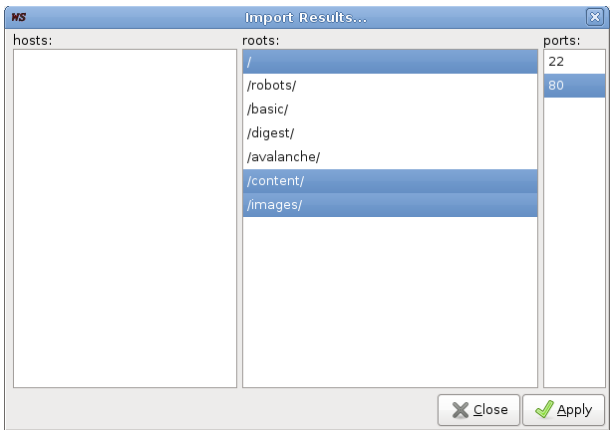
/* Graphical User Interface */

To run the URL scanner, select the USCAN tab and fill in the required parameters. The target(s) are given by the target and port fields and the root directory(ies) can be specified by the root field. All these fields can contain one or several (comma separated values). The skip string field can be used to provide a string causing results to be ignored. In more details if the skip string is present in the server response, the result is ignored. The target webserver can be forced using the available list of servers. If this option is set to default (auto) the server is automatically detected.



Note that an Import button allows to automatically import results from previous (PSCAN, INFO, SPIDER) modules. Selected results are directly inserted in the correct fields.

9 http://www.cirt.net/code/nikto.shtml



/* Command Line Interface */

To run this module in CLI mode, run webshag with **-C** option and set the **-m** option to **uscan**.

```
./webshag.py -C -m uscan 10.51.41.36,10.51.41.37
```

The optional port(s), root directory(ies), skip string and server parameters can be specified using **-p**, **-r**, **-k** and **-s** options respectively.

```
./webshag.py -C -m uscan -p 80,81 -r /,/img/ -s apache -k 'skip me'
10.51.41.36,10.51.41.37
```

0x0111 FUZZ Module

/* Basics */

The file fuzzing module can be used to discover hidden (unliked) files on the server. It can be used in two distinct modes of operation: list mode and generator mode. The list mode relies on lists of common filenames (and directory names) and exhaustively tries to request all the entries. These lists are configured in program configuration. The user can specify a default extension to be added to all filenames (e.g. *.php* or *.html*).

The second mode uses a "generator" string to generate several filenames according to a given format. For instance the generator string "*image_[0-9]{2}.jpg*" is expanded as "*image_00.jpg*" ... "*image_99.jpg*". The advantage of this mode is that it allows to perform context aware fuzzing. Indeed it allows to incorporate some knowledge about the environment in fuzzing process.

The syntax of the generator parts to be expanded is: **[charset]{size}**. This means that such substrings will be replaced by all possible values and that the remaining of the generator string will remain unchanged. The table below summarizes the available character sets.

expression	elements
0-9	0123456789
a-z	abcdefghijklmnopqrstuvwxyz
A-Z	ABCDEFGHIJKLMNOPQRSTUVWXYZ
a-Z	abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

Note that an expression containing any unsupported character set will not be expanded but will be kept as is in filename (e.g. "*[x-5]{*3}" will be taken as is!).

Note also that you can combine several generating expressions in a generator string (e.g. the generator *image.[a-z]{1}_[0-9]{2}.jpg* is perfectly valid) however, due to multi-threaded design of the module, all the possible filenames have to be generated at once at the beginning of the scan. Thus be careful with the sizes unless you have a large amount of RAM (e.g. *[a-Z]{6}* => 120GB!!).

