A Comparison of the Effectiveness of Different Web Vulnerability Application Scanners

Year 3 Mini Project

Alexandra Cherry
1700315@uad.ac.uk

CMP320: Ethical Hacking 3

BSc Ethical Hacking - Year 3

2019/20

.

Abstract

This paper reviews a number of popular scanners which are free to use, easy to locate online and with a relatively low skill barrier to entry, comparing ease of use, comprehensiveness of the report and scan, and ease of accessing and interpreting the results.

The scanners, Zed Attack Proxy, Wapiti and Nikto2, were run using default configurations, generating an html report, and, in the case of Wapiti and Zed Attack Proxy, were run at a more intense level for a more thorough evaluation of the scanners in question.

This investigation revealed that Open Web Application Security Project's (OWASP) Zed Attack Proxy is easiest to use of the three tested. This is mainly due to the highly intuitive graphical user interface and provides the most detailed and technical reports. The reports show where the vulnerabilities are and provide some information on how to fix them, which is useful to a web application administrator.

Nikto2 came appeared to be the second most useful scanner as it was focused more on reconnaissance and searching for default configurations in the web application than providing a detailed scan of the vulnerabilities.

Wapiti was the least most useful scanner despite producing highly intuitive reports as it missed multiple vulnerabilities that were caught by Zed Attack Proxy.

.

Contents

.

Table of Figures

1 INTRODUCTION

1.1 BACKGROUND

Web applications are dynamic interactive applications accessed through a web browser (eg webmail, social media, Wikipedia, online banking, etc).

Web applications have become integral to modern life and are increasingly prevalent as more commercial processes and individuals' personal data are online (banking, shopping, paying bills/utilities etc.). It would be difficult to run many businesses without them and they make consumers' lives easier.

Due to the high occurrence of web applications in modern life, companies that run web applications need to be on top of their web application security on an ongoing basis to reduce the impact of an attack.

These applications are attractive to malicious actors due to potential easy access to personal and private data when compared to a corporate network. By their nature, web applications are usually public facing, whereas corporate networks can be run behind powerful firewalls and do not need to be accessible to the general public.

The consequences of an attack on a web application can include data breaches – which can lead to fines, negative publicity in media, damaged reputation and loss of sales due to customer distrust of the company; loss of access to finances for customers which can affect payment of bills/rent/mortgages; fraud due to customers' banking and personal information being stolen; there is a potential breach of corporate network if the web application is hosted on the same network – this is inadvisable but does happen; and financial losses due to customers choosing to take their business elsewhere. It can take companies years to recover frm loss of reputation and loss of public confidence. This can have significant effects on share prices among other matters.

According to OWASP (OWASP, n.d.), some of the most common security risks to a web application include: code injection – this happens when untrusted user input is sent to an interpreter as part of a command or query, this can cause the execution of code or provide access to secure data; broken authentication – this is caused by improperly implemented authentication, potentially compromising/allowing the bypass of admin credentials; sensitive data exposure – incorrectly protected sensitive data can be stolen by attackers for use in fraud including credit card fraud and identity theft; broken access control – a significant number of web applications do not utilise the principle of least privilege which results in accounts having the ability to access sensitive data that they should not; cross-site scripting (XSS) – Cross site scripting (XSS) attacks are an injection attack in which malicious scripts are inserted into web applications that do not sanitise the input from the user (OWASP, 2018);

and insufficient logging & monitoring – when combined with insufficient incident response can allow malicious actors to maintain access and/or attack further systems potentially leading to the deletion/modification of sensitive data.

Part of this security process for safeguarding data is by regularly and actively checking the web application for vulnerabilities and risks to mitigate against, thus preventing/reducing the severity of attacks. This is vital, but does not always happen, rendering the web application more vulnerable to malicious attacks.

Web application scanners are one method of finding vulnerabilities in web applications. Web application scanners are applications which are used to detect security vulnerabilities in web applications by examining them. (OWASP, n.d.)

The scanners compared in this paper are Zed Attack Proxy (ZAP) – created and managed by Open Web Application Security Project (OWASP); Wapiti is a web app scanner created by Nicolas Surribas; Nikto2 is a web app scanner associated with CIRT and sullo on GitHub.

The criteria used to evaluate and compare the scanners were: the ease of use of the scanner – how easy and intuitive is the initial use; the features of the scanner – is it modular, what report/export options are there, what does the scanner focus on including foot-printing and reconnaissance, cross-site scripting, SQL injection, directory browsing and path traversal; and the categories and number of vulnerabilities discovered.

1.2 AIMS

The aim of this investigation is to provide a balanced and detailed comparison of Zed Attack Proxy, Nikto2 and Wapiti. The scanners were compared on the ease of use of the scanner and interpretation of the report, the total and types of vulnerabilities discovered and the main focus of the scanner.

2 Procedure and Results

2.1 Setup

The web application tested, Astley's Autos, was provided by my lecturer (Colin McLean) and was hosted using Uniform Server – a WAMP (Windows: Apache + MySQL + PHP) web hosting solution for windows (Anon., n.d.) – on a Windows 10 virtual machine. The web application was a mock-up of a car sales web application and had multiple features typical of many web applications.

This paper compared three open-source scanners. The first two scanners used in this paper, Nikto2 and Zed Attack Proxy, are available with a fresh installation of Kali Linux (Offensive Security, n.d.) and are maintained on GitHub. Wapiti, the third scanner used in this paper, is available for download from source forge (Wapiti, n.d.). These are scanners which are commonly used and readily accessible to anyone who wishes to use them.

For full reports of Wapiti and Zed Arrack Proxy please view https://github.com/Aliisace/Mini-Project-Scanner-Reports (GitHub Repository with reports) as they are too long to include in this paper.

2.2 Procedure Part 1 –Zed Attack Proxy

2.2.1 Zed Attack Proxy Introduction

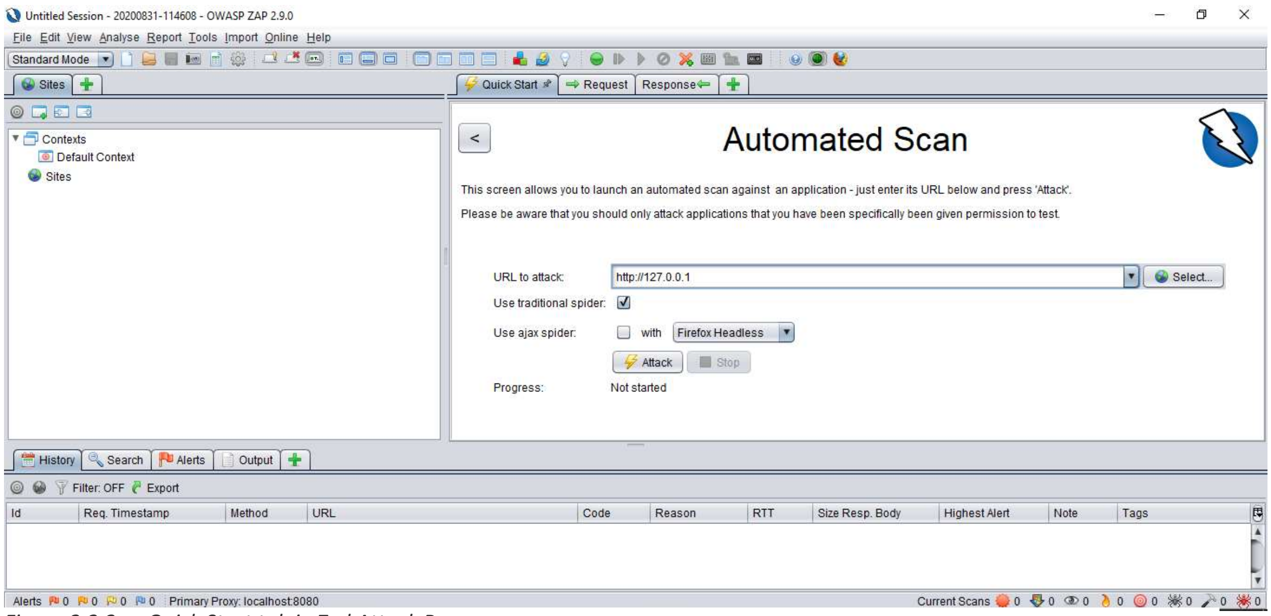
Zed Attack Proxy is an open-source web application penetration tool maintained by the Open Source Web Application Project (OWASP) (OWASP, n.d.) and is available on GitHub.

