

Wireshark – SYN Scan – Packet Spoofing

Outcomes:

- Refresh Wireshark skills and knowledge of protocol communication, packet data/ transfer.
- Concepts behind packet spoofing
- Concepts behind SYN-SCAN and SYN FLOOD

Open your Win10 VM which we have used previously.

Install Wireshark (download 64-bit installer from <https://www.wireshark.org>)

1a. Wireshark sample

On our canvas page, download http.cap, then open it in Wireshark.

The wireshark display includes 3 main windows/sections, each with a different level of detail:

The screenshot shows the Wireshark 1.12.1 interface with the file http.cap loaded. The interface is divided into three main sections:

- Upper Window: Packet List**: Displays a list of captured packets. The first packet is highlighted, showing details like Time, Source, Destination, Protocol, Length, and Info.
- Middle Window: Packet Detail for specific packets**: Provides a detailed view of the selected packet, showing the protocol stack (Ethernet II, Internet Protocol Version 4, Transmission Control Protocol) and the raw data.
- Lower Window: Hex + binary raw data**: Displays the raw data of the selected packet in hexadecimal and binary format.

The status bar at the bottom indicates: File: "C:\Users\jhoag\Documents\n... Packets: 43 - Displayed: 43 (100.0%)... Profile: Default

1b. Exploring Packets

In the first packet, what is the source (browser, or web client) IP address? **145.254.160.237**

What is the destination (web server) IP address? **65.208.228.223**

What is the length? (value in length Column)? **62**

Click in the middle Packet Detail section of the Wireshark window:

- Notice the frame length matches the value you just recorded

In the top Packet List section, right-click on the first packet. Choose the option for Conversation filter (TCP). There are actually 2 TCP conversations in this stream. We want to concentrate on the first one.

Can you find the TCP 3-way handshake?

What packet numbers does it use? **1-3**

What packet does the HTTP protocol show up in? **4**

This is the start of the http conversation

After that, there is a series of TCP segments containing the web page data

Packet 38 is the end of the http conversation (HTTP 200 OK)

Packets 40-43 are the TCP FIN sequence to end the connection

What Web Server application is in use here? (Hint: Packet Details) **apache**

1c. Headers

Let's examine some of the other fields and data in the capture.

Each protocol represents a portion of the entire capture. Each protocol also has a header section that provides information regarding source + destination + what to do with the data. Selecting a particular packet allows you to examine data in the protocol's header fields, as well as the data sent.

In the first packet, click on the Ethernet II header.

What is the source MAC address? **_00:00:01:00:00:00_**

What is the destination Mac address? **_fe:ff:20:00:01:00_**

What is the length of this header (Side note: Look in the lower Raw Data window ... each pair of hex values in the lower frame is a byte)? **_62_**

Select the IP header

What is the TTL? **128**

What is the Total length? **48**

What is the Protocol field? (just the protocol, not the number) **TCP**

In the 4th packet (HTTP), click on the TCP header to expand it.

What TCP ports were used in this conversation? **3372, 80**

What is the size of the TCP header? **20**

How much data is sent? **479**

1d. Statistics: Now let's explore some statistics of the conversation.

Click on the Statistics tab at the top menu, and select Capture File Properties.

In the bottom of the window, capture statistics are presented regarding the conversation.

Under Capture File Properties, record the following statistics:

Average packets per second (pps) **1.4**

Average packet size (Bytes) **25091**

1e. Get a baseline of normal traffic.

Start a Wireshark [capture](#), and do normal internet activity for 1-2 minutes. Stop the capture.

Under Statistics/Capture File Properties, record the following statistics:

Packets	26612
Time spans	275.598
Average pps	96.6
Average packet size, B	205
Bytes	5456549
Average bytes/s	19k
Average bits/s	158k

Under Statistics/Packet Lengths, what are the 2 most common categories? **80-159, 160-319**

