Hochschule
Augsburg University of
Applied Sciences

OWASP Offensive Web Testing Framework

Master Industrial Security

University of Applied Sciences Augsburg

Tool Evaluation

Table 1: Student author

Student	**Student ID**
Michael Wager	2081894

Abstract

Penetration testers often need to work under time pressure. Companies may have limited budget but still need high quality results quickly. Therefore it is a major goal for penetration testers to work as efficient as possible and integrate the use of sophisticated and comprehensive tools to reach this goal.

In the context of the lecture *Network Penetration Testing* at the University of Applied Sciences, Augsburg, a collection of such tools will be presented in this report: the Offensive Web Testing Framework (OWTF) from the Open Web Application Security Project (OWASP).

First, the motivation of this project will be explained followed by a feature overview. After a short technical analysis, installation instructions will be given. Afterwards a demonstration of the usage and the tools' possibilities will be presented. Finally, a conclusion related the productive use of the tool completes this document.

Contents

1 Motivation

As penetration testers often work under time pressure, good tools are needed to work efficiently and deliver high quality results. One such tool which will be analyzed in this document is the Offensive Web Testing Framework from OWASP. OWTF is a flagship project and open source tool initiated by the OWASP foundation and focuses on efficiency and alignment of security tests to security standards like the OWASP Testing Guide ([1]), the OWASP Top 10 ([2]), Penetration Testing Execution Standard (PTES) ([3]) and the NIST Technical Guide to Information Security Testing and Assessment (NIST SP 800-115 [4]). [5]

It tries to reach the following goals:

- See the big picture and think out of the box
- More efficiently find, verify and combine vulnerabilities
- Have time to investigate complex vulnerabilities like business logic
- Perform more tactical/targeted fuzzing on seemingly risky areas
- Demonstrate true impact despite the short timeframes we are typically given to test.

The tool is highly configurable and it is possible to create own plugins or add new tests in the configuration files without having any development experience. As it is aligned with to security standards, these standards should be introduced in the following sub sections.

1.1 OWASP Top 10

The Open Web Application Security Project (OWASP) is a non-profit foundation dedicated to improving the security of software. The **OWASP Top 10** is a standard awareness document for developers and web application security. It represents a broad consensus about the most critical security risks to web applications ([2]). It defines the top 10 most critical web application security risks, based on a consensus from security experts around the world. Every 2-3 years the list is updated according to changes related the field of security. OWTF contains plugins for each of this vulnerabilities.

1.2 OWASP Web Security Testing Guide

The **OWASP Testing Guide** (current iteration: v4) is a comprehensive guide related testing the security of web services and applications ([1]). It gives developers, testers and management a structured approach for questions like "When, Why & What to test" - specifically related to the Software Development Life Cycle (SDLC). Lots of plugins from OWTF are directly aligned to the tests defined in this guide.

1.3 Penetration Testing Execution Standard

The PTES ([3]) is a standard which defines the process of penetration testing. It is structured into 7 phases:

1. Pre-engagement Interactions
2. Intelligence Gathering
3. Threat Modeling
4. Vulnerability Analysis
5. Exploitation
6. Post Exploitation
7. Reporting

OWTF is trying to follow this standard and all of its phases.

1.4 NIST SP 800-115

The SP 800-115 Technical Guide to Information Security Testing and Assessment from the National Institute of Standards and Technology (NIST) (NIST SP 800-115) is a guide to the basic technical aspects of conducting information security assessments [4]. Some plugins from OWTF are also directly aligned to the tests defined in this guide.

2 Features

Some features provided by the tool:

- Resilience: If one plugin/command crashes, the tool will move on to the next
- Tests Separation
  - Passive : No traffic goes to the target
  - Semi Passive : Normal traffic to target
  - Active: Direct vulnerability probing
- Extensive REST API

- Aligned with OWASP Testing Guide(v3, v4), Top 10
- Simple intuitive web interface: manage large penetration engagements easily
- Report generation and export

2.1 Workflow

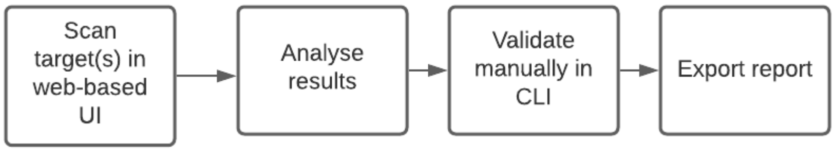


Figure 1: Workflow ([6])

As shown in figure 1, the basic workflow for usage of the tool is pretty straightforward:

1. Add target URL(s)
2. Run plugins
3. Analyse scan results
4. Copy commands from web UI to CLI
5. Run CLI commands
6. Analyse/verify the CLI results
7. Add notes in the web UI
8. Generate (export) the report

3 Technical analysis

This section gives a short overview of the technical specification, the tools running under the hood and the architecture of OWTF.