Zed Attack Proxy functions as a man in the middle proxy between the web browser and the web application intercepting (and modifying) traffic between the browser and application before forwarding the packets to their destination.

The web application was scanned using the automated scan feature to expedite testing. Zed Attack Proxy has four modes: Attack, Standard, Safe and Protected. The modes were used in this paper were Standard Mode - this mode allows users to perform all attacks on pages that are in scope and Attack Mode - this mode performs active scans on all pages that are in scope as they appear. Safe and protected mode were not used because they are less powerful (as they are designed to be more cautious) so have more limited scope. Safe mode does not allow the user to do anything dangerous and Protected mode limits certain risky actions to those nodes explicitly in the scope.

Before scans were performed, the VM was (deliberately) disconnected from the internet to remove the risk of the scanners attacking anything outside of the target web application.

2.2.2 Zed Attack Proxy Procedure



This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://127.0.0.1

Use traditional spider: ✓

Use ajax spider: ☐ with Firefox Headless

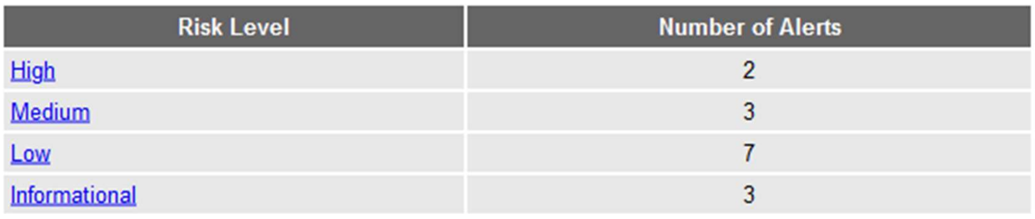
Progress: Not started

Figure 2.2.2-a: Quick Start tab in Zed Attack Proxy

Both of the scans done with Zed Attack Proxy were started using the “Quick Start” tab in the GUI, as seen in figure 2.2.2-a. This was intuitive and easy to use.

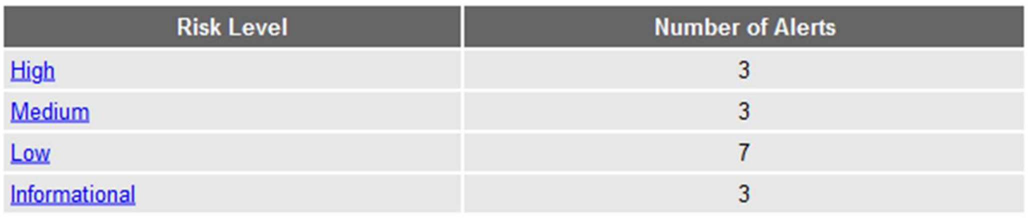
Both scans used the default options, with the only difference between the two scans being one used Standard mode and the other used Attack mode. This setting can easily be changed using the drop down menu in the top left of the scanner graphical user interface.

2.2.3 Results



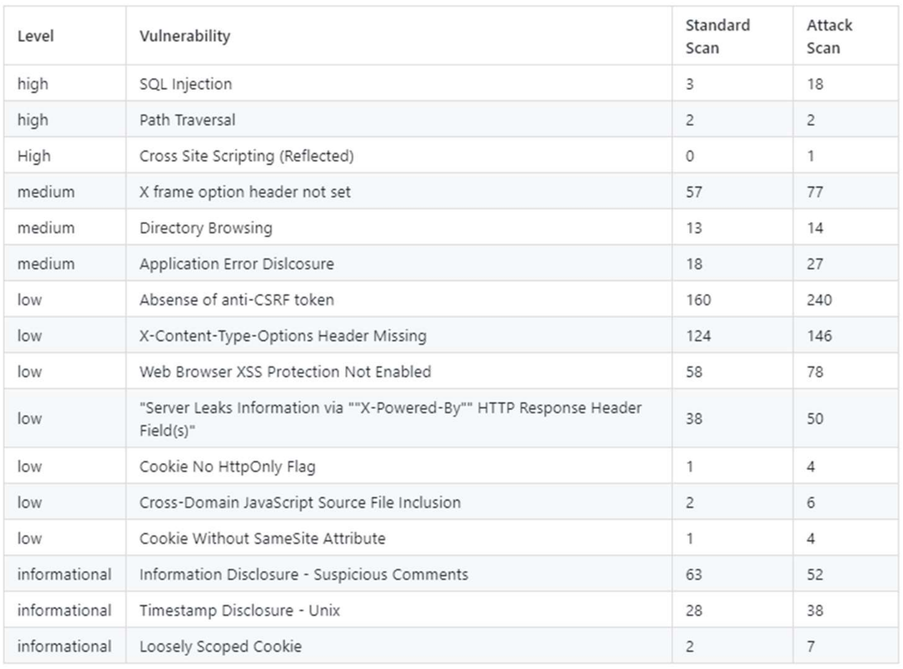
Risk Level	Number of Alerts
High	2
Medium	3
Low	7
Informational	3

Figure 2.2.33-a: Results of Standard Zed Attack Proxy Scan



Risk Level	Number of Alerts
High	3
Medium	3
Low	7
Informational	3

Figure 2.2.33-b: Results of Attack Zed Attack Proxy Scan



Level	Vulnerability	Standard Scan	Attack Scan
high	SQL Injection	3	18
high	Path Traversal	2	2
High	Cross Site Scripting (Reflected)	0	1
medium	X frame option header not set	57	77
medium	Directory Browsing	13	14
medium	Application Error Dislcosure	18	27
low	Absense of anti-CSRF token	160	240
low	X-Content-Type-Options Header Missing	124	146
low	Web Browser XSS Protection Not Enabled	58	78
low	"Server Leaks Information via ""X-Powered-By"" HTTP Response Header Field(s)"	38	50
low	Cookie No HttpOnly Flag	1	4
low	Cross-Domain JavaScript Source File Inclusion	2	6
low	Cookie Without SameSite Attribute	1	4
informational	Information Disclosure - Suspicious Comments	63	52
informational	Timestamp Disclosure - Unix	28	38
informational	Loosely Scoped Cookie	2	7

Figure 2.2.33-c: Breakdown and Comparison of Standard and Attack Zed Attack Proxy Scans

Zed Attack Proxy categorises vulnerabilities based on the alert level which is helpful for administrators of the web application as they can see what vulnerabilities have the highest risk associated with them. This enables efficient use of potentially limited resources targeting the most significant vulnerabilities

As shown in figures 2.2.3-a, 2.2.3-b and 2.2.3-c, the Standard scan and the Attack scan found different results for all the vulnerabilities discovered. The most dramatic difference is with the number of SQL Injection vulnerabilities found, with the Attack scan finding six times more instances than the Standard scan. The Attack scan also found a cross site scripting vulnerability that was not found by the Standard scan. This is potentially significant. Attack scan is more thorough and produces more detailed results, but this mode took approximately an hour longer. Unless the scope of Attack Scan is tightly controlled, it has the potential to attack an unwanted target.

