

Kadejha Jones
Shawn Ching
Shane McDonough

Lab 12

Quick Qanda

1. What is meant by a *half-open* scan?
 - a. A scan is half open if the client does not ack the servers response.
2. What is the significance of a port number?
 - a. Some port numbers typically used by specific protocols. For example ssh usually uses port 22. Http uses port 80, and https uses 443.
3. Is there anything stopping you from running SSH on something other than port 22?
 - a. No smiley face
4. How could a firewall easily detect a port scan?
 - a. If someone tries port 1, then port 2, then port 3, etc. in sequential order the firewall could see that and then add them to the deny list.
5. In what ways could an attacker make a port scan stealthier?
 - a. The attacker could scan the ports in a random order.

Local Enumeration

It is very important to be aware of what services are running on your servers. Since there is a local firewall running, you may also want to scan from an external host to see which ports are "open" to the public.

- A. Install `nmap` (network mapper) on `node1` and `main`
- B. What ports are open locally on `node1`?
 - a. `sudo nmap -sS -p1-4096 127.0.0.1`
- C. Does this match the output from the `ss` command?

- a. Almost all of the ports match but we can see an extra port in ss. We see 53 in ss but not in nmap.
- D. Using `ss`, are there any UDP sockets listening?
- a. There are no listening UDP sockets, they are all “UNCONN”.

Non-local Enumeration

When you run a port scan, you are attempting to connect to the host. Many firewall appliances can detect a port scan, can you?

1. On `node1`, open Wireshark and keep an eye out for someone scanning you.
 1. Can you tell that your lab partner is port scanning this machine?
 - i. yes
 2. Is there an order to which ports are checked by `nmap`?
 - i. It looks pretty random
 3. On ports that were open, did nmap complete the three-way handshake?
 - i. No.
2. Meanwhile, from `main`, scan TCP ports 1-4096 on `node1`
 1. What ports are open?
 - i. 22, 80, 443
 2. Are the results different than the scan results in *Local Enumeration*?
 - i. Yea. We found less ports.
3. nmap claims to have some super sneaky scan methods, look through `nmap --help` and try out various options
 1. What options did you try?
 - i. -S to spoof the source address
 - ii. --badsum to include a bogus “checksum”
 2. What shows up in Wireshark for each one?
 - i. When we spoofed, it showed the address we spoofed started port scanning instead of our own address. Professor Scrivnor port scanned us :(.

tcp && ip.addr == 10.100.100.141							Expression...	+
No.	Time	Source	Destination	Protocol	Length	Info		
1	0.000000000	10.100.100.100	10.100.100.141	TCP	58	58928 → 3 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
2	0.004924274	10.100.100.100	10.100.100.141	TCP	58	58927 → 2585 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
3	0.004971810	10.100.100.100	10.100.100.141	TCP	58	58927 → 163 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
4	0.007360554	10.100.100.100	10.100.100.141	TCP	58	58928 → 2874 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
5	0.007411533	10.100.100.100	10.100.100.141	TCP	58	58928 → 1359 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
6	0.007432920	10.100.100.100	10.100.100.141	TCP	58	58928 → 948 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
7	0.007451870	10.100.100.100	10.100.100.141	TCP	58	58928 → 2519 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
8	0.007469920	10.100.100.100	10.100.100.141	TCP	58	58928 → 157 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
9	0.007490102	10.100.100.100	10.100.100.141	TCP	58	58928 → 3855 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
10	0.009881250	10.100.100.100	10.100.100.141	TCP	58	58927 → 3482 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
11	0.100131979	10.100.100.100	10.100.100.141	TCP	58	58926 → 3817 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
12	0.105090995	10.100.100.100	10.100.100.141	TCP	58	58928 → 2585 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
13	0.107402308	10.100.100.100	10.100.100.141	TCP	58	58928 → 163 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
14	0.107438722	10.100.100.100	10.100.100.141	TCP	58	58926 → 2067 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
15	0.109746608	10.100.100.100	10.100.100.141	TCP	58	58926 → 261 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
16	0.109777066	10.100.100.100	10.100.100.141	TCP	58	58926 → 1004 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
17	0.109796683	10.100.100.100	10.100.100.141	TCP	58	58926 → 1101 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
18	0.109814311	10.100.100.100	10.100.100.141	TCP	58	58926 → 3840 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
19	0.109831175	10.100.100.100	10.100.100.141	TCP	58	58926 → 2952 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
20	0.112119138	10.100.100.100	10.100.100.141	TCP	58	58928 → 3482 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
21	0.200286028	10.100.100.100	10.100.100.141	TCP	58	58927 → 3817 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
22	0.205226288	10.100.100.100	10.100.100.141	TCP	58	58926 → 2421 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
23	0.207562090	10.100.100.100	10.100.100.141	TCP	58	58926 → 616 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
24	0.209915819	10.100.100.100	10.100.100.141	TCP	58	58927 → 261 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
25	0.209946159	10.100.100.100	10.100.100.141	TCP	58	58927 → 2067 [SYN] Seq=0 Win=1024 Len=0 MSS=146		

