Network Measurement Lab. Individual Lab.
Analysis Using Wireshark

Group7: Behnam Sobhani Nadri

January 2022

Abstract

Network Mapper or Nmap is a software tool that sends packets using protocols like TCP, UDP, ICMP etc. to determine the availability of the hosts in a network and find information about the services and operating systems running on them. The approach is based on host discovery and then port recognition to distinguish the active IP addresses and their connections. The objective of this experiment is to analyze the mechanism of nmap by using Wireshark packet tracing tool.

1 Network Setup

The experiment is carried out at home using Ubuntu terminal which is running on a Lenovo ideaPad Z510 which is running Ubuntu 20.04 LTS and the target host is Galaxy Tab A7. Both hosts are conncted to a wireless access point on a Samsung Galaxy A30s cellphone and the cellphone connected to a wireless router located in the building corridor. Internet Service Provider(ISP) and connection type is Vodafone Italia and ADSL respectively. Figure 1 is showing the network under the experiment.

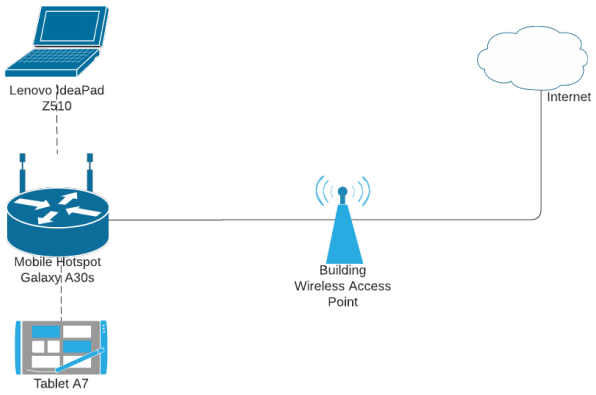

Figure 1: Block Diagram of the Tested Experiment

Table 1: Network setup

	IP Address	MAC Address
Lenovo Z510	192.168.128.181	FC:F8:AE:CB:21:27
Tablet A7	192.168.128.141	26:C1:30:76:BE:42
Galaxy A30	192.168.128.252	3A:18:2F:EF:3A:30

Table 2: Physical Layer Configuration on PC

Type	Duplex Status	Interface Speed	Link Status
Wireless	HALF-DUP	144.4Mbps	Detected

2 Experiment

2.1 Host Discovery

In the first step nmap search the local network to find active hosts. Nmap in unprivileged mode uses two frequently-used protocols to find the hosts in the network. The most interesting ones are http (port 80) and https (port 443) and they are likely running on most of active hosts. In unprivileged mode Nmap discover the target by scanning port 80 and 443. It sends two `SYN` on port 80 and 443. Target responding with a `RST` is surely active otherwise it is likely unavailable. It should be noted that because of IP layer, the attacker must send `arp` to find the MAC address of the target.(look at figure 3. In the privileged mode, host is discovered only by the help of the initial `arp` request. However, the attacker send `arp` twice to be sure about target availability.

When the TCP scan is unprivileged, nmap by default does a `connect` system call towards the target to get information from the target port. while if it runs the command as a privileged user, by default a `SYN` scan is done towards the target. Two approaches are slightly different in terms of mechanism and the speed and probability of being found by the target firewall.

In general three possible scenarios could happen in TCP scan:

- First, the target port doesn't respond to the SYN, when the target's Firewall blocks `SYN`s. In this case nmap send the `SYN` again and if it doesn't receive any response it consider the port as `Filtered`.
- Second, the target port send `RST` response. TCP generate `RST` packets when the received `SYN` from a non-existing connection with wrong header.
- Third, TCP responds with `SYN,ACK` and it means the port is open.

2.2 Unprivilaged TCP Scan

As we run the command `nmap 192.168.128.141 -p 7,22,445`, nmap by default run a `connect` system call (or by `-sT` option) and send non-raw packets (usually of size 74bytes, 54 bytes of header and 20 bytes of application payload) toward the target. As the attackers send a SYN to handshake with the target, if the target port is `open` it replies with a `[SYN,ACK]` to establish a connection. The attacker acknowledges this message and the target open a connection with the attacker with a data packet (e.g a SSH protocol, depending on the port number). Immediately after receiving a data packet, the attacker kills the connection with the target with a `RST` packet and mark the port as `open`.