2.3 Procedure Part 2 – Nikto2

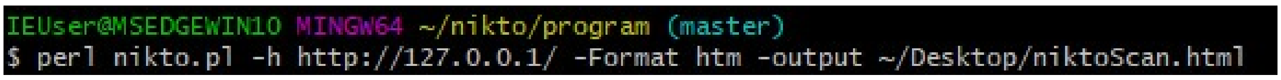
2.3.1 Nikto Introduction

The second scanner used in this paper is Nikto2, an open-source web application scanner commonly used to carry out detailed scans against web servers. It is frequently used to uncover default content and perform simple foot-printing scans of the server hosting the web application (CIRT, n.d.). Foot-printing scans are reconnaissance scans used to gather information about a target before doing a more in-depth scan/attack.

Nikto2 performs detailed scans for multiple problems, including checks for potentially dangerous files, outdated server versions, existence of index files, contents of *robots.txt* (a file used by crawlers that search the web which tells the bots what pages they are not allowed to access to prevent them appearing in search results), server options, and will attempt to identify installed web servers and software. (sullo, 2018)

Nikto2's default setting is to use all the plugins, but this can be customised

2.3.2 Nikto2 Procedure



```
IEUser@MSEDGEWIN10 MINGW64 ~/nikto/program (master)
$ perl nikto.pl -h http://127.0.0.1/ -Format htm -output ~/Desktop/niktoScan.html
```

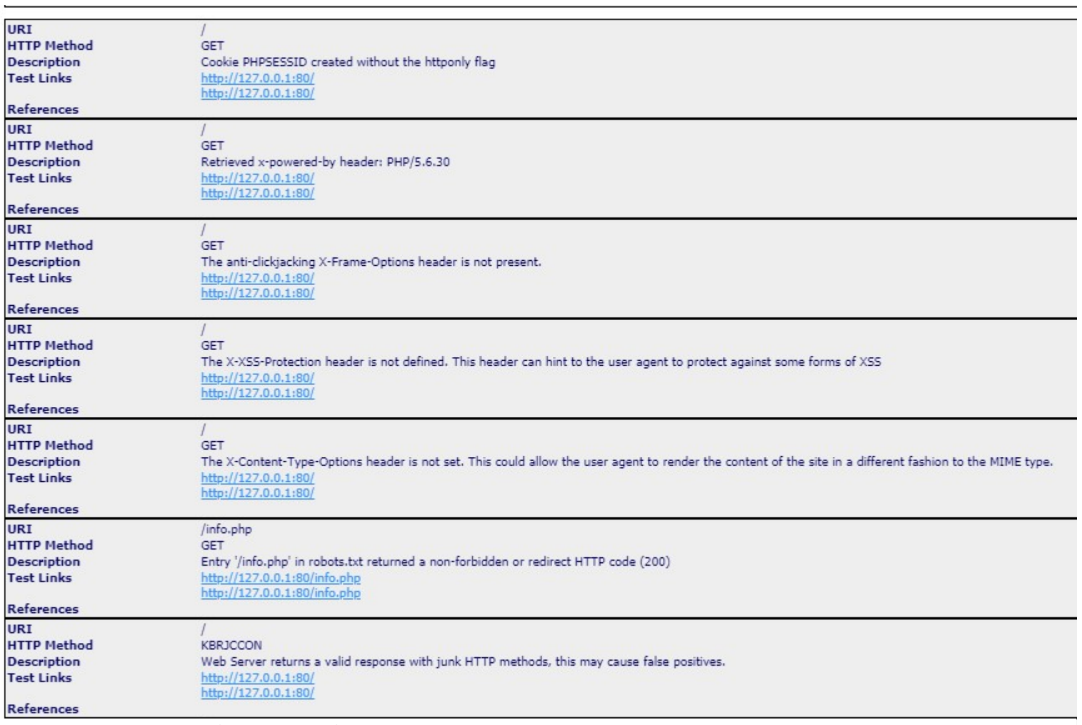
Figure 2.3.2-a: Nikto2 Scan

The Nikto2 scan was run using perl in Git Bash for windows, as shown in figure 2.3.2-a. Git Bash is an emulation of Bourne Again Shell for Windows which comes packaged with git.

The Nikto2 scan used the default configuration (using all the plugins) and generated a report of the results in an html format.

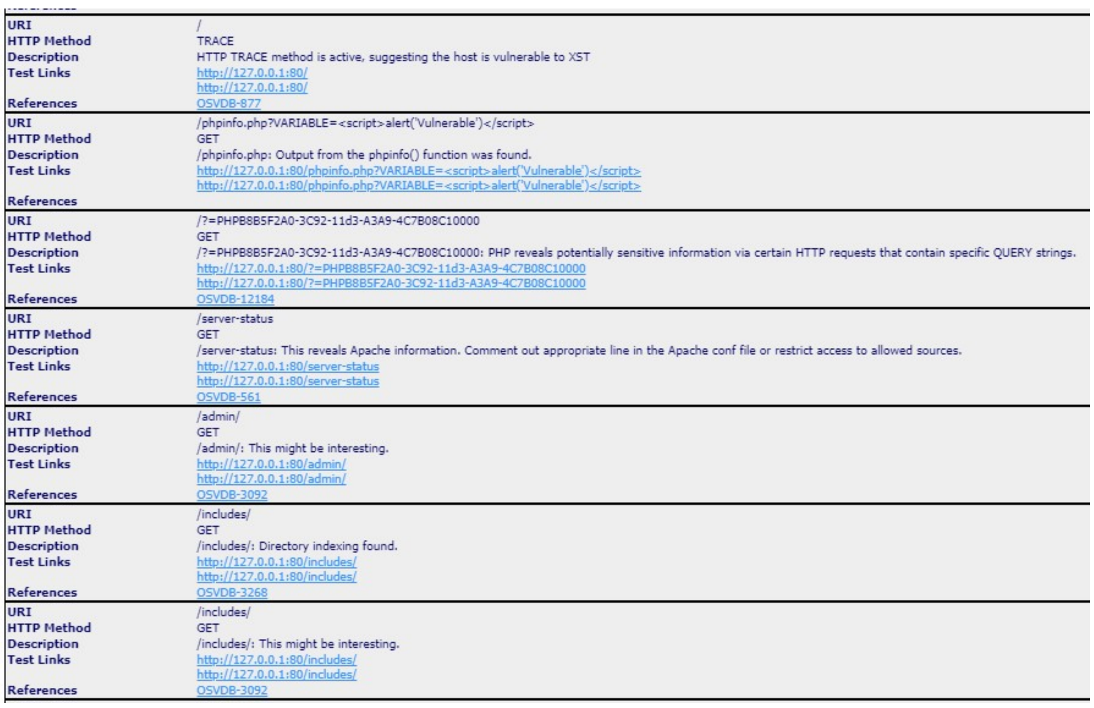
The report included information on potentially interesting pages, directory listing and default configuration of the server and/or web application.