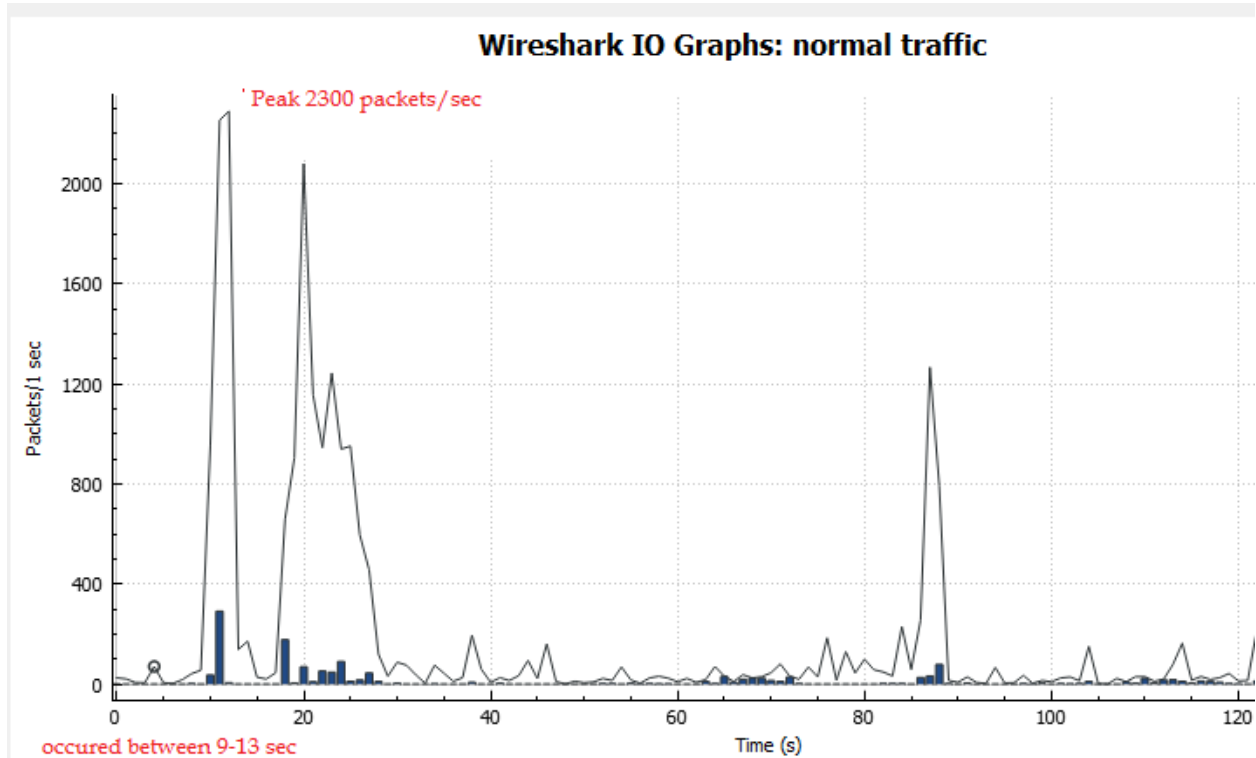
Look at Statistics/I-O graph.

What is the peak value (see example below)? ~200 pps

When in the capture did it occur? 193 seconds

How long did it last? Less than 3 seconds

In this example, you can see the peak occurred from about 9-13 secs and was 2,300 packets/sec.



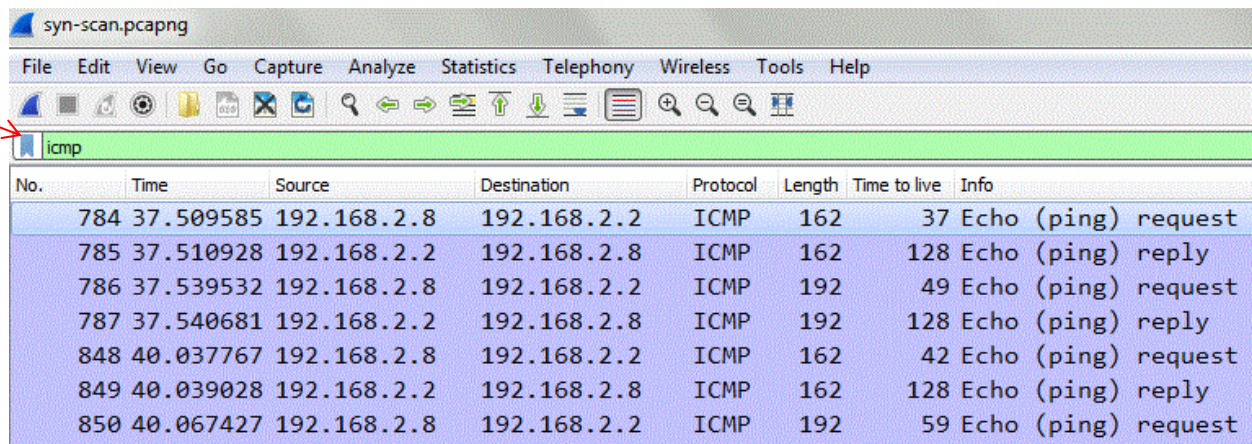
2. IP address spoof:

2a. Use Wireshark to capture a ping communication between your VM and your host.

Ping is an implementation of the Internet Control Message Protocol ([ICMP](#)).

In Wireshark, create a filter for ICMP & take a Snip.

The resulting data should look like this:



syn-scan.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

icmp

No.	Time	Source	Destination	Protocol	Length	Time to live	Info
784	37.509585	192.168.2.8	192.168.2.2	ICMP	162	37	Echo (ping) request
785	37.510928	192.168.2.2	192.168.2.8	ICMP	162	128	Echo (ping) reply
786	37.539532	192.168.2.8	192.168.2.2	ICMP	192	49	Echo (ping) request
787	37.540681	192.168.2.2	192.168.2.8	ICMP	192	128	Echo (ping) reply
848	40.037767	192.168.2.8	192.168.2.2	ICMP	162	42	Echo (ping) request
849	40.039028	192.168.2.2	192.168.2.8	ICMP	162	128	Echo (ping) reply
850	40.067427	192.168.2.8	192.168.2.2	ICMP	192	59	Echo (ping) request

You should have the 4 ping requests with a matching ping reply.

2b. Spoof

Now you are going to spoof the source IP address. What happens if you say the source is an IP different from your own?

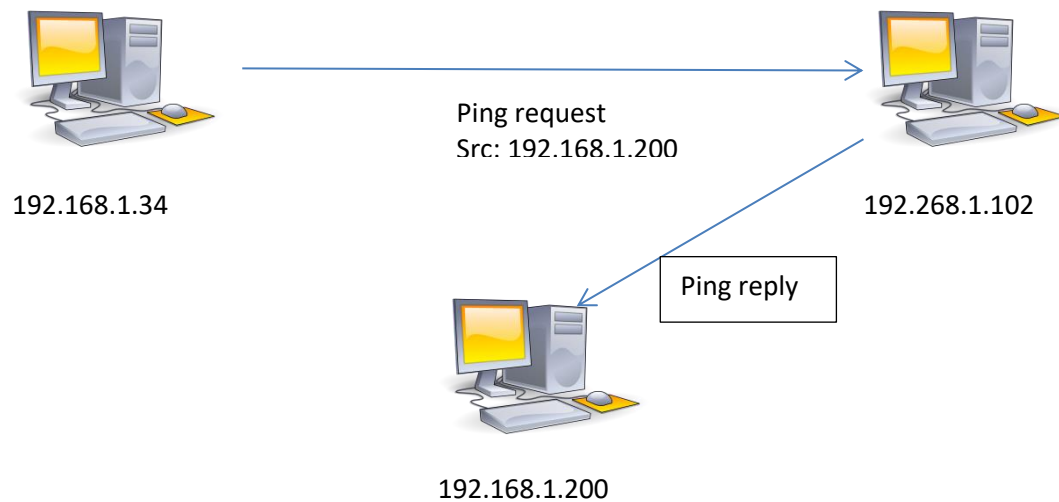
Download [nmap](#) on your VM, install it, then open a [CMD](#) line window and navigate to the nmap directory. (should be c:\Program Files (x86)\nmap)

[nping](#) is a utility that lets you modify the parameters of a ping request.

- Run `nping -h` to see all the options.
- Run `nping neighbor_IP` to see what normal output looks like.
- **Snip** your nping's normal output of your neighbor's IP.

