ETHICAL HACKING

LAB: NETWORK RECON

OVERALL LEARNING OBJECTIVES

- Scan the network with nmap tool
- Network discovery with nmap

The 10 First Principles Covered by Lesson 4

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

LEARNING CONTENT

Exercise 1 Network Discovery with Nmap

Nmap is by far one of the most popular tools in the world of information security. This popularity can be attributed to may factors. One of which is the fact that it is extremely effective. Nmap was introduced as a port scanner, but it's far outgrown that title a this point. We will be using it in this exercise to do basic Network Discovery. We will start with a ping scan. Enter the following to discover all the devices on your network. Remember your network might be in a different range than the example. So make sure you're scanning your actual network range.

nmap -sP 192.168.180.0/24

```
root@kali:~# nmap -sP 192.168.180.0/24

Starting Nmap 7.60 ( https://nmap.org ) at 2018-01-02 16:10 CST
Nmap scan report for 192.168.180.2
Host is up (0.000099s latency).
MAC Address: 00:50:56:F3:B0:B2 (VMware)
Nmap scan report for 192.168.180.133
Host is up (0.000076s latency).
MAC Address: 00:0C:29:85:DD:9F (VMware)
Nmap scan report for 192.168.180.254
Host is up (0.000067s latency).
MAC Address: 00:50:56:E4:BA:70 (VMware)
Nmap scan report for 192.168.180.132
Host is up.
Nmap scan report for 192.168.180.134
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 2.05 seconds
root@kali:~#
```

Figure 1- nmap ping scan

As we can see, there appears to be 5 hosts up (you should have 4 or 5 depending on which VM's you have running). This is the most basic Nmap scan and it's designed strictly to tell which hosts are actually "up". We'll now run the same command, except this time we'll use the *packet_trace* option. This option is good for learning about the different types of nmap scans as you run them. It essentially causes nmap to output to the screen all requests that it sends to targets, as well as responses from the targets. Enter the command as follows.

nmap -sP 192.168.180.0/24 –packet_trace

You should see three key thing in the output. First you should see that Nmap was actually sending out ARP's and not pings. This is because you were scanning your own internal network, and Nmap is smart enough to know that arps will work since we're scanning the same layer 2 network. See the first part of the scan output below.

```
root@kali:~# nmap -sP 192.168.180.0/24 --packet_trace

Starting Nmap 7.60 ( https://nmap.org ) at 2018-01-02 16:12 CST
SENT (0.0404s) ARP who-has 192.168.180.1 tell 192.168.180.132
SENT (0.0405s) ARP who-has 192.168.180.2 tell 192.168.180.132
SENT (0.0406s) ARP who-has 192.168.180.3 tell 192.168.180.132
SENT (0.0406s) ARP who-has 192.168.180.4 tell 192.168.180.132
SENT (0.0407s) ARP who-has 192.168.180.5 tell 192.168.180.132
SENT (0.0408s) ARP who-has 192.168.180.6 tell 192.168.180.132
SENT (0.0409s) ARP who-has 192.168.180.7 tell 192.168.180.132
SENT (0.0409s) ARP who-has 192.168.180.8 tell 192.168.180.132
```

Figure 2 – First part of ping scan with ARP's show using the packet trace option.

If you continue to scroll past all the arps, you'll eventually see responses from the machines on the network that are up. See below.

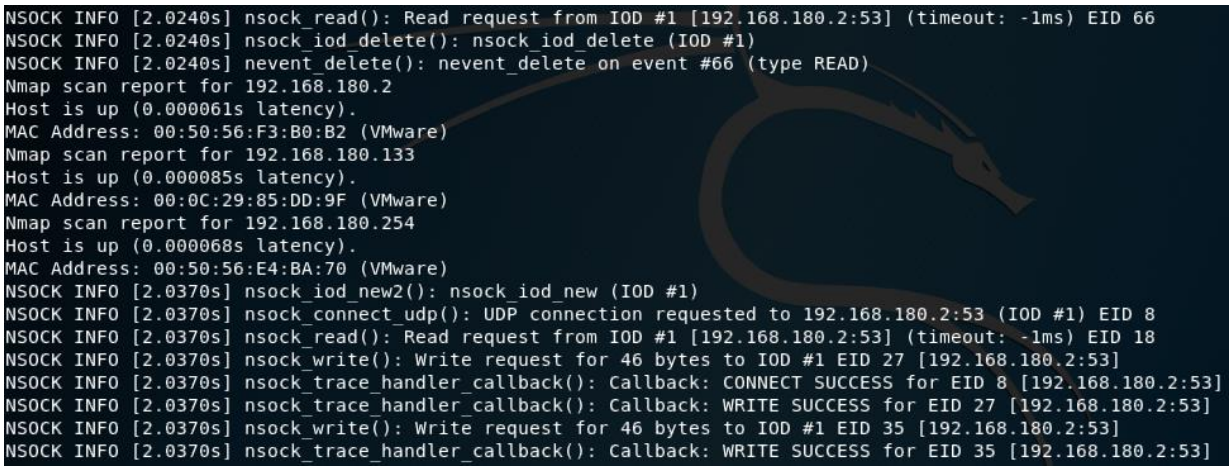

Figure 3 – Nmap ping scan showing return traffic from targets using the packet trace option.