2.3.3 Results



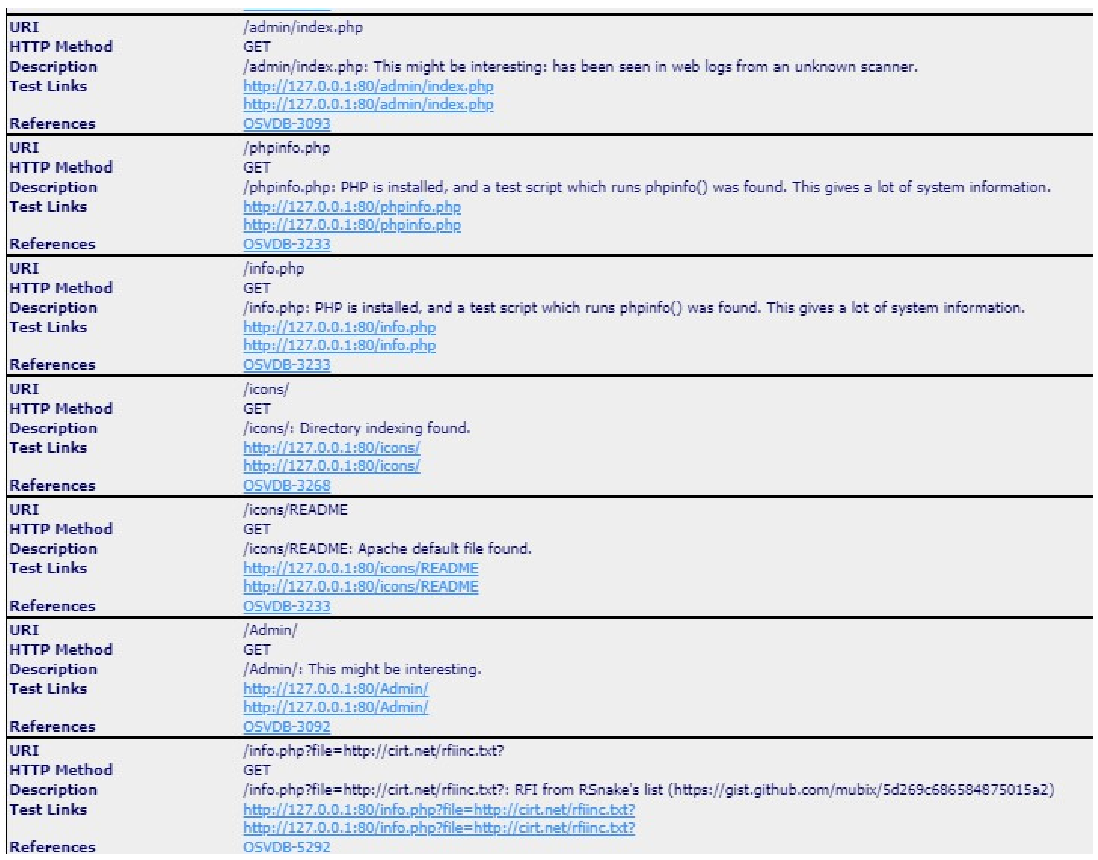
URI	/
HTTP Method	GET
Description	Cookie PHPSESSID created without the httponly flag
Test Links	http://127.0.0.1:80/ http://127.0.0.1:80/
References	
URI	/
HTTP Method	GET
Description	Retrieved x-powered-by header: PHP/5.6.30
Test Links	http://127.0.0.1:80/ http://127.0.0.1:80/
References	
URI	/
HTTP Method	GET
Description	The anti-clickjacking X-Frame-Options header is not present.
Test Links	http://127.0.0.1:80/ http://127.0.0.1:80/
References	
URI	/
HTTP Method	GET
Description	The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
Test Links	http://127.0.0.1:80/ http://127.0.0.1:80/
References	
URI	/
HTTP Method	GET
Description	The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type.
Test Links	http://127.0.0.1:80/ http://127.0.0.1:80/
References	
URI	/info.php
HTTP Method	GET
Description	Entry '/info.php' in robots.txt returned a non-forbidden or redirect HTTP code (200)
Test Links	http://127.0.0.1:80/info.php http://127.0.0.1:80/info.php
References	
URI	/
HTTP Method	KBRJCCON
Description	Web Server returns a valid response with junk HTTP methods, this may cause false positives.
Test Links	http://127.0.0.1:80/ http://127.0.0.1:80/
References	

Figure 2.3.2-a: Part 1 of the results for Nikto



URI	/
HTTP Method	TRACE
Description	HTTP TRACE method is active, suggesting the host is vulnerable to XST
Test Links	http://127.0.0.1:80/ http://127.0.0.1:80/
References	OSVDB-877
URI	/phpinfo.php?VARIABLE=<script>alert('Vulnerable')</script>
HTTP Method	GET
Description	/phpinfo.php: Output from the phpinfo() function was found.
Test Links	http://127.0.0.1:80/phpinfo.php?VARIABLE=<script>alert('Vulnerable')</script> http://127.0.0.1:80/phpinfo.php?VARIABLE=<script>alert('Vulnerable')</script>
References	
URI	/?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000
HTTP Method	GET
Description	/?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
Test Links	http://127.0.0.1:80/?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000 http://127.0.0.1:80/?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000
References	OSVDB-12184
URI	/server-status
HTTP Method	GET
Description	/server-status: This reveals Apache information. Comment out appropriate line in the Apache conf file or restrict access to allowed sources.
Test Links	http://127.0.0.1:80/server-status http://127.0.0.1:80/server-status
References	OSVDB-561
URI	/admin/
HTTP Method	GET
Description	/admin/: This might be interesting.
Test Links	http://127.0.0.1:80/admin/ http://127.0.0.1:80/admin/
References	OSVDB-3092
URI	/includes/
HTTP Method	GET
Description	/includes/: Directory indexing found.
Test Links	http://127.0.0.1:80/includes/ http://127.0.0.1:80/includes/
References	OSVDB-3268
URI	/includes/
HTTP Method	GET
Description	/includes/: This might be interesting.
Test Links	http://127.0.0.1:80/includes/ http://127.0.0.1:80/includes/
References	OSVDB-3092

Figure 2.3.3-b: Part 2 of the results for Nikto



URI	/admin/index.php
HTTP Method	GET
Description	/admin/index.php: This might be interesting: has been seen in web logs from an unknown scanner.
Test Links	http://127.0.0.1:80/admin/index.php
	http://127.0.0.1:80/admin/index.php
References	OSVDB-3093
URI	/phpinfo.php
HTTP Method	GET
Description	/phpinfo.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of system information.
Test Links	http://127.0.0.1:80/phpinfo.php
	http://127.0.0.1:80/phpinfo.php
References	OSVDB-3233
URI	/info.php
HTTP Method	GET
Description	/info.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of system information.
Test Links	http://127.0.0.1:80/info.php
	http://127.0.0.1:80/info.php
References	OSVDB-3233
URI	/icons/
HTTP Method	GET
Description	/icons/: Directory indexing found.
Test Links	http://127.0.0.1:80/icons/
	http://127.0.0.1:80/icons/
References	OSVDB-3268
URI	/icons/README
HTTP Method	GET
Description	/icons/README: Apache default file found.
Test Links	http://127.0.0.1:80/icons/README
	http://127.0.0.1:80/icons/README
References	OSVDB-3233
URI	/Admin/
HTTP Method	GET
Description	/Admin/: This might be interesting.
Test Links	http://127.0.0.1:80/Admin/
	http://127.0.0.1:80/Admin/
References	OSVDB-3092
URI	/info.php?file=http://cirt.net/rfiinc.txt?
HTTP Method	GET
Description	/info.php?file=http://cirt.net/rfiinc.txt?: RFI from RSnake's list (https://gist.github.com/mubix/5d269c686584875015a2)
Test Links	http://127.0.0.1:80/info.php?file=http://cirt.net/rfiinc.txt?
	http://127.0.0.1:80/info.php?file=http://cirt.net/rfiinc.txt?
References	OSVDB-5292

Figure 3.3.3-c: Part 3 of the results for Nikto

The results of the Nikto2 scan show a significant amount of default configuration in the web application, which makes it easier for an attack to succeed. Default configuration is generally well known and there can be many known ways of bypassing default configuration. It also shows that the web application lists the contents of directories that do not have an index page configured (directory listing).