Table 3: IP layer configuration on PC

IP Address	Netmask	Default Gateway	#PCs_on_the_Subnet
192.168.128.181	255.255.255.0	192.168.128.141	1

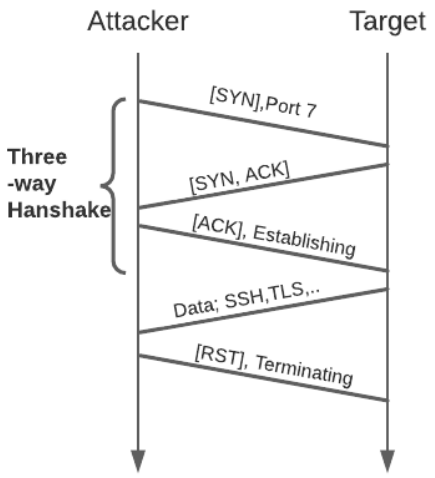

Figure 2: Timing Diagram of nmap *connect* System call for port 7

While if the connection is closed, the target interrupt the request with `RST` message. This type of scan usually takes more time than `SYN` scan because it complete a three-way handshake with the target TCP port. The diagram showing in Figure 2 showing the procedure. The difference between a normal TCP connection and nmap *connect* system call is that in a normal connection the connection is closed with three-way handshake (`[FIN], [FIN,ACK]`) as well as establishing it. While in nmap, the connection is terminated by the attacker with `RST` message. Looking at the trace of packets in figure 3 Wireshark showing that `SYN` requests from nmap were sent from the same ports. In response, all target ports terminate the connection using `RST` meaning that they were closed.

	Time	Source	Destination	Protocol	Length	Info
15	57.460205	IntelCor_cb:21:27	Broadcast	ARP	42	Who has 192.168.128.141? Tell 192.168.128.181
16	57.564923	26:c1:30:76:be:42	IntelCor_cb:21:27	ARP	42	192.168.128.141 is at 26:c1:30:76:be:42
17	57.564937	192.168.128.181	192.168.128.141	TCP	74	33702 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1619138587 TSecr=0 WS=128
18	57.564954	192.168.128.181	192.168.128.141	TCP	74	36366 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1619138587 TSecr=0 WS=128
19	57.753215	26:c1:30:76:be:42	Broadcast	ARP	42	Who has 192.168.128.181? Tell 192.168.128.141
20	57.753229	IntelCor_cb:21:27	26:c1:30:76:be:42	ARP	42	192.168.128.181 is at fc:f8:ae:cb:21:27
21	57.760082	192.168.128.141	192.168.128.181	TCP	54	80 → 33702 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
22	57.760083	192.168.128.141	192.168.128.181	TCP	54	443 → 36366 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
23	57.761358	192.168.128.181	192.168.128.252	DNS	88	Standard query 0xc8d4 PTR 141.128.168.192.in-addr.arpa
24	60.011027	192.168.128.141	224.0.0.251	MDNS	103	Standard query 0x000a PTR _CC32E753._sub._googlecast._tcp.local, "QM" question PTR _googlecas

Figure 3: The trace of TCP packets sent by nmap as unprivilaged user

2.3 Privilaged TCP Scan

Upon a privileged TCP scan, `sudo nmap 192.168.128.141 -p 7,22,445`, nmap by default run a `SYN` scan towards the target (or using (`-sS` option in other case). In `SYN` scan, nmap sends raw packets to the target port and wait for any response. If it receives a `[SYN,ACK]` it mark the target port as `open` and terminate the connection immediately (by `RST`) without a complete

three-way handshake. In result, `SYN` scan time the scan is pretty much faster than *connect* system call which does a *complete* three-way handshake. Figure 4 shows the timing diagram of nmap `SYN` scan.

