

Vulnerability Scanners in the Web Application Space

Benjamin Moeller

EECS 465

University of Kansas

3/1/2022

1 Introduction

Vulnerability scanners have the unique property of allowing automated scans against a certain system, network, or application. This extends to web applications as well, and there are a variety of different types of scanners that have both comparable and differing functionality. This paper will primarily focus on a Web Application scanner called Wapiti, but other tools will be discussed as well in order to compare functionality. The rest of the article is structured as follows. The next section gives the motivation as to why Vulnerability Scanners are important. Section 3 will contain discussion around Wapiti and other tools similar to it. Section 4 will show a demo of how Wapiti can be used to scan a web application. Section 5 will go over some of the useful commands that Wapiti can offer, and then Section 6 will conclude the paper.

2 Motivation

Studying vulnerability scanners is important because they allow us to validate the security of our applications. For the purpose of

this paper, we are more interested in accessing the validity of online web applications. Making sure our web applications are secure is pivotal due to the sheer load of requests and interactions users make with websites every day. It is in the best interest of companies around the world to keep their sites both online and secure in the hopes of retaining customers. Even just one security breach or outage caused by an attack can turn a customer away forever, therefore it is ideal for a business to keep their applications online as much as possible. These attacks can be prevented ahead of time by using vulnerability scanners to detect any defects within a web application, and from the output given by a tool a plan can be made to correct whatever vulnerability is present. For this reason it is both necessary to know how and when to use a given tool.

3 Tool Analysis

The goal of this section is to describe the features and functionality of the Wapiti tool, as well as a couple of similar tools for comparison.

3.1 Wapiti

Wapiti is an open source web application scanner that specializes in black-box scans, written in python. According to its official website, it does not study the source code of a site, but instead “[crawls] the webpages of the deployed webapp, looking for scripts and forms where it can inject data”[1]. It has a variety of modules that can detect vulnerabilities in areas such as SQL Injection, Cross Site Scripting (XSS), and in recent updates Log4Shell vulnerability detection has even been added. It has more than just these, and they can be easily be changed and customized through the use of flags. After the completion of the tool, it will generate a report in either json, html, txt, or xml format depending on the user’s choice. In this report it will tell you the number of vulnerabilities found for each module used, descriptions of the types of scans used, and even some recommended solutions to try and fix some of the vulnerabilities.

3.2 Vega

Vega is another open source web application scanner. Like Wapiti it also has modules that can scan for SQL Injection and Cross Site Scripting vulnerabilities. It “works on Linux, OSX and Windows platform since its detection modules are written in JavaScript and additional modules can be developed by using its API’s”[4]. The main differences between Wapiti and Vega come from the fact that Vega has its own Graphical User Interface (GUI) and Application Programming Interface (API) infrastructure. Due to this, Vega is a bit more user friendly out the box, and if one is knowledgeable enough to make their own JavaScript de-

tection modules that option is available to them. Wapiti has the advantage of still receiving updates to the tool as recent as February 2022, while Vega hasn’t been updated since June 2016 according to its last github commit. Both tools have their own strengths though, and I would say Vega specializes in its user friendliness and customizability.

3.3 w3af

The final tool being discussed in this paper is w3af, which like Wapiti is also written in python. It has both a GUI and Shell interface, and like the previous two tools it features a plethora of different modules for a user to scan with. It also advertises itself as being “easy to use and extend”[3], which could be helpful for any developers looking to create their own types of scans. It looks very convenient for doing repeated testing, as different types of scanning configurations can be saved for later use.

3.4 Concluding Tool Discussion

Overall all three of these tools are similar in terms of the types of scans they can apply to web applications. Wapiti is an intuitive command line tool that has a variety of scanning modules that are still updated to this day. Vega on the other hand hasn’t had recent updates like Wapiti, but is still powerful and features a nice graphical interface. A good compromise between the two is w3af, which features both a command line and graphical interface, as well as a large variety of scanning modules. The next section will give an in depth demo of how to use Wapiti.

