DAY7_advent_of_the_cyber

Learning Objectives

- What are IP Addresses & how are they assigned.
- Understanding TCP/IP and UDP
  - 3-way handshake
- Wireshark Crash Course (where is it used and why)
  - Basic Filtering and operators
- Analysing our first few PCAPS
  - HTTP
  - SMB
- Challenge

Made with ❤ by CMNatic

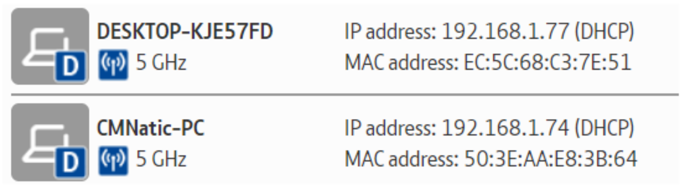
What is an IP Address?

You'll hear talk of the term "IP address" frequently throughout the information technology field - not just TryHackMe. Short for an Internet Protocol address, I like to explain this fundamental of networking using the same way that a postal/mail system works in real life.

When sending a letter, you must provide the address for where the letter should go, and it is best practice to include your address as the return address in case the letter is lost (or you wish to let the recipient know how to reply). An IP address serves the same purpose but for devices connected to a network! Devices connected to the internet will have two of these addresses -a public and a private address. Think of a private address as the name of the recipient at a business i.e. Joe Smith, and the public address being the location of this business i.e. 160 Kemp Road, London.

Let's say you are accessing the Internet through your computer. Your computer will be a part of two networks, and in turn, will use both public and private IP address:

- A private IP address to identify itself amongst other devices (such as smartphones, TV's and other computers) within the network of your house. In the screenshot below, the two devices have the following private IP address:



DESKTOP-KJE57FD 5 GHz	IP address: 192.168.1.77 (DHCP) MAC address: EC:5C:68:C3:7E:51
CMNatic-PC 5 GHz	IP address: 192.168.1.74 (DHCP) MAC address: 50:3E:AA:E8:3B:64

Device Name	IP Address	IP Address Type
DESKTOP-KJE57FD	192.168.1.77	Private
CMNatic-PC	192.168.1.74	Private

TryHackMe similarly uses these private addresses. For you to access other TryHackMe devices such as the instances that you deploy in this room, you will need to be on the same private network as these instances are not connected to the internet. This is why you must use a client such as OpenVPN to connect to the network.

MuirlandOracle explains how these private IP addresses work in his Intro to Networking room.

- A public IP address was given by your Internet Service Provider (ISP) that identifies your house on the Internet (the Internet is just many, many networks connected). Using our example from above, the two devices will share a public IP address to identify themselves on the Internet:

My Public IPv4 is: 86.157.52.21

Location: ENG GB

ISP: Secure Communications TLD

Device Name	IP Address	IP Address Type
DESKTOP-KJE57FD	86.157.52.21	Public
CMNatic-PC	86.157.52.21	Public

This is achieved through NATting, however, detailing how this works exactly this works is a bit beyond the scope of today.

Protocols 101

With the internet predicted to have 50 billion devices connected by the end of 2020 (https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf), chaos quickly ensues if there are no ground-rules in how devices should communicate with each other.

If this is a bit confusing - I don't blame you, just bear with me here. Think of it this way: You use protocols in everyday life! When talking to someone, you will both use the same set of protocols...otherwise, no one will understand each other. At the very least, all parties wishing to converse will use the same language - this is a protocol! Other protocols may also include the context or topic of the conversation. If anyone strays from these protocols - they risk not being understood! This is the same for network-connected devices.

- Back to the technical stuff...

Enter protocols such as the TCP/IP & UDP/IP models. With TCP/IP being the most common-place today, we'll discuss this further. TCP/IP is a protocol that ensures that any data sent is processed in the same order. Going back to our postal system, your letter will go to many places - even when sending domestically. Your computer traffic does the same, going from device to device in a process called routing. One device could deliver data quicker (and a result, in a different order) then another causing a headache for the situation where accuracy is important such as the following:

- Downloading files
- Visiting a website in your browser
- Sending emails

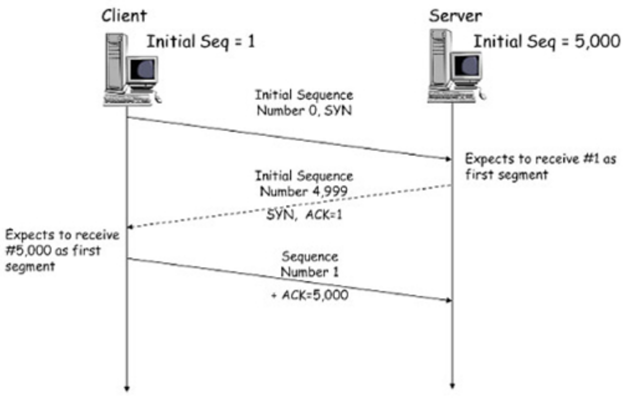
This is unlike the UDP protocol where having all packets is not quite as important (making the protocol a lot quicker than TCP/IP) which is why applications like video streaming make use of UDP (i.e. Skype). We don't really care if a few packets are lost as we can still see a majority of the picture.

How does TCP/IP send data? The Three-Way Handshake:

The three-way handshake is the method that makes TCP reliable. Any data that is sent is given a random number sequence and is reconstructed using this number sequence and incrementing by 1. Both computers must agree on the same number sequence for data to be sent in the correct order. This order is agreed upon during three steps.

In the diagram below, "Client" has the initial sequence number (ISN) of "0" where the "Server" has "5,000". Any data sent from "Client" and received on "Server" will be initial sequence + 1. If this is the first packet from "Client" this would be "0 + 1". I've shown three packets being sent from "Client" in the table below to help demonstrate this:

Device	Initial Sequence Number (IN)	Final Sequence Number	
Client (Sender)	0	0 + 1 = 1	
Client (Sender)	1	1 + 1 = 2	
Client (Sender)	2	2 + 1 = 3	


Client
Initial Seq = 1
Server
Initial Seq = 5,000
Initial Sequence Number 0, SYN
Expects to receive #1 as first segment
Initial Sequence Number 4,999
SYN, ACK=1
Expects to receive #5,000 as first segment
Sequence Number 1
+ ACK=5,000

1. SYN - Client: Here's my initial number sequence (ISN) to **SYN**chronise with (0)
2. SYN/ACK - Server: Here my Initial Number Sequence (ISN) to **SYN**chronise with (5,000) and I **ACK**nowledge your initial number sequence (0)
3. ACK - Client: I **ACK**nowledge your Initial Number Sequence (ISN) of (5,000) here is some data that is my ISN+1

When the data is received, it is reassembled by the receiver. Let's show the conditions of which reassembly is required:

Examples of TCP reassembling data:

No reassembly required:

If the "Server" were to receive data that was received in the exact order it was sent from "Client":

1. Sent 1st - Received 1st
2. Sent 2nd - Received 2nd
3. Sent 3rd - Received 3rd

Then no reassembly is needed as the data is received in the exact order it was sent.

Reassembly required:

For example, if the "Server" was to receive all the data, but in a different order then what was sent, reassembly is required:

1. Sent 1st - Received 1st
2. Sent 2nd - Received 3rd
3. Sent 3rd - Received 2nd

Because all data is received, just in a different order, it can be reassembled using the agreed sequence numbers that would have been exchanged during the three-way-handshake.

The connection is dropped:

If the "Client" was to send **three** packets, but the "Server" only receives **two** out of three packets, they are disconnected from each other as the data sent is corrupt:

1. Sent 1st - Received 1st
2. Sent 2nd - Not received
3. Sent 3rd - Received 2nd

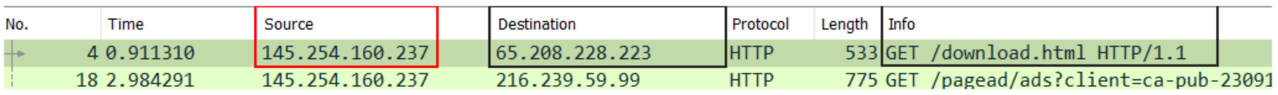
The data will not be processed by "Server" as the packet that was sent 2nd by the "Client" was never received by the "Server"#2, meaning there was a loss of data along the way.

Crash Course in Monitoring Network Traffic:

Being able to capture exactly what is travelling across a network and understanding this is an important skill in information technology. From diagnosing to capturing credentials, positions in IT ranging from system administrators to digital forensics and us pentesters all use network traffic in their own ways. For example, since data sent via HTTP or FTP is unencrypted, a pentester might be able to capture usernames and passwords being entered into a website.