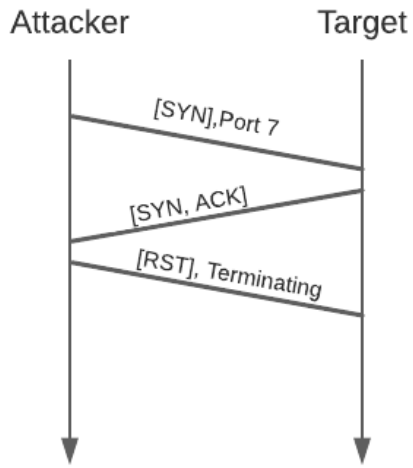

Figure 4: Timing Diagram of nmap `SYN` Scan for port 7

Looking at wireshark packet traces (Figure 5) shows that in packets of 58bytes size are sent to the target to scan the target ports. It is observed that nmap chooses a random port and for the subsequent ports, increments it.

	Time	Source	Destination	Protocol	Length	Info
45	112.549073	IntelCor_cb:21:27	Broadcast	ARP	42	Who has 192.168.128.141? Tell 192.168.128.181
46	112.749389	IntelCor_cb:21:27	Broadcast	ARP	42	Who has 192.168.128.141? Tell 192.168.128.181
47	112.800266	26:c1:30:76:be:42	IntelCor_cb:21:27	ARP	42	192.168.128.141 is at 26:c1:30:76:be:42
48	112.845682	192.168.128.181	192.168.128.252	DNS	88	Standard query 0x6566 PTR 141.128.168.192.in-addr.arpa
49	112.848842	192.168.128.252	192.168.128.181	DNS	88	Standard query response 0x6566 No such name PTR 141.128.168.192.in-addr.arpa
50	112.860740	26:c1:30:76:be:42	IntelCor_cb:21:27	ARP	42	192.168.128.141 is at 26:c1:30:76:be:42
51	112.905171	192.168.128.181	192.168.128.141	TCP	58	47765 → 22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
52	112.905220	192.168.128.181	192.168.128.141	TCP	58	47765 → 445 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
53	112.905242	192.168.128.181	192.168.128.141	TCP	58	47765 → 7 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
54	112.970146	192.168.128.141	192.168.128.181	TCP	54	445 → 47765 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
55	112.973711	192.168.128.141	192.168.128.181	TCP	54	7 → 47765 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
56	112.973711	192.168.128.141	192.168.128.181	TCP	54	22 → 47765 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Figure 5: The trace of TCP packets sent by nmap as privileged user

```
(base) snadri@snadri-Lenovo-IdeaPad-Z510:~/Desktop$ sudo nmap 192.168.128.141 -p 7,22,445
[sudo] password for snadri:
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-20 20:49 CET
Nmap scan report for 192.168.128.141
Host is up (0.056s latency).

PORT    STATE  SERVICE
7/tcp   closed echo
22/tcp  closed ssh
445/tcp closed microsoft-ds
MAC Address: 26:C1:30:76:BE:42 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.56 seconds
```

Figure 6: Printed Result showing on Terminal

2.4 TCP Scan Ports (50 - 150)

In the next step, nmap scans all TCP ports from 50 to 150 on the tablet. `sudo nmap 192.168.128.141 -p 50-150` It is showing that all ports on tablet respond with tcp flag RST and nmap recognizes

all ports are as `Closed` on the target. The test takes roughly 2 seconds.

2.5 UDP Scan Ports (50 - 150)

UDP scan is a bit different with TCP since UDP doesn't have handshake mechanism and packet losses are often higher. As the attacker run `sudo nmap 192.168.128.141 -p 50-150 -sU`, UDP start sending packets toward the ports 50-150. Depending on the UDP service on the Target, some requests may have empty payload and some filled with payload (Each service describe which possible payload needs to respond to a UDP request). When doing a UDP scan, four possible result could be observed:

- The target port is open; when the service running on the target respond to the request, nmap consider it as`open`
- Port is whether open or filtered `open|filtered`; some services drop UDP requests without payload. While also target firewall can drop in-going UDP requests and therefore it could be very difficult for the nmap to distinguish if the target port is open. Although nmap is developing its database with necessary payloads for UDP services, some UDP services might not be offered.
- Target return ICMP destination unreachable error (code 3) which is a standard when UDP port is `closed` (Figure 8) .
- Target firewall doesn't permit to penetrate and nmap returns the port scan result as *filtered*

