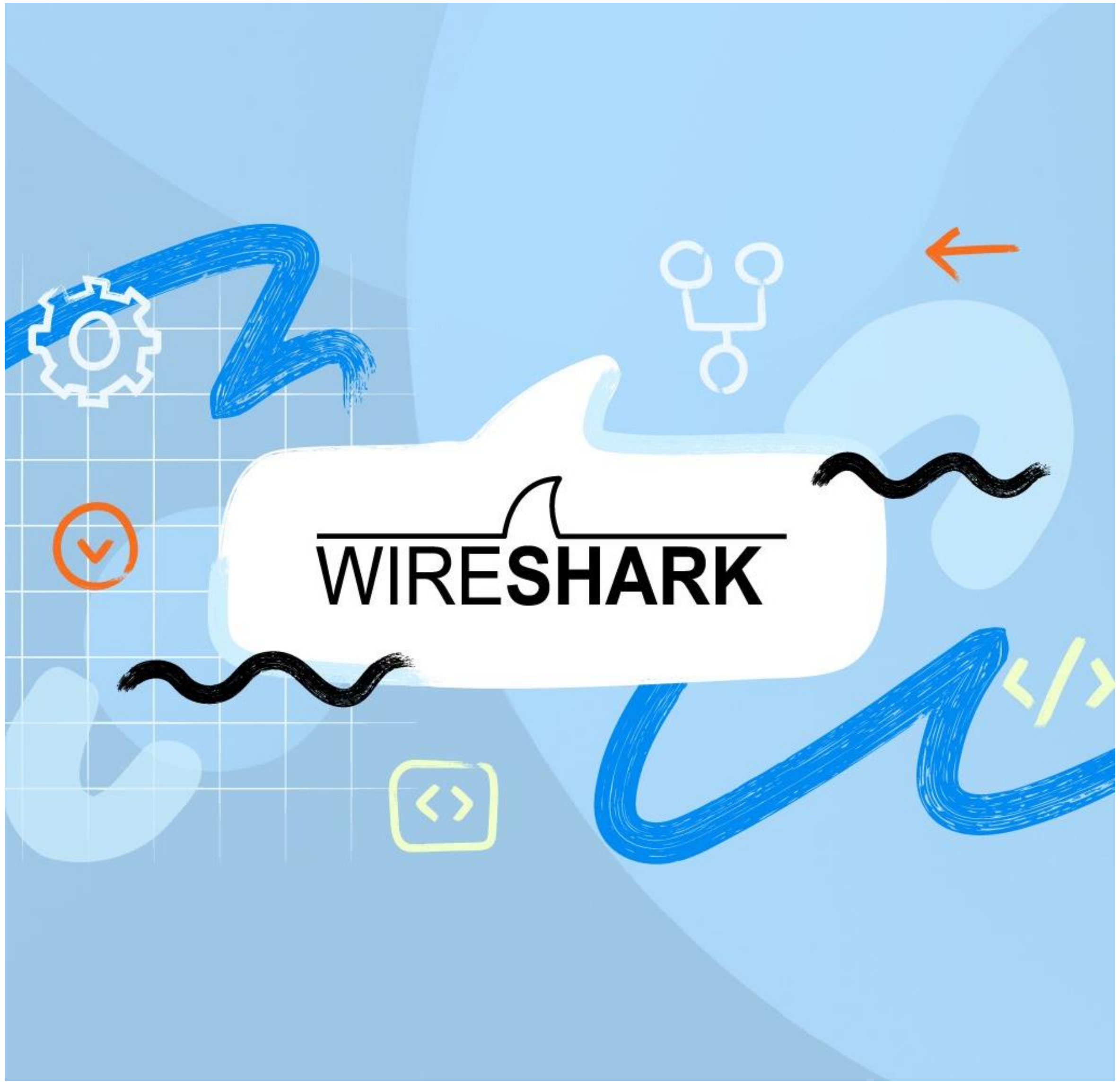
Source: Freepik | Available at: https://tinyurl.com/4pvxz7su

