Bug Hunting Methodology and Enumeration

Summary

- Enumerate all subdomains
    - Subbrute
    - KnockPy
    - GoogleDorks
    - EyeWitness
    - Sublist3r
    - Aquatone
- Passive Recon
    - Shodan
    - Wayback Machine
    - The Harvester
- Active Recon
    - Nmap
    - Nmap Script
    - RPCClient
    - Enum4all
- List all the subdirectories and files
    - Gobuster
    - Backup File Artifacts Checker
- Web Vulnerabilities
    - Repository Github
    - Burp
    - Web Checklist
    - Nikto
    - Payment functionality

Enumerate all subdomains (only if the scope is

*.domain.ext)

Using Subbrute

```
git clone https://github.com/TheRook/subbrute
python subbrute.py domain.example.com
```

Using KnockPy with Daniel Miessler's SecLists for subdomain "/Discover/DNS"

```
git clone https://github.com/guelfoweb/knock
git clone https://github.com/danielmiessler/SecLists.git
knockpy domain.com -w subdomains-top1mil-110000.txt
```

Using Google Dorks and Google Transparency Report

You need to include subdomains ;) https://www.google.com/transparencyreport/https/ct/?hl=en-US#domain=[DOMAIN]g&incl_exp=true&incl_sub=true
(https://www.google.com/transparencyreport/https/ct/?hl=en-US#domain=[DOMAIN]g&incl_exp=true&incl_sub=true)

```
site:*.domain.com -www
site:domain.com filetype:pdf
site:domain.com inurl:'&'
site:domain.com inurl:login,register,upload,logout,redirect,redir,goto,admin
site:domain.com ext:php,asp,aspx,jsp,jspa,txt,swf
site:*.*.domain.com
```

Subdomain take over using HostileSubBruteForcer

```
git clone https://github.com/nahamsec/HostileSubBruteforcer
chmox +x sub_brute.rb
./sub_brute.rb
```

EyeWitness and Nmap scans from the KnockPy and enumall scans

```
git clone https://github.com/ChrisTruncer/EyeWitness.git
./setup/setup.sh
./EyeWitness.py -f filename -t optionaltimeout --open (Optional)
./EyeWitness -f urls.txt --web
./EyeWitness -x urls.xml -t 8 --headless
./EyeWitness -f rdp.txt --rdp
```

Using Sublist3r

```
To enumerate subdomains of specific domain and show the results in realtime:
```

```
python sublist3r.py -v -d example.com

To enumerate subdomains and enable the bruteforce module:
python sublist3r.py -b -d example.com

To enumerate subdomains and use specific engines such Google, Yahoo and Virustotal
engines
python sublist3r.py -e google,yahoo,virustotal -d example.com

python sublist3r.py -b -d example.com
```

Using Aquatone

```
gem install aquatone
```

```
Discover subdomains : results in ~/aquatone/example.com/hosts.txt
aquatone-discover --domain example.com
aquatone-discover --domain example.com --threads 25
aquatone-discover --domain example.com --sleep 5 --jitter 30
aquatone-discover --set-key shodan o1hyw8pv59vSVjrZU3Qaz6ZQqgM91ihQ

Active scans : results in ~/aquatone/example.com/urls.txt
aquatone-scan --domain example.com
aquatone-scan --domain example.com --ports 80,443,3000,8080
aquatone-scan --domain example.com --ports large
aquatone-scan --domain example.com --threads 25

Final results
aquatone-gather --domain example.com
```

Passive recon

- Using Shodan (https://www.shodan.io/ (https://www.shodan.io/)) to detect similar app

```
can be integrated with nmap (https://github.com/glennzw/shodan-hq-nse)
nmap --script shodan-hq.nse --script-args 'apikey=<yourShodanAPIKey>,target=<
hackme>'
```

- Using The Wayback Machine (https://archive.org/web/ (https://archive.org/web/)) to detect forgotten endpoints

```
look for JS files, old links
```

- Using The Harvester (https://github.com/laramies/theHarvester (https://github.com/laramies/theHarvester))

```
python theHarvester.py -b all -d domain.com
```