3.1 Tech specs

- python 2.7
- PostgreSQL database backend
- Installation in Kali linux (docker available too)

- REST API
- web based user interface to control the tool

3.2 Under the hood

- curl
- Arachni ([7]) - feature-full, modular, high-performance Ruby framework aimed towards helping penetration testers and administrators evaluate the security of modern web applications
- w3af ([8]) - Web Application Attack and Audit Framework
- Skipfish ([9]) - active web application security reconnaissance tool
- DirBuster ([10]) - multi threaded java application designed to brute force directories
- Other well known tools like nmap, hydra, wpscan, whatweb, wapiti, metasploit, etc...

3.3 Architecture

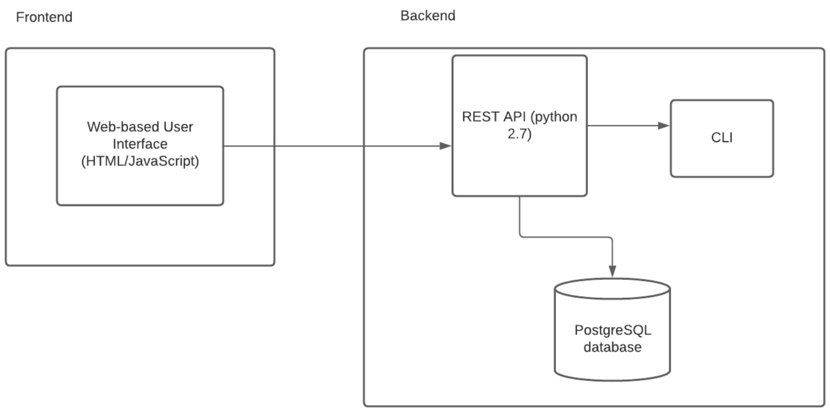
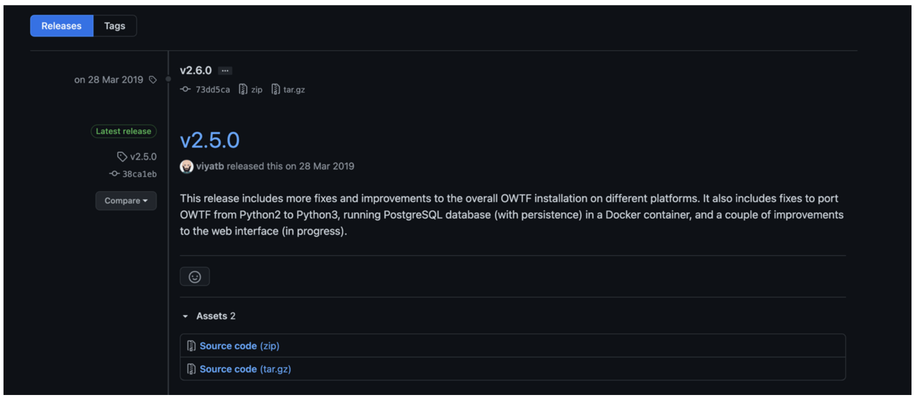


Figure 2: OWTF basic architecture diagram

As shown in figure 2, basic architecture is split into a **Frontend**, containing a web based user interface for controlling and using the tool written in JavaScript using the popular React.js Framework from Facebook. A simple Express.js based server is responsible for serving the built assets to the client. The **Backend** implements a REST-based API written in python and communicates with a PostgreSQL database and the command line for executing different commands under the hood when the user runs plugins. It is possible to think of OWTF as a kind of collection of sophisticated tools for penetration testers, put together behind a nice user interface for controlling everything.

4 Installation



on 28 Mar 2019

v2.6.0

73dd5ca zip tar.gz

v2.5.0

38ca1eb

v2.5.0

viyatb released this on 28 Mar 2019

This release includes more fixes and improvements to the overall OWTF installation on different platforms. It also includes fixes to port OWTF from Python2 to Python3, running PostgreSQL database (with persistence) in a Docker container, and a couple of improvements to the web interface (in progress).

Source code (zip)

Source code (tar.gz)

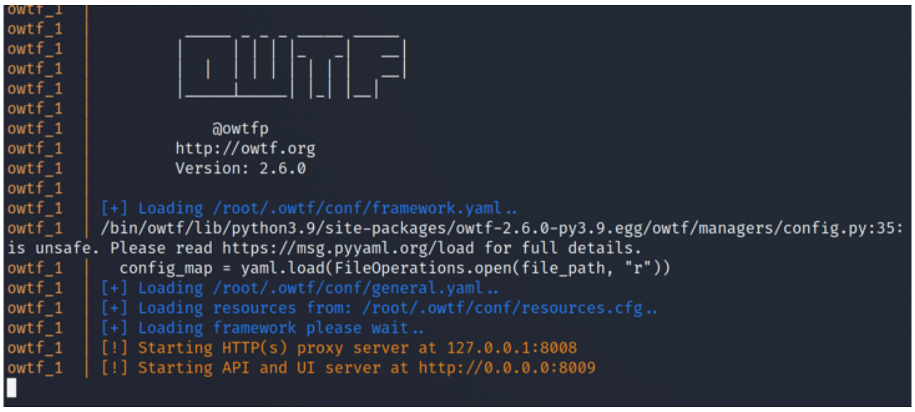
Figure 3: Latest release page on GitHub

It is possible to install all the dependencies on a kali linux machine, but the project repository also provides configuration files for a docker-based installation, so for reasons of simplicity the docker based installation will be used. Also the docker based installation is recommended by the contributors.

