ETHICAL HACKING

LAB: STEALTHY SCANNING

LEARNING OBJECTIVES

- Understand scanning network with nmap

The 10 First Principles Covered by Lesson 4

- Data\Information Hiding
- Resource Encapsulation
- Simplicity
- Domain separation
- Process separation

LEARNING CONTENT

Exercise 1 – Stealthy Network Recon -Speed

One of the first things triggers that sets off an IDS when scanning is the number of packets sent in such a short amount of time. Nmap is know for being a very fast port scanner, and ironically, this speed is sometimes the very thing that makes your scan detectable.

Nmap has several "canned" speed adjustment options. If we wanted to slow our scan down we could use the -T option to specify one of these canned speeds. Scan your Windows Server 2012 for port 80 and 135 with a speed timing option of sneaky. Enter the following command.

nmap -sT 192.168.X.X -p 80,135 -T sneaky

Take note of the time it takes for this scan to finish (mine took about 45 seconds). See the result below.

```
root@kali:~# nmap -sT 192.168.180.133 -p 80,135 -T sneaky

Starting Nmap 7.60 ( https://nmap.org ) at 2018-01-02 23:46 CST
Nmap scan report for 192.168.180.133
Host is up (0.00036s latency).

PORT    STATE SERVICE
80/tcp  open  http
135/tcp open  msrpc
MAC Address: 00:0C:29:85:DD:9F (VMware)

Nmap done: 1 IP address (1 host up) scanned in 45.11 seconds
root@kali:~#
```

Figure 1 – TCP connect scan with sneaky timing options.

Now repeat the scan, except this time use the timing option of insane, which is must faster.

nmap -sT 192.168.X.X -p 80, 135 -T insane

```
root@kali:~# nmap -sT 192.168.180.133 -p 80,135 -T insane

Starting Nmap 7.60 ( https://nmap.org ) at 2018-01-02 23:48 CST
Nmap scan report for 192.168.180.133
Host is up (0.00031s latency).

PORT    STATE SERVICE
80/tcp  open  http
135/tcp open  msrpc
MAC Address: 00:0C:29:85:DD:9F (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.14 seconds
root@kali:~#
```

Figure 2 – Scan port 80, 135 with -T insane

For me that one took about 0.14 seconds! Much faster right? Which scan do you think is less likely to set off an IDS? As we can see the -T option changes the rate at which the scan finishes. This is because it's slowing down the speed at which the probing packets are being sent. The canned speeds are paranoid, sneaky, polite, normal, aggressive, and insane. It this is not granular enough, Nmap allows you to specify your scan delay speed using the --scan_delay option. Go ahead and enter the command below to send a probe every 5 seconds.

Nmap -sT 192.168.X.X -p 80, 135 --scan_delay 5s

```
root@kali:~# nmap -sT 192.168.180.133 -p 80,135 --scan_delay 5s

Starting Nmap 7.60 ( https://nmap.org ) at 2018-01-02 23:52 CST
Nmap scan report for 192.168.180.133
Host is up (0.00031s latency).

PORT    STATE SERVICE
80/tcp  open  http
135/tcp open  msrpc
MAC Address: 00:0C:29:85:DD:9F (VMware)

Nmap done: 1 IP address (1 host up) scanned in 15.10 seconds
root@kali:~#
```

Figure 3 – ports scan with --scan_delay 5s

That's about 5 seconds per port which in this case was two ports + an initial 5 delay. That means 15 seconds. Now change the scan delay speed to 1 second.

nmap -sT 192.168.X.X -p 80,135 --scan_delay 1s

```
root@kali:~# nmap -sT 192.168.180.133 -p 80,135 --scan_delay 1s

Starting Nmap 7.60 ( https://nmap.org ) at 2018-01-02 23:50 CST
Nmap scan report for 192.168.180.133
Host is up (0.00043s latency).

PORT    STATE SERVICE
80/tcp  open  http
135/tcp open  msrpc
MAC Address: 00:0C:29:85:DD:9F (VMware)

Nmap done: 1 IP address (1 host up) scanned in 3.11 seconds
root@kali:~#
```

Figure 4 – ports scan with --scan_delay 1s

It should be noticeably faster, somewhere around 3 seconds. We can also pass the speed argument in the form of milliseconds. Try this

nmap -sT 192.168.X.X -p 80,135 --scan_delay 5ms

```
root@kali:~# nmap -sT 192.168.180.133 -p 80,135 --scan_delay 5ms

Starting Nmap 7.60 ( https://nmap.org ) at 2018-01-02 23:53 CST
Nmap scan report for 192.168.180.133
Host is up (0.0036s latency).

PORT    STATE SERVICE
80/tcp  open  http
135/tcp open  msrpc
MAC Address: 00:0C:29:85:DD:9F (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.17 seconds
root@kali:~#
```

Figure 5 - ports scan with --scan_delay 5ms

That is 5 milliseconds and you should notice it finishes much faster. In my case it finished it about .17 seconds. The actual speeds could vary based on network conditions, configuration, etc. So if you speeds don't match mine exactly, it's all good.