Now, you are going to use another live "spoofed" station's IP address as the source IP in a ping to your neighbor. Your neighbor will be running Wireshark to capture the ping requests and responses.

Example:



Target: run Wireshark

Attacker: Run `nping -S spoofed_IP target_IP`

Target: stop Wireshark capture & then filter by icmp. This should show ICMP replies going to the spoofed system's IP.

The attacker's CMD window should show the ICMP requests, but no replies. Why? _____

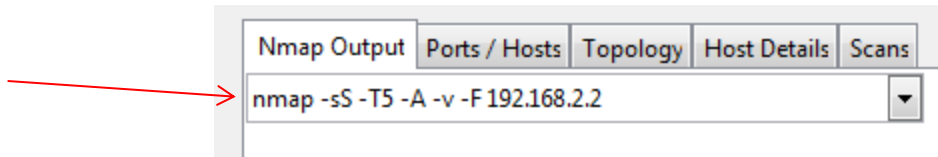
Attacker: Upload a screenshot of the CMD nping window.

Target: Snip the Wireshark capture filtered to show only ICMP.

3. SYN-SCAN/FLOOD

A SYN scan uses the 3-way handshake to see what ports are open. A SYN flood uses this same technique to overwhelm a device. A SYN packet is sent. If a port is open, it will send a SYN/ACK back. If the final ACK is never sent, the connection is not opened.

To do this, you'll use nmap. We used Zenmap previously which is the GUI version. You'll recall that Zenmap generated commands based on scan selections



It builds character to use command line, so today you will use nmap from CMD. And you are already in the directory in the CMD window.

- Type `nmap -h` to see what the possible options are.

Get ready for the scan/attack:

Target: start Wireshark capture

Attacker: run nmap from CMD → `nmap -sS -T4 -v target_IP`.

*The `-sS` = a SYN scan, `-T4` = speed of packets, `-v` = verbose output

Target: stop Wireshark

Analyze traffic. You should see a number of SYN requests.

SYN/ACK replies should correspond to the open ports listed in the nmap output.

(p.addr eq 192.168.2.8 and ip.addr eq 192.168.2.2)							
No.	Time	Source	Destination	Protocol	Length	Time to live	Info
440	25.405154	192.168.2.8	192.168.2.2	TCP	58	→	38 42656→h323hostcall(1720) [SYN] Seq=0 Win=1024 Len=0 MSS=1460
441	25.405662	192.168.2.8	192.168.2.2	TCP	58		58 42656→blackjack(1025) [SYN] Seq=0 Win=1024 Len=0 MSS=1460
442	25.451135	192.168.2.8	192.168.2.2	TCP	58		53 42655→smux(199) [SYN] Seq=0 Win=1024 Len=0 MSS=1460
443	25.451850	192.168.2.8	192.168.2.2	TCP	58	→	53 42655→netbios-ssn(139) [SYN] Seq=0 Win=1024 Len=0 MSS=1460
444	25.452421	192.168.2.8	192.168.2.2	TCP	58		56 42655→domain(53) [SYN] Seq=0 Win=1024 Len=0 MSS=1460
445	25.453043	192.168.2.8	192.168.2.2	TCP	58		41 42655→imaps(993) [SYN] Seq=0 Win=1024 Len=0 MSS=1460
446	25.453078	192.168.2.2	192.168.2.8	TCP	60	→	128 netbios-ssn(139)→42655 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
447	25.453119	192.168.2.8	192.168.2.2	TCP	54		128 42655→netbios-ssn(139) [RST] Seq=1 Win=0 Len=0
448	25.453635	192.168.2.8	192.168.2.2	TCP	58		52 42655→http-alt(8080) [SYN] Seq=0 Win=1024 Len=0 MSS=1460
449	25.455169	192.168.2.8	192.168.2.2	TCP	58		43 42655→microsoft-ds(445) [SYN] Seq=0 Win=1024 Len=0 MSS=1460
450	25.455709	192.168.2.8	192.168.2.2	TCP	58		56 42655→ddi-tcp-1(8888) [SYN] Seq=0 Win=1024 Len=0 MSS=1460

Maneuver to a set of packets that show both unacknowledged SYNs and SYN/ACKS similar to the image above. **Get a screenshot.**

Attacker: **Snip** nmap output on target.