The latest official release of OWTF is from march 2019 as shown in figure 3. Installing from docker in this release was not possible due to docker image authentication errors, so instead the latest state of the repository "develop" branch was used. Following the installation instructions from the official project README file, the installation routine using docker is quite simple and can be executed using the following commands:

```
$ git clone https://github.com/owtf/owtf
$ cd owtf
$ make compose
```

The command **make compose** pulls a kali linux image and installs all the additional dependencies needed by the tool. After fetching and installing everything it finally starts the tool as shown in figure 4. Now the database is up and running, a proxy to the API and a webserver is listening for requests and the web based user interface can be accessed via a browser at the URL **http://localhost:8009/**.



```
owtf_1 |
owtf_1 |
owtf_1 |
owtf_1 |
owtf_1 |
owtf_1 |
owtf_1 |
owtf_1 |          @owtfp
owtf_1 |         http://owtf.org
owtf_1 |         Version: 2.6.0
owtf_1 |
owtf_1 | [+] Loading /root/.owtf/conf/framework.yaml..
owtf_1 | /bin/owtf/lib/python3.9/site-packages/owtf-2.6.0-py3.9.egg/owtf/managers/config.py:35:
is unsafe. Please read https://msg.pyyaml.org/load for full details.
owtf_1 |   config_map = yaml.load(FileOperations.open(file_path, "r"))
owtf_1 | [+] Loading /root/.owtf/conf/general.yaml..
owtf_1 | [+] Loading resources from: /root/.owtf/conf/resources.cfg..
owtf_1 | [+] Loading framework please wait..
owtf_1 | [!] Starting HTTP(s) proxy server at 127.0.0.1:8008
owtf_1 | [!] Starting API and UI server at http://0.0.0.0:8009
```

Figure 4: OWTF startup terminal output

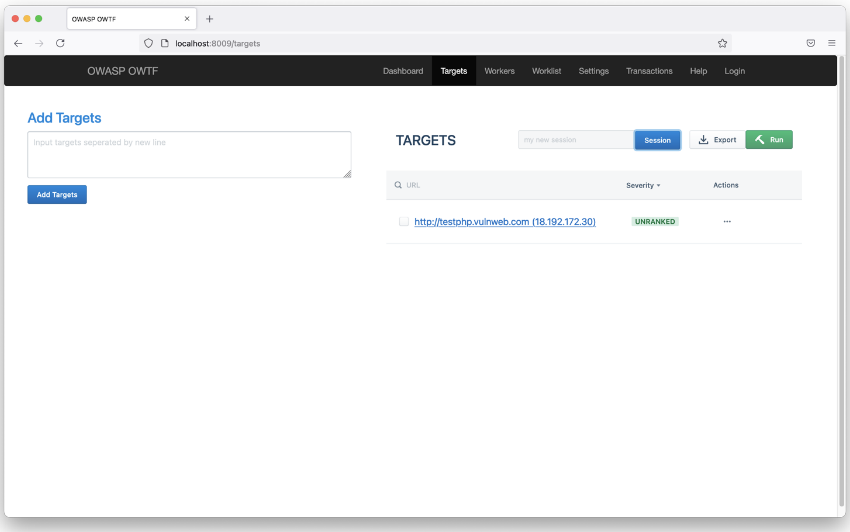
5 Usage

The following sections will walk the reader through the different screens according to the workflow described above.

5.1 User interface

After installing and running the tool, the web based user interface is available and can be accessed via a web browser. First we want to add a target to scan.

5.1.1 Adding targets



URL	Severity	Actions
http://testphp.vulnweb.com (18.192.172.30)	UNRANKED	···

Figure 5: Screenshot of the UI screen after adding a target URL

Figure 5 shows a screenshot of the UI after adding the target URL to our vulnerable target (**http://testphp.vulnweb.com/** [11])), a web application with a vast number of intended security vulnerabilities, developed for educational purposes. As this web application has lots of known vulnerabilities, the author decided to use this application as the target for all further demonstration.

5.1.2 Selecting and running plugins

TODO: how many plugins exist?

After adding a target, plugins may be started. Plugins have the following structure:

OWTF-IG-004 Web Application Fingerprint

Under the hood, plugins use command line tools to scan the target(s). For example, the plugin "Web Application Fingerprint" is using curl and whatweb to do its job. The phrase "OWTF-IG-004" is a reference to the OWASP Testing Guide ([1]) and a relation to the chapter "Information gathering" and section "Fingerprint Web Server".

Other plugins like for example **PTES-001 ftp** reference the Penetration testing execution standard ([3]). Behind the scenes, this plugin uses the "msfconsole" command line tool from the popular metasploit framework.

Figure 6 shows the modal window for selecting the plugins to execute against the target. It is possible to launch plugins in groups or individual. On individual selection, text based search is possible to explicitly search for vulnerabilities like "SQL injection" or "Cross site scripting".