Exercise 2 -Stealthy TCP Scanning

Nmap provides several methods to perform stealth tcp scanning. We'll be taking an in depth look at several of them. First up is the most popular Syn scan. Let's see what the Nmap man page has to say about the Syn Scan.

"This technique is often referred to an half-open scanning, because you don't open all full TCP connection. You send a SYN packet, as if you are going to open a real connection and then wait for a responses. A SYM/ACK indicates the port is listening (open), while a RST (reset) is indicative of a non-listener. If no response is received after several retransmissions, the port is marked as filtered. The port is also marked filtered if an ICMP unreachable error (type 3, code 1,2,3,9,10 or 13) is received. The port is also consider open if a SYN packet (without the ACK

flag) is received in response. This can be due to an extremely rare TCP feature known as a simultaneous open or split handshake connection."

Reference: https://svn.nmap.org/nmap/docs/nmap.1

STEP 1

Try performing a Syn scan against your Windows Server 2012 VM

nmap -sS 192.168.X.X

Now compare the results to a basic TCP Connect Scan.

nmap -sT 192.168.X.X

Results should be the same.

STEP 2

Next we'll look at Stealth FIN, Xmas Tree, and Null scans. They all exploit subtle behaviors in the TCP protocol if the protocol is implemented based on the RFC.

The TCP RFC document says the following

"if the [destination] port state is CLOSED …. an incoming segment not containing a RST causes a RST to be sent in response."

It further discuss packets sent to open ports without the SYN, RST, or ACK bits set, starting that:

" you are unlikely to get here, but if you do, drop the segment, and return. When scanning systems compliant with this RFC text, ay packet not containing SYN, RST, or ACK bits will result in a returned RST if the port is closed and no response at all if the port is open."

So the prototypes for the following scans are: Null has no bits set; FIN has just the FIN bit set; and Xmas is all bits sets. Going back to the excerpt form the manual, all three scans sets of responses will be treated similar.

RST received == closed

ICMP unreacable == filtered

Other == open or filtered

These scans can be a bit more error prone since operating systems implement RFC 793 differently. Referring back to the Nmap man page we find the following.

"A number of systems send RST responses to the proves regardless of whether the port is open or not. This causes all of the ports to be labeled closed. Major operating system that do this are Microsoft Windows, many Cisco devices, BSDI, and IBM OS/400."

It goes on to explain that this scan type does work against most Unix based systems. Try the FIN scan against your 2012 Windows VM.

Nmap -sF 192.168.X.X

Does it appear to work properly? If you notice, it came back and said all ports are closed. And we know for a fact this is not true concerning the 2012 VM. So basically we can say that it doesn't. We'll see the same results with a Xmas scan and Null scan. Run them both against the Windows Server 2012 VM, and you should notice that all ports come back as closed.

nmap -sX 192.168.X.X

nmap -sN 192.168.X.X

STEP 3

In this step you will be performing a decoy scan or what is also called a spoof scan. The general idea behind the decoy scan is to gorge form addresses in order to add other origin point for the scanning activity. This is essentially making logs more difficult to parse and the scanning more difficult to attribute. There are active response method to defeat this, but they aren't generally implemented. The reasoning there is, once you cross a certain threshold the traffic log become noise. As important note, also from the manual, is the use of the ME keyword. Some logging products start filtering the logging output after a certain number of concurrent scans are detected. This is an attempt to reduce noise in the logs. The manual recommends placing the ME keyword in the sixth or later position in order to try and take advantage of this behavior.

First let's set start Wireshark on our Windows Server 2008 VM so that we can prove the IP address that are doing the scan from the victim's perspective.