And then finally you should see the same output you see if you would have left off the packet trace option.

Exercise 2 Scanning Outside Networks

NOTE: MAKE SURE YOU HAVE INTERNET ACCESS ON YOUR VM'S.

Nmap includes a scan called the List Scan which will resolve the given IP address or range of IP address, to a host name. Let's try it on the Purdue University Northwest network. Pick another network if you'd like. However, don't try and use the classroom address range as there are probably no reverse lookup entries anywhere in our internal DNS. The below scan will not work if you don't have internet access.

nmap -sL 205.215.116.0/24

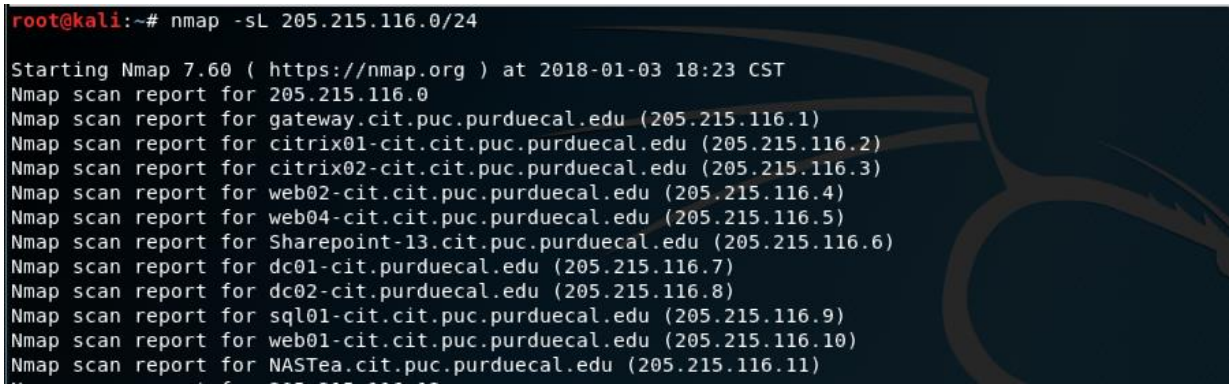

Figure 4 List scan of the 205.215.116.0 network.

Sometimes we might not have nmap handy. We can still do some basic outside discovery using tools that ship with most operating system

including Windows and most Linux distros. One example of this is *traceroute*. It allows us to see the shortest route to a host. Do a *traceroute* to www.pnw.edu

traceroute www.pnw.edu

Did you make it to pnw.edu? How many hops did it take? Note: The way that VMW are networking is set up can impact your result here. Having the Linux server set up as "Bridged" for the networking type will work the best.

Exercise 3 Host Discovery

One of the fist steps in host discovery could be to discover other hosts on the network, that might be in a range different than the range you've been checking. To validate this we'll use a powerful arp based tool named Netdiscover. It is considered to be an active/passive tool because it not only use arps to discover devices, but it also passively monitors other traffic and tries to identify hosts based on this traffic. Run it by simply typing *netdiscover* from a command shell.

netdiscover

It will take it a few minutes to check the entire possible network range (it default to class B). But when it's finished, you should see something like this following;

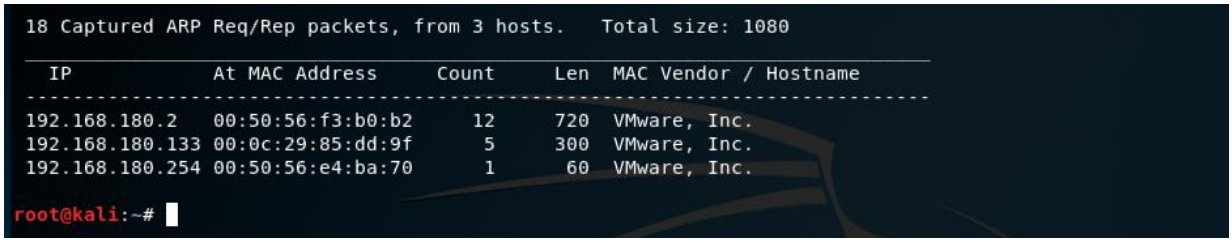

Figure 5 – Results of running netdiscover.

In the results above, you can see that my NAT'd network is in 192.168.180.0 range. Your will like be different, probably 192.168.120.0. But looking at the device above, 192.168.180.133 is my windows 2012 server VM, 192.168.180.2 is my NAT'd network router to the outside world (gateway), and 192.168.180.254 is another virtual router/dhcp server. It's safe to say *netdiscover* can be very userful! Hit *Ctrl + C* if you need to stop it at any time.