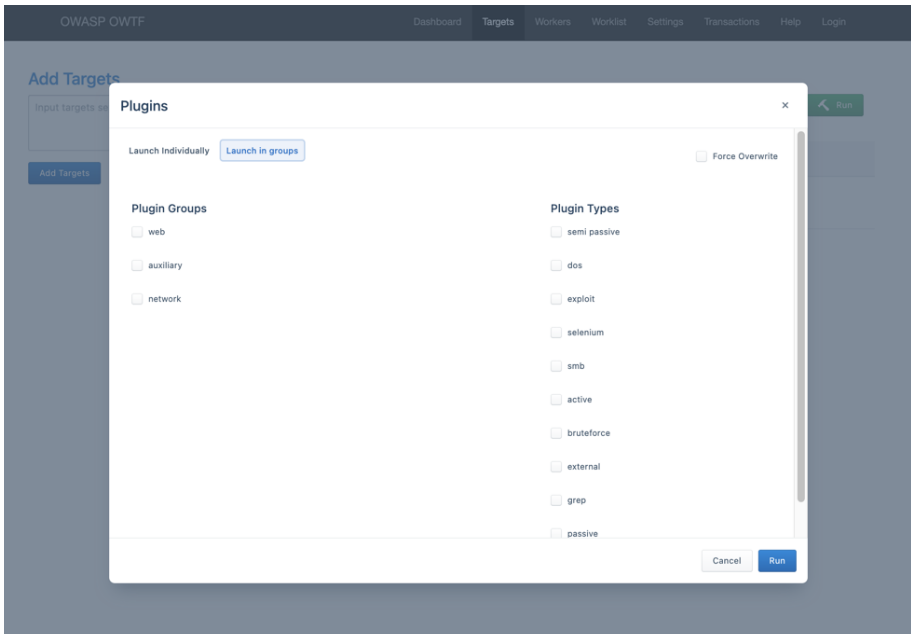


Figure 6: Screenshot of the UI modal window for selecting plugins

Plugin groups

This section explains the plugin groups and types based on the official documentation [12].

- **web**
  This group contains plugins related basic web capabilities like beb application fingerprint or SQL injection
- **auxiliary**
  Bruteforce and actual explotation tools
- **network**
  Network based scanning toole like DirBuster

Plugin types

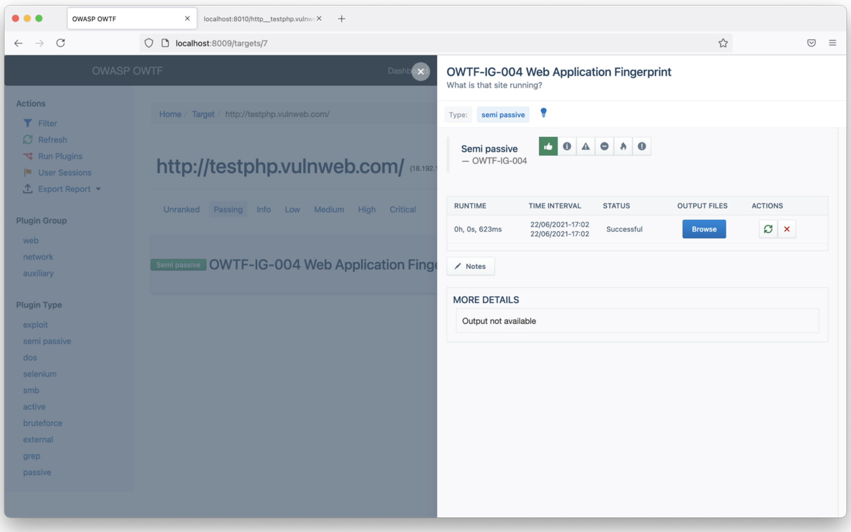
- **passive**
- **semi passive**
- **active**
- **dos**
- **exploit**

- **selenium**
- **bruteforce**
- **external**
- **grep**

First it would be nice to gather some basic information about our target like the IP address, the server on which the site is running on or some information about the framework and tools used by the application. Therefore we start the plugin "OWTF-IG-004 Web Application Fingerprint", which uses "whatweb" and "curl" under the hood".

5.1.3 Report page - analyzing the scan results

In Figure 7 the detail page of a target after running the fingerprint plugin is shown. The left sidebar provides options for filtering plugin results based on plugin groups and plugin types and generating and downloading the final report. In the middle, run plugins are rendered including the plugin type and the title of the plugin. Clicking on one of the plugins opens a flyover window on the right of the screen for viewing the plugin output or adding custom notes. It is possible to categorize the plugin in the following states: **Unranked**, **Passing**, **Info**, **Low**, **Medium**, **High** and **Critical**. This is helpful for filtering later and to better organize our work. Clicking on the **Browse** button opens the results of the scan in a new window.