Observing the results of the test on TCP and UDP ports showing (Figure 7 that all ports were closed.

```
(base) snadri@snadri-Lenovo-IdeaPad-Z510:~/Desktop$ sudo nmap 192.168.128.141 -p 50-150
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-20 20:51 CET
Nmap scan report for 192.168.128.141
Host is up (0.056s latency).
All 101 scanned ports on 192.168.128.141 are filtered (54) or closed (47)
MAC Address: 26:C1:30:76:BE:42 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.97 seconds
(base) snadri@snadri-Lenovo-IdeaPad-Z510:~/Desktop$ sudo nmap 192.168.128.141 -p 50-150 -sU
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-20 20:51 CET
Nmap scan report for 192.168.128.141
Host is up (0.090s latency).
All 101 scanned ports on 192.168.128.141 are closed
MAC Address: 26:C1:30:76:BE:42 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 104.71 seconds
```

Figure 7: Scanning ports from 50-150 on both TCP and UDP

	Time	Source	Destination	Protocol	Length	Info
521	255.371983	192.168.128.181	192.168.128.141	UDP	42	35810 → 95 Len=0
522	255.402716	192.168.128.141	192.168.128.181	ICMP	70	Destination unreachable (Port unreachable)
523	256.172837	192.168.128.181	192.168.128.141	UDP	42	35810 → 126 Len=0
524	256.973157	192.168.128.181	192.168.128.141	UDP	42	35824 → 135 Len=0
525	257.041947	192.168.128.141	192.168.128.181	ICMP	70	Destination unreachable (Port unreachable)
526	257.773984	192.168.128.181	192.168.128.141	UDP	42	35811 → 126 Len=0
527	257.866808	192.168.128.141	192.168.128.181	ICMP	70	Destination unreachable (Port unreachable)
528	258.574781	192.168.128.181	192.168.128.141	UDP	42	35810 → 64 Len=0
529	258.681516	192.168.128.141	192.168.128.181	ICMP	70	Destination unreachable (Port unreachable)

Figure 8: UDP scan; When the target port is `closed`, ICMP error request is sent by the target

3 Results and Analysis

The result of TCP test and UDP shows that TCP retransmit a request at most once for those ports which didn't respond at all. While UDP should retransmit for some ports up to 7,8 or even 10 times to make sure about them. This causes TCP scan much faster than UDP. The UDP test took roughly 105 seconds while the TCP lasts only 2 seconds. Another reason that could keep UDP slower is that ICMP response rate could be limited by Felix service framework Linux based systems.

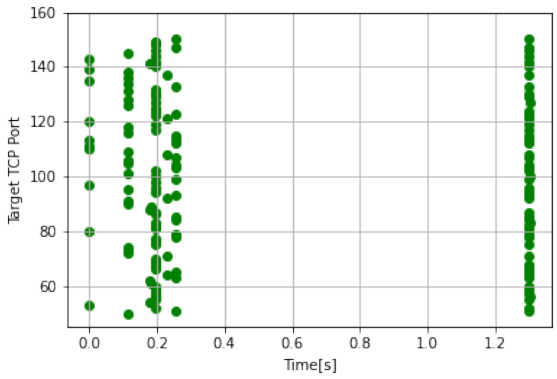

Figure 9: Scanning ports from 50-150 on TCP

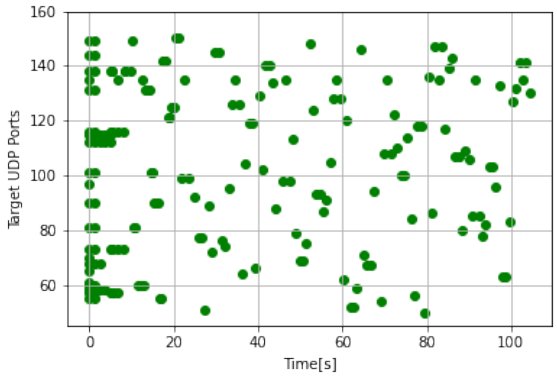

Figure 10: Scanning ports from 50-150 on UDP

The scripts used in Wireshark to filter the UDP and TCP test separately:

```
(ip.src==192.168.128.141 &&
ip.dst==192.168.128.181) ||
(ip.src==192.168.128.181 &&
ip.dst==192.168.128.141)
ip.src==192.168.128.181 &&
ip.dst==192.168.128.141
frame.time <= "Jan 20, 2022 20:53:22.9"
for TCP:
```

