Configuration Manual

MSc Research Project
Cyber Security

Mayuresh Gadekar
Student ID: x21179280

School of Computing
National College of Ireland

Supervisor: Imran Khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Mayuresh Rajesh Gadekar		
Student ID:	X21179280		
Programme:	MSc Cyber Security	**Year:**	2022 - 2023
Module:	MSc Research Project/Internship		
Lecturer:	Imran Khan		
Submission Due Date:	18th September 2023		
Project Title:	Consolidating reconnaissance tools and techniques for penetration testing under a single platform		
Word Count:	666	**Page Count:**	8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Mayuresh Gadekar
Date:	15th September 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Mayuresh Gadekar
Student ID: x21179280

1 System Configuration

The setup is installed on Virtualbox VMs and following is the configuration details.

	Memory (RAM)	Storage (HDD)	CPU	Operating System
Kali linux	4 GB	60.55 GB	2	5.10.0-kali3-amd64

2 Installation of Assetfinder

If you have Go installed and configured (i.e. with $GOPATH/bin in your $PATH):

```
go get -u github.com/tomnomnom/assetfinder
```

Use above command to install assetfinder.

Usage :

```
assetfinder [--subs-only] <domain>
```

Replace any domain name in place of <domain> and use assetfinder.

(hudson, n.d.)

3 Installation of httprobe

```
▶ go install github.com/tomnomnom/httprobe@latest
```

Use above command to install httprobe.

Usage:

```
▶ cat recon/example/domains.txt
example.com
example.edu
example.net
▶ cat recon/example/domains.txt | httprobe
http://example.com
http://example.net
http://example.edu
https://example.com
https://example.edu
https://example.net
```

Above screenshot shows demonstration on how to use httprobe.

(hudson, n.d.)

4 Setting up nDorker:

```
git clone https://github.com/nerrorsec/nDorker
cd nDorker
```

Download the github repository of nDorker and change the directory into downloaded nDorker repository.

Usage :

```
python3 nDorker.py -d example.com
```

Above command shows how to use nDorker.

(Khatiwada, n.d.)

5 Setting up Secretfinder:

```
$ git clone https://github.com/m4ll0k/SecretFinder.git secretfinder
$ cd secretfinder
$ python -m pip install -r requirements.txt or pip install -r requirements.txt
$ python3 SecretFinder.py
```

Download the github repository of secretfinder and change the directory into downloaded secretfinder repository. Install the requirements using the third command.

Usage :

```
python3 SecretFinder.py -i https://example.com/ -e
```

Above command demonstrates how to use Secretfinder.py.

(mallock, n.d.)

6 Setting up ParamSpider:

```
$ git clone https://github.com/devanshbatham/ParamSpider
$ cd ParamSpider
$ pip3 install -r requirements.txt
$ python3 paramspider.py --domain hackerone.com
```

Download the github repository of ParamSpider and change the directory into downloaded ParamSpider repository. Install the requirements using the third command. In the forth command paramspider usage is demonstrated.

(Batham, n.d.)

7 Installation of qsreplace:

```
▶ go install github.com/tomnomnom/qsreplace@latest
```

Install qsreplace using above command.

Usage:

```
▶ cat urls.txt | qsreplace newval
https://example.com/path?one=newval&two=newval
https://example.com/pathtwo?one=newval&two=newval
https://example.net/a/path?one=newval&two=newval
```

Above screenshot demonstrates usage of qsreplace.
(Hudson, n.d.)

8 nmap:

Once kali linux is installed it has nmap pre-installed in it. Its not needed to be installed explicitly.

9 ffuf : (Fuzz Faster U Fool)

```
git clone https://github.com/ffuf/ffuf ; cd ffuf ; go get ; go build
```

Install ffuf using command shown in above screenshot.

Usage:

```
ffuf -w /path/to/wordlist -u https://target/FUZZ
```

Use ffuf using command shown in above screenshot.

(Hoikkala, n.d.)