TRAFFIC ANALYSIS USING WIRESHARK

Name: Shah Abu Kawsar Rafi

B00 Number: B00968292

ID Number: 10403451



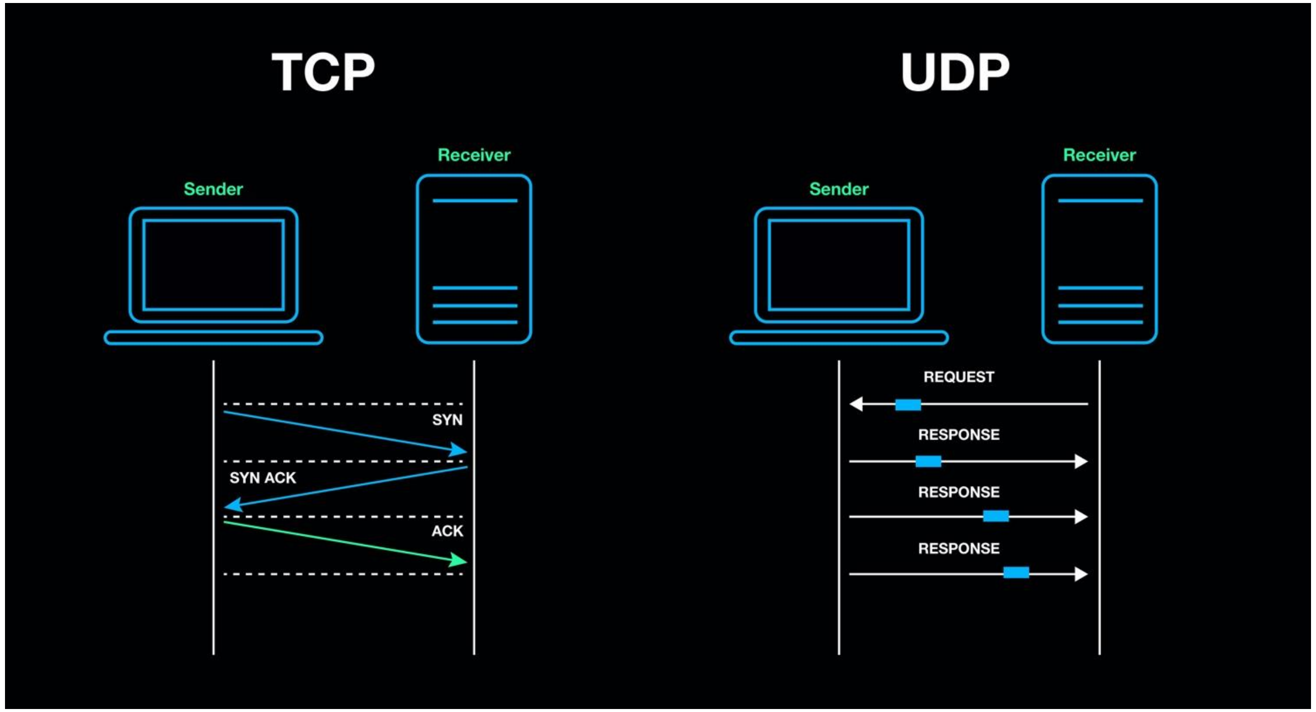
Source: InvGate | Available at: https://tinyurl.com/y54he8ef

INTRODUCTION

In today's digital world, network security is critical to maintaining the integrity and availability of systems. With increasing cyber-attacks targeting various network components, it is vital to understand the behaviour of these attacks and how to defend against them.

This presentation explains the TCP/IP protocol suite, including TCP and UDP protocols, their differences, and the distinction between TCP and IP. It also covers the TCP 3-Way Handshake, the definitions and differences between DoS and DDoS attacks, and an analysis of SYN Flood, HTTP Flood, and Fragmentation attacks using Wireshark, a network protocol analyser. Additionally, it explores mitigation strategies to protect against these types of threats.

TCP (TRANSMISSION CONTROL PROTOCOL) AND UDP (USER DATAGRAM PROTOCOL)

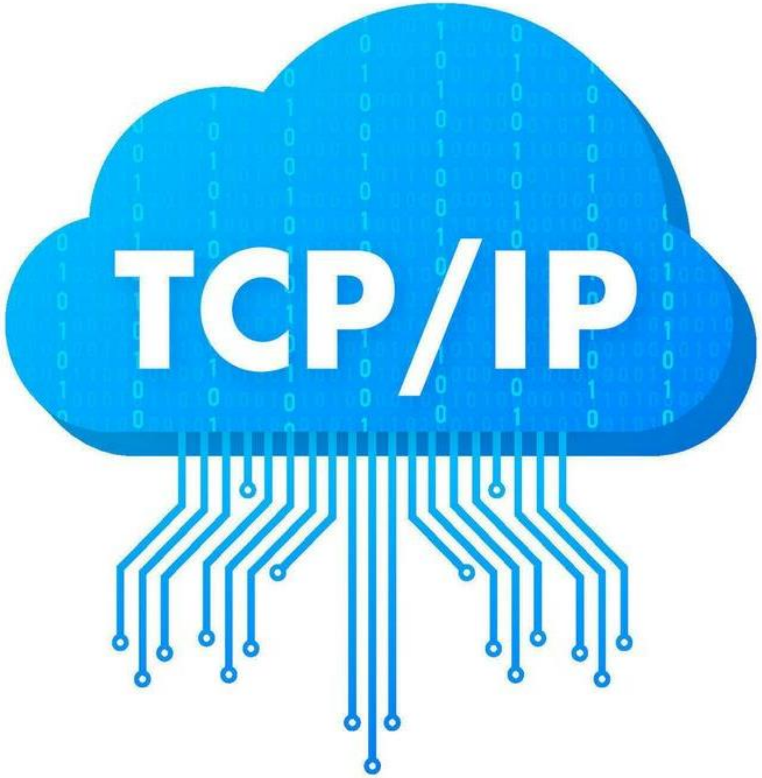


Source: Ebyte | Available at: https://tinyurl.com/3wawehe5

- **TCP (Transmission Control Protocol)**: A connection-oriented protocol that ensures reliable data transmission with error checking and retransmission. Suitable for applications like web browsing and email.
- **UDP (User Datagram Protocol)**: A connectionless protocol offering faster transmission without error checking or retransmission, making it ideal for streaming and online gaming.

THE DIFFERENCE BETWEEN THE TWO PROTOCOLS

TCP	UDP
TCP is connection oriented	UDP is connectionless
It uses acknowledgement to prevent and correct errors.	It cannot correct errors.
TCP data packets have a sequencing number in the header to maintain the order of transmission.	UDP data packets arrive in no fixed order, and incorrect sequencing cannot be detected or corrected.
It has a longer latency time and consumes more resources.	It starts the connection faster, delivers data at lower latency, and consumes fewer resources.
TCP’s most significant advantage is that it is highly reliable.	Its architecture is designed in a manner that makes it inherently unreliable.
It is suitable for use cases where data integrity, including images, web pages, data files, etc. matters more than transmission speed.	It is ideal for live data transmission (e.g., media), where transmission is so fast that a few dropped packets do not matter.