Introducing Wireshark:

Wireshark is capable of recording a log of all the packets sent and received on a computer's network adapter. For example, we can see how a computer (highlighted in red) connected to a computer (highlighted in black) that was running a web server via HTTP, in this case, it was the web page: web_server/download.HTML, which we can export and view for ourselves:



```
No.     Time        Source            Destination       Protocol  Length  Info
     4  0.911310    145.254.160.237   65.208.228.223    HTTP         533  GET /download.html HTTP/1.1
    18  2.984291    145.254.160.237   216.239.59.99     HTTP         775  GET /pagead/ads?client=ca-pub-23091
```

Networks are, however, rather noisy...Wireshark captured 2,648 packets after a single minute on my machine. This makes analysing very hard. Thankfully, we can use filters to narrow down the results. We can filter by many things, but we'll only cover a couple of important ones in the table below. Note that all the examples below use the `==` operator to see if the filter **exactly** matches the value we give it.

Filter	Description	Example
ip.src	Show all packets that originate from the specified IP address	`ip.src == 192.168.1.1`
ip.dst	Show all packets that are destined to the specified IP address	`ip.dst == 192.168.1.1`
tcp/udp.port	Show all packets that are sent via the protocol and port specified	`tcp.port == 22 / udp.port == 67`
protocol.request.method	Show all packets that use a specific method of the protocol given. For example, HTTP allows for both a `GET` and `POST` to retrieve and submit data accordingly.	`http.request.method == GET / POST`

Combining Filters With Operators

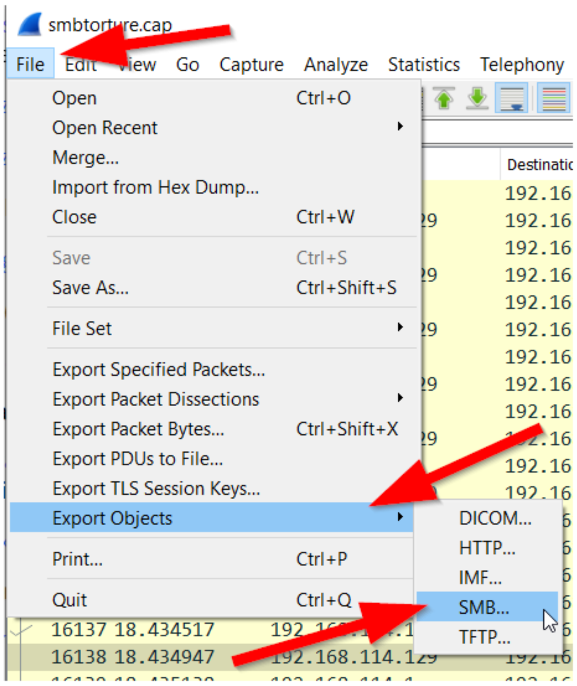
Operator	Description	Example
==	You'd use this operator to check if the filter exactly matches the value given in all packets	`ip.addr == 192.168.1.1` will show all packets with the IP address `192.168.1.1` (this could be source or destination)
!=	This operator checks if the filter does not match the value given in all packets	`ip.addr != 192.168.1.10` will show all packetsthat does not include the IP address `192.168.1.10` (this could be source or destination)
&&	Use this operator to combine multiple filters together.	For example, to show all packets associated with two different IP addresses `ip.addr == 192.168.1.1 && ip.addr == 192.168.1.10` will only show packets with the source or destination IP addresses of `192.168.1.1` or `192.168.1.10`

Exporting data from Wireshark:

As previously shown, Wireshark is capable of exporting data from protocols such as HTTP by navigating to "File → Export Objects" and selecting the protocol available. In the screenshots below, we are listing objects that can be exported from the file-sharing SMB protocol.

Exporting data from Wireshark:

As previously shown, Wireshark is capable of exporting data from protocols such as HTTP by navigating to "File → Export Objects" and selecting the protocol available. In the screenshots below, we are listing objects that can be exported from the file-sharing SMB protocol.


smbtorture.cap
File
Edit
View
Go
Capture
Analyze
Statistics
Telephony
Open Ctrl+O
Open Recent
Merge...
Import from Hex Dump...
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
File Set
Export Specified Packets...
Export Packet Dissections
Export Packet Bytes... Ctrl+Shift+X
Export PDUs to File...
Export TLS Session Keys...
Export Objects
Print... Ctrl+P
Quit Ctrl+Q
DICOM...
HTTP...
IMF...
SMB...
TFTP...
Destinatio
16137 18.434517
16138 18.434947 192.168.114.129