RUNTIME	TIME INTERVAL	STATUS	OUTPUT FILES	ACTIONS
0h, 0s, 623ms	22/06/2021-17:02 22/06/2021-17:02	Successful	Browse	

Output not available

Figure 7: Screenshot of the UI page of a target after running the plugin "Web Application Fingerprint"

The result of the underlying "whatweb" command looks like this:

```
http://testphp.vulnweb.com/ [200 OK] ActiveX[D27CDB6E-AE6D-11cf-96B8-444553540000]
Adobe-Flash, Country[UNITED STATES][US]
Email[wvs@acunetix.com]
HTTPServer[nginx/1.19.0]
IP[18.192.172.30]
Object[http://download.macromedia.com/...
PHP[5.6.40-38+ubuntu20.04.1+deb.sury.org+1]
Script[text/JavaScript]
Title[Home of Acunetix Art]
UncommonHeaders[x-http-reason]
X-Powered-By[PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1]
nginx[1.19.0]
```

So among other information, this tells us that the site is served by an nginx server in version 1.19.0, it runs on PHP 5.6.40 and it is using JavaScript. Looks like a regular web application, so we are going to try some active vulnerability scanning. We select the plugin **OWTF-WVS-006 Skipfish Unauthenticated**. The plugin unfortunately runs into an error, but it tells us which command it wants to run. We now want to try the command manually.

5.1.4 Running and analyzing CLI commands

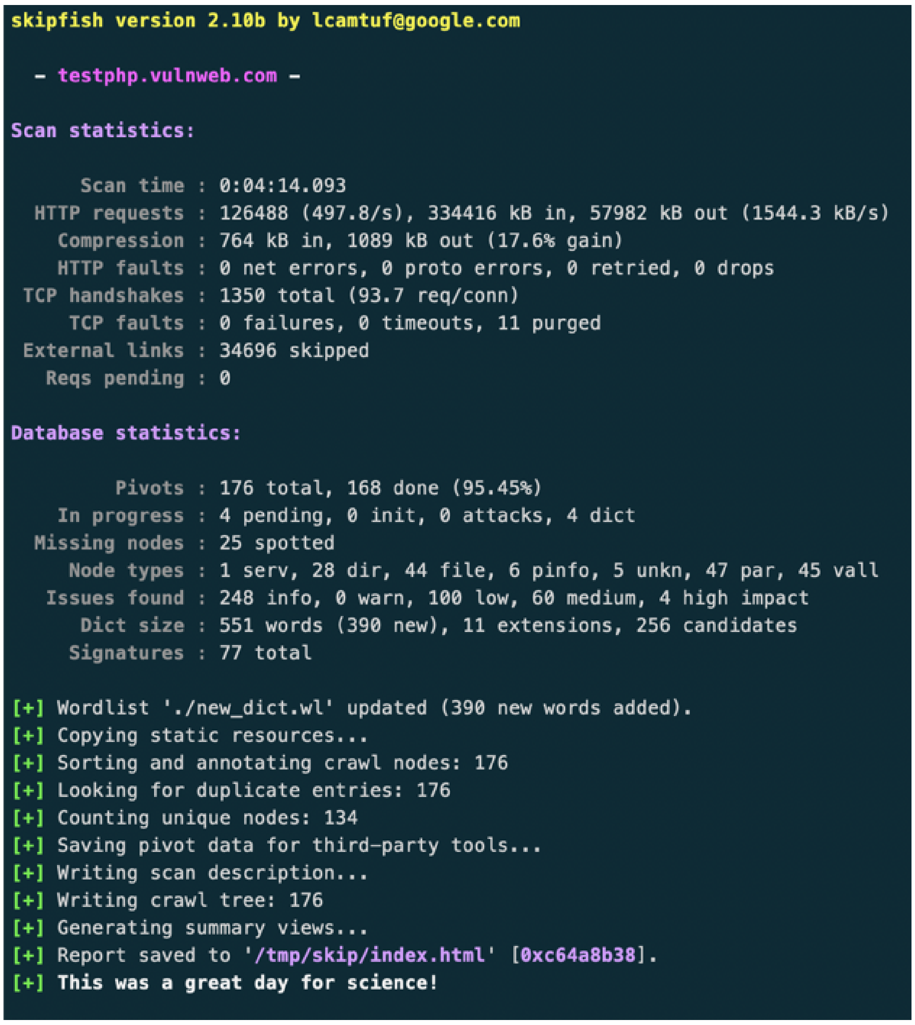
We open a terminal inside the docker container with the following command:

```
$ docker exec -it docker_owtf_1 bash
```

Now that we have a shell inside the kali linux docker container we can execute the commands just like OWTF would. After some tweaking we try the following:

```
touch new_dict.wl
skipfish -t 90 -i 90 -w 90 -f1000 -b f -o /tmp/skip -W ./
   new_dict.wl http://testphp.vulnweb.com
```