Since we're confident that we know which network we're on, let's move on to actual host discovery now. We'll be using *nmap* and *hping3.*

Hping3 is a versatile custom packet crafting program, which ca be installed as long as you have write access to any directory on the Linux box you want to run it on, and you have access to some of the common libraries. Hping3 is capable of sending ICMP packets. Hping3 can handle fragmentation, arbitrary packet body contents and size. It can be used in order to transfer files encapsulated under certain protocols. We will use *hping3* to determine if a specific port is open on your Windows 2012 Server. We'll check to see if port 23 is open. Do that by entering the following command (replacing your server's IP address)

Hping3 -S 192.168.X.X -p 23

Here are what the results should be similar to:

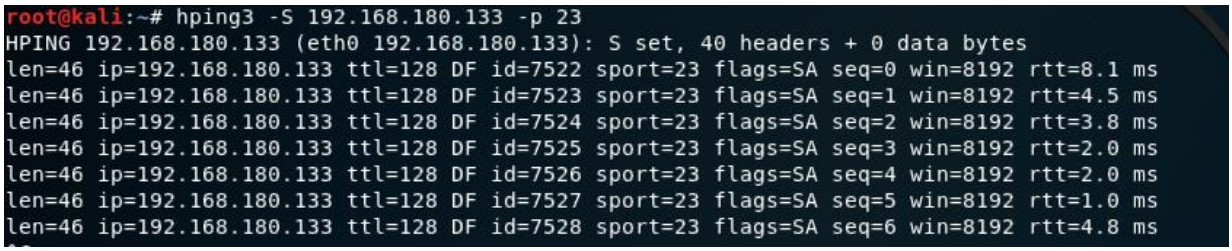

Figure 6 – Hping3 results against the 2012 VM's port 23

Basically, the -S is telling hping3 to use TCP Syn packets for sending. The -p 23 at the end instructs hping3 to send these TCP Syn packets to port 23 on the target IP address. Below is what the results should look like. Hit Ctrl+C to stop it. In the graphic above, the SA flags on the return packets let us know the port in question (23), is probably open, hence the SA (SYN/ACK) responses.

Try the same scan with hping3 against port 20 on the 2012 VM. It should look like the following:

Hping3 -S 192.168.X.X -p 20

You should see that now instead of SA(Syn/Ack) packets coming back, RA (RESET/ACK) packets are coming back. This tells us that port 20 must be closed.

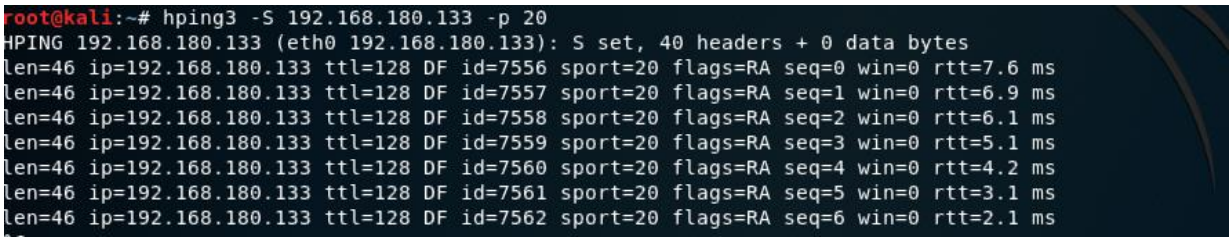

Figure 7 – Hping3 results against the 2012 VM's port 20

We can also use hping3 to start at one port, then sequentially increase to the next port which each additional packet it sends out. Do that by entering the following command:

Hping3 -S 192.168.X.X -p ++20

As you can see in my example, hping3 is sending Syn packets to the target starting at port 20 or after 20 depending on the server port status then going to the next port, and the next port. Notice the SA coming back for port 22 vs RA for other ports shows in the graphic.

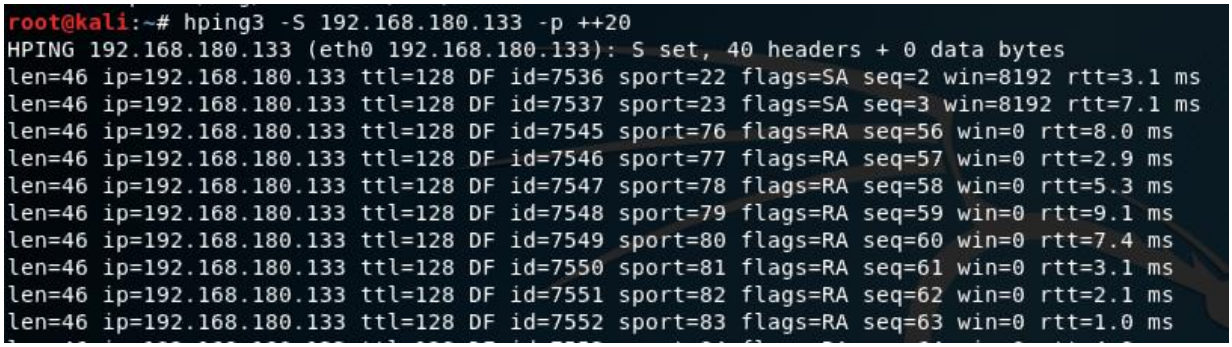

Figure 8 – Using Hping3 to scan ports sequentially.

One really powerful feature of Hping3 is its ability to spoof IP packets. In other words, it can forge the source address in the packet and make it appear to be coming form any source IP we choose. To illustrate this we will run Wireshark on our 2012 VM and watch the packets as they come over. Open your 2012 Windows VM and open the Wireshark from your start windows. Once it's started, click the connected network interface or the blue Shark fin icon as show in the figure.