Active recon

- Basic NMAP

```
sudo nmap -sSV -p- 192.168.0.1 -oA OUTPUTFILE -T4
sudo nmap -sSV -oA OUTPUTFILE -T4 -iL INPUTFILE.csv

• the flag -sSV defines the type of packet to send to the server and tells Nm
ap to try and determine any service on open ports
• the -p- tells Nmap to check all 65,535 ports (by default it will only check
 the most popular 1,000)
• 192.168.0.1 is the IP address to scan
• -oA OUTPUTFILE tells Nmap to output the findings in its three major formats
 at once using the filename "OUTPUTFILE"
• -iL INPUTFILE tells Nmap to use the provided file as inputs
```

- CTF NMAP This configuration is enough to do a basic check for a CTF VM

```
nmap -sV -sC -oA ~/nmap-initial 192.168.1.1

-sV : Probe open ports to determine service/version info
-sC : to enable the script
-oA : to save the results

After this quick command you can add "-p-" to run a full scan while you work
 with the previous result
```

- Aggressive NMAP

```
nmap -A -T4 scanme.nmap.org
• -A: Enable OS detection, version detection, script scanning, and traceroute
• -T4: Defines the timing for the task (options are 0-5 and higher is faster
)
```

- NMAP and add-ons
  - Using searchsploit to detect vulnerable services

```
nmap -p- -sV -oX a.xml IP_ADDRESS; searchsploit --nmap a.xml
```

  - Generating nice scan report

```
nmap -sV IP_ADDRESS -oX scan.xml && xsltproc scan.xml -o "`date +%m%d%y
`_report.html"
```

- NMAP Scripts

```
nmap -sC : equivalent to --script=default
```

```
nmap --script 'http-enum' -v web.xxxx.com -p80 -oN http-enum.nmap
PORT   STATE SERVICE
80/tcp open  http
| http-enum:
|   /phpmyadmin/: phpMyAdmin
|   /.git/HEAD: Git folder
|   /css/: Potentially interesting directory w/ listing on 'apache/2.4.10 (de
bian)'
|_  /image/: Potentially interesting directory w/ listing on 'apache/2.4.10 (
debian)'

nmap --script smb-enum-users.nse -p 445 [target host]
Host script results:
| smb-enum-users:
|   METASPLOITABLE\backup (RID: 1068)
|     Full name:   backup
|     Flags:       Account disabled, Normal user account
|   METASPLOITABLE\bin (RID: 1004)
|     Full name:   bin
|     Flags:       Account disabled, Normal user account
|   METASPLOITABLE\msfadmin (RID: 3000)
|     Full name:   msfadmin,,,
|     Flags:       Normal user account

List Nmap scripts : ls /usr/share/nmap/scripts/
```

- RPCClient

```
└─$ rpcclient -U "" [target host]
rpcclient $> querydominfo
Domain: WORKGROUP
Server: METASPLOITABLE
Comment: metasploitable server (Samba 3.0.20-Debian)
Total Users: 35

rpcclient $> enumdomusers
user:[games] rid:[0x3f2]
user:[nobody] rid:[0x1f5]
user:[bind] rid:[0x4ba]
```

- Enum4all

```
Usage: ./enum4linux.pl [options]ip
-U        get userlist
-M        get machine list*
-S        get sharelist
-P        get password policy information
-G        get group and member list
-d        be detailed, applies to -U and -S
-u user   specify username to use (default “”)
-p pass   specify password to use (default “”
```

```
-a        Do all simple enumeration (-U -S -G -P -r -o -n -i).
-o        Get OS information
-i        Get printer information
==============================
|   Users on XXX.XXX.XXX.XXX |
==============================
index: 0x1 Account: games Name: games Desc: (null)
index: 0x2 Account: nobody Name: nobody Desc: (null)
index: 0x3 Account: bind Name: (null) Desc: (null)
index: 0x4 Account: proxy Name: proxy Desc: (null)
index: 0x5 Account: syslog Name: (null) Desc: (null)
index: 0x6 Account: user Name: just a user,111,, Desc: (null)
index: 0x7 Account: www-data Name: www-data Desc: (null)
index: 0x8 Account: root Name: root Desc: (null)
```