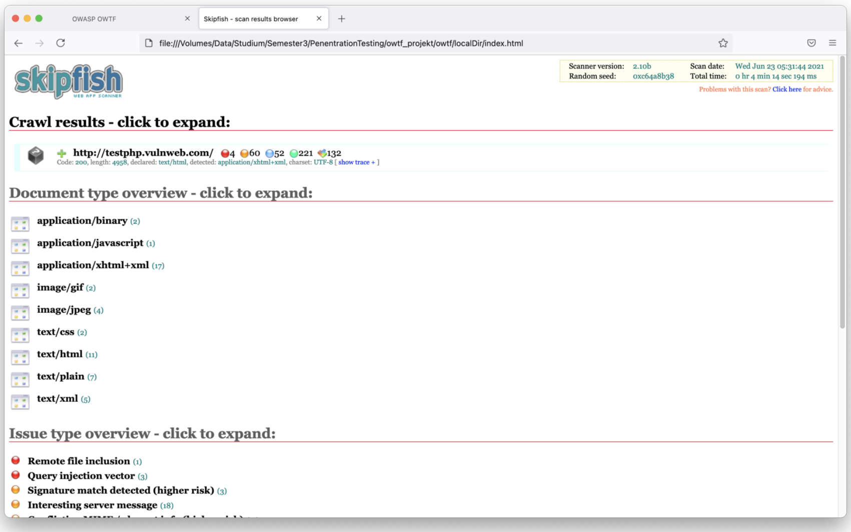
Figure 8 shows a screenshot of the command terminal output.



```
skipfish version 2.10b by lcamtuf@google.com

  - testphp.vulnweb.com -

Scan statistics:

      Scan time : 0:04:14.093
  HTTP requests : 126488 (497.8/s), 334416 kB in, 57982 kB out (1544.3 kB/s)
    Compression : 764 kB in, 1089 kB out (17.6% gain)
    HTTP faults : 0 net errors, 0 proto errors, 0 retried, 0 drops
 TCP handshakes : 1350 total (93.7 req/conn)
     TCP faults : 0 failures, 0 timeouts, 11 purged
 External links : 34696 skipped
   Reqs pending : 0

Database statistics:

         Pivots : 176 total, 168 done (95.45%)
    In progress : 4 pending, 0 init, 0 attacks, 4 dict
  Missing nodes : 25 spotted
     Node types : 1 serv, 28 dir, 44 file, 6 pinfo, 5 unkn, 47 par, 45 vall
   Issues found : 248 info, 0 warn, 100 low, 60 medium, 4 high impact
      Dict size : 551 words (390 new), 11 extensions, 256 candidates
     Signatures : 77 total

[+] Wordlist './new_dict.wl' updated (390 new words added).
[+] Copying static resources...
[+] Sorting and annotating crawl nodes: 176
[+] Looking for duplicate entries: 176
[+] Counting unique nodes: 134
[+] Saving pivot data for third-party tools...
[+] Writing scan description...
[+] Writing crawl tree: 176
[+] Generating summary views...
[+] Report saved to '/tmp/skip/index.html' [0xc64a8b38].
[+] This was a great day for science!
```

Figure 8: Screenshot of the Terminal output of the skipfish command

After the command run, it generated an html report which we download to our host for inspection. Figure 9 shows a screenshot of the report.


OWASP OWTF
Skipfish - scan results browser
file:///Volumes/Data/Studium/Semester3/PenentrationTesting/owtf_projekt/owtf/localDir/index.html
skipfish
WEB APP SCANNER

Scanner version: 2.10b   Scan date: Wed Jun 23 05:31:44 2021
Random seed: 0xc64a8b38   Total time: 0 hr 4 min 14 sec 194 ms
Problems with this scan? Click here for advice.

Crawl results - click to expand:

http://testphp.vulnweb.com/ 🔴4 🟠60 🔵52 🟢221 132
Code: 200, length: 4958, declared: text/html, detected: application/xhtml+xml, charset: UTF-8 [show trace +]

Document type overview - click to expand:

- **application/binary** (2)
- **application/javascript** (1)
- **application/xhtml+xml** (17)
- **image/gif** (2)
- **image/jpeg** (4)
- **text/css** (2)
- **text/html** (11)
- **text/plain** (7)
- **text/xml** (5)

Issue type overview - click to expand:

- 🔴 **Remote file inclusion** (1)
- 🔴 **Query injection vector** (3)
- 🟠 **Signature match detected (higher risk)** (3)
- 🟠 **Interesting server message** (18)

Figure 9: Screenshot of the skipfish report

Analyzing the report we see 16 XSS vectors. The web application allows users to list products filtering for a category. The parameter payload for the URL **http://testphp.vulnweb.com/listproducts.php** looks like this:

```
?cat=%22.$.htaccess.aspx--%3E%22%3E%27%3E%27%22%3Csfi000107v352075%3E
```

Navigating to that URL renders a MySQL exception, which is a hint that we may inject code:

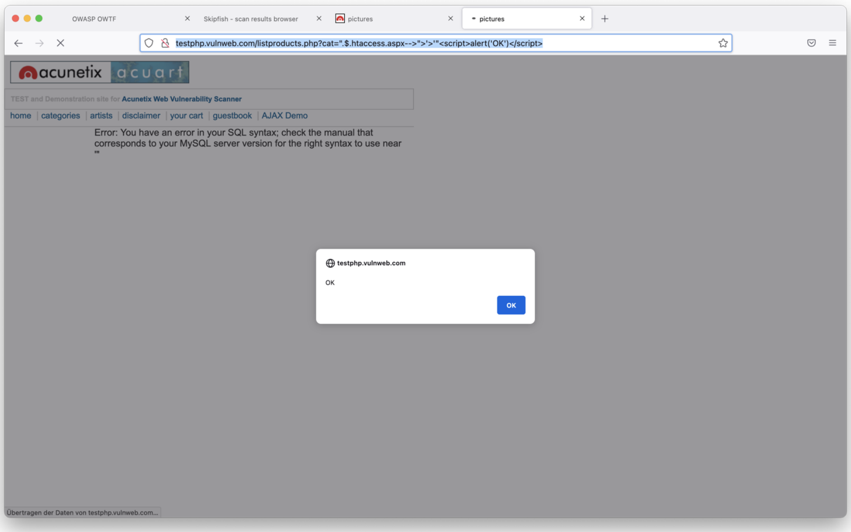
```
Error: You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to
use near '"' at line 1 Warning: mysql_fetch_array() expects
parameter 1 to be resource, boolean given in
/hj/var/www/listproducts.php on line 74
```