Source: Vecteezy | Available at: https://tinyurl.com/yd6fk2cm

TCP/IP OVERVIEW

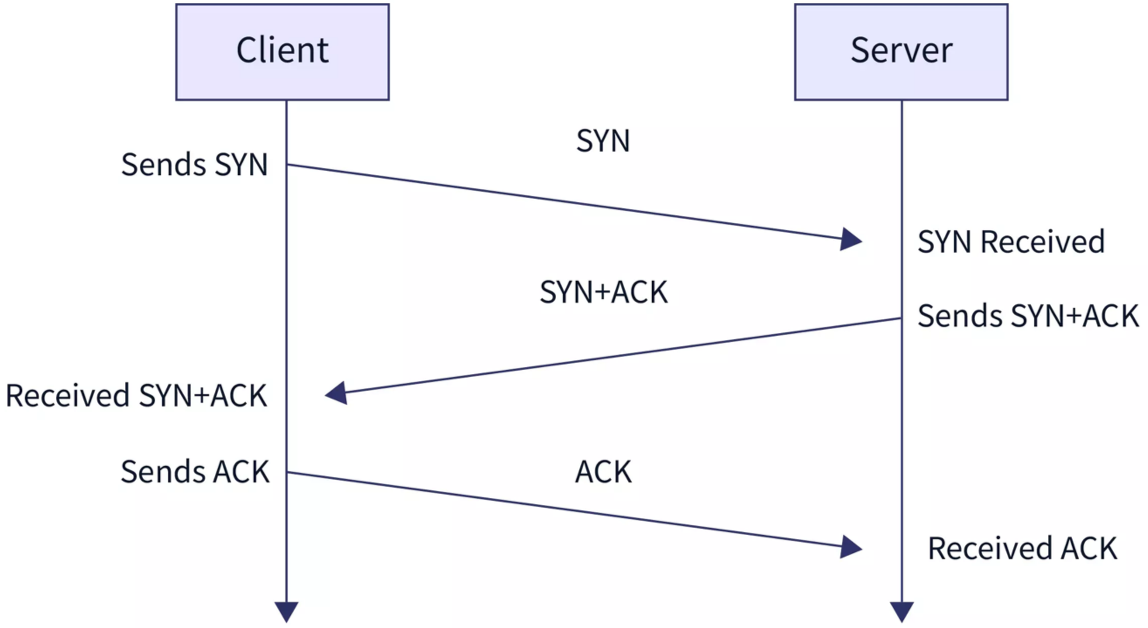
TCP/IP (Transmission Control Protocol/Internet Protocol) is the backbone of internet communication. **TCP** ensures reliable communication by establishing a connection and ensuring data integrity, while **IP** is responsible for routing packets across the network. Understanding these protocols is essential for detecting abnormal traffic patterns that may indicate an attack.

Difference Between TCP and IP:

- **TCP** operates at the **transport layer**, ensuring reliability in data transmission.
- **IP** operates at the **network layer**, handling addressing and routing packets between networks.

TCP 3-WAY HANDSHAKE PROCESS

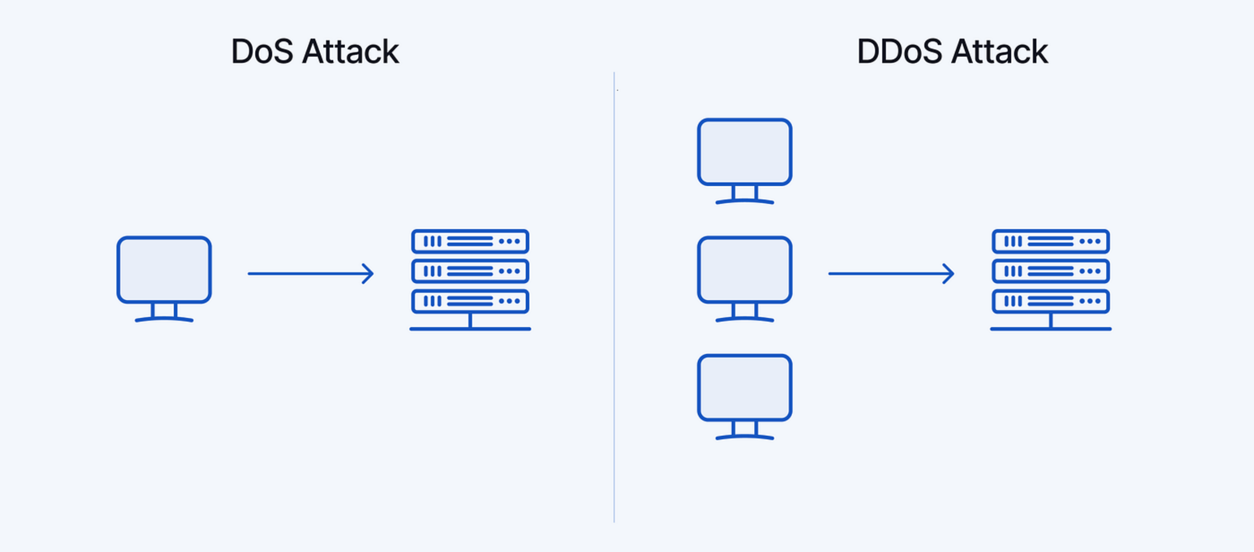
- Step 1: **SYN** – The client sends a synchronisation request to the server.
- Step 2: **SYN-ACK** – The server acknowledges the request.
- Step 3: **ACK** – The client sends an acknowledgment, and the connection is established.



Source: Turkhackteam | Available at: https://tinyurl.com/mskac9xh

DENIAL-OF-SERVICE (DOS) AND DISTRIBUTED DENIAL-OF-SERVICE (DDOS) ATTACKS

- **DoS Attack**: A single attacker floods a target system with excessive requests, disrupting service availability.
- **DDoS Attack**: Multiple compromised systems (botnets) overwhelm the target with traffic, making it harder to mitigate.