10 recon.sh: (Automation Script)

```
*/root/Thesis/recon.sh - Mousepad
File  Edit  Search  View  Document  Help
Warning: you are using the root account. You m
#!/bin/bash

if [ -z "$1" ]
then
        echo "Usage: ./recon.sh <domain>"
        exit 1
fi

if [ ! -d "recon_output" ]; then
        mkdir recon_output
fi

if [ ! -d "recon_output/thirdlevels" ]; then
        mkdir recon_output/thirdlevels
fi


if [ ! -d "recon_output/scans" ]; then
        mkdir recon_output/scans
fi

if [ ! -d "recon_output/Dirb" ]; then
        mkdir recon_output/Dirb
fi


if [ ! -d "recon_output/Dorks" ]; then
        mkdir recon_output/Dorks
fi


if [ ! -d "recon_output/Javascript_Enum" ]; then
        mkdir recon_output/Javascript_Enum
fi


if [ ! -d "recon_output/OWASP_Top_10/SQLI" ]; then
        mkdir -p recon_output/OWASP_Top_10/SQLI
fi

if [ ! -d "recon_output/OWASP_Top_10/XSS" ]; then
        mkdir -p recon_output/OWASP_Top_10/XSS
fi


if [ ! -d "recon_output/OWASP_Top_10/LFI" ]; then
        mkdir -p recon_output/OWASP_Top_10/LFI
fi
```

Code snippet: 1

The code in the Code snippet 1 screenshot makes directories for storing results in organised manner.

```
*/root/Thesis/recon.sh - Mousepad
File  Edit  Search  View  Document  Help
Warning: you are using the root account. You may harm your system.

#starting assetfinder
echo "————————————————————————————Gathering subdomains with assetfinder————————————————————————————"
time assetfinder -subs-only $1 > recon_output/domains.txt
printf '\n\n'

#sorting duplicate subdomains
echo "————————————————————————————Compiling third-level domains————————————————————————————"
time cat recon_output/domains.txt | grep -Po "(\w+\.\w+\.\w+)$" | sort -u >> recon_output/third-level.txt
printf '\n\n'

#Gathering third level subdomain
echo "————————————————————————————Gathering full third level domains with assetfinder————————————————————————————"
time for domain in $(cat recon_output/third-level.txt); do assetfinder -subs-only $domain in > recon_output/thirdlevels/$domain.txt; cat recon_output/thirdlevels/$domain.txt | sort -u >> recon_output/final.txt;done
printf '\n\n'

echo "————————————————————————————Probing for alive third levels————————————————————————————"
time cat recon_output/final.txt | sort -u | httprobe  > recon_output/probed_og.txt
time cat recon_output/final.txt | sort -u | httprobe  |  sed 's/https\?:\/\///' | sort -u > recon_output/probed.txt
printf '\n\n'
```

Code Snippet: 2

The code in the code snippet 2 gathers all subdomains with **assetfinder** and stores it in the text file called **domains.txt** then removes the duplicate subdomain entries from the **domains.txt** by using **sort command** and stores the result in **third-level.txt.** After that, for each subdomains entry in the **third-level.txt** it gathers subdomains and store it in the required text file associated to that domain and it sorts third-level.txt again and stores its result in **final.txt**. Further to it carries out probing on every domain in final.txt using **httprobe** and stores the results in probed_og.txt with prefixes **http:**// or **https:**// about the active domains.

Also, another file called just **probed.txt** is made to just store the active domains without the **http://** prefixes.

```
echo "————————————————————————————Scanning for Google dorks, Github dorks and Shodan dorks for each domain————————————————————————————"
time for domain in $(cat recon_output/probed.txt); do python3 nDorker.py -d $domain;
done
printf '\n\n'
```

Code snippet: 3

The code in code snippet 3 performs Google dorking and Github dorking on all the subdomains in probed.txt one by one.

```
echo "————————————————————————————Finding secrets in Javascript file————————————————————————————"
time for domain in $(cat recon_output/probed_og.txt); do
outfile="$( tr -s -c 'A-Za-z0-9' '_' <<<"$domain" )"
python3 SecretFinder.py -i $domain -e -o cli >> recon_output/Javascript_Enum/$outfile.txt;
#result= '$( tr -s ' ' '\n' < Javascript_Enum/$outfile.txt | grep -c "possible_Creds" )'
result=$(tr -s ' ' '\n' < "recon_output/Javascript_Enum/$outfile.txt" | grep -c "possible_Creds")
echo "$result secrets found in $domain"
done
printf '\n\n'
```