273	3.670754	184.171.153.136	184.171.153.6	TCP	58 [TCP Out-Of-Order] 55579 → 513 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
274	3.670796	184.171.153.136	184.171.153.6	TCP	58 55579 → 10000 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
275	3.670799	184.171.153.136	184.171.153.6	TCP	58 [TCP Out-Of-Order] 55579 → 10000 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
276	3.670841	184.171.153.136	184.171.153.6	TCP	58 55579 → 5051 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
277	3.670844	184.171.153.136	184.171.153.6	TCP	58 [TCP Out-Of-Order] 55579 → 5051 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
278	3.670885	184.171.153.136	184.171.153.6	TCP	58 55579 → 5000 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
279	3.670888	184.171.153.136	184.171.153.6	TCP	58 [TCP Out-Of-Order] 55579 → 5000 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
280	3.670930	184.171.153.136	184.171.153.6	TCP	58 55579 → 544 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
281	3.670932	184.171.153.136	184.171.153.6	TCP	58 [TCP Out-Of-Order] 55579 → 544 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
282	3.670974	184.171.153.136	184.171.153.6	TCP	58 55579 → 2121 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
283	3.670976	184.171.153.136	184.171.153.6	TCP	58 [TCP Out-Of-Order] 55579 → 2121 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
284	3.671018	184.171.153.136	184.171.153.6	TCP	58 55579 → 49153 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
285	3.671021	184.171.153.136	184.171.153.6	TCP	58 [TCP Out-Of-Order] 55579 → 49153 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
286	3.671062	184.171.153.136	184.171.153.6	TCP	58 55579 → 1029 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
287	3.671065	184.171.153.136	184.171.153.6	TCP	58 [TCP Out-Of-Order] 55579 → 1029 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
288	3.671106	184.171.153.136	184.171.153.6	TCP	58 55579 → 49152 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
289	3.671109	184.171.153.136	184.171.153.6	TCP	58 [TCP Out-Of-Order] 55579 → 49152 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
290	3.671151	184.171.153.136	184.171.153.6	TCP	58 55579 → 5190 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
291	3.671153	184.171.153.136	184.171.153.6	TCP	58 [TCP Out-Of-Order] 55579 → 5190 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
292	3.671199	184.171.153.136	184.171.153.6	TCP	58 55579 → 49154 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
293	3.671201	184.171.153.136	184.171.153.6	TCP	58 [TCP Out-Of-Order] 55579 → 49154 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
294	3.671244	184.171.153.136	184.171.153.6	TCP	58 55579 → 88 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
295	3.671246	184.171.153.136	184.171.153.6	TCP	58 [TCP Out-Of-Order] 55579 → 88 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
296	3.671288	184.171.153.136	184.171.153.6	TCP	58 55579 → 1433 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
297	3.671291	184.171.153.136	184.171.153.6	TCP	58 [TCP Out-Of-Order] 55579 → 1433 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
298	3.671332	184.171.153.136	184.171.153.6	TCP	58 55579 → 106 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
299	3.671335	184.171.153.136	184.171.153.6	TCP	58 [TCP Out-Of-Order] 55579 → 106 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
300	3.671376	184.171.153.136	184.171.153.6	TCP	58 55579 → 548 [SYN] Seq=0 Win=1024 Len=0 MSS=1460

It might be helpful to know you are being scanned or attacked. An attack in this case would be a more intense number of SYN packets.

Can this action be spotted /identified by the target machine as it is happening? **yes**

Find the following data from the Wireshark capture.

Under Capture File Properties, record the following statistics:

Packets	8573
Time span s	155.836
Average pps	55.0
Average packet size, B	176
Bytes	1511078
Average bytes/s	9696
Average bits/s	77k

What are the two most common packet lengths? **40-79, 16-319**

Look at statistics/I-O graph

What is the peak value? **1864**

When in the capture did it occur? **149s**

How long did it last? **Less than a few seconds**

How does this data compare with the normal traffic statistics you obtained earlier? What are the differences that might be able to identify a SYN SCAN/FLOOD? **The peaks are much more noticeable and the types and sizes of the packets are different from the normal.**