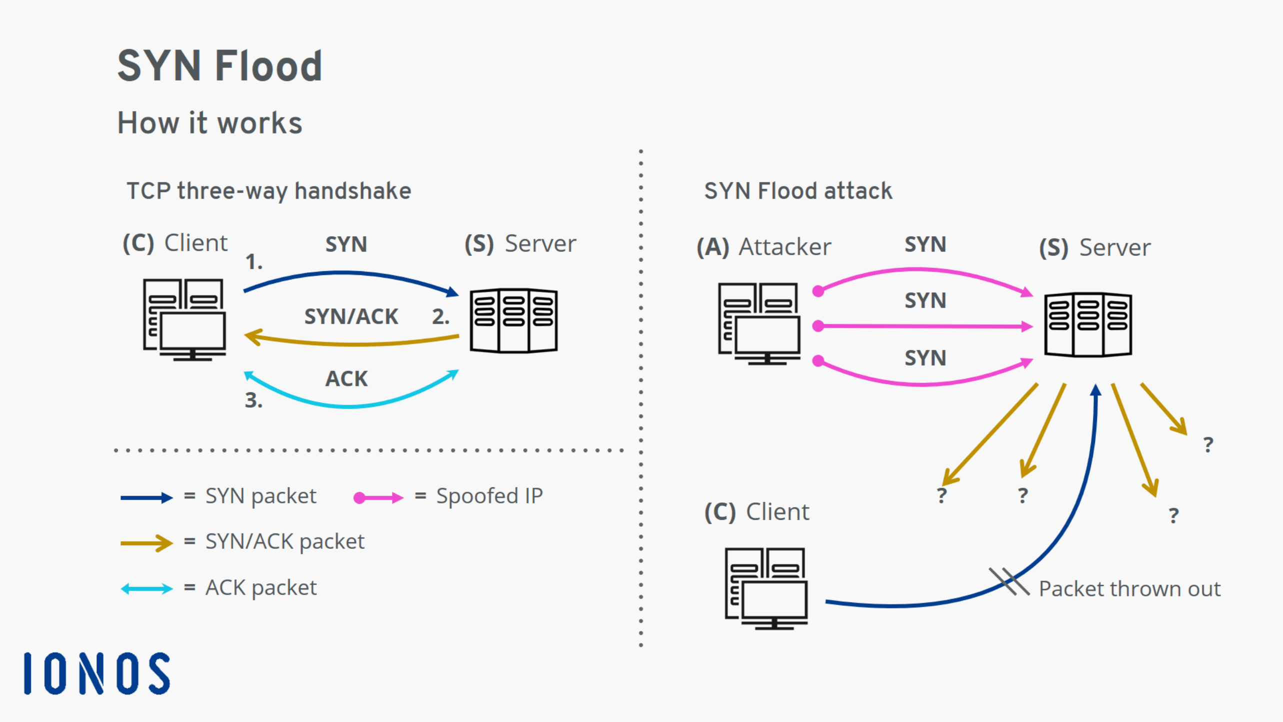


Source: Linkedin | Available at: https://tinyurl.com/bde8uaw9

DENIAL-OF-SERVICE (DOS) VS. DISTRIBUTED DENIAL-OF-SERVICE (DDOS) ATTACKS

DoS	DDoS
A DoS attack is an attempt to make an online service unavailable by overwhelming it with traffic from a single source	A DDoS attack is an attempt to make an online service unavailable by overwhelming it with traffic from multiple sources
Single source, usually one machine or IP	Multiple sources, coordinated botnets, or compromised IPs
Relatively simple and easy to execute	More complex, requires coordination and larger resources
Lower volume of traffic	Significantly higher volume of traffic
Easier to detect and mitigate as traffic comes from a single source	Harder to detect and mitigate due to traffic coming from many different sources