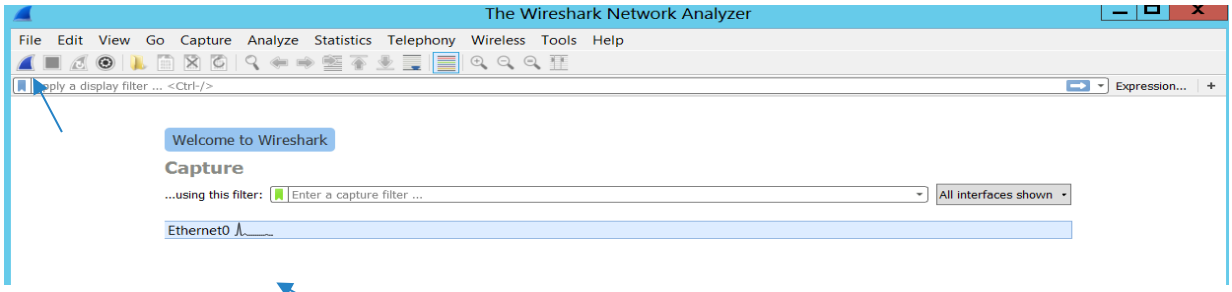

Figure 9 – Start Wireshark.

Now go back to your kali Linux and enter the following hping3 command from a terminal.

Hping3 -a 216.58.193.132 -S 192.168.X.X -p 80

```
root@kali:~# hping3 -a 216.58.193.132 -S 192.168.180.134 -p 80
HPING 192.168.180.134 (eth0 192.168.180.134): S set, 40 headers + 0 data bytes
len=44 ip=192.168.180.134 ttl=64 DF id=0 sport=80 flags=SA seq=0 win=29200 rtt=8.0 ms
len=44 ip=192.168.180.134 ttl=64 DF id=0 sport=80 flags=SA seq=1 win=29200 rtt=6.1 ms
len=44 ip=192.168.180.134 ttl=64 DF id=0 sport=80 flags=SA seq=2 win=29200 rtt=6.2 ms
```

Figure 10 – Hping3 output on linux server.

Now go at your Wireshark capture on your 2012 VM, and notice where the packets appear to be coming from. You might need to stop your Wireshark capture if the packet are scrolling too fast for you to make sense of what you're seeing. In the graphic, we can clearly see the 216 address appearing to send the SYN (Synchronize Request) packets and we can see our victim responding back to the 216 address with SYN/ACK's

No.	Time	Source	Destination	Protocol	Length	Info
3	0.489860	205.215.103.166	192.168.180.133	TCP	60	1467 → 80 [SYN] Seq=0 Win=512 Len=0
4	0.489962	192.168.180.133	205.215.103.166	TCP	58	80 → 1467 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
5	0.490042	205.215.103.166	192.168.180.133	TCP	60	1467 → 80 [RST] Seq=1 Win=32767 Len=0
6	1.487868	205.215.103.166	192.168.180.133	TCP	60	1468 → 80 [SYN] Seq=0 Win=512 Len=0
7	1.487919	192.168.180.133	205.215.103.166	TCP	58	80 → 1468 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
8	1.487995	205.215.103.166	192.168.180.133	TCP	60	1468 → 80 [RST] Seq=1 Win=32767 Len=0
9	2.485607	205.215.103.166	192.168.180.133	TCP	60	1469 → 80 [SYN] Seq=0 Win=512 Len=0
10	2.485663	192.168.180.133	205.215.103.166	TCP	58	80 → 1469 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
11	2.485745	205.215.103.166	192.168.180.133	TCP	60	1469 → 80 [RST] Seq=1 Win=32767 Len=0

Figure 11 – Wireshark showing the hping3 IP spoof happening.

Now you should also notice that there doesn't appear to be any responses coming back from the victim from the Linux Server perspective. Notice how your hping3 command appears to just be hung? This is because we spoofed the packets we are sending to the victim, and the victim is responding to the spoofed address, not ours! We shouldn't expect to get response back.

Exercise 4 Port Scanning with Nmap

In this exercise we will do basic port scanning. These exercise are designed to get you more familiar with the process of port scanning using Nmap. Here's the definition of connect scanning form the nmap website.

"*This is the most basic form of TCP scanning. The connect() system call provided by your operating system it used to open a connection to every interesting port on the machine. If the port is listening, connect() will succeed, otherwise the port isn't reachable. One strong advantage to this technique is that you don't need any special privileges. Any user on most UNIX boxes is free to use this call."*

Reference: http://nmap.org/nmap_doc.html

STEP 1

The scan described is easily detected as the target host will usually log a large number of connections and connection attempts (to closed ports). Scan your 2012 Windows VM with the following command. Observe the output.

Nmap -sT 192.168.X.X –packet_trace

Just a reminder, make sure you've scanned your own 2012 Windows VM IP address and not the example IP shown here. As for the results you should see a pretty long list of open ports. Most of these ports are standard ports for a Windows 2012 Active Directory Service Domain controller, Which is what your 2012 VM is. Some common one are 21 for ftp, 22 for ssh, 23 for telnet, 53 dns, 80 http, 88 Kerberos, 135, 139 and 445 which are common ports, 389 ldap, and several other for Global Catalog services.

```
CONN (4.8916s) TCP localhost > 192.168.180.133:1812 => Operation now in progress
CONN (4.8917s) TCP localhost > 192.168.180.133:14442 => Operation now in progres
CONN (4.8918s) TCP localhost > 192.168.180.133:8300 => Operation now in progress
Nmap scan report for 192.168.180.133
Host is up (0.00036s latency).
Not shown: 996 filtered ports
PORT    STATE SERVICE
22/tcp  open  ssh
23/tcp  open  telnet
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 5.04 seconds
```

Figure 12 – nmap output with packet trace option for port scan.

