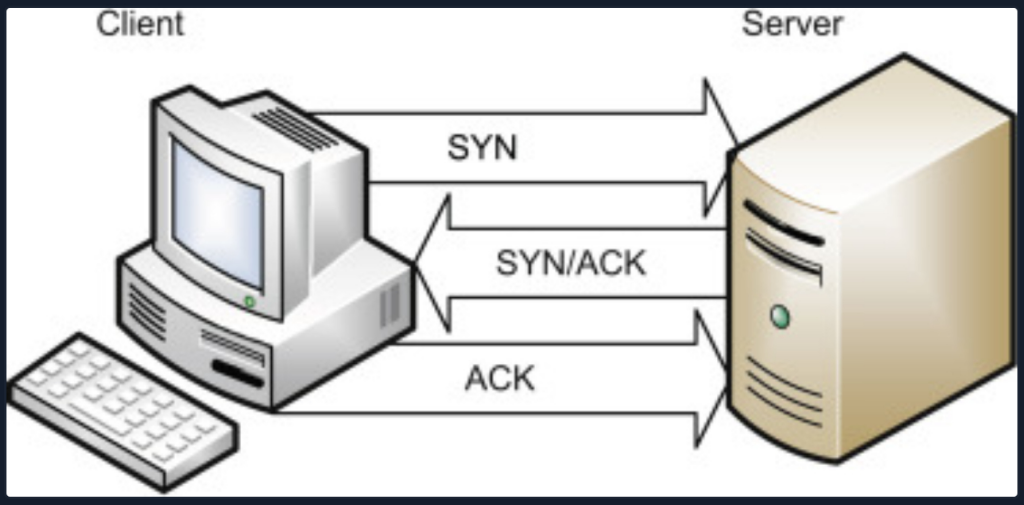
TCP Handshake Abnormalities

Innately, when attackers are gaining information on our TCP services, we might notice a few odd behaviors during our traffic analysis efforts. Firstly, let's consider how normal TCP connections work with their 3-way handshake.



To initiate a TCP connection for whatever purpose the client first sends the machine it is attempting to connect to a TCP SYN request to begin the TCP connection.

If this port is open, and in fact able to be connected to, the machine responds with a TCP SYN/ACK to acknowledge that the connection is valid and able to be used. However, we should consider all TCP flags.

Flags	Description
`URG (Urgent)`	This flag is to denote urgency with the current data in stream.
`ACK (Acknowledgement)`	This flag acknowledges receipt of data.
`PSH (Push)`	This flag instructs the TCP stack to immediately deliver the received data to the application layer, and bypass buffering.
`RST (Reset)`	This flag is used for termination of the TCP connection (we will dive into hijacking and RST attacks soon).
`SYN (Synchronize)`	This flag is used to establish an initial connection with TCP.
`FIN (Finish)`	This flag is used to denote the finish of a TCP connection. It is used when no more data needs to be sent.
`ECN (Explicit Congestion Notification)`	This flag is used to denote congestion within our network, it is to let the hosts know to avoid unnecessary re-transmissions.

As such, when we are performing our traffic analysis efforts we can look for the following strange conditions:

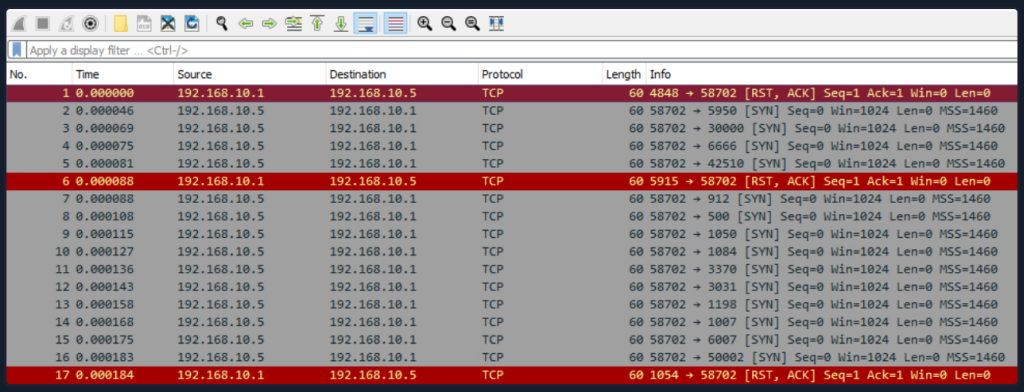
1. `Too many flags of a kind or kinds` - This could show us that scanning is occurring within our network.
2. `The usage of different and unusual flags` - Sometimes this could indicate a TCP RST attack, hijacking, or simply some form of control evasion for scanning.
3. `Solo host to multiple ports, or solo host to multiple hosts` - Easy enough, we can find scanning as we have done before by noticing where these connections are going from one host. In a lot of cases, we may even need to consider decoy scans and random source attacks.

Excessive SYN Flags

Related PCAP File(s):

- `nmap_syn_scan.pcapng`

Right away one of the traffic patterns that we can notice is too many SYN flags. This is a prime example of nmap scanning. Simply put, the adversary will send TCP SYN packets to the target ports. In the case where our port is open, our machine will respond with a SYN-ACK packet to continue the handshake, which will then be met by an RST from the attackers scanner. However, we can get lost in the RSTs here as our machine will respond with RST for closed ports.



Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.10.1	192.168.10.5	TCP	60	4848 → 58702 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
2	0.000046	192.168.10.5	192.168.10.1	TCP	60	58702 → 5950 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
3	0.000069	192.168.10.5	192.168.10.1	TCP	60	58702 → 30000 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
4	0.000075	192.168.10.5	192.168.10.1	TCP	60	58702 → 6666 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
5	0.000081	192.168.10.5	192.168.10.1	TCP	60	58702 → 42510 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
6	0.000088	192.168.10.1	192.168.10.5	TCP	60	5915 → 58702 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
7	0.000088	192.168.10.5	192.168.10.1	TCP	60	58702 → 912 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
8	0.000108	192.168.10.5	192.168.10.1	TCP	60	58702 → 500 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
9	0.000115	192.168.10.5	192.168.10.1	TCP	60	58702 → 1050 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
10	0.000127	192.168.10.5	192.168.10.1	TCP	60	58702 → 1084 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
11	0.000136	192.168.10.5	192.168.10.1	TCP	60	58702 → 3370 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
12	0.000143	192.168.10.5	192.168.10.1	TCP	60	58702 → 3031 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
13	0.000158	192.168.10.5	192.168.10.1	TCP	60	58702 → 1198 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
14	0.000168	192.168.10.5	192.168.10.1	TCP	60	58702 → 1007 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
15	0.000175	192.168.10.5	192.168.10.1	TCP	60	58702 → 6007 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
16	0.000183	192.168.10.5	192.168.10.1	TCP	60	58702 → 50002 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
17	0.000184	192.168.10.1	192.168.10.5	TCP	60	1054 → 58702 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

However it is worth noting that there are two primary scan types we might detect that use the SYN flag. These are:

1. `SYN Scans` - In these scans the behavior will be as we see, however the attacker will pre-emptively end the handshake with the RST flag.
2. `SYN Stealth Scans` - In this case the attacker will attempt to evade detection by only partially completing the TCP handshake.