4 Wapiti Demo

4.1 Experimental Setup

In order to setup the experiment I used a Kali 2021 (Version 5.10.0) virtual machine with an Open Web Application Security Project Broken Web Applications (OWASPBWA) virtual machine (Version 2.6.32) to serve as a target for the experiment. Both machines were run using Oracle VM Virtual Box, and both were also set to Host-Only Adapter mode to ensure a safe environment. Wapiti comes preinstalled with Kali Linux, so it wasn't necessary to do any installation.

4.2 Launching a scan

To start first make sure that your OWASPBWA virtual machine is launched. If you don't know the IP Address of the OWASPBWA system, you can use the `ifconfig` command inside the terminal to get the address. Once you know the address and your OWASPBWA machine is on, launch the Kali virtual machine and open the command prompt. For this test we are going to use the default scanning tests provided by Wapiti (If you want to learn more about some helpful Wapiti commands, continue to section 5). In order to do this we are going to use the following command:

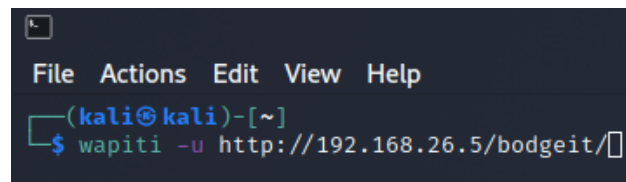
```
wapiti -u [SITE TO SCAN]
```

In this case the site to scan is going to be inside of the OWASPBWA virtual machine. It has a variety of different web applications on it that are intentionally left vulnerable. For this test we are going to target the site known as the bodgeit store, which according to the developer's github page contains significant vulnerabilities in "Cross Site Scripting, SQL Injection,

Hidden Content"[5], as well as other vulnerabilities. So in order to reach this webpage, it is only necessary to provide:

```
http:// + TARGET IP ADDRESS + /bodgeit
```

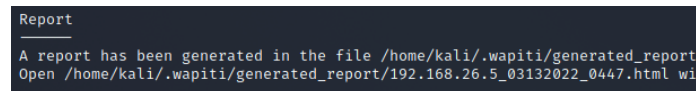
If we were targeting an actual webpage it wouldn't be necessary to deal with finding the site on an ip address, but since this test uses two machines in Host-Only mode this is necessary to provide the correct path to the webpage. Here is an example of what the full command will look like in the terminal:

A terminal window with a dark background. The title bar shows a window icon and the text "File Actions Edit View Help". The prompt is "(kali㉿kali)-[~]". The command being entered is "\$ wapiti -u http://192.168.26.5/bodgeit/".

```
(kali㉿kali)-[~]  
$ wapiti -u http://192.168.26.5/bodgeit/
```

Table 1: Default command for Wapiti

When this command is run, it will start all the default scans provided by Wapiti. This is excluding some of the other modules, but they can be included if the user desires (discussed in next section). Once this command starts running, the tool will start going through all of the different modules included. After that the tool will finish, and it will output a report of what the tool found in html format by default.

A screenshot of a report generated by Wapiti. The title is "Report". The text says: "A report has been generated in the file /home/kali/.wapiti/generated_report. Open /home/kali/.wapiti/generated_report/192.168.26.5_03132022_0447.html wi".

```
Report  
A report has been generated in the file /home/kali/.wapiti/generated_report  
Open /home/kali/.wapiti/generated_report/192.168.26.5_03132022_0447.html wi
```

Table 2: Wapiti report location

The report is sent by default to a folder in the home directory, but the tool will let you know where the file was sent. In this report, it first starts by providing a summary of how many vulnerabilities of each type were found by the scanner.

Summary

Category	Number of vulnerabilities found
Backup file	0
Blind SQL Injection	0
Weak credentials	0
CRLF Injection	0
Content Security Policy Configuration	2
Cross Site Request Forgery	0
Potentially dangerous file	0
Command execution	0
Path Traversal	0
Htaccess Bypass	0
HTTP Secure Headers	8
HttpOnly Flag cookie	3
Open Redirect	0
Secure Flag cookie	3
SQL Injection	0
Server Side Request Forgery	0
Cross Site Scripting	1
XML External Entity	0
Internal Server Error	160

Table 3: Wapiti report summary

If a vulnerability was found, you can look into the details and see description of the vulnerability, and if applicable the HTTP request or cURL command used by the tool.