Now try running a TCP Connect scan again two devices in your network. Use your 2012 Windows VM and your Linux server VM (yes Nmap can even scan the server from which it's running). See the example below.

Nmap -sT 192.168.180.133, 134

In the above example, we're scanning the IP address of 192.168.180.133 and 192.168.180.134, my Windows 2012 server and Linux server. It's important to note that Nmap is not actually attempting to scan all ports. The ports scanned by default are the 1000 most commonly used ports. You should see that for your Windows 2012 VM IP address, ,most of the ports you saw earlier are open. On the Linux server, you should see that it reports few open ports compare to Windows 2012 VM.

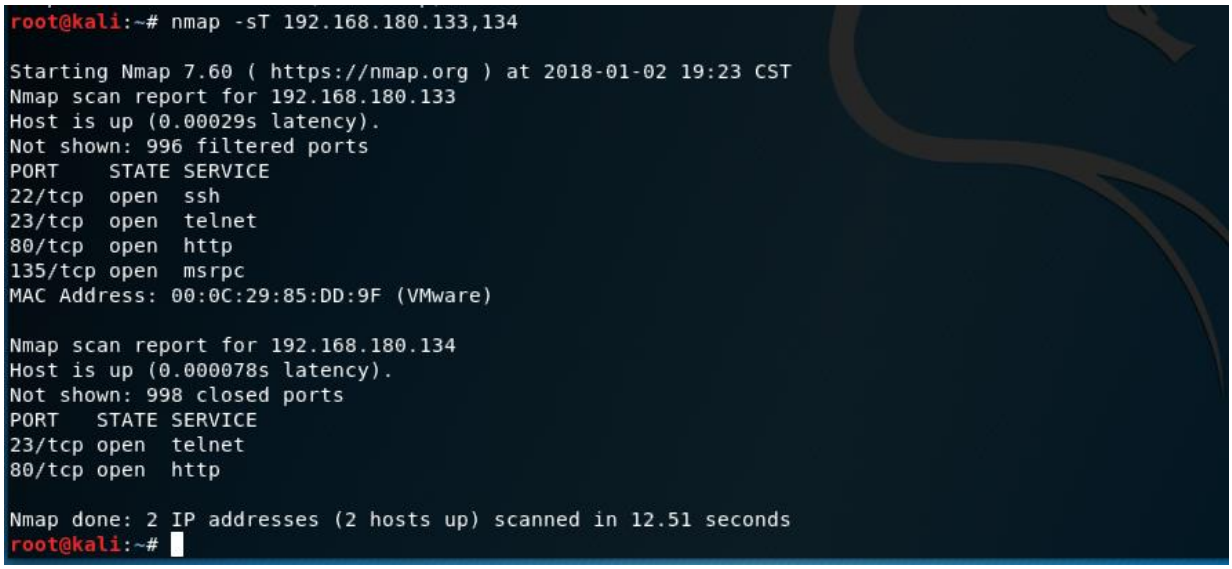

Figure 13 – TCP Connect Scan of multiple IP address.

Now lets stop the Apache server on Linux server with following command.

service apache2 stop

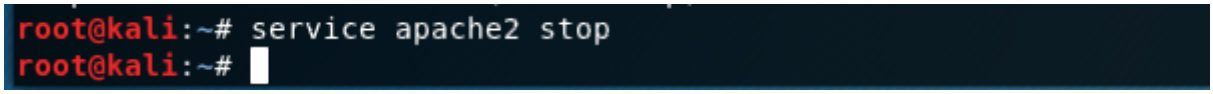

Figure 14 – Stopping Apache web server on Linux.

We can also specify specific ports for Nmap to scan if we don't want it to scan all ports. Scan your 2012 Windows or Linux server port 80. The syntax is below.

Nmap -sT 192.168.X.X -p 80

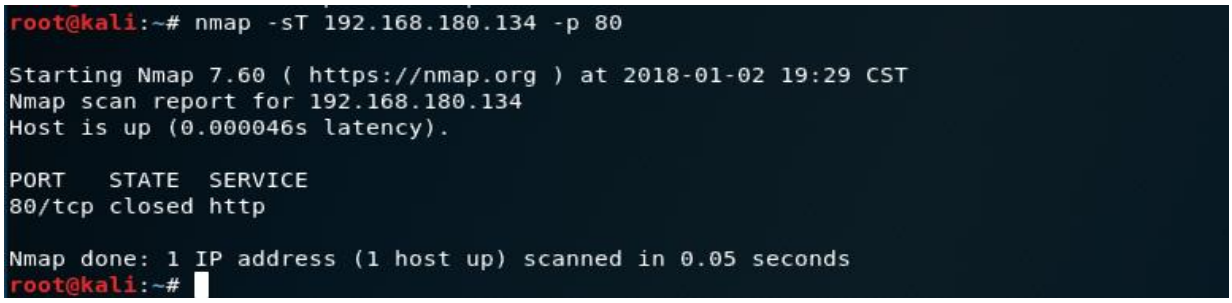

Figure 15 – Scan specific port on network host IP address with nmap.