Code snippet: 4

The code in code snippet 4 performs Javascript enumeration using SecretFinder.py on every domain in probed_og.txt and stores in the Javascript_enum directory with appropriate text file name.

```
*/root/Thesis/recon.sh - Mousepad
File  Edit  Search  View  Document  Help
Warning: you are using the root account. You may harm your system.
printf '\n\n'

echo "————————————————————————————Find owasp top 10 vulnerabilities————————————————————————————"
time for url in $(cat recon_output/probed_og.txt); do
outfile="$( tr -s -c 'A-Za-z0-9' '_' <<<"$url" )"
python3 paramspider.py -d $url > recon_output/$outfile.txt
sed -i -e '/32/d' -e '/0m/d' recon_output/$outfile.txt
cat recon_output/$outfile.txt | qsreplace '"><script>confirm(1)</script>' | while read host; do curl --silent --path-as-is --insecure "$host" |tac | tac| grep -qs "<script>confirm(1)" && echo -en "$host \e\033[32mVulnerable\033[0m\n" >>
result=$(tr -s ' ' '\n' < "recon_output/OWASP_Top_10/XSS/$outfile.txt" | grep -c "Vulnerable")
echo "$result XSS vulnerabilities found in $url"
cat recon_output/$outfile.txt | qsreplace '../../etc/passwd'| while read host; do curl --silent --path-as-is --insecure "$host"|tac | tac | grep -qs "root:x" && echo -en "$host Vulnerable\n" >> recon_output/OWASP_Top_10/LFI/$outfile.txt
result=$(tr -s ' ' '\n' < "recon_output/OWASP_Top_10/LFI/$outfile.txt" | grep -c "Vulnerable")
echo "$result LFI linux server vulnerabilities found in $url"
cat recon_output/$outfile.txt | qsreplace 'C:\boot.ini'| while read host; do curl --silent --path-as-is --insecure "$host"|tac | tac | grep -qs "Windows" && echo -en "$host Vulnerable Parameter\n" >> recon_output/OWASP_Top_10/LFI/$outfi
result=$(tr -s ' ' '\n' < "recon_output/OWASP_Top_10/LFI/$outfile.txt" | grep -c "Vulnerable")
echo "$result LFI windows server vulnerabilities found in $url"
cat recon_output/$outfile.txt | qsreplace "'"| while read host; do curl --silent --path-as-is --insecure "$host"|tac | tac | grep -qs "SQL" && echo -en "$host Vulnerable Parameter\n" >> recon_output/OWASP_Top_10/SQLI/$outfile.txt || ech
result=$(tr -s ' ' '\n' < "recon_output/OWASP_Top_10/SQLI/$outfile.txt" | grep -c "Vulnerable")
echo "$result SQLI vulnerabilities found in $url"
cat recon_output/$outfile.txt | qsreplace 'http://www.google.com'| while read host; do curl --silent --path-as-is --insecure "$host"|tac | tac | grep -qs "google.com" && echo -en "$host \e\033[32mVulnerable\033[0m\n" >> recon_output/OWA
result=$(tr -s ' ' '\n' < "recon_output/OWASP_Top_10/SSRF/$outfile.txt" | grep -c "Vulnerable")
echo "$result SSRF vulnerabilities found in $url"
cat recon_output/$outfile.txt | qsreplace 'ping 127.0.0.1'| while read host; do curl --silent --path-as-is --insecure "$host"|tac | tac | grep -qs "64 bytes" && echo -en "$host Vulnerable Parameter\n" >> recon_output/OWASP_Top_10/Commar
result=$(tr -s ' ' '\n' < "recon_output/OWASP_Top_10/Command_injection/$outfile.txt" | grep -c "Vulnerable")
echo "$result Command injecion vulnerabilities found in $url"
rm recon_output/$outfile.txt
printf '\n\n'
done
printf '\n\n'
```