OVERVIEW: SYN FLOOD ATTACK

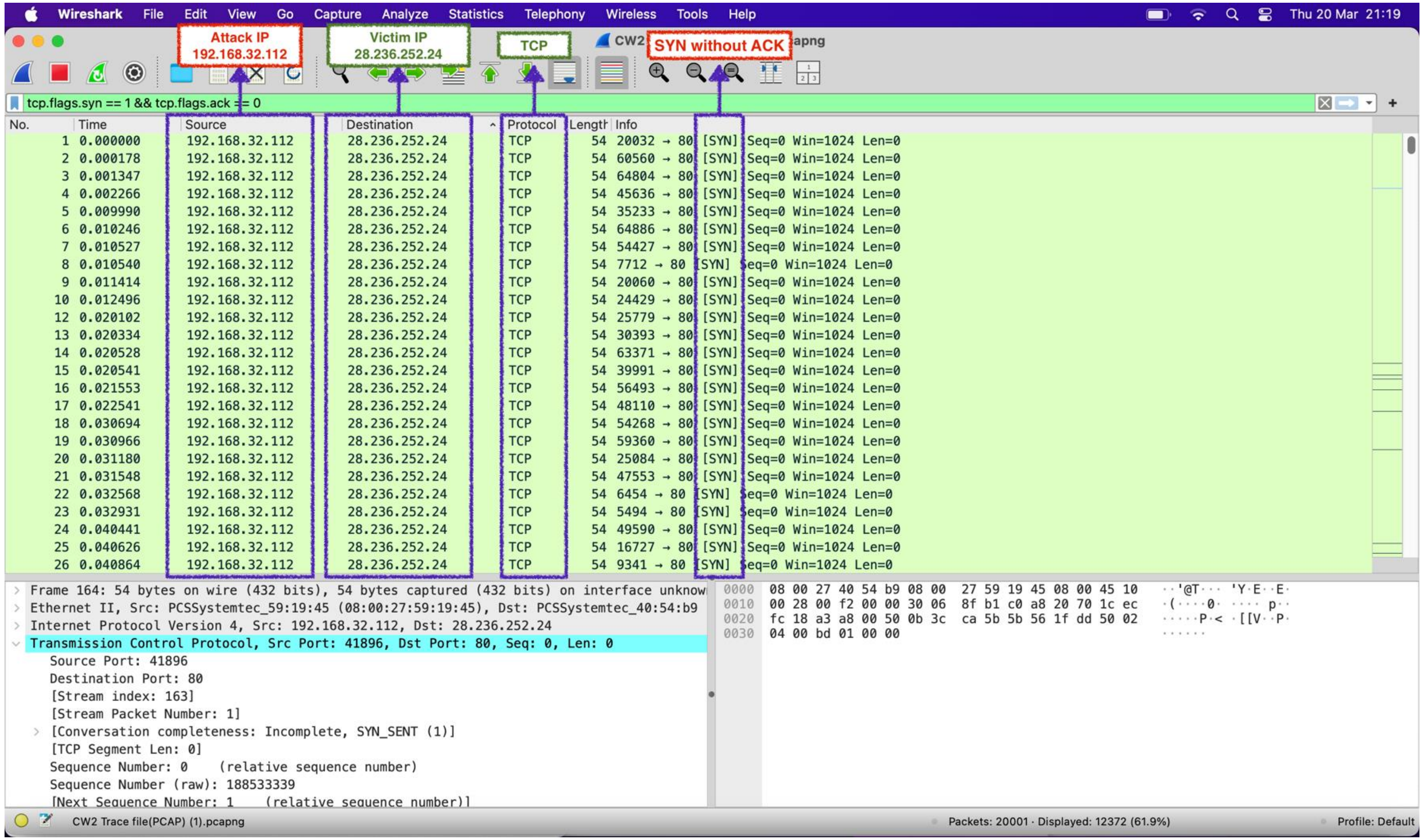


Source: Ionos | Availabe at: https://tinyurl.com/46hbby9j

A SYN Flood attack is a type of Denial-of-Service (DoS) attack that targets the TCP handshake process. In this attack, an attacker sends a flood of SYN packets to a target system without completing the handshake, exhausting the target's resources and preventing legitimate users from establishing connections.

The main goal of a SYN Flood is to overwhelm the target system, leading to denial of service.

WIRESHARK TRAFFIC ANALYSIS: SYN FLOOD ATTACK



```
tcp.flags.syn == 1 && tcp.flags.ack == 0
```

```
No.  Time      Source          Destination    Protocol Length Info
  1 0.000000  192.168.32.112  28.236.252.24  TCP      54 20032 → 80 [SYN] Seq=0 Win=1024 Len=0
  2 0.000178  192.168.32.112  28.236.252.24  TCP      54 60560 → 80 [SYN] Seq=0 Win=1024 Len=0
  3 0.001347  192.168.32.112  28.236.252.24  TCP      54 64804 → 80 [SYN] Seq=0 Win=1024 Len=0
  4 0.002266  192.168.32.112  28.236.252.24  TCP      54 45636 → 80 [SYN] Seq=0 Win=1024 Len=0
  5 0.009990  192.168.32.112  28.236.252.24  TCP      54 35233 → 80 [SYN] Seq=0 Win=1024 Len=0
  6 0.010246  192.168.32.112  28.236.252.24  TCP      54 64886 → 80 [SYN] Seq=0 Win=1024 Len=0
  7 0.010527  192.168.32.112  28.236.252.24  TCP      54 54427 → 80 [SYN] Seq=0 Win=1024 Len=0
  8 0.010540  192.168.32.112  28.236.252.24  TCP      54 7712 → 80 [SYN] Seq=0 Win=1024 Len=0
  9 0.011414  192.168.32.112  28.236.252.24  TCP      54 20060 → 80 [SYN] Seq=0 Win=1024 Len=0
 10 0.012496  192.168.32.112  28.236.252.24  TCP      54 24429 → 80 [SYN] Seq=0 Win=1024 Len=0
 12 0.020102  192.168.32.112  28.236.252.24  TCP      54 25779 → 80 [SYN] Seq=0 Win=1024 Len=0
 13 0.020334  192.168.32.112  28.236.252.24  TCP      54 30393 → 80 [SYN] Seq=0 Win=1024 Len=0
 14 0.020528  192.168.32.112  28.236.252.24  TCP      54 63371 → 80 [SYN] Seq=0 Win=1024 Len=0
 15 0.020541  192.168.32.112  28.236.252.24  TCP      54 39991 → 80 [SYN] Seq=0 Win=1024 Len=0
 16 0.021553  192.168.32.112  28.236.252.24  TCP      54 56493 → 80 [SYN] Seq=0 Win=1024 Len=0
 17 0.022541  192.168.32.112  28.236.252.24  TCP      54 48110 → 80 [SYN] Seq=0 Win=1024 Len=0
 18 0.030694  192.168.32.112  28.236.252.24  TCP      54 54268 → 80 [SYN] Seq=0 Win=1024 Len=0
 19 0.030966  192.168.32.112  28.236.252.24  TCP      54 59360 → 80 [SYN] Seq=0 Win=1024 Len=0
 20 0.031180  192.168.32.112  28.236.252.24  TCP      54 25084 → 80 [SYN] Seq=0 Win=1024 Len=0
 21 0.031548  192.168.32.112  28.236.252.24  TCP      54 47553 → 80 [SYN] Seq=0 Win=1024 Len=0
 22 0.032568  192.168.32.112  28.236.252.24  TCP      54 6454 → 80 [SYN] Seq=0 Win=1024 Len=0
 23 0.032931  192.168.32.112  28.236.252.24  TCP      54 5494 → 80 [SYN] Seq=0 Win=1024 Len=0
 24 0.040441  192.168.32.112  28.236.252.24  TCP      54 49590 → 80 [SYN] Seq=0 Win=1024 Len=0
 25 0.040626  192.168.32.112  28.236.252.24  TCP      54 16727 → 80 [SYN] Seq=0 Win=1024 Len=0
 26 0.040864  192.168.32.112  28.236.252.24  TCP      54 9341 → 80 [SYN] Seq=0 Win=1024 Len=0
```

```
> Frame 164: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface unknow
> Ethernet II, Src: PCSSystemtec_59:19:45 (08:00:27:59:19:45), Dst: PCSSystemtec_40:54:b9
> Internet Protocol Version 4, Src: 192.168.32.112, Dst: 28.236.252.24
v Transmission Control Protocol, Src Port: 41896, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 41896
    Destination Port: 80
    [Stream index: 163]
    [Stream Packet Number: 1]
  > [Conversation completeness: Incomplete, SYN_SENT (1)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 188533339
    [Next Sequence Number: 1    (relative sequence number)]
```

```
0000  08 00 27 40 54 b9 08 00  27 59 19 45 08 00 45 10   ··'@T··· 'Y·E··E·
0010  00 28 00 f2 00 00 30 06  8f b1 c0 a8 20 70 1c ec   ·(····0· ···· p··
0020  fc 18 a3 a8 00 50 0b 3c  ca 5b 5b 56 1f dd 50 02   ·····P·< ·[[V··P·
0030  04 00 bd 01 00 00                                  ······
```

```
CW2 Trace file(PCAP) (1).pcapng        Packets: 20001 · Displayed: 12372 (61.9%)        Profile: Default
```