No Flags

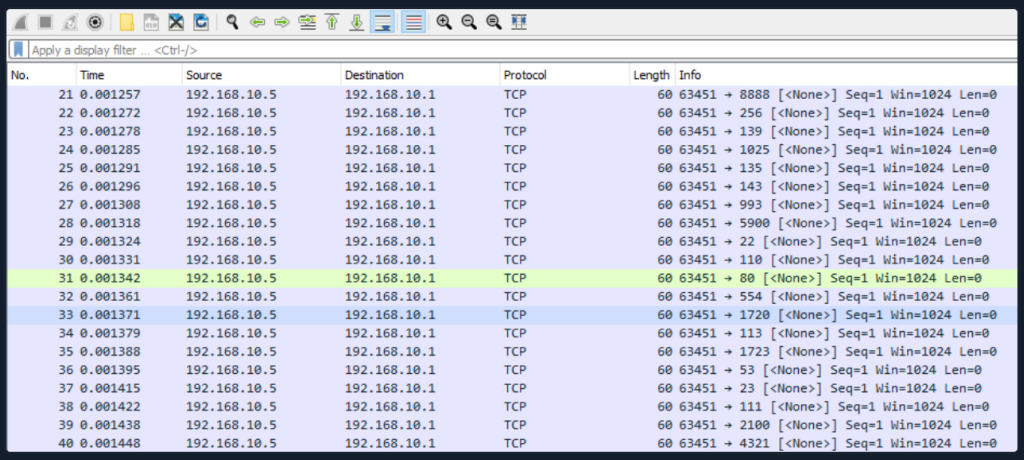
Related PCAP File(s):

- `nmap_null_scan.pcapng`

On the opposite side of things, the attacker might send no flags. This is what is commonly referrred to as a NULL scan. In a NULL scan an attacker sends TCP packets with no flags. TCP connections behave like the following when a NULL packet is received.

1. `If the port is open` - The system will not respond at all since there is no flags.
2. `If the port is closed` - The system will respond with an RST packet.

As such a NULL scan might look like the following.



Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
21	0.001257	192.168.10.5	192.168.10.1	TCP	60	63451 → 8888 [<None>] Seq=1 Win=1024 Len=0
22	0.001272	192.168.10.5	192.168.10.1	TCP	60	63451 → 256 [<None>] Seq=1 Win=1024 Len=0
23	0.001278	192.168.10.5	192.168.10.1	TCP	60	63451 → 139 [<None>] Seq=1 Win=1024 Len=0
24	0.001285	192.168.10.5	192.168.10.1	TCP	60	63451 → 1025 [<None>] Seq=1 Win=1024 Len=0
25	0.001291	192.168.10.5	192.168.10.1	TCP	60	63451 → 135 [<None>] Seq=1 Win=1024 Len=0
26	0.001296	192.168.10.5	192.168.10.1	TCP	60	63451 → 143 [<None>] Seq=1 Win=1024 Len=0
27	0.001308	192.168.10.5	192.168.10.1	TCP	60	63451 → 993 [<None>] Seq=1 Win=1024 Len=0
28	0.001318	192.168.10.5	192.168.10.1	TCP	60	63451 → 5900 [<None>] Seq=1 Win=1024 Len=0
29	0.001324	192.168.10.5	192.168.10.1	TCP	60	63451 → 22 [<None>] Seq=1 Win=1024 Len=0
30	0.001331	192.168.10.5	192.168.10.1	TCP	60	63451 → 110 [<None>] Seq=1 Win=1024 Len=0
31	0.001342	192.168.10.5	192.168.10.1	TCP	60	63451 → 80 [<None>] Seq=1 Win=1024 Len=0
32	0.001361	192.168.10.5	192.168.10.1	TCP	60	63451 → 554 [<None>] Seq=1 Win=1024 Len=0
33	0.001371	192.168.10.5	192.168.10.1	TCP	60	63451 → 1720 [<None>] Seq=1 Win=1024 Len=0
34	0.001379	192.168.10.5	192.168.10.1	TCP	60	63451 → 113 [<None>] Seq=1 Win=1024 Len=0
35	0.001388	192.168.10.5	192.168.10.1	TCP	60	63451 → 1723 [<None>] Seq=1 Win=1024 Len=0
36	0.001395	192.168.10.5	192.168.10.1	TCP	60	63451 → 53 [<None>] Seq=1 Win=1024 Len=0
37	0.001415	192.168.10.5	192.168.10.1	TCP	60	63451 → 23 [<None>] Seq=1 Win=1024 Len=0
38	0.001422	192.168.10.5	192.168.10.1	TCP	60	63451 → 111 [<None>] Seq=1 Win=1024 Len=0
39	0.001438	192.168.10.5	192.168.10.1	TCP	60	63451 → 2100 [<None>] Seq=1 Win=1024 Len=0
40	0.001448	192.168.10.5	192.168.10.1	TCP	60	63451 → 4321 [<None>] Seq=1 Win=1024 Len=0