After some manual playing with this payload we found a working reflected XSS injection using the following payload:

```
?cat=".$.htaccess.aspx-->">'>'"<script>alert('OK')</
   script>
```

Here is a link to the full working exploit URL: Link

Figure 10 shows a screenshot of the successful exploit.



testphp.vulnweb.com/listproducts.php?cat=".$.htaccess.aspx-->">'>'"<script>alert('OK')</script>

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '"'

testphp.vulnweb.com

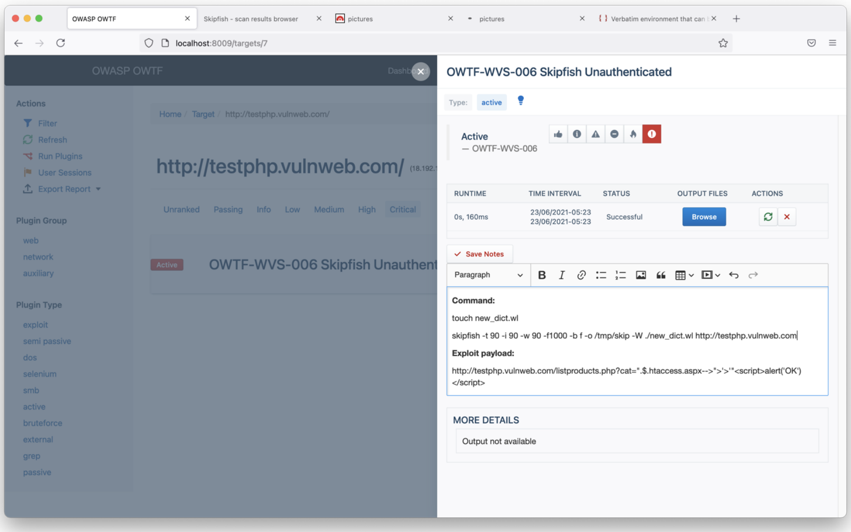
OK

OWASP OWTF
Skipfish - scan results browser
pictures
pictures
acunetix acuart
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
OK
Übertragen der Daten von testphp.vulnweb.com...

Figure 10: Successful exploit of a reflected XSS vector

5.1.5 Adding notes in the web UI

Now that we found a successful exploit we go back to the user interface and add some notes to our plugin and categorize the plugin run as critical. See Figure 11 for a detailed screenshot.



OWTF-WVS-006 Skipfish Unauthenticated

Type: active

Active
— OWTF-WVS-006

RUNTIME	TIME INTERVAL	STATUS	OUTPUT FILES	ACTIONS
0s, 160ms	23/06/2021-05:23 23/06/2021-05:23	Successful	Browse	

Command:

touch new_dict.wl

skipfish -t 90 -i 90 -w 90 -f1000 -b f -o /tmp/skip -W ./new_dict.wl http://testphp.vulnweb.com

Exploit payload:

http://testphp.vulnweb.com/listproducts.php?cat=".$.htaccess.aspx-->">'>'"<script>alert('OK')</script>

MORE DETAILS

Output not available

Figure 11: Screenshot of adding manual notes to a plugin

5.1.6 Generating (export) the report

Unfortunatly, exporting the report seems to be broken. After some analysis of the inner workings, an issue on GitHub was created by the author:

https://github.com/owtf/owtf/issues/1149

5.1.7 Dashboard

Finally we take a look at the dashboard which could also be very helpful for reporting our work. Figure 12 shows a screenshot of the dashboard. As seen in the screenshot, the used categorization inside the plugin views are represented here in a nice looking chart.