With internet access we can try to scan the internet host port like port 80 on the Purdue Northwest University webserver.

Nmap -sT www.pwn.edu -p 80

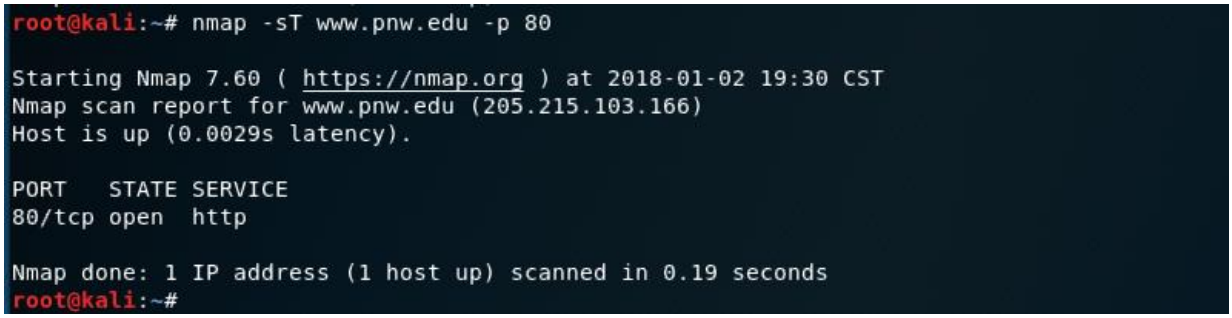

Figure 16 – Internet host port scan with nmap

Does it come back with result that say port 80 is open at www.pnw.edu? It should because we know for sure there is a web server running there that servers out the Purdue Northwest home page. If you were scanning a target that has ICMP Echo Request disable (not allowing ping), then you might have to tell Nmap not to ping a target before doing its scan. By default Nmap will only scan targets that respond to ping. This is to make Nmap run faster. The logic is why scan a target that's not up? That would be wasting time, and bandwidth. So regardless of the scan type selected, Nmap will first try and "discover" the target, then move on to do the instructed scan type only if the target responds to discovery pings. To tell Nmap not to ping we issue -P0 (zero) option. If Purdue Northwest were blocking ICMP we'd have to issue the following command.

nmap -sT www.pnw.edu -p 80 -P0

Notice the P is capital and the 0 is a zero, not the letter O.

Ping disable host

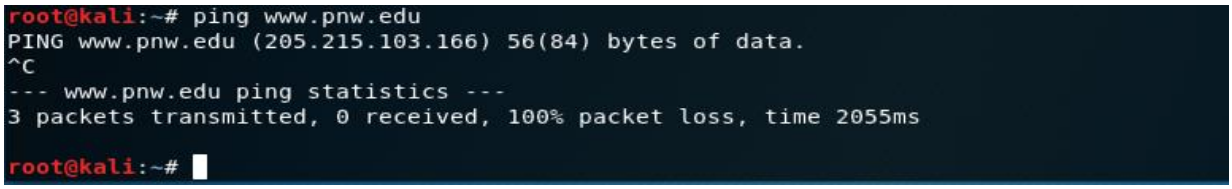

Figure 17 – Ping disable host

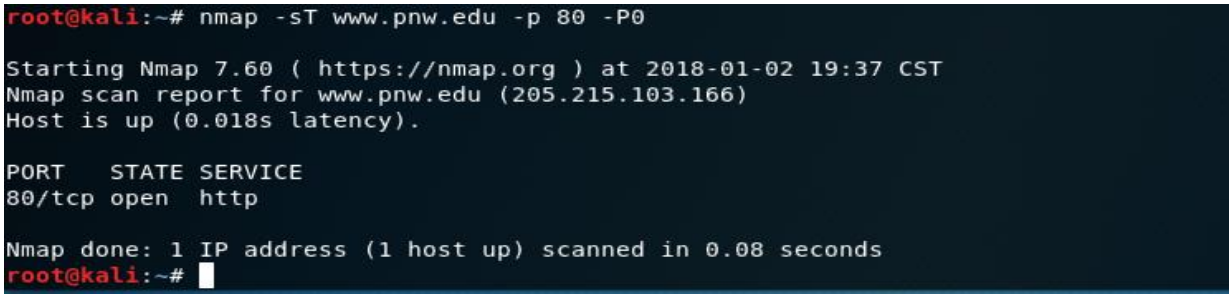

Figure 18 – Nmap scan for ping disable host.

STEP 2

Next we'll learn how to perform scans to find which UDP ports are in use. Nmap does this by sending a zero byte UDP packets to each port on the target machine or device. If it received an ICMP port unreachable message, it determines the port is closed. Otherwise it assumes the port is open.

The problem with this technique is that it's not uncommon for network equipment to look unreachable messages. This can lead to false positive. Due to these factors, it can be difficult to determine real open UDP ports vs. filtered false positive ones. To UDP scan your 2012 Windows VM enter the following.

nmap -sU 192.168.X.X -T insane

We slipped in a new scanning option there. The -T option is for timing, and we picked the canned scanning speed of insane which really fast. UDP scans can take a considerable amount of time, which is why we sped the scan up with -T insane. Try scanning just UDP port 161 for SNMP.

nmap -sU 192.168.X.X -p 161

You should see that it comes back as open|filtered. This means nmap wasn't able to determine for sure if the port is actually open or filtered (firewalled).