Start Wireshark as we previously did by double clicking the connected interface or the blue sharkfin icon on menu bar on your Windows Server 2012 VM Desktop. Now go back to your Kali Linux Server and type the following command to do a decoy scan.

nmap -sS 192.168.X.X -D10.10.10.10, 11.11.11.11, 1.1.1.1, 8.8.8.8

You should see that your scan results come back accurately based on previous scan results. More importantly, check your Wireshark capture. You should be able

to find packets which indicates that the IP address we entered after the D option have been scanning . See the result below on screenshot.

Figure 6 – Output of decoy scan on Linux server.

Packet captured on Wireshark.

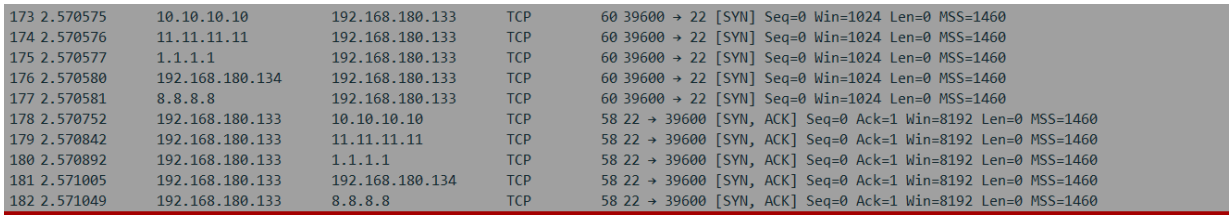

Figure 7 – Results of the Decoy scan against the Windows Server 2012.

Nmap basically sends packets that spoof each of the IP address entered after the -D option. Additionally it sends a packet with the source being the true IP address, since we do need responses back to determine port state.

Exercise 3 – Stealthy ICMP Scanning

Many admins today turn off ping responses (ICMP type 8) on most devices on their network, especially the ones reachable from the internet. They may have noticed that hosts which have ping disabled will be skipped by scanners such as Nmap. However, this is only true when Nmap is used with its default settings.

One way to find hosts not using ping is to use other ICMP codes and types. For example, we could use an ICMP Timestamp Request (type 13) packet to find listening hosts.

Remember if we're on the same network, nmap will use ARP's which would lead to the device being discovered even if ping is blocked. So we will use send_ip to get around this behavior and have nmap send ICMP requests. The switch -PP specifies timestamp request as the ICMP type. The -PE option is what actually enable the feature that allows us to send custom ICMP.

nmap -sP -PE –PP 192.168.X.X --send_ip

```
root@kali:~# nmap -sP -PE -PP 192.168.180.133 --send_ip

Starting Nmap 7.60 ( https://nmap.org ) at 2018-01-03 00:10 CST
Nmap scan report for 192.168.180.133
Host is up (0.00094s latency).
MAC Address: 00:0C:29:85:DD:9F (VMware)
Nmap done: 1 IP address (1 host up) scanned in 0.14 seconds
root@kali:~#
```

Figure 8 – nmap scan with custom ICMP request discovered status of Local host with MAC Address.

Exercise 4 -Idle Scanning

What if we wanted to not send our IP address to the target host at all ? In this case we would use the Idle scan type. It takes advantage of the IPID incrementing behavior on a no traffic host (idle host or zombie). The behavior being taken advantage of is that the IPID should increment for every packet the idle host sends. If it doesn't actually send any packets, every response we get form the host, the IPID should increment by 1.

The steps of the scan are:

1. Get the intitial IPID
2. Forge a packet to the target such that the response goes to the idle host (zombie)
3. IPID increments only if the idle host responds to the packet.
    a. It would not respond to a RST.

Here is a description of how it gets the initial IPID

1. Send a packet to our idle host.

2. Wait.
3. Send another packet to our idle host.
4. Verify that the IPID increment by 1

If it increments by more than one, you need a different idle host. Now we think we have a predictable IPID progression.

Spoof the idle host and send a single packet

If we spoof this idle host such that an open port is indicated by a non-RST response and closed port is indicated by a RST or no response, then we have this scan type. The IPID should only be incremented when the idle host send a packet.

Send a single packet to the idle host and harvest the IPID

Since non-RST should required a response and RST should not, we then have an open closed indicator on the zombie. If the IPID skips one, we believe the port to be open. If the IPID skips more than one, we may need to pick another zombie.

So now that you understand how the scan works, why might you use it?

- You absolutely MUST not have your IP shown in the target system log
- Abusing existing trust relationship to determine open ports.
- Consider what happens if you use a zombie that the target implicitly trusts.

