

Ethical Student Hackers

Enumeration



The Legal Bit

- The skills taught in these sessions allow identification and exploitation of security vulnerabilities in systems. We strive to give you a place to practice legally, and can point you to other places to practice. These skills should not be used on systems where you do not have explicit permission from the owner of the system. It is VERY easy to end up in breach of relevant laws, and we can accept no responsibility for anything you do with the skills learnt here.
- If we have reason to believe that you are utilising these skills against systems where you are not authorised you will be banned from our events, and if necessary the relevant authorities will be alerted.
- Remember, if you have any doubts as to if something is legal or authorised, just don't do it until you are able to confirm you are allowed to.
- Relevant UK Law: <https://www.legislation.gov.uk/ukpga/1990/18/contents>



Code of Conduct

- Before proceeding past this point you must read and agree to our Code of Conduct - this is a requirement from the University for us to operate as a society.
- If you have any doubts or need anything clarified, please ask a member of the committee.
- Breaching the Code of Conduct = immediate ejection and further consequences.
- Code of Conduct can be found at
<https://shefesh.com/downloads/SESH%20Code%20of%20Conduct.pdf>



What is Enumeration?

Enumeration is one of the of the most important methodologies in cybersecurity - especially for offensive security

In a nutshell, it's all about information gathering

Not only automated scanning, but manual exploration of

- Webpages
- Networks
- File Systems

Enumeration is recursive.



Why do we do it?

To figure out the kind of system we're dealing with

- Whether it's networked (outgoing and incoming connections, protocols)
- What the OS is (Any known exploits? Vulnerable kernel versions?)
- What its purpose is (web server? Domain controller?)

To figure out a way in

- Vulnerable software versions or exposed ports
- Users and credentials left lying around

To spot anything out of the ordinary

- Interesting files and unusual processes
- Remote connections to other services/machines
- Privilege Escalation



What are we looking for?

Open ports

- Useful services like SMB, SSH, and Windows Services like LDAP & Kerberos

Config info, users & credentials

- Default creds for common services
- Passwords lying around in public documents, config files & exposed databases
- Version numbers, package info etc

Running services and funky processes

- Machines communicating with each other
- Processes downloading things from a server
- Stuff running as root
- Anything that runs periodically is a possible path to a foothold/privesc



Nmap

Nmap is one of the first steps in a security assessment - it scans the most common ports (or a specific list of ports), checks if they are open, and tries to discover services on each of them. It comes preinstalled on Kali Linux

A standard command to run nmap on the most common ports is: `nmap -sC -sV [ip]`

- `sC` is use default scripts
- `sV` is enumerate versions/services
- `oA [file directory]` can be used to output the data in all formats to a directory (why?)

You can also specify ports with the `-p` flag (use `-p-` to scan all 65535 ports) and control the speed of the scan with the `-Tx` flag where `x` is the intensity from 0 to 5 (5 is highest!) (why?)

The `-O` flag discovers the operating system. Another tip is to run an all ports scan in the background while you test (use `nmap -p- [ip]`)



Nmap

Scan an IP/Domain Name: `nmap [IP/DOMAIN]`

Scan all ports: `nmap -p- [IP/DOMAIN]`

Scan specific ports: `nmap -p 1-1000,8080,9001`

Don't do ping probing: `nmap -Pn [IP/DOMAIN]`

Run standard scripts and version detection: `nmap -sC -sV [IP/DOMAIN]`

Scan UDP: `nmap -sU [IP/DOMAIN]`

Run a specific script: `nmap --script=[SCRIPT_NAME] [IP/DOMAIN]`

Save your results: `nmap -oA [path/to/file] [IP/DOMAIN]`

Use verbose mode to see ports as they appear/diagnose issues: `nmap -v [IP/DOMAIN]`

You can, of course, combine these flags - e.g. scanning specific ports with `-sC` and `-sV` flags



Gobuster

Gobuster is a tool that is used for enumerating multiple services, most notably HTTP/S services. However it also supports DNS and vhost enumeration.

After an nmap scan it's always worth having some form of enumeration running in the background while you actively search for other exploitation paths. An example of this would be to run gobuster file/directory enumeration against the server you're exploiting.

- `gobuster dir -w [file/directory wordlist] -u [http:// + ip]`
 - `-x` can be used to specify extensions, e.g. `-x php,html,txt`
 - `-s` & `-b` can be used to add or remove response codes from the filter list

Gobuster can also be used for DNS enumeration for subdomains as well as virtual host enumeration.

Can also use dirbuster and Wfuzz



Wfuzz

Allows injection of payloads into HTTP requests (similar to the Burp Intruder module)

Payload positions are marked by the FUZZ word - for example:

- wfuzz -u http://example.com/FUZZ -w wordlist/general/common.txt will replace FUZZ with all words in the specified wordlist (useful for URL discovery)
- wfuzz -u http://example.com/login.php -d 'email=FUZZ&password=testpass' -w /path/to/email-list -p 127.0.0.1:8080:HTTP will use the -d parameter to pass data to a POST request and enumerate possible emails that we can login with - it also passes the request through a proxy, so we can see what it's doing
- wfuzz -u http://example.com/search.php?search=FUZZ -w /path/to/search-terms -b 'PHPSESSID=12345678912345678912345678' performs a search, passing a cookie with -b

Remember to add box URLs to your /etc/hosts folder if wfuzz is struggling to connect!