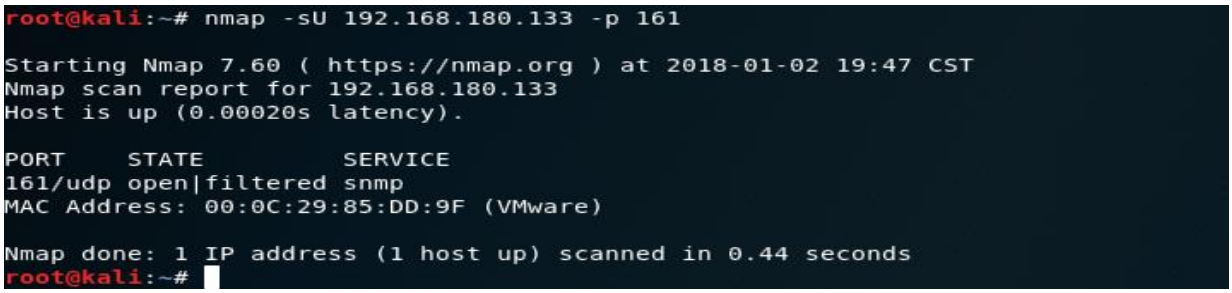

Figure 19 – UDP port 161 on the Windows 2012 VM.

Nmap also allows the ability to combine certain types of scans as long as they are compatible scan types. For example, we can combine TCP connect and UDP scans. Let's add packet trace to the scan along with insane speed option. You will now see firsthand just how much traffic is being generated with this one scan. Be patient, this scan might take up to a minutes to complete.

nmap -sU -sT 192.168.X.X -packet_trace -T insane

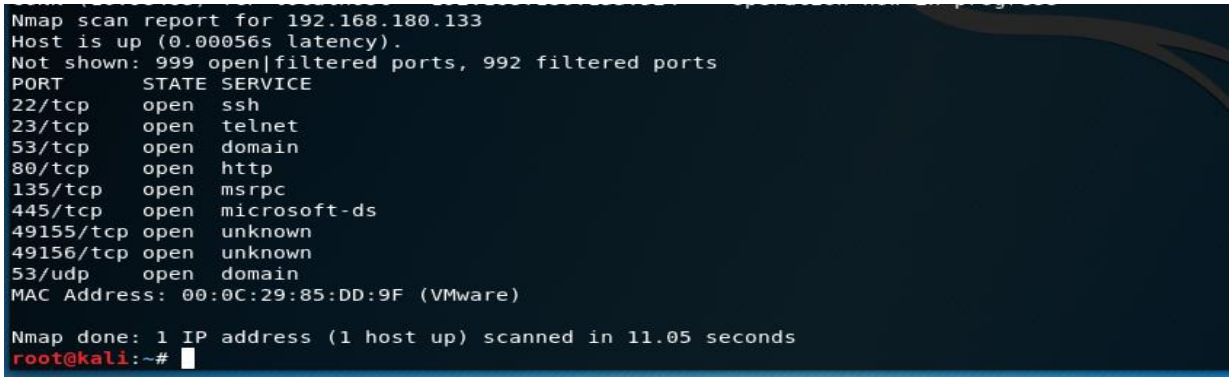

Figure 20- Combining TCP connect scan and UDP scan.

By default Nmap outputs its results to a somewhat flat text format. We're going to learn how to output to a greppable format using the -oG option. Enter the following Nmap command to run a ping scan and write the result to a greppable file. Remember to scan your network and not the one in the example.

Nmap -sP 192.168.X.0/24 -oG pingscan.out

Now cat the pingscan.out file nmap just created and notice how it is nicely delimited the output.

```
root@kali:~# nmap -sP 192.168.180.0/24 -oG pingscan.out

Starting Nmap 7.60 ( https://nmap.org ) at 2018-01-02 20:18 CST
Nmap scan report for 192.168.180.2
Host is up (0.00063s latency).
MAC Address: 00:50:56:F3:B0:B2 (VMware)
Nmap scan report for 192.168.180.133
Host is up (0.0051s latency).
MAC Address: 00:0C:29:85:DD:9F (VMware)
Nmap scan report for 192.168.180.254
Host is up (0.000059s latency).
MAC Address: 00:50:56:E4:BA:70 (VMware)
Nmap scan report for 192.168.180.134
Host is up.
Nmap done: 256 IP addresses (4 hosts up) scanned in 2.20 seconds
root@kali:~# ls
Desktop  Documents  Downloads  lab  Music  Pictures  pingscan.out  Public  Templates  Videos
root@kali:~#
```

Figure 21 – Output to a greppable format using the -oG option with Nmap.

cat pingscan.out

```
root@kali:~# cat pingscan.out
# Nmap 7.60 scan initiated Tue Jan  2 20:18:39 2018 as: nmap -sP -oG pingscan.out 192.168.180.0/24
Host: 192.168.180.2 ()  Status: Up
Host: 192.168.180.133 ()        Status: Up
Host: 192.168.180.254 ()        Status: Up
Host: 192.168.180.134 ()        Status: Up
# Nmap done at Tue Jan  2 20:18:41 2018 -- 256 IP addresses (4 hosts up) scanned in 2.20 seconds
root@kali:~#
```

Figure 22 – Reading the pingscan.out file.