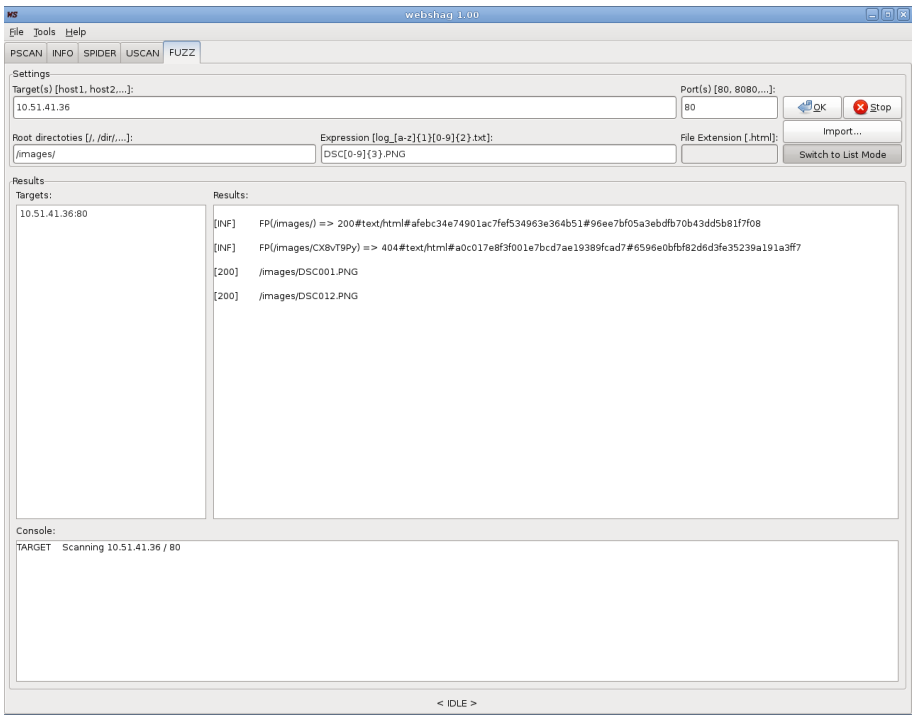
Finally, note that file fuzzer uses the same page fingerprinting technique than URL scanner to identify false positives.

/* Graphical User Interface */

To run the file fuzzing module, select the FUZZ tab and fill in the required parameters. The target(s) are given by the target and port fields and the root directory(ies) can be specified by the root field. All these fields can contain one or several (comma separated values).

When using the list mode, a default extension can be specified in *File Extension* field. When using generator mode, the generator string has to be inserted in *Expression* field.

Note that it is possible to directly import results from PSCAN, INFO and SPIDER modules using the *Import* button.



/* Command Line Interface */

To run this module in CLI mode, run webshag with **-C** option and set the **-m** option to **fuzz**. The mode of operation is specified by **-n** option (-n **list** or -n **gen**) and is completed by **-e** (extension) or **-g** (generator) depending on the chosen mode

```
./webshag.py -C -m fuzz -n list -e '.html' 10.51.41.36
```

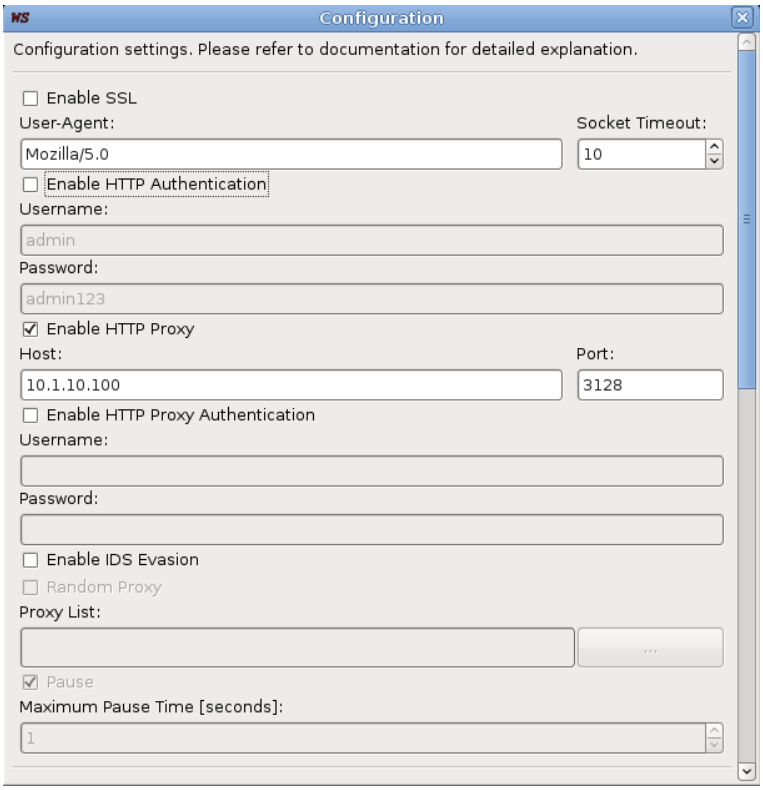
The optional port(s) and root directory(ies) can be specified using **-p** and **-r** options respectively.

```
./webshag.py -C -m fuzz -p 80,81 -r /log/ -n gen -g 'log_[a-z]{1}-[0-9]{2}.txt'
10.51.41.36
```

0x1000 Configuration

/* Configuration Menu */

All the application-wide settings (as well as variables that are not supposed to vary often) can be modified through program configuration menu (***Tools > Config...***). These include the various paths to database files, the HTTP settings (proxy, authentication, IDS evasion), the number of threads to use per module, the *Live Search AppID*, ...



When modifying configuration, **do not forget to confirm the changes by clicking Apply button at the bottom of the dialog window**.