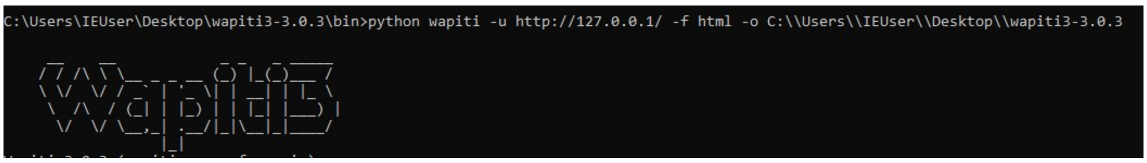
2.4 PROCEDURE PART 3 – WAPITI

2.4.1 Wapiti Introduction

The third and final scanner used in this paper was Wapiti. It is a modular black box web application scanner that searches for input fields in the application before fuzzing them by injecting payloads to check for vulnerabilities. Wapiti also checks for potential security breaches due to errors in the configuration of *.httaccess* files and/or backups stored on the server. (Wapiti, 2020)

Wapiti has several levels of intensity of scan - paranoid, sneaky, polite, normal, aggressive, and insane. Scans were performed at normal and aggressive strength – insane strength was attempted but did not complete after 48 hours, so was stopped and excluded from results. Scans which take too long are not user friendly and may be less likely to be used.

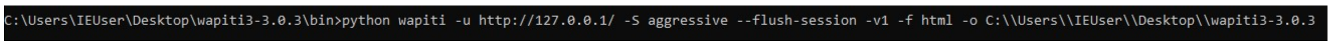
2.4.2 Wapiti Procedure



```
C:\Users\IEUser\Desktop\wapiti3-3.0.3\bin>python wapiti -u http://127.0.0.1/ -f html -o C:\\Users\\IEUser\\Desktop\\wapiti3-3.0.3

 __    __            _ _   _ _____
/ / /\ \ \__ _ _ __ (_) |_(_)___ /
\ \/  \/ / _` | '_ \| | __| | |_ \
 \  /\  / (_| | |_) | | |_| |___) |
  \/  \/ \__,_| .__/|_|\__|_|____/
              |_|
```

Figure 2.4.22-2.4.2-b: Basic Wapiti Scan



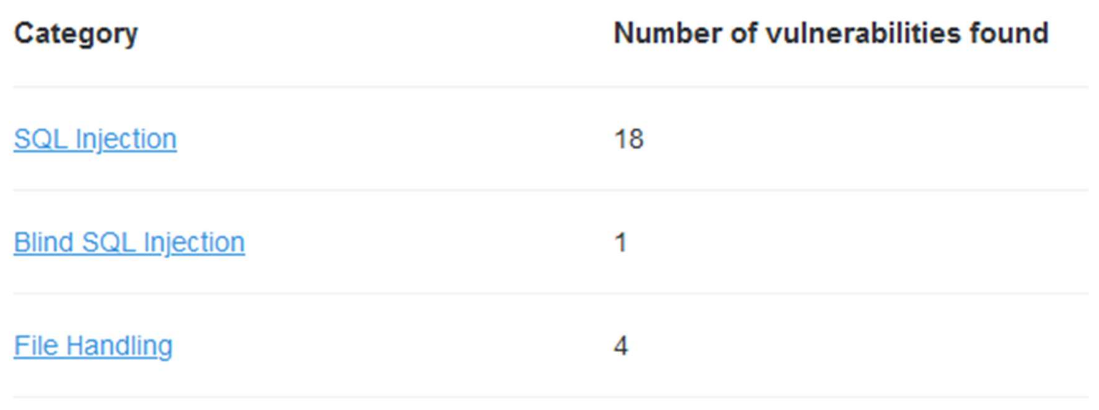
```
C:\Users\IEUser\Desktop\wapiti3-3.0.3\bin>python wapiti -u http://127.0.0.1/ -S aggressive --flush-session -v1 -f html -o C:\\Users\\IEUser\\Desktop\\wapiti3-3.0.3
```

Figure 2.4.1-a: Aggressive intensity Wapiti scan

Wapiti was run using the windows command line (figure 2.4.2-a). The scanner was run twice – on both occasions with the default configurations and generated a report of the results in an html format.

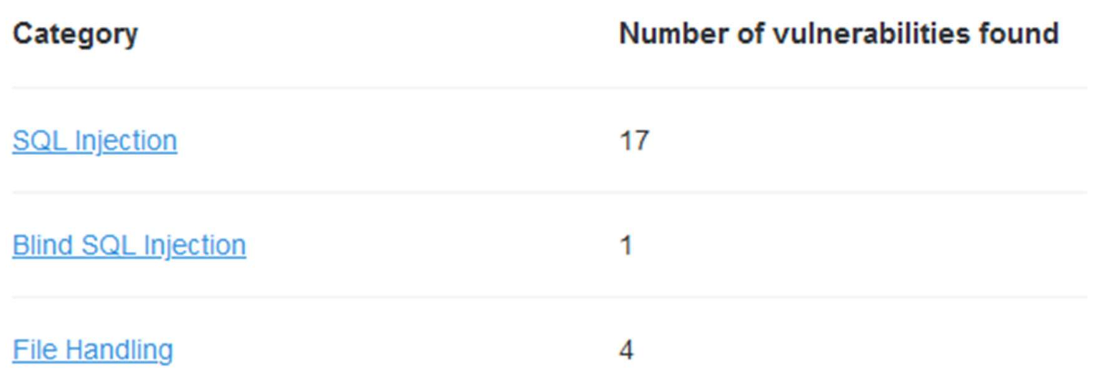
The first scan was run at normal intensity and the second scan was run at aggressive intensity (figure 2.4.2-b).

2.4.3 Results



Category	Number of vulnerabilities found
SQL Injection	18
Blind SQL Injection	1
File Handling	4

Figure 2.4.2-a: Results of Normal Wapiti Scan



Category	Number of vulnerabilities found
SQL Injection	17
Blind SQL Injection	1
File Handling	4

Figure 2.4.3-b: Results of Aggressive Strength Scan

The results of a normal strength scan with Wapiti (figure 2.4.1-a) show that in total, Wapiti discovered 23 vulnerabilities in the web application – 18 SQL Injection, 1 Blind SQL Injection and 4 File Handing errors.