Too Many ACKs

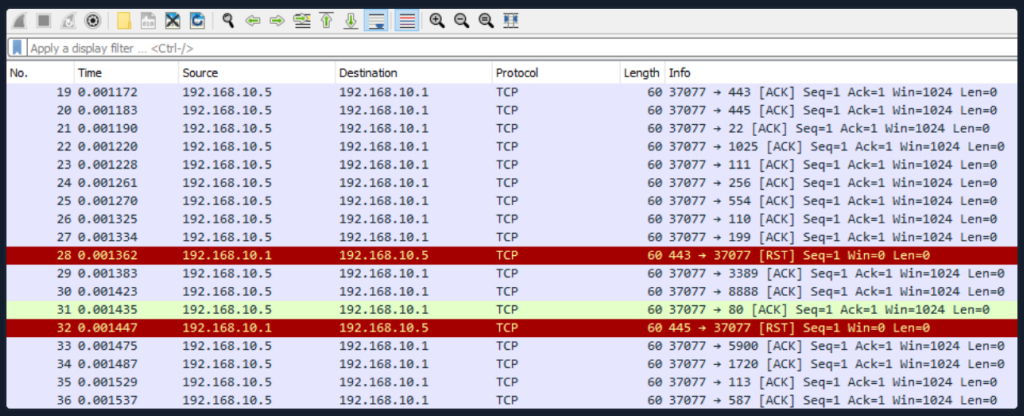
Related PCAP File(s):

- `nmap_ack_scan.pcapng`

On the other hand, we might notice an excessive amount of acknowledgements between two hosts. In this case the attacker might be employing the usage of an ACK scan. In the case of an ACK scan TCP connections will behave like the following.

1. `If the port is open` - The affected machine will either not respond, or will respond with an RST packet.
2. `If the port is closed` - The affected machine will respond with an RST packet.

So, we might see the following traffic which would indicate an ACK scan.



Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
19	0.001172	192.168.10.5	192.168.10.1	TCP	60	37077 → 443 [ACK] Seq=1 Ack=1 Win=1024 Len=0
20	0.001183	192.168.10.5	192.168.10.1	TCP	60	37077 → 445 [ACK] Seq=1 Ack=1 Win=1024 Len=0
21	0.001190	192.168.10.5	192.168.10.1	TCP	60	37077 → 22 [ACK] Seq=1 Ack=1 Win=1024 Len=0
22	0.001220	192.168.10.5	192.168.10.1	TCP	60	37077 → 1025 [ACK] Seq=1 Ack=1 Win=1024 Len=0
23	0.001228	192.168.10.5	192.168.10.1	TCP	60	37077 → 111 [ACK] Seq=1 Ack=1 Win=1024 Len=0
24	0.001261	192.168.10.5	192.168.10.1	TCP	60	37077 → 256 [ACK] Seq=1 Ack=1 Win=1024 Len=0
25	0.001270	192.168.10.5	192.168.10.1	TCP	60	37077 → 554 [ACK] Seq=1 Ack=1 Win=1024 Len=0
26	0.001325	192.168.10.5	192.168.10.1	TCP	60	37077 → 110 [ACK] Seq=1 Ack=1 Win=1024 Len=0
27	0.001334	192.168.10.5	192.168.10.1	TCP	60	37077 → 199 [ACK] Seq=1 Ack=1 Win=1024 Len=0
28	0.001362	192.168.10.1	192.168.10.5	TCP	60	443 → 37077 [RST] Seq=1 Win=0 Len=0
29	0.001383	192.168.10.5	192.168.10.1	TCP	60	37077 → 3389 [ACK] Seq=1 Ack=1 Win=1024 Len=0
30	0.001423	192.168.10.5	192.168.10.1	TCP	60	37077 → 8888 [ACK] Seq=1 Ack=1 Win=1024 Len=0
31	0.001435	192.168.10.5	192.168.10.1	TCP	60	37077 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
32	0.001447	192.168.10.1	192.168.10.5	TCP	60	445 → 37077 [RST] Seq=1 Win=0 Len=0
33	0.001475	192.168.10.5	192.168.10.1	TCP	60	37077 → 5900 [ACK] Seq=1 Ack=1 Win=1024 Len=0
34	0.001487	192.168.10.5	192.168.10.1	TCP	60	37077 → 1720 [ACK] Seq=1 Ack=1 Win=1024 Len=0
35	0.001529	192.168.10.5	192.168.10.1	TCP	60	37077 → 113 [ACK] Seq=1 Ack=1 Win=1024 Len=0
36	0.001537	192.168.10.5	192.168.10.1	TCP	60	37077 → 587 [ACK] Seq=1 Ack=1 Win=1024 Len=0

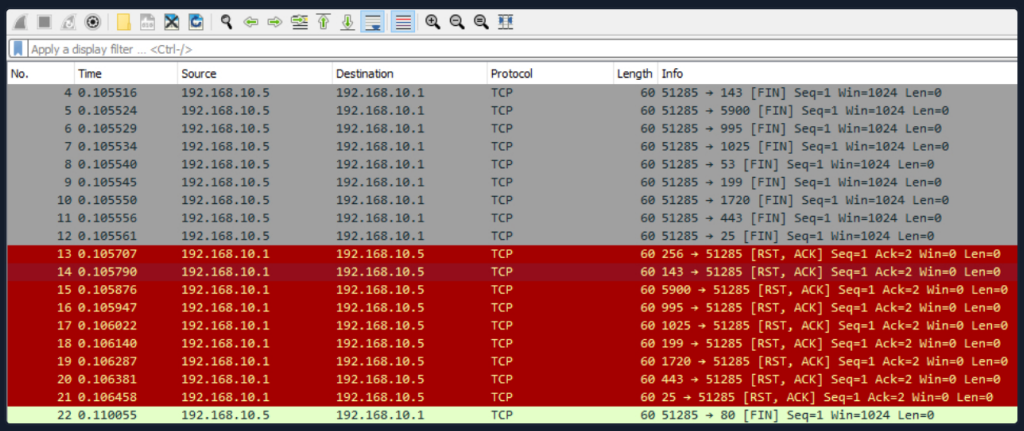
Excessive FINs

Related PCAP File(s):

- `nmap_fin_scan.pcapng`

Using another part of the handshake, an attacker might utilize a FIN scan. In this case, all TCP packets will be marked with the FIN flag. We might notice the following behavior from our affected machine.

1. `If the port is open` - Our affected machine simply will not respond.
2. `If the port is closed` - Our affected machine will respond with an RST packet.



Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
4	0.105516	192.168.10.5	192.168.10.1	TCP	60	51285 → 143 [FIN] Seq=1 Win=1024 Len=0
5	0.105524	192.168.10.5	192.168.10.1	TCP	60	51285 → 5900 [FIN] Seq=1 Win=1024 Len=0
6	0.105529	192.168.10.5	192.168.10.1	TCP	60	51285 → 995 [FIN] Seq=1 Win=1024 Len=0
7	0.105534	192.168.10.5	192.168.10.1	TCP	60	51285 → 1025 [FIN] Seq=1 Win=1024 Len=0
8	0.105540	192.168.10.5	192.168.10.1	TCP	60	51285 → 53 [FIN] Seq=1 Win=1024 Len=0
9	0.105545	192.168.10.5	192.168.10.1	TCP	60	51285 → 199 [FIN] Seq=1 Win=1024 Len=0
10	0.105550	192.168.10.5	192.168.10.1	TCP	60	51285 → 1720 [FIN] Seq=1 Win=1024 Len=0
11	0.105556	192.168.10.5	192.168.10.1	TCP	60	51285 → 443 [FIN] Seq=1 Win=1024 Len=0
12	0.105561	192.168.10.5	192.168.10.1	TCP	60	51285 → 25 [FIN] Seq=1 Win=1024 Len=0
13	0.105707	192.168.10.1	192.168.10.5	TCP	60	256 → 51285 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
14	0.105790	192.168.10.1	192.168.10.5	TCP	60	143 → 51285 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
15	0.105876	192.168.10.1	192.168.10.5	TCP	60	5900 → 51285 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
16	0.105947	192.168.10.1	192.168.10.5	TCP	60	995 → 51285 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
17	0.106022	192.168.10.1	192.168.10.5	TCP	60	1025 → 51285 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
18	0.106140	192.168.10.1	192.168.10.5	TCP	60	199 → 51285 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
19	0.106287	192.168.10.1	192.168.10.5	TCP	60	1720 → 51285 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
20	0.106381	192.168.10.1	192.168.10.5	TCP	60	443 → 51285 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
21	0.106458	192.168.10.1	192.168.10.5	TCP	60	25 → 51285 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
22	0.110055	192.168.10.5	192.168.10.1	TCP	60	51285 → 80 [FIN] Seq=1 Win=1024 Len=0

Just too many flags

Related PCAP File(s):

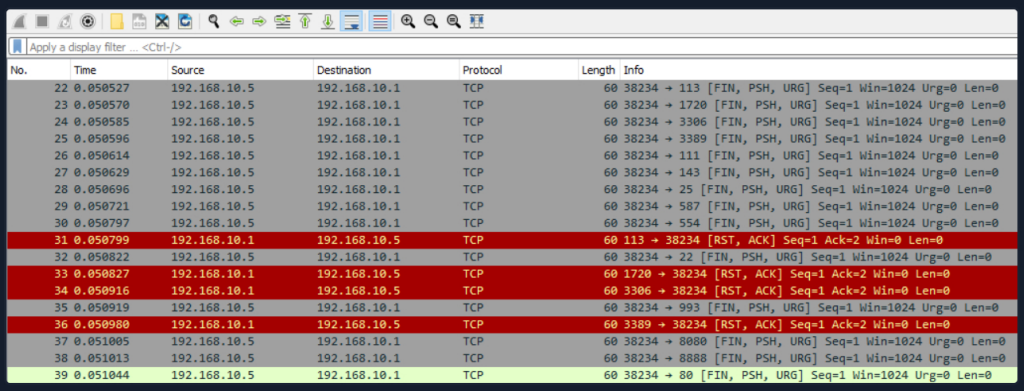
- `nmap_xmas_scan.pcapng`

Let's say the attacker just wanted to throw spaghetti at the wall. In that case, they might utilize a Xmas tree scan, which is when they put all TCP flags on their transmissions. Similarly, our affected host might respond like the following when all flags are set.

1. `If the port is open` - The affected machine will not respond, or at least it will with an RST packet.

2. `If the port is closed` - The affected machine will respond with an RST packet.

Xmas tree scans are pretty easy to spot and look like the following.