In this attack, I observed a high number of SYN packets. The screenshot from Wireshark shows the filtered traffic where SYN flags are set but without the corresponding ACK responses. These half-open connections tie up resources on the target system.

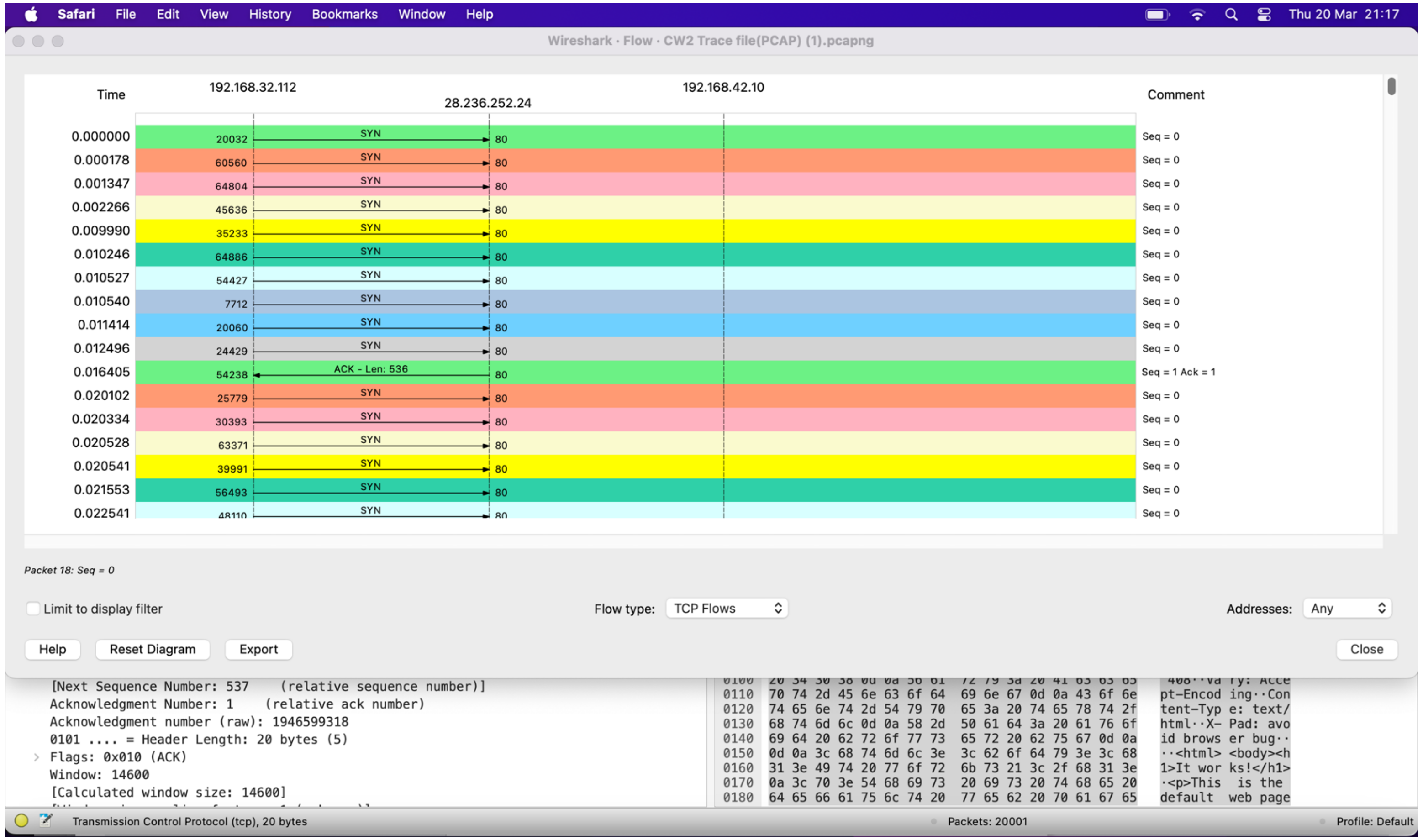
Filter: (tcp.flags.syn == 1 && tcp.flags.ack == 0)