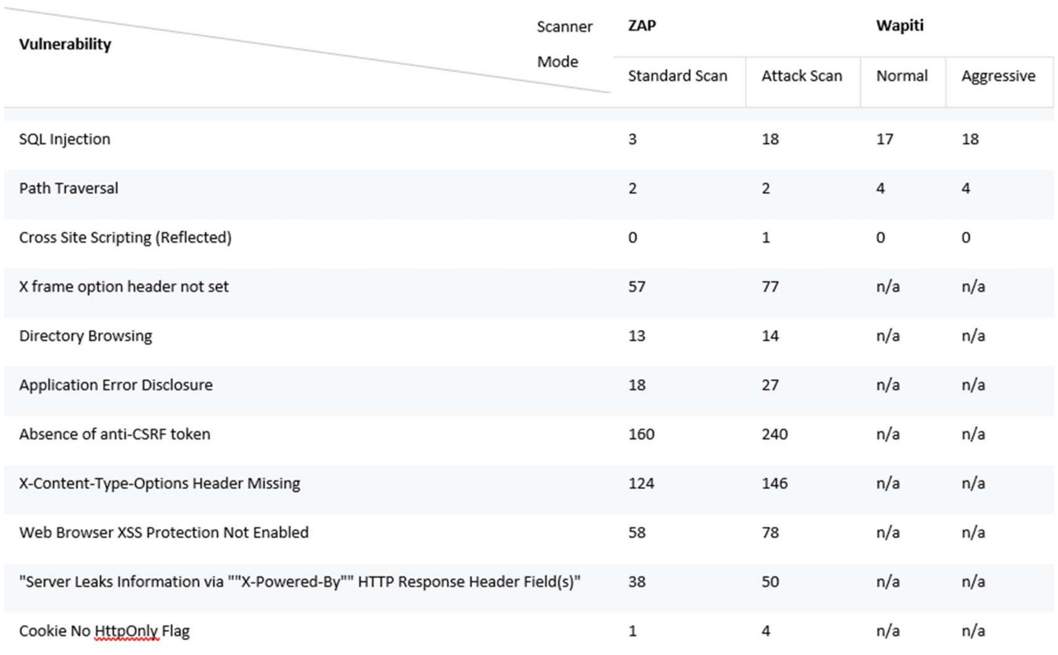
The results of an aggressive strength scan of Wapiti (figure 2.4.1-b) show that in total, Wapiti detected 17 SQL Injection, 1 Blind SQL Injection and 4 File Handing errors – a total of 22 vulnerabilities.

This result would appear counter intuitive with the aggressive scan detecting fewer vulnerabilities than the normal scan.

3 DISCUSSION

3.1 COMPARISON

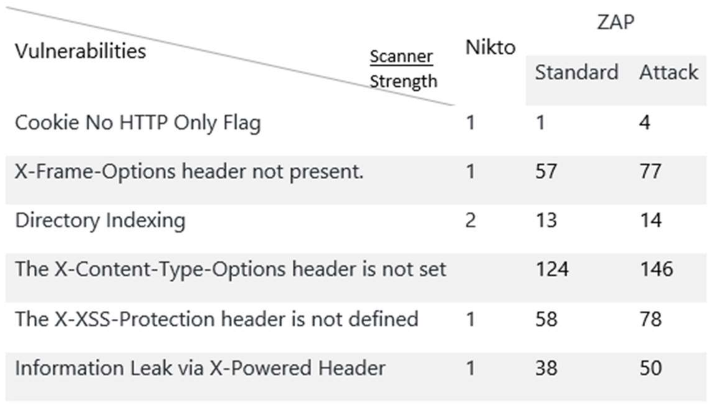
3.1.1 Results



Vulnerability / Scanner Mode	ZAP		Wapiti	
	Standard Scan	Attack Scan	Normal	Aggressive
SQL Injection	3	18	17	18
Path Traversal	2	2	4	4
Cross Site Scripting (Reflected)	0	1	0	0
X frame option header not set	57	77	n/a	n/a
Directory Browsing	13	14	n/a	n/a
Application Error Disclosure	18	27	n/a	n/a
Absence of anti-CSRF token	160	240	n/a	n/a
X-Content-Type-Options Header Missing	124	146	n/a	n/a
Web Browser XSS Protection Not Enabled	58	78	n/a	n/a
"Server Leaks Information via ""X-Powered-By"" HTTP Response Header Field(s)"	38	50	n/a	n/a
Cookie No HttpOnly Flag	1	4	n/a	n/a

Figure 3.1.1.1-3.1.1-a: Comparison of Wapiti and Zed Attack Proxy

Comparing scanner results directly was easiest with Zed Attack Proxy and Wapiti. This is because they both look for similar vulnerabilities in a scan. Both Wapiti scans missed the reflected Cross Site Scripting vulnerability in the web application that was picked up by the Attack scan with Zed Attack Proxy, but interestingly picked up a Blind SQL Injection Vulnerability that both Zed Attack Proxy scans missed.



Vulnerabilities / Scanner Strength	Nikto	ZAP	
		Standard	Attack
Cookie No HTTP Only Flag	1	1	4
X-Frame-Options header not present.	1	57	77
Directory Indexing	2	13	14
The X-Content-Type-Options header is not set		124	146
The X-XSS-Protection header is not defined	1	58	78
Information Leak via X-Powered Header	1	38	50

Figure 3.1.1-b: Comparison of Nikto and Zed Attack Proxy

Due to Nikto2 having a strong focus on foot-printing and highlighting default configurations, it is harder to compare with the other scanners used in this project. Nikto2 and both Zed Attack Proxy scans picked up: issues with the configuration of directories (directory listing and directory browsing) which potentially allows a malicious actor to access areas of the web application that they shouldn't; missing X Content Type Options header which prevents content sniffing (Mozilla, 2020); absence of anti-clickjacking X-Frame Options header – clickjacking is when a malicious actor deceives the user of an web application into clicking on something different to what the user sees, this could potentially allow access to the victim's machine; information leak due to X-Powered header.

3.1.2 Timing

The Zed Attack Proxy Standard scan took 5 hours 32 minutes and 21 seconds.

The Zed Attack Proxy Attack scan took 6 hours 41 minutes and 17 seconds – this is the longest recorded exact length of a scan, but it is possible that either of the two Wapiti scans took longer.

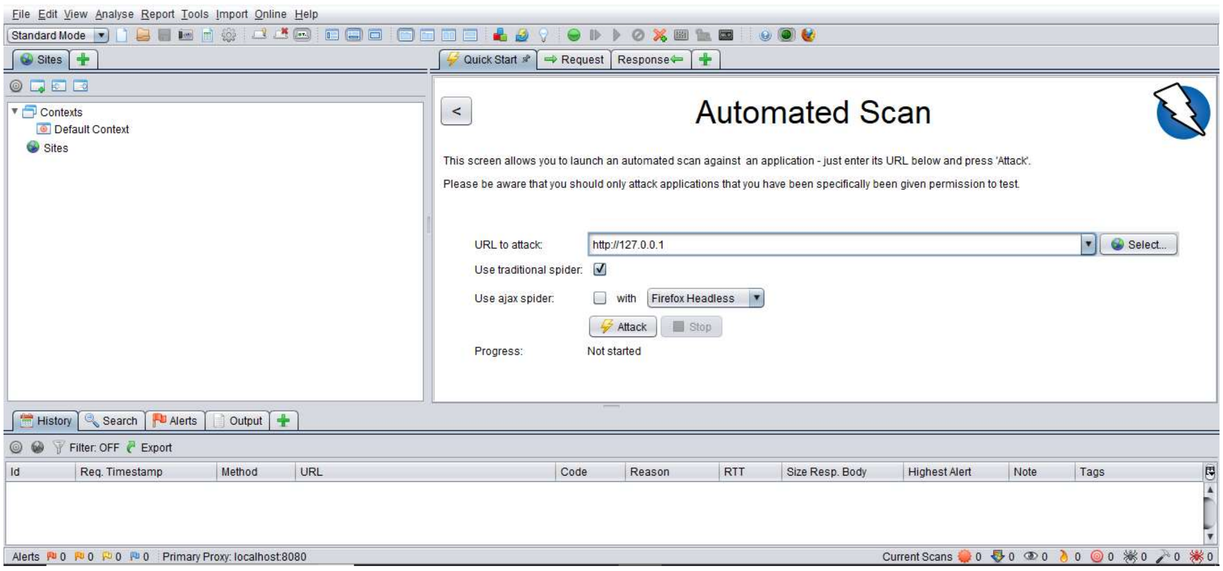
There is no exact length for the Wapiti scans as, due to time constraints in this project, the scans were run overnight, and the generated reports do not contain any indication of the timing of the scans. The scans were both less than 8 hours.