/* Configuration File */

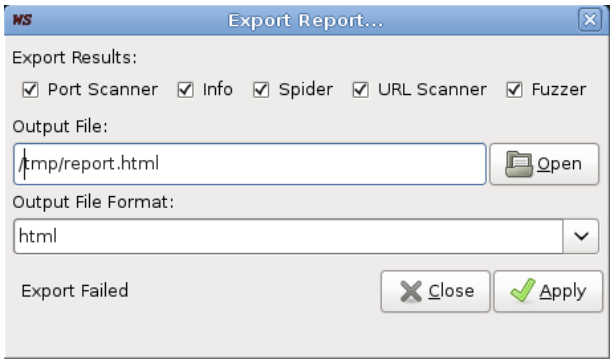
The configuration settings visible through configuration interface (in GUI mode) are actually stored in a configuration file (*<webshag directory>/config/webshag.conf*). When using webshag in CLI mode, these parameters can be updated by directly editing the configuration file.

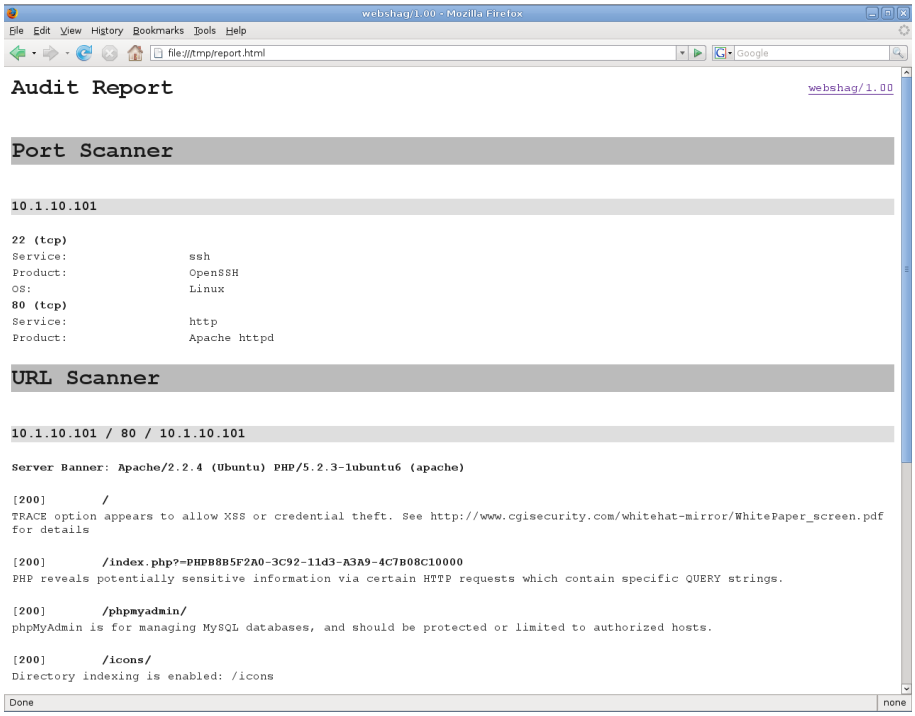
```
[...]
[module_urlscan]
scan_show_codes = 200,301,302,401,500
use_db_custom = True
use_db_nikto = True
scan_threads = 2

[core_http]
user_agent = Mozilla/5.0
ids_rp_list =
proxy_host =
[...]
```

0x1001 Reporting

/* Graphical User Interface */

Webshag allows exporting scan reports in three different formats: **XML**, **HTML** and **TXT**. This reports can summarize the results of a single module or of any combination of modules (including all). To export scan reports, use the ***File > Export...*** menu. Then simply choose the destination file, the elements to include in the report and the format.





/* Command Line Interface */

When using CLI, it is also possible to export scan reports (but summarizing only the results of a single module, as CLI mode only allows to run one module at a time). This is controlled by options **-x** (trigger export) **-o** (output format) and **-f** (file) For instance to run URL scanner, exporting the results as ah HTML file, use the following command.

```
./webshag.py -C -m uscan -x -o html -f '/tmp/wsreport.html' 10.1.10.101
```

0x1010 Acknowledgments

/* Thanks To */

Chris Sullo (CIRT.net[10]/OSVDB[11]) for granting us the right to use *Nikto* vulnerability database.

/* Credits */

Webshag is distributed with files from *Nikto* vulnerability database.

Webshag is distributed with (fuzzing) directory lists taken from *OWASP DirBuster*[12] Project.

Webshag Windows installer has been built using *py2exe*[13] and Jordan Russell's *Inno Setup*[14].

10 http://www.cirt.net/
11 http://osvdb.org/
12 http://www.owasp.org/index.php/Category:OWASP_DirBuster_Project
13 http://www.py2exe.org/
14 http://www.jrsoftware.org/isinfo.php