Code snippet: 5

The code in code snippet 5 carries out vulnerability scanning on every domain using paramspider and qsreplace. Vulnerabilities like Cross site scripting, Server side request forgery, SQL injection, local file inclusion and command injection are identified and stored in the OWASP_TOP_10 directory within the appropriate text file.

```
echo "————————————————————————————Scanning for open ports————————————————————————————"
time for domain in $(cat recon_output/probed.txt); do nmap -A -Pn -T5 $domain > recon_output/scans/$domain.txt
result=$(tr -s ' ' '\n' < "recon_output/scans/$domain.txt" | grep -c "open")
echo "$result open ports found of $domain" printf '\n\n';done
printf '\n\n'
```

Code snippet: 6

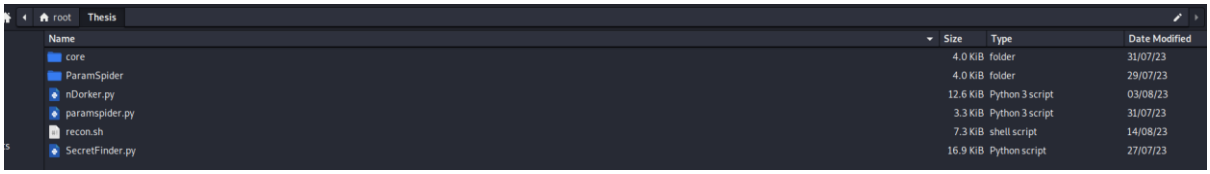
The code in code snippet 6 performs port scanning using nmap on every domain and stores result in the text file in scans directory.

```

echo "————————————————————————————Looking for directories————————————————————————————"
time while read -r url; do
        outfile="$( tr -s -c 'A-Za-z0-9' '_' <<<"$url" )"
        ffuf -w /usr/share/wordlists/dirb/common.txt -u $url/FUZZ -mc 200 -s  > recon_output/Dirb/$outfile.txt
        awk -v prefix="$url/" '{print prefix $0}' recon_output/Dirb/$outfile.txt > recon_output/Dirb/drb_$outfile.txt
        result=$(wc -l "recon_output/Dirb/drb_$outfile.txt")
        echo "$result "
        printf '\n\n'
        rm recon_output/Dirb/$outfile.txt
done <recon_output/probed_og.txt
printf '\n\n'
```

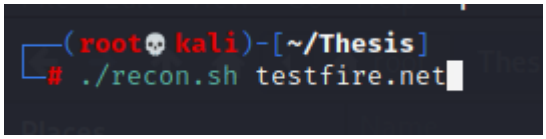
Code snippet: 7

The code in code snippet 7 carries out directory brute force using **ffuf** on every domain in probed_og.txt and stores the results in appropriate text file in Dirb directory.



Directory structure to run the automation script

Keep the recon.sh automation script within the directory structure demonstrated in the above screenshot.



Run recon.sh

Run the recon.sh as demonstrated in above screenshot.

References

Batham, D., n.d. Mining parameters from dark corners of Web Archives. [Online]
Available at: https://github.com/devanshbatham/ParamSpider

Hoikkala, J., n.d. Fast web fuzzer written in Go. [Online]
Available at: https://github.com/ffuf/ffuf

Hudson, T., n.d. Accept URLs on stdin, replace all query string values with a user-supplied value. [Online]
Available at: https://github.com/tomnomnom/qsreplace

hudson, T., n.d. Find domains and subdomains related to a given domain. [Online]
Available at: https://github.com/tomnomnom/assetfinder

hudson, T., n.d. Take a list of domains and probe for working HTTP and HTTPS servers. [Online]
Available at: https://github.com/tomnomnom/httprobe

Khatiwada, N., n.d. Automate dorking while doing bug bounty or other stuffs.. [Online]
Available at: https://github.com/nerrorsec/Google-Dorker

mallock, n.d. SecretFinder - A python script for find sensitive data (apikeys, accesstoken,jwt,..) and search anything on javascript files. [Online]
Available at: https://github.com/m4ll0k/SecretFinder