Nikto2 scan had a length of 10 minutes and 6 seconds – this is the shortest of length of scans from the scanners used in this investigation.

3.1.3 Ease of Use/Accessibility

The scanners were also assessed on how easy it was to start a basic scan, and how easy it was to access and interpret results of the scan as this is important for users so they can make informed decisions about the security of their web application.

3.1.3.1 Starting a Scan



Automated Scan

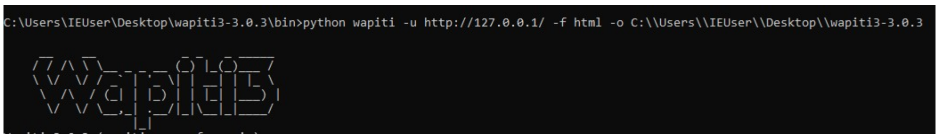
This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://127.0.0.1 Select...
Use traditional spider: ✓
Use ajax spider: ☐ with Firefox Headless
Attack Stop
Progress: Not started

Figure 3.1.3.3-3.1.3-a: Zed Attack Proxy GUI

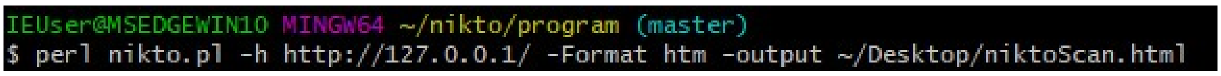
Zed Attack Proxy has a graphical user interface (figure 3.1.3.1-a) which is makes it easy to start and customise a scan. It has a quick start tab that can be used to launch automatic scans of a web application. The graphical user interface also makes it simple to track progress of different stages of a scan.



```
C:\Users\IEUser\Desktop\wapiti3-3.0.3\bin>python wapiti -u http://127.0.0.1/ -f html -o C:\\Users\\IEUser\\Desktop\\wapiti3-3.0.3
```

Figure 3.1.3.1-b: Wapiti Scan

Wapiti is a command line application (figure 3.1.3.1-b) and is fairly easy to start a basic scan.



```
IEUser@MSEDGEWIN10 MINGW64 ~/nikto/program (master)
$ perl nikto.pl -h http://127.0.0.1/ -Format htm -output ~/Desktop/niktoScan.html
```

Figure 3.1.3.3-c: Nikto Scan

Nikto2 is also a command line application and is again easy to start a basic scan. It also has interactive features that can be used during an active scan to interact with the scan in progress. These include pause, update on progress and other features.

3.1.3.2 Reports

Zed Attack Proxy reports are easy to generate through the fairly intuitive graphical user interface. It has multiple available formats and options which makes it easy to compare with other scanners and assess how vulnerable the web application is. The default report options result in a well laid out report with colour coding the vulnerabilities based on risk However, there is a potentially overwhelming amount of information, which could intimidate a new user due to the reports including every instance of each vulnerability found. It is possible to remove this data manually, but it would be helpful if the scanner provided that option as manual removal is potentially time consuming.

Wapiti reports can be easily generated by specifying the format and location in the initial command to run a scan They are laid out in an unintimidating and user friendly manner which makes them easy to absorb the information from.

Nikto2 reports are easy to generate, again by specifying the format and location in the initial command to run a scan. However, when compared with the other scanners used in this report, they are less accessible to a lay person as they are quite technical and also lack the information that is provided with the reports generated by the other scanners.

For full reports of Wapiti and Zed Arrack Proxy please view https://github.com/Aliisace/Mini-Project-Scanner-Reports (GitHub Repository with reports) as they are too long to include in this paper.

3.2 CONCLUSION

The different scanners had advantages and disadvantages, including ease of use, comprehensiveness, and ease of absorbing the information the report produces. The time taken to run the report is also important, as is the formats that the reports are available in. All produced html reports but reports were also produced in other formats. However, html formatting allows easier comparison between scanner reports.

In conclusion, Zed Attack Proxy appears to be better than Nikto2 and Wapiti for an intensive, detailed scan of a web application. If possible, penetration testers should use the Attack mode for scans as it performs a thorough scan of a web application. While the reports for Zed Attack Proxy are lengthy and detailed, it is possible to manually remove the individual instances listed in the report quickly, in order to make it a less intimidating report to process.

Nikto2 is a great scanner for foot-printing a web application and checking for default configurations, but it should not be used as the sole vulnerability scanner on a web application due to this focus. The reports produced are quite sparse on technical detail and are visually unappealing when compared to both Zed Attack Proxy and Wapiti.

Wapiti produces the most accessible reports but is let down by missing the majority of vulnerabilities picked up by the other scanners used in this investigation.

It is worthwhile using more than one scanner as they detected different potential problems, but the level of user expertise may dictate which scanner is used. Other scanners are available, and the situation is constantly changing. Malicious actors are developing new skills and methods of exploiting vulnerabilities, so new vulnerabilities are constantly being created and exploited.

3.3 FUTURE WORK

If more time were available, additional testing could be done using different configurations of the scanners used (Zed Attack Proxy, Wapiti, Nikto2). This would include, but not be limited to, Safe and Protected modes in Zed Attack Proxy and Paranoid, Sneaky Polite and Insane force in Wapiti.

Additional scanners could also have been used, such as Port Swigger's Burp Suite Proxy, OWASP Web Scarab.

Scanners are evolving over time, so repeating the work using the same scanners at the same settings might produce different results.

References

Anon., n.d. *The Uniform Server.* [Online]
Available at: https://www.uniformserver.com/
[Accessed 25 August 2020].

CIRT, n.d. *Nikto2.* [Online]
Available at: http://www.cirt.net/nikto2
[Accessed 18 August 2020].

Mozilla, 2020. *X-Content-Type-Options - HTTP | MDN.* [Online]
Available at: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options
[Accessed 29 August 2020].

Offensive Security, n.d. *Kali Linux Tools Listing | Penetration Testing Tools.* [Online]
Available at: https://tools.kali.org/tools-listing
[Accessed 25 August 2020].

OWASP, 2018. *Cross-site Scripting (XSS) - OWASP.* [Online]
Available at: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

OWASP, n.d. *OWASP Top Ten Web Application Security Risks | OWASP.* [Online]
Available at: https://owasp.org/www-project-top-ten/
[Accessed 20 August 2020].