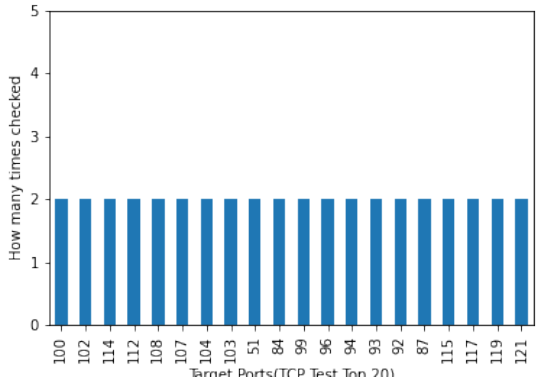

Figure 11: How many times TCP sent o each target port

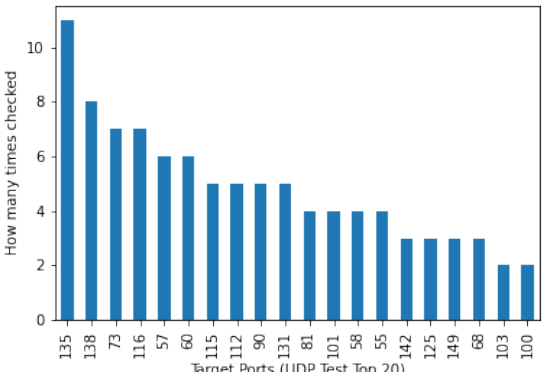

Figure 12: How many times UDP sent o each target port

```
frame.time <= "Jan 20, 2022 20:51:10.9" &&
frame.time >= "Jan 20, 2022 20:50:00"
for UDP:
frame.time <= "Jan 20, 2022 20:53:22.9" &&
frame.time >= "Jan 20, 2022 20:51:10.9"
```

The Scripts used in terminal to extract the time and port values:

TCP:

```
cat tcp.csv |cut -d '"' -f 4 >tcp_time.csv
cat tcp.csv |cut -d '"' -f 14 |cut -d '>' -f 2  |cut -d '[' -f 1
>tcp_port.csv
paste tcp_time.csv tcp_port.csv
> A7_Target_TCP.csv
```

UDP:

```
cat udp.csv |cut -d '"' -f 4 >udp_time.csv
cat udp.csv |cut -d '"' -f 14 |cut -d '>' -f 2  |cut -d '[' -f 1
>udp_port.csv
paste udp_time.csv udp_port.csv
```

> A7_Target_UDP.csv

The Scripts that are used in pandas to plot:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
pd.set_option("display.max_columns",180)
pd.set_option("display.max_rows",180)
df_a7_tcp=pd.read_csv('A7_Target_TCP.csv')
df_a7_udp=pd.read_csv('A7_Target_UDP.csv')

a7_tcp_iteration=df_a7_tcp.groupby(['Source Port']).size()
a7_tcp_iteration.sort_values(ascending=False).head(20).plot(kind='bar')
plt.xlabel("Target Ports(TCP Test,Top 20)")
plt.ylabel("How many times checked")
plt.ylim(top=5)
plt.savefig('a7_tcp_iteration.png')

plt.scatter(df_a7_udp['Time']-df_a7_udp.iloc[0,0],
df_a7_udp['Source Port'],c='g')
plt.ylabel("Target UDP Ports")
plt.ylim(top=160)
plt.xlabel("Time[s]")
plt.grid()
plt.savefig('df_a7_udp.png')

plt.scatter(df_a7_tcp['Time']-df_a7_tcp.iloc[0,0],
df_a7_tcp['Source Port'],c='g')
plt.ylabel("Target TCP Port")
#plt.yscale("log")
plt.xlabel("Time[s]")
plt.ylim(top=160)
plt.grid()
plt.savefig('df_a7_tcp.png')

a7_udp_iteration=df_a7_udp.groupby(['Source Port']).size()
a7_udp_iteration.sort_values(ascending=False).head(20).plot(kind='bar')
plt.xlabel("Target Ports (UDP Test,Top 20)")
plt.ylabel("How many times checked")
plt.savefig('a7_udp_iteration.png')
```