Attacker IP: 192.168.32.122

Victim IP: 28.236.252.24

Figure 01: Wireshark screenshot

IMPACT ON TARGET SYSTEM: SYN FLOOD ATTACK


Safari File Edit View History Bookmarks Window Help
Thu 20 Mar 21:17
Wireshark · Flow · CW2 Trace file(PCAP) (1).pcapng

Time	192.168.32.112	28.236.252.24		192.168.42.10	Comment
0.000000	20032	SYN →	80		Seq = 0
0.000178	60560	SYN →	80		Seq = 0
0.001347	64804	SYN →	80		Seq = 0
0.002266	45636	SYN →	80		Seq = 0
0.009990	35233	SYN →	80		Seq = 0
0.010246	64886	SYN →	80		Seq = 0
0.010527	54427	SYN →	80		Seq = 0
0.010540	7712	SYN →	80		Seq = 0
0.011414	20060	SYN →	80		Seq = 0
0.012496	24429	SYN →	80		Seq = 0
0.016405	54238	← ACK - Len: 536	80		Seq = 1 Ack = 1
0.020102	25779	SYN →	80		Seq = 0
0.020334	30393	SYN →	80		Seq = 0
0.020528	63371	SYN →	80		Seq = 0
0.020541	39991	SYN →	80		Seq = 0
0.021553	56493	SYN →	80		Seq = 0
0.022541	48110	SYN →	80		Seq = 0

Packet 18: Seq = 0

☐ Limit to display filter   Flow type: TCP Flows   Addresses: Any

```
    [Next Sequence Number: 537    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 1946599318
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window: 14600
    [Calculated window size: 14600]
```

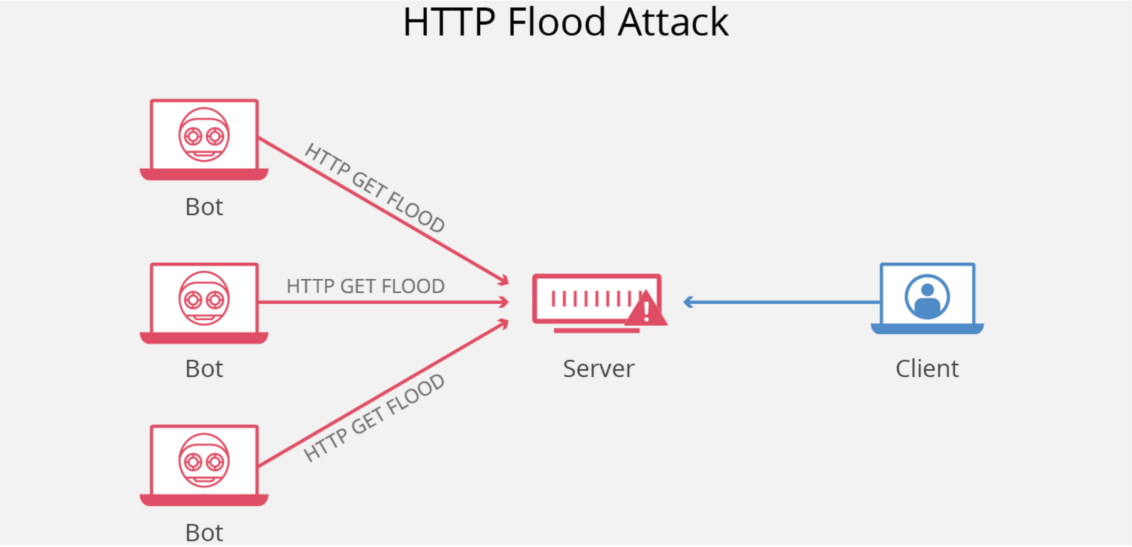
```
0100  20 34 30 38 0d 0a 56 61  72 79 3a 20 41 63 63 65   408··Va ry: Acce
0110  70 74 2d 45 6e 63 6f 64  69 6e 67 0d 0a 43 6f 6e   pt-Encod ing··Con
0120  74 65 6e 74 2d 54 79 70  65 3a 20 74 65 78 74 2f   tent-Typ e: text/
0130  68 74 6d 6c 0d 0a 58 2d  50 61 64 3a 20 61 76 6f   html··X- Pad: avo
0140  69 64 20 62 72 6f 77 73  65 72 20 62 75 67 0d 0a   id brows er bug··
0150  0d 0a 3c 68 74 6d 6c 3e  3c 62 6f 64 79 3e 3c 68   ··<html> <body><h
0160  31 3e 49 74 20 77 6f 72  6b 73 21 3c 2f 68 31 3e   1>It wor ks!</h1>
0170  0a 3c 70 3e 54 68 69 73  20 69 73 20 74 68 65 20   ·<p>This  is the 
0180  64 65 66 61 75 6c 74 20  77 65 62 20 70 61 67 65   default  web page
```

Transmission Control Protocol (tcp), 20 bytes   Packets: 20001   Profile: Default

The target system is forced to allocate resources to manage incomplete connections (half-open connections). This leads to resource depletion, causing the system to be unable to process new, legitimate requests. The system may either slow down or crash, resulting in denial of service.