OWASP, n.d. *OWASP ZAP – Getting Started.* [Online]
Available at: https://www.zaproxy.org/getting-started/
[Accessed 19 August 2020].

OWASP, n.d. *Vulnerability Scanning Tools | OWASP.* [Online]
Available at: https://owasp.org/www-community/Vulnerability_Scanning_Tools
[Accessed 23 August 2020].

pentestit, 2020. [Online]
Available at: https://medium.com/@Pentestit_ru/wapiti-free-web-application-vulnerability-scanner-ce7712adf644
[Accessed 23 August 2020].

sullo, 2018. *Overview & Description · sullo/nikto Wiki · GitHub.* [Online]
Available at: https://github.com/sullo/nikto/wiki/Overview-&-Description
[Accessed 23 August 2020].

Wapiti, 2020. *Wapiti : a Free and Open-Source web-application vulnerability scanner in Python for Windows, Linux, BSD, OSX.* [Online]
Available at: https://wapiti.sourceforge.io/
[Accessed 30 August 2020].

Wapiti, n.d. *Wapiti : a Free and Open-Source web-application vulnerability scanner in Python for Windows, Linux, BSD, OSX.* [Online]
Available at: https://wapiti.sourceforge.io/
[Accessed 20 August 2020].

APPENDICES

APPENDIX A – NIKTO2 SCAN REPORT

URI	/
HTTP Method	
Description	No web server found on 127.0.0.1:80
Test Links	https://127.0.0.1:80/ https://127.0.0.1:80/
References	

Scan Summary

Software Details	Nikto 2.1.6
CLI Options	-h http://127.0.0.1/ -Format htm -output /c/Users/IEUser/Desktop/niktoScan.html
Hosts Tested	0
Start Time	Fri Aug 28 11:49:50 2020
End Time	Fri Aug 28 11:49:56 2020
Elapsed Time	6 seconds

© 2008 Chris Sullo

127.0.0.1 / 127.0.0.1 port 80

Target IP	127.0.0.1
Target hostname	127.0.0.1
Target Port	80
HTTP Server	Apache
Site Link (Name)	http://127.0.0.1:80/
Site Link (IP)	http://127.0.0.1:80/
URI	/
HTTP Method	GET
Description	Cookie PHPSESSID created without the httponly flag

Test Links	http://127.0.0.1:80/ http://127.0.0.1:80/
References	
URI	/
HTTP Method	GET
Description	Retrieved x-powered-by header: PHP/5.6.30
Test Links	http://127.0.0.1:80/ http://127.0.0.1:80/
References	
URI	/
HTTP Method	GET
Description	The anti-clickjacking X-Frame-Options header is not present.
Test Links	http://127.0.0.1:80/ http://127.0.0.1:80/
References	
URI	/
HTTP Method	GET
Description	The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
Test Links	http://127.0.0.1:80/ http://127.0.0.1:80/
References	
URI	/
HTTP Method	GET
Description	The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type.
Test Links	http://127.0.0.1:80/ http://127.0.0.1:80/
References	
URI	/info.php
HTTP Method	GET
Description	Entry '/info.php' in robots.txt returned a non-forbidden or redirect HTTP code (200)

Test Links	http://127.0.0.1:80/info.php http://127.0.0.1:80/info.php
References	
URI	/
HTTP Method	KBRJCCON
Description	Web Server returns a valid response with junk HTTP methods, this may cause false positives.
Test Links	http://127.0.0.1:80/ http://127.0.0.1:80/
References	
URI	/
HTTP Method	TRACE
Description	HTTP TRACE method is active, suggesting the host is vulnerable to XST
Test Links	http://127.0.0.1:80/ http://127.0.0.1:80/
References	OSVDB-877
URI	/phpinfo.php?VARIABLE=<script>alert('Vulnerable')</script>
HTTP Method	GET
Description	/phpinfo.php: Output from the phpinfo() function was found.
Test Links	http://127.0.0.1:80/phpinfo.php?VARIABLE=<script>alert('Vulnerable')</script> http://127.0.0.1:80/phpinfo.php?VARIABLE=<script>alert('Vulnerable')</script>
References	
URI	/?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000
HTTP Method	GET
Description	/?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
Test Links	http://127.0.0.1:80/?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000 http://127.0.0.1:80/?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000
References	OSVDB-12184
URI	/server-status
HTTP Method	GET
Description	/server-status: This reveals Apache information. Comment out appropriate line in the Apache conf file or restrict access to allowed sources.

Test Links	http://127.0.0.1:80/server-status http://127.0.0.1:80/server-status
References	OSVDB-561
URI	/admin/
HTTP Method	GET
Description	/admin/: This might be interesting.
Test Links	http://127.0.0.1:80/admin/ http://127.0.0.1:80/admin/
References	OSVDB-3092
URI	/includes/
HTTP Method	GET
Description	/includes/: Directory indexing found.
Test Links	http://127.0.0.1:80/includes/ http://127.0.0.1:80/includes/
References	OSVDB-3268
URI	/includes/
HTTP Method	GET
Description	/includes/: This might be interesting.
Test Links	http://127.0.0.1:80/includes/ http://127.0.0.1:80/includes/
References	OSVDB-3092
URI	/admin/index.php
HTTP Method	GET
Description	/admin/index.php: This might be interesting: has been seen in web logs from an unknown scanner.
Test Links	http://127.0.0.1:80/admin/index.php http://127.0.0.1:80/admin/index.php
References	OSVDB-3093
URI	/phpinfo.php
HTTP Method	GET
Description	/phpinfo.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of system information.

Test Links	http://127.0.0.1:80/phpinfo.php http://127.0.0.1:80/phpinfo.php
References	OSVDB-3233
URI	/info.php
HTTP Method	GET
Description	/info.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of system information.
Test Links	http://127.0.0.1:80/info.php http://127.0.0.1:80/info.php
References	OSVDB-3233
URI	/icons/
HTTP Method	GET
Description	/icons/: Directory indexing found.
Test Links	http://127.0.0.1:80/icons/ http://127.0.0.1:80/icons/
References	OSVDB-3268
URI	/icons/README
HTTP Method	GET
Description	/icons/README: Apache default file found.
Test Links	http://127.0.0.1:80/icons/README http://127.0.0.1:80/icons/README
References	OSVDB-3233
URI	/Admin/
HTTP Method	GET
Description	/Admin/: This might be interesting.
Test Links	http://127.0.0.1:80/Admin/ http://127.0.0.1:80/Admin/
References	OSVDB-3092
URI	/info.php?file=http://cirt.net/rfiinc.txt?
HTTP Method	GET
Description	/info.php?file=http://cirt.net/rfiinc.txt?: RFI from RSnake's list (https://gist.github.com/mubix/5d269c686584875015a2)

Test Links	http://127.0.0.1:80/info.php?file=http://cirt.net/rfiinc.txt? http://127.0.0.1:80/info.php?file=http://cirt.net/rfiinc.txt?
References	OSVDB-5292

Host Summary

Start Time	2020-08-31 11:50:51
End Time	2020-08-31 11:59:07
Elapsed Time	496 seconds
Statistics	8883 requests, 0 errors, 21 findings

Scan Summary

Software Details	Nikto 2.1.6
CLI Options	-h http://127.0.0.1/ -Format htm -output /c/Users/IEUser/Desktop/niktoScan.html
Hosts Tested	1
Start Time	Fri Aug 28 11:50:48 2020
End Time	Fri Aug 28 11:59:07 2020
Elapsed Time	499 seconds