tcp && ip.addr == 10.100.100.141							Expression...	+
No.	Time	Source	Destination	Protocol	Length	Info		
41	19.920568149	10.100.100.1	10.100.100.141	TCP	58	58091 → 110 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
42	19.920598420	10.100.100.1	10.100.100.141	TCP	58	58091 → 587 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
43	19.920615559	10.100.100.1	10.100.100.141	TCP	58	58091 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
44	19.920632768	10.100.100.1	10.100.100.141	TCP	58	58091 → 139 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
45	19.920649633	10.100.100.1	10.100.100.141	TCP	58	58091 → 993 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
46	19.920665788	10.100.100.1	10.100.100.141	TCP	58	58091 → 256 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
47	19.920682473	10.100.100.1	10.100.100.141	TCP	58	58091 → 995 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
48	19.920699374	10.100.100.1	10.100.100.141	TCP	58	58091 → 443 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
49	19.920716307	10.100.100.1	10.100.100.141	TCP	58	58091 → 143 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
50	19.920733276	10.100.100.1	10.100.100.141	TCP	58	58091 → 111 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
52	21.0216727930	10.100.100.1	10.100.100.141	TCP	58	58092 → 111 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
53	21.021675778	10.100.100.1	10.100.100.141	TCP	58	58092 → 143 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
54	21.021699803	10.100.100.1	10.100.100.141	TCP	58	58092 → 443 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
55	21.021722855	10.100.100.1	10.100.100.141	TCP	58	58092 → 995 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
56	21.021745314	10.100.100.1	10.100.100.141	TCP	58	58092 → 256 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
57	21.021767516	10.100.100.1	10.100.100.141	TCP	58	58092 → 993 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
58	21.021789520	10.100.100.1	10.100.100.141	TCP	58	58092 → 139 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
59	21.021811662	10.100.100.1	10.100.100.141	TCP	58	58092 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
60	21.021833369	10.100.100.1	10.100.100.141	TCP	58	58092 → 587 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
61	21.021858392	10.100.100.1	10.100.100.141	TCP	58	58092 → 110 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
62	21.121687865	10.100.100.1	10.100.100.141	TCP	58	58091 → 53 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
63	21.124109406	10.100.100.1	10.100.100.141	TCP	58	58091 → 199 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
64	21.124154348	10.100.100.1	10.100.100.141	TCP	58	58091 → 3389 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
65	21.124176924	10.100.100.1	10.100.100.141	TCP	58	58091 → 1025 [SYN] Seq=0 Win=1024 Len=0 MSS=146		
66	21.124200198	10.100.100.1	10.100.100.141	TCP	58	58091 → 445 [SYN] Seq=0 Win=1024 Len=0 MSS=146		

- ii. The checksums of the messages sent did not add up correctly.