List all the subdirectories and files

- Using BFAC (Backup File Artifacts Checker): An automated tool that checks for backup artifacts that may disclose the web-application's source code.

```
git clone https://github.com/mazen160/bfac

Check a single URL
bfac --url http://example.com/test.php --level 4

Check a list of URLs
bfac --list testing_list.txt
```

- Using DirBuster or GoBuster

```
./gobuster -u http://buffered.io/ -w words.txt -t 10
-u url
-w wordlist
-t threads

More subdomain :
./gobuster -m dns -w subdomains.txt -u google.com -i

gobuster -w wordlist -u URL -r -e
```

- Using a script to detect all phpinfo.php files in a range of IPs (CIDR can be found with a whois)

```
#!/bin/bash
for ipa in 98.13{6..9}.{0..255}.{0..255}; do
wget -t 1 -T 3 http://${ipa}/phpinfo.php; done &
```

- Using a script to detect all .htpasswd files in a range of IPs

```
#!/bin/bash
for ipa in 98.13{6..9}.{0..255}.{0..255}; do
wget -t 1 -T 3 http://${ipa}/.htpasswd; done &
```

Looking for Web vulnerabilities

- Look for private information in GitHub repos with GitRob

```
git clone https://github.com/michenriksen/gitrob.git
gitrob analyze johndoe --site=https://github.acme.com --endpoint=https://gith
ub.acme.com/api/v3 --access-tokens=token1,token2
```

- Explore the website with a proxy (ZAP/Burp Suite)

    1. Start proxy, visit the main target site and perform a Forced Browse to discover files and directories
    2. Map technologies used with Wappalyzer and Burp Suite (or ZAP) proxy
    3. Explore and understand available functionality, noting areas that correspond to vulnerability types

```
Burp Proxy configuration on port 8080 (in .bashrc):
alias set_proxy_burp='gsettings set org.gnome.system.proxy.http host "h
ttp://localhost";gsettings set org.gnome.system.proxy.http port 8080;gs
ettings set org.gnome.system.proxy mode "manual"'
alias set_proxy_normal='gsettings set org.gnome.system.proxy mode "none
"'

then launch Burp with : java -jar burpsuite_free_v*.jar &
```

- Checklist for Web vulns (http://mdsec.net/wahh/tasks.html)
- Subscribe to the site and pay for the additional functionality to test
- Launch a Nikto scan in case you missed something

```
nikto -h http://domain.example.com
```

- Payment functionality - @gwendallecoguic (https://twitter.com/gwendallecoguic/status/988138794686779392)

    > *if the webapp you're testing uses an external payment gateway, check the doc to find the test credit numbers, purchase something and if the webapp didn't disable the test mode, it will be free*

    From https://stripe.com/docs/testing#cards (https://stripe.com/docs/testing#cards) : "Use any of the following test card numbers, a valid expiration date in the future, and any random CVC number, to create a successful payment. Each test card's billing country is set to U.S. " e.g :

Test card numbers and tokens

NUMBER	BRAND	TOKEN
4242424242424242	Visa	tok_visa
4000056655665556	Visa (debit)	tok_visa_debit
5555555555554444	Mastercard	tok_mastercard

International test card numbers and tokens

NUMBER	TOKEN	COUNTRY	BRAND
4000000400000008	tok_at	Austria (AT)	Visa
4000000560000004	tok_be	Belgium (BE)	Visa
4000002080000001	tok_dk	Denmark (DK)	Visa
4000002460000001	tok_fi	Finland (FI)	Visa
4000002500000003	tok_fr	France (FR)	Visa

Thanks to

- [BugBounty] Yahoo phpinfo.php disclosure – Patrik Fehrenbach (http://blog.it-securityguard.com/bugbounty-yahoo-phpinfo-php-disclosure-2/)
- Nmap CheatSheet – HackerTarget (https://hackertarget.com/nmap-cheatsheet-a-quick-reference-guide/)