Now cat it and grep for the string "UP".

Cat pingscan.out | grep Up

```
root@kali:~# cat pingscan.out | grep Up
Host: 192.168.180.2 ()  Status: Up
Host: 192.168.180.133 ()        Status: Up
Host: 192.168.180.254 ()        Status: Up
Host: 192.168.180.134 ()        Status: Up
root@kali:~#
```

Figure 22 – Combining cat and grep on nmap output.

Now send those results to an output file.

Cat pingscan.out | grep Up > pingscan.out1

Now cat the resulting file. It should match the output form the first grep infused cat command, however note that the same output is now in a file! What we rally need here is for this file to contain just a list of IP address.

.

```
root@kali:~# cat pingscan.out | grep Up >> pingscan.out1
root@kali:~# ls
Desktop  Documents  Downloads  lab  Music  Pictures  pingscan.out  pingscan.out1  Public  Templates  Videos
root@kali:~#
```

Figure 22 – Creating new pingscan.out1 file with grep up status from pingscan.out file.

So if you went ahead and did a cat on the pingscan.out1 file, it should have looked like this

```
root@kali:~# cat pingscan.out1
Host: 192.168.180.2 ()  Status: Up
Host: 192.168.180.133 ()        Status: Up
Host: 192.168.180.254 ()        Status: Up
Host: 192.168.180.134 ()        Status: Up
root@kali:~#
```

Figure 23 – Reading pingscan.out1 file.

Now let's add the Linux command "cut" to it to give us a nice list of IP address.

cat pingscan.out1 | cut -f2 -d " " > pingscan.out2

In this cut part of the command we are saying "keep" field 2 and we're using a single space as our delimiter, hence the -d " ", which is -d followed by double quotes, a space, then another set of double quotes. Then finally we output the results to a file named pingscan.out2. Cat the newly created file and you should see that you're left with a list of IP address.

cat pingscan.out2

```
root@kali:~# cat pingscan.out1 | cut -f2 -d" " > pingscan.out2
root@kali:~# cat pingscan.out2
192.168.180.2
192.168.180.133
192.168.180.254
192.168.180.134
root@kali:~#
```

Figure 24 – Final results after cutting out just IP.

STEP 3

Often times, we'll only need to scan for specific protocols instead of open ports. For example it's sometimes useful to enumerate what protocols are in use on a router and other broader devices to aid in finger printing them. For this Nmap includes the protocol scan. Here's what the Nmap man pages say about the Protocol Scan.

"If Nmap receives any response in any protocol form the target host, Nmap marks that protocol as open. An ICMP protocol unreachable erro (type3, code 2) causes the protocol to be marked as closed. Other ICMP unreachable error (type 3, code 1, 3, 9, 10 or 13) cause the protocol to be marked filtered (though they prove that ICMP is open at the same time). If o response is received after retransmissions, the protocol is marked open|filtered."

Let's run a protocol scan against the Windows server 2012 VM.

nmap -sO 192.168.X.X -T insane

The results should look similar to the output below. Remember, the numbers you see are not port numbers in this case, but hey are protocol numbers. Each protocol has a protocol number that represents that protocol. You can find a protocol number lookup chart here.

https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml

Below is a sample output

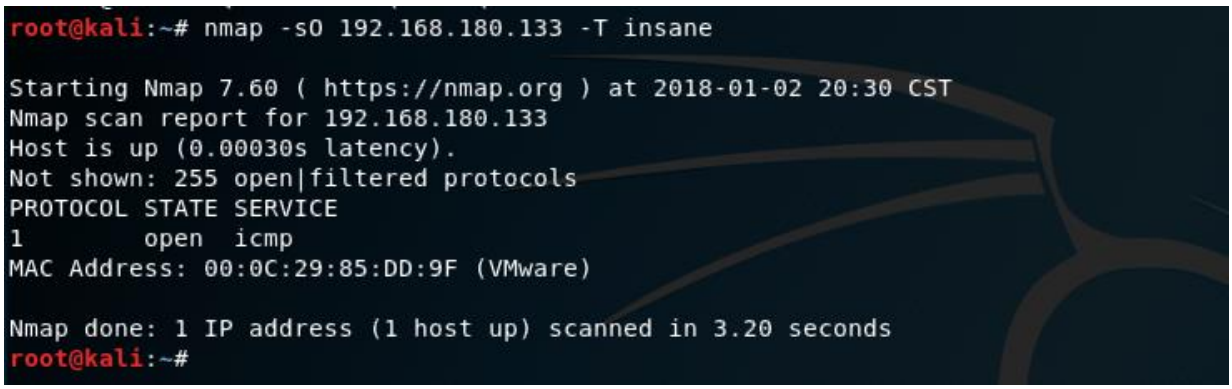

Figure 24 – Nmap protocol scan results.

All the scans we've covered so far are basic which are not really designed for stealth, but reliability. We will work on stealth scanning lab on upcoming lab sessions.

WHAT TO SUBMIT

Submit your response with detailed screenshots.