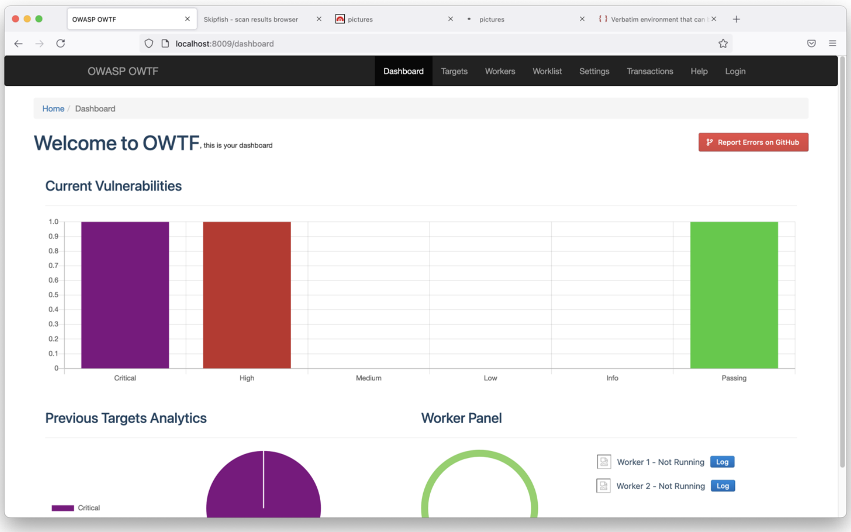


Figure 12: Screenshot of the dashboard

6 Conclusion

The first impression of the tool including its motivation, goals and features looked really impressive. Having a tool supporting the complete process of penetration testing, aligned to security standards and containing support for automatic report generation really could enhance the life of a security tester. Setup and installation was quite simple and pretty straightforward. Supporting a docker based installation deserves some bonus points too. But after exploring and trying to use the tool in a productive way, **using** the overall experience was pretty disappointing.

Lots of features just don't work as expected and running into an error. The following list gives an overview and documents the author's findings:

- **Starting plugins**
  Starting plugins, especially multiple at the same time or in groups, randomly leads to an error with the message "Unable to add Error: Bad Request". The POST request returns a status code 400 Bad Request with the error "Plugin list should not be empty", even though the checkboxes for selecting plugin (groups) was selected before clicking the "Run" button.

- **Looking at the plugin results**
  Looking at the plugin results is not possible in most cases. Clicking on the "Browse" button to view a plugins' results navigates to an URL like **http://localhost:8010/null/** with a status code 404 Not Found.

- **Report generation and download**
  Clicking on the "Export Report" button leads to a white screen and a JavaScript error in the DevTools console saying "The menu has no menu items". An issue at the git repository on GitHub was created by the author: https://github.com/owtf/owtf/issues/1149

- **Tools not found**
  Sometimes when running a plugin, the tools does not seem to be installed inside the docker based virtual machine (e.g. theHarvester: command not found)

Nevertheless it should be kept in mind that OWTF is a free, open source solution containing lots of work (the repository was created in January 2012), as can be seen looking at the git source repository history. Also, it is under active development and there is a web page for addressing and optimizing some pain points in the context of the google summer of code 2021 [13]. For example they want to enhance the user experience and are searching for developers implementing some overall improvements. The author had some discussion on the development slack channel and the maintainers responded quickly, were thankful for reporting these issues and want to address them very soon.

Finally, in the opinion of the author, at current stage there are too many issues and lots of debugging needed, so the tool just not seem to be ready for production use. But as the tool is still under active development, is is possible that the contributors add more documentation and fix the found issues in the foreseeable future and it is definitely worth **following** the development of this promising solution.

7 Abbreviations

NIST	National Institute of Standards and Technology
OWASP	Open Web Application Security Project
OWTF	Offensive Web Testing Framework
PTES	Penetration Testing Execution Standard
SDLC	Software Development Life Cycle

8 References

[1] OWASP, *OWASP Testing Guide*, 11.06.2021. [Online]. Available: `https://github.com/OWASP/wstg/tree/master/document/`.

[2] OWASP, *OWASP Top 10*, 11.06.2021. [Online]. Available: `https://owasp.org/www-project-top-ten/`.

[3] *Penentration Testing Execution Standard*, 11.06.2021. [Online]. Available: `http://www.pentest-standard.org/index.php/Main_Page/`.

[4] *NIST SP 800-115 Information Security Testing and Assessment*, 29.06.2021. [Online]. Available: `https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf`.

[5] *OWTF at GitHub*, 11.06.2021. [Online]. Available: `https://github.com/owtf/owtf/`.

[6] J. Brennen, *Automating Security Testing with the OWTF by Jerod Brennen*, 11.06.2021. [Online]. Available: `https://www.youtube.com/watch?v=03o2FdNK4IY`.

[7] *Arachni - Web Application Security Scanner Framework*, 11.06.2021. [Online]. Available: `https://www.arachni-scanner.com/`.

[8] *w3af - Web Application Attack and Audit Framework*, 11.06.2021. [Online]. Available: `http://w3af.org/`.

[9] *Skipfish - active web application security reconnaissance tool*, 11.06.2021. [Online]. Available: `https://tools.kali.org/web-applications/skipfish/`.

[10] *DirBuster - multi threaded java application designed to brute force directories and files names on web/application servers*, 11.06.2021. [Online]. Available: `https://tools.kali.org/web-applications/dirbuster/`.

[11] *example PHP application, intentionally vulnerable to web attacks*, 11.06.2021. [Online]. Available: `http://testphp.vulnweb.com/`.

[12] *OWTF official documentation*, 11.06.2021. [Online]. Available: `https://owtf.readthedocs.io/`.

[13] *Google Sommer of code 2021 - Ideas*, 11.06.2021. [Online]. Available: `https://owasp.org/www-community/initiatives/gsoc/gsoc2021ideas/`.