HTTP Secure Headers

Description

HTTP security headers tell the browser how to behave when handling the website's content.

Vulnerability found in /bodgeit/

Description	HTTP Request	cURL command line
X-Frame-Options is not set		

Table 4: Wapiti vulnerability found

After going through all of the vulnerabilities listed in the report, this final line will give some tips if it can on how to fix any of the vulnerabilities. It will also list any references it thinks it would be helpful.

Solutions

More information about the error should be found in the server logs.

References

- [Wikipedia: List of 5xx HTTP status codes](#)
- [OWASP: Improper Error Handling](#)

Table 5: Wapiti helpful tips

Using Wapiti with the default options is a good way to scan a web application without too much difficulty. Wapiti also has a variety of different flags and options available to it though, and those will be discussed in the next section.

5 Wapiti options

Wapiti has a variety of different options that give users more customizability when launching a scan. The following subsections will list some of the flags that I found to be the most important when I read through the manual.

5.1 The `—scope` flag

With the scope flag, the user can define how deep the scanner should go. By default the “url” value is used, which only scans the base url provided to the tool. If “folder” is inputted into the tool instead, it “will scan and attack every URL starting with the base URL value” [6]. If “domain” is passed in, it “will scan and attack every URL whose domain name match the one from the base URL” [6]. Finally, and most dangerous, the “punk” option will scan every URL that it can find while the tool is running. While these could be useful, they should be used sparingly as if not careful the tool could scan a webpage that it is not authorized to scan.

5.2 The -m flag (modules)

Wapiti has many different modules that it can use for scanning purposes, many of which it uses by default. If a user wants to be more selective with what modules are used, they can call the -m flag with the modules they want separated by commas. The command -list-modules can also be used to see what the current modules available are.

5.3 The -a flag

The -a flag or -auth-type is a flag to set the login credentials of a particular site if needed for a scan. They should be input “in the form login%password (% is used as a separator)” [6]

5.4 The -f flag

The -f or -format flag is a helpful flag to determine the output type of the final report. By default it makes a html file for the report, but json, txt, and xml files are all also supported.

6 Conclusion

In conclusion, Wapiti is a helpful vulnerability scanner that can help check the security status of webpages. While it doesn't have a GUI, its command line interface is very easy to use while also providing a large amount of customizability options. It has some very comparable competition in Vega and w3af, and while the tools have similar functionalities, it is nice that there are different options for users to pick based on what suites their needs. I think wapiti is a very versatile tool that can provide a variety of scan types, but if a user feels more comfortable with a tool that has a GUI, w3af and Vega are avail-

able as well. It was definitely impressive to see how Wapiti had Log4Shell scans available after what happened earlier this year, and I think if it continues to keep the tool updated as time goes on it could be a tool to keep an eye on.

References

- [1] Wapiti: A Free and Open-Source web-application vulnerability scanner. (2022, February 23rd). Wapiti. <https://wapiti-scanner.github.io/>
- [2] Kritikos, K., Magoutis, K., Papoutsakis, M., & Ioannidis, S. (2019, September 1). A survey on vulnerability assessment tools and databases for cloud-based web applications. ScienceDirect. <https://www.sciencedirect.com/science/article/pii/S2590005619300116>
- [3] w3af - Open Source Web Application Security Scanner. (2013). W3af. <https://w3af.org/>
- [4] Tundis, A., Mazurczyk, W., & Muhlhauser, M. (2018, August 27). A review of network vulnerabilities scanning tools: types, capabilities and functioning. ACM Digital Library. <https://doi.org/10.1145/3230833.3233287>
- [5] Bodgeit. (2018, January 8). <https://github.com/psiinon/bodgeit>
- [6] Wapiti Manual. (2021, May). <https://wapiti.sourceforge.io/wapiti.1.html>