Webserver Enumeration

Fuzzing endpoints with `ffuf`:

- Spray usernames with a found password: `ffuf -u [URL]/search -X POST -d username=FUZZ&password=KNOWN_PASS -w /path/to/usernames`
- Try bad data in a POST: `ffuf -u [URL]/search -X POST -d searchterm=FUZZ -w /usr/share/seclists/Fuzzing/big-list-of-naughty-strings.txt`
- Try special characters in a URL parameter: `ffuf -u [URL]/profile?id=FUZZ -w /usr/share/seclists/Fuzzing/special-chars.txt`
- Fuzz a cookie: `ffuf -u [URL] -b "auth=FUZZ"`
- Check for LFI: `ffuf -u [URL]?page=FUZZ -w /usr/share/seclists/Fuzzing/LFI/LFI-gracefulsecurity-linux.txt`

Vulnerability scanning with Nikto: `nikto -host=[URL]`

Manual things to enumerate:

- Tech stacks leaked in response headers (`Server`, `X-Powered-By`, `PHPSESSID`, etc...)
- Test whether / responds to `/index.html`, `/index.php`, `/index.asp`, etc..
- Provoke errors with unexpected data/characters, unknown URLs...
- Read SSL certificates to look for addresses, domains, etc



Useful Resources

Seclists

- Very useful list of lists - ranging from passwords to directories to usernames

PrivEsc Scripts Suite

- <https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite>
- <https://book.hacktricks.xyz/> is a really useful guide for explaining how WinPeas and LinPeas work, and why they flag up the things they do

<https://ippsec.rocks>

- Just search the tool/service you want to learn about - e.g. "LDAP", "nmap", "LinPeas"
- Enumeration of a website: Tabby, Admirer
- Windows Enumeration: CTF, Forest, Resolute, Sauna

Find all these links and more at <https://www.shefesh.com/wiki/resources>



Hydra

Hydra is a tool for brute forcing usernames and passwords for different services. It has support for 50 different protocols.

Brute forcing is also very 'loud' - it can be easy to see the system is under attack.

We will use Hydra when we have a go at hacking a wifi network.

```
mole@Darth-Kali:~$ hydra -l roary -P /usr/share/wordlists/rockyou.txt 192.168.186.138 ssh -t 32
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes
```

```
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-11-13 11:37:06
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 32 tasks per 1 server, overall 32 tasks, 14344398 login tries (l:1/p:14344398), ~448263 tries per task
[DATA] attacking ssh://192.168.186.138:22/
[22][ssh] host: 192.168.186.138 login: roary password: tigers
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 19 final worker threads did not complete until end.
[ERROR] 19 targets did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-11-13 11:37:57
```



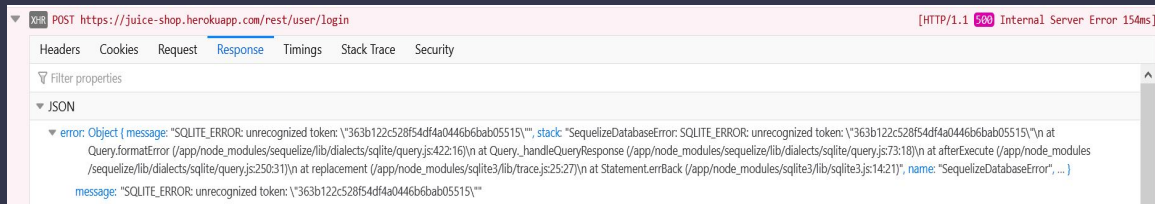
Enumerating SQL

Looking for potentially vulnerable inputs

- Look for areas on a site/service that might interact with a database - login forms, comment systems, search fields - anything else?
- This methodology also applies to looking for potential XSS entry points

Provoking error messages to enumerate version

- Try various control characters for different versions of SQL - " , ' ;
- If you provoke an error message, it may reveal something about the flavour of SQL/version that the database is using
- Prevent this by catching exceptions!



```
POST https://juice-shop.herokuapp.com/rest/user/login [HTTP/1.1 500 Internal Server Error 154ms]
Headers Cookies Request Response Timings Stack Trace Security
Filter properties
JSON
error: Object { message: "SQLITE_ERROR: unrecognized token: '\363b122c528f54df4a0446b6bab05515'", stack: "SequelizeDatabaseError: SQLITE_ERROR: unrecognized token: '\363b122c528f54df4a0446b6bab05515'" at Query.formatError (/app/node_modules/sequelize/lib/dialects/sqlite/query.js:422:16)\n at Query.handleQueryResponse (/app/node_modules/sequelize/lib/dialects/sqlite/query.js:73:18)\n at afterExecute (/app/node_modules/sequelize/lib/dialects/sqlite/query.js:250:31)\n at replacement (/app/node_modules/sequelize/lib/trace.js:252:7)\n at Statement.errBack (/app/node_modules/sequelize/lib/sqlite3.js:142:1)", name: "SequelizeDatabaseError", ... }
message: "SQLITE_ERROR: unrecognized token: '\363b122c528f54df4a0446b6bab05515'"
```

Running SQLmap

- SQLmap can automatically search for and execute SQL Injection attacks
- It has a huge range of functionality, supporting full database dumps, automatic blind injection, HTTP authentication, and can be configured directly from a saved HTTP request
- We won't cover it today in depth - but read more here: <https://github.com/sqlmapproject/sqlmap/wiki/Usage>



Practical

- <https://tryhackme.com/room/easypeasyctf>
 - Make an account
 - Connect using the VPN or use the attack box
 - If it doesn't work go to the /access page and change the region
 - Scan the target machine
 - You don't have to do the cron job bit
- If you finish try:
 - <https://tryhackme.com/room/ice>
 - <https://tryhackme.com/room/nmap01>
 - <https://tryhackme.com/room/kenobi>



Upcoming Sessions

What's up next?

www.shefesh.com/sessions

Any Questions?



www.shefesh.com
Thanks for coming!