For this lab you can use your either your lab Windows Host machine NAT'd ip address or pnw webserver IP address. The point is, When we look at Wireshark after doing our idle scan, the address that should show up as having done the scan is the zombie IP address (i.e. the host NAT'd IP address or the pnw webserver IP address) .

First we'll need to start a new capture in Wireshark on the Windows Server 2012 VM. Once your capture is running, go to your kali Linux Server and enter the following command on your terminal

nmap -sI 192.168.Y.Y:135 192.168.X.X

The first ip address is that of your zombie, which we're using either the host or the webserver as the zombie. This IP is followed by a colon and a port number. We're using port 135 since it's likely to always be open a window machine. However, if you were using a public web server as your zombie, you'd probably want to specify port 80. Another port that often work will be port 53 on the VMware IP that is your DNS server. The send IP address is your actual target, which in this case is Windows Server 2012 VM which has Wireshark running. Go ahead and enter your scan. Once the scan is finished, you should see that the results are goo. Now go to the Windows Server 2012 VM and look for the scan

traffic in Wiresharek. To make this easier, you'll probably want to stop the Wireshark window. Now let's create a filter for our zombie IP and target IP in Wireshark. In the filter area of Wireshark enter the following

ip.addr == 192.168.Y.Y && ip.addr == 192.168.X.X

Make sure you've entered your zombie and target IP address. For example my target IP is 192.168.180.133 and the zombie I used was pnw webserver IP address i.e. 205.215.103.116. See the screen shot below.

```
root@kali:~# nmap -sI 205.215.103.116:80 192.168.180.133
WARNING: Many people use -Pn w/Idlescan to prevent pings from their true IP.  On the other ha
nd, timing info Nmap gains from pings can allow for faster, more reliable scans.

Starting Nmap 7.60 ( https://nmap.org ) at 2018-01-04 15:44 CST
Idle scan using zombie 205.215.103.116 (205.215.103.116:80); Class: Incremental
Nmap scan report for 192.168.180.133
Host is up (0.051s latency).
Not shown: 990 closed|filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
23/tcp    open  telnet
53/tcp    open  domain
80/tcp    open  http
135/tcp   open  msrpc
445/tcp   open  microsoft-ds
49154/tcp open  unknown
49156/tcp open  unknown
49157/tcp open  unknown
49158/tcp open  unknown
MAC Address: 00:0C:29:85:DD:9F (VMware)

Nmap done: 1 IP address (1 host up) scanned in 12.80 seconds
root@kali:~#
```

Figure 9 – Idle scan output at kali Linux server.

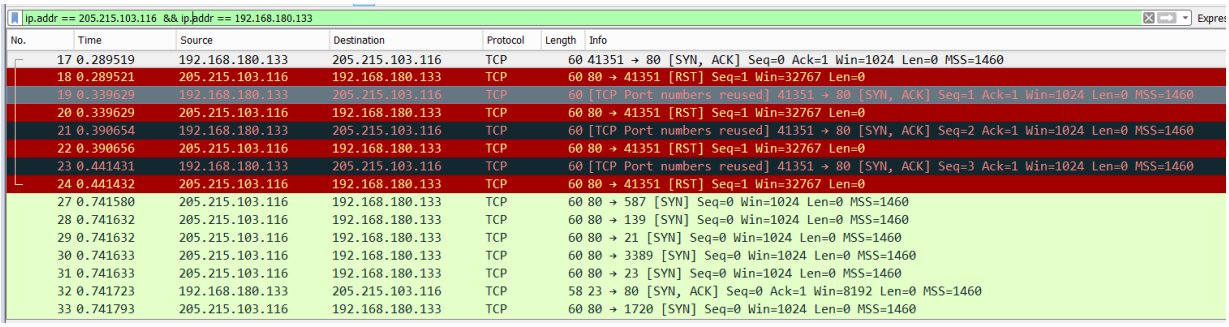

Figure 10 – Capture filter packets on Wireshark.

You should also notice that there are many packet between the Windows Server 2012 VM and the zombie. What you are seeing here is the actual port scan taking place. Ans if you were the victim in this case, you would certainly think the zombie was scanning you and not the actual attacker, which is the Kali Linux server in our case. This is why the idle scan is considered to be the only true "blind" port scan. Most of the stealth techniques we've learned can be combined to increase stealth even more.

WHAT TO SUBMIT

Submit your response with detailed screenshots.