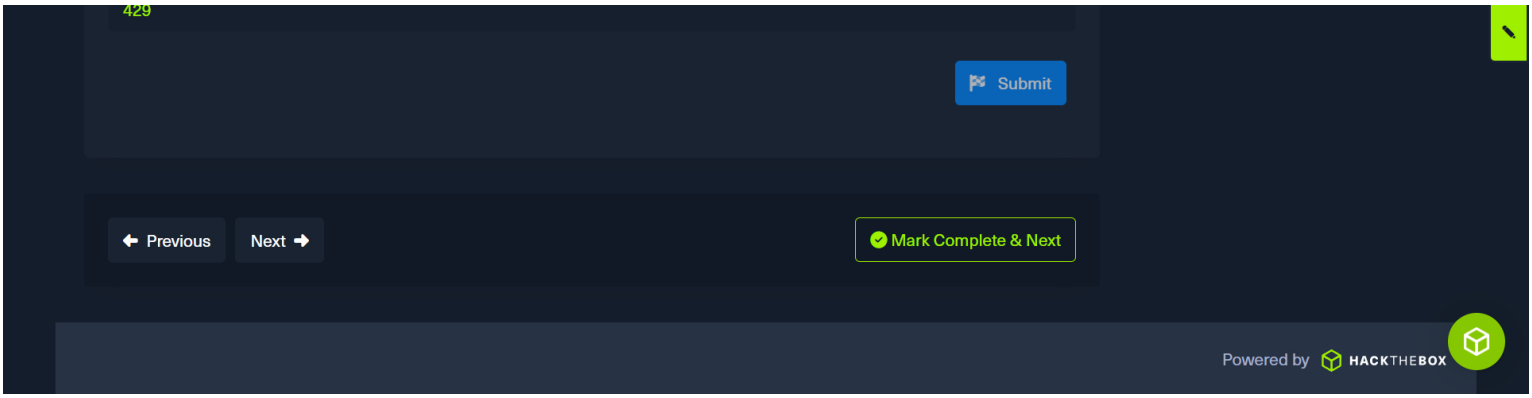
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
22	0.050527	192.168.10.5	192.168.10.1	TCP	60	38234 → 113 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
23	0.050570	192.168.10.5	192.168.10.1	TCP	60	38234 → 1720 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
24	0.050585	192.168.10.5	192.168.10.1	TCP	60	38234 → 3306 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
25	0.050596	192.168.10.5	192.168.10.1	TCP	60	38234 → 3389 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
26	0.050614	192.168.10.5	192.168.10.1	TCP	60	38234 → 111 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
27	0.050629	192.168.10.5	192.168.10.1	TCP	60	38234 → 143 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
28	0.050696	192.168.10.5	192.168.10.1	TCP	60	38234 → 25 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
29	0.050721	192.168.10.5	192.168.10.1	TCP	60	38234 → 587 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
30	0.050797	192.168.10.5	192.168.10.1	TCP	60	38234 → 554 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
31	0.050799	192.168.10.1	192.168.10.5	TCP	60	113 → 38234 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
32	0.050822	192.168.10.5	192.168.10.1	TCP	60	38234 → 22 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
33	0.050827	192.168.10.1	192.168.10.5	TCP	60	1720 → 38234 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
34	0.050916	192.168.10.1	192.168.10.5	TCP	60	3306 → 38234 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
35	0.050919	192.168.10.5	192.168.10.1	TCP	60	38234 → 993 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
36	0.050980	192.168.10.1	192.168.10.5	TCP	60	3389 → 38234 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
37	0.051005	192.168.10.5	192.168.10.1	TCP	60	38234 → 8080 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
38	0.051013	192.168.10.5	192.168.10.1	TCP	60	38234 → 8888 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0
39	0.051044	192.168.10.5	192.168.10.1	TCP	60	38234 → 80 [FIN, PSH, URG] Seq=1 Win=1024 Urg=0 Len=0

Questions

Answer the question(s) below to complete this Section and earn cubes!

+1 Inspect the nmap_syn_scan.pcapng file, part of this module's resources, and enter the total count of packets that have the TCP ACK flag set as your answer.


429
Submit
Previous
Next
Mark Complete & Next
Powered by HACKTHEBOX