Figure 02: Wireshark Flow (TCP)

OVERVIEW: HTTP FLOOD ATTACK

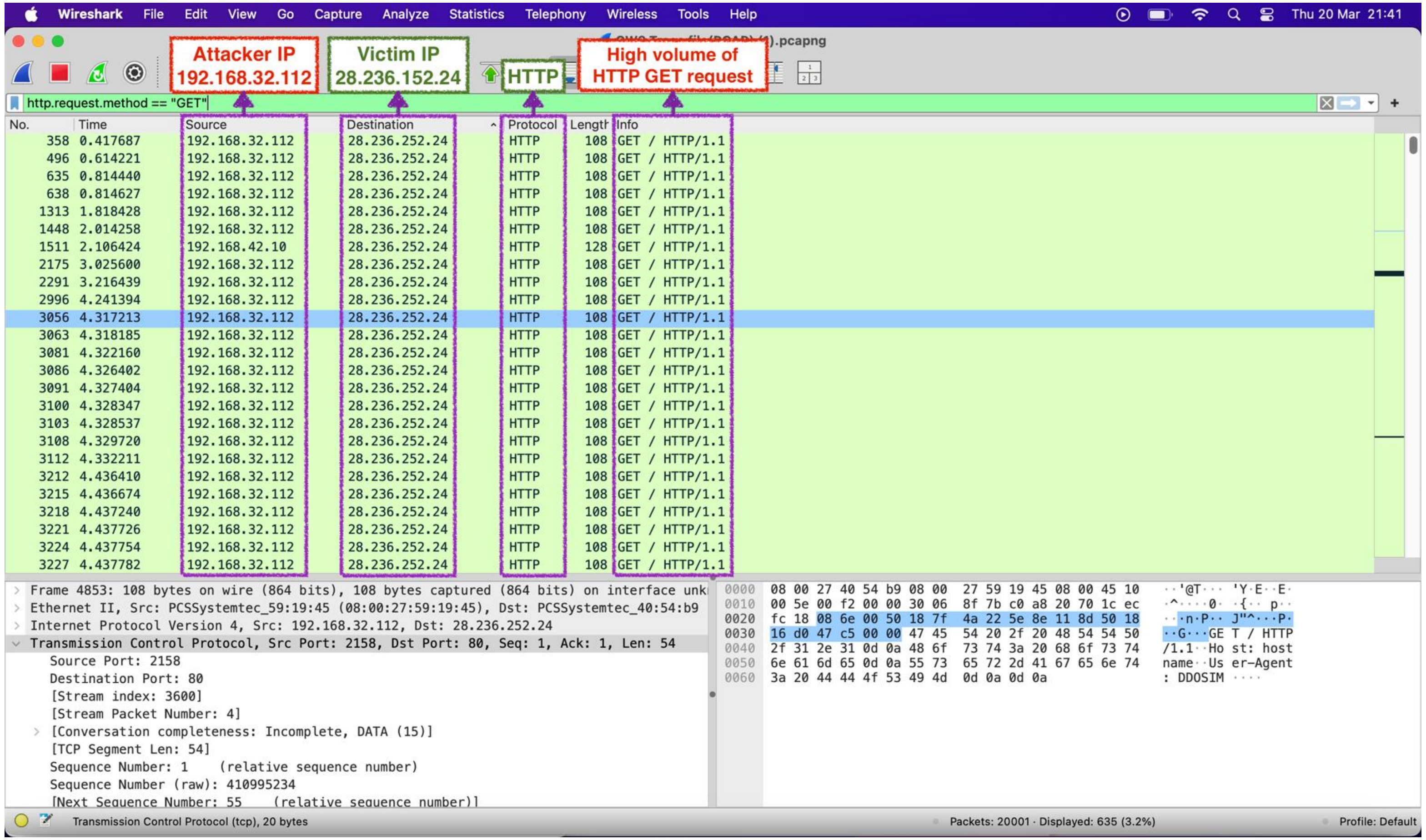


Source: Researchgate | Available at: https://tinyurl.com/h6dphnud

An HTTP Flood attack is a type of Distributed Denial of Service (DDoS) attack that targets a web server by overwhelming it with excessive HTTP GET or POST requests. These requests consume server resources (CPU, memory), making the server unable to process legitimate requests.

The goal of this attack is to exhaust the web server's resources, causing downtime or reduced functionality.

WIRESHARK TRAFFIC ANALYSIS: HTTP FLOOD ATTACK



http.request.method == "GET"

```
No.   Time      Source          Destination    Protocol Length Info
  358 0.417687  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
  496 0.614221  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
  635 0.814440  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
  638 0.814627  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
 1313 1.818428  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
 1448 2.014258  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
 1511 2.106424  192.168.42.10   28.236.252.24  HTTP     128 GET / HTTP/1.1
 2175 3.025600  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
 2291 3.216439  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
 2996 4.241394  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
 3056 4.317213  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
 3063 4.318185  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
 3081 4.322160  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
 3086 4.326402  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
 3091 4.327404  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
 3100 4.328347  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
 3103 4.328537  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
 3108 4.329720  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
 3112 4.332211  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
 3212 4.436410  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
 3215 4.436674  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
 3218 4.437240  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
 3221 4.437726  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
 3224 4.437754  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
 3227 4.437782  192.168.32.112  28.236.252.24  HTTP     108 GET / HTTP/1.1
```

```
> Frame 4853: 108 bytes on wire (864 bits), 108 bytes captured (864 bits) on interface unk
> Ethernet II, Src: PCSSystemtec_59:19:45 (08:00:27:59:19:45), Dst: PCSSystemtec_40:54:b9
> Internet Protocol Version 4, Src: 192.168.32.112, Dst: 28.236.252.24
v Transmission Control Protocol, Src Port: 2158, Dst Port: 80, Seq: 1, Ack: 1, Len: 54
    Source Port: 2158
    Destination Port: 80
    [Stream index: 3600]
    [Stream Packet Number: 4]
  > [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 54]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 410995234
    [Next Sequence Number: 55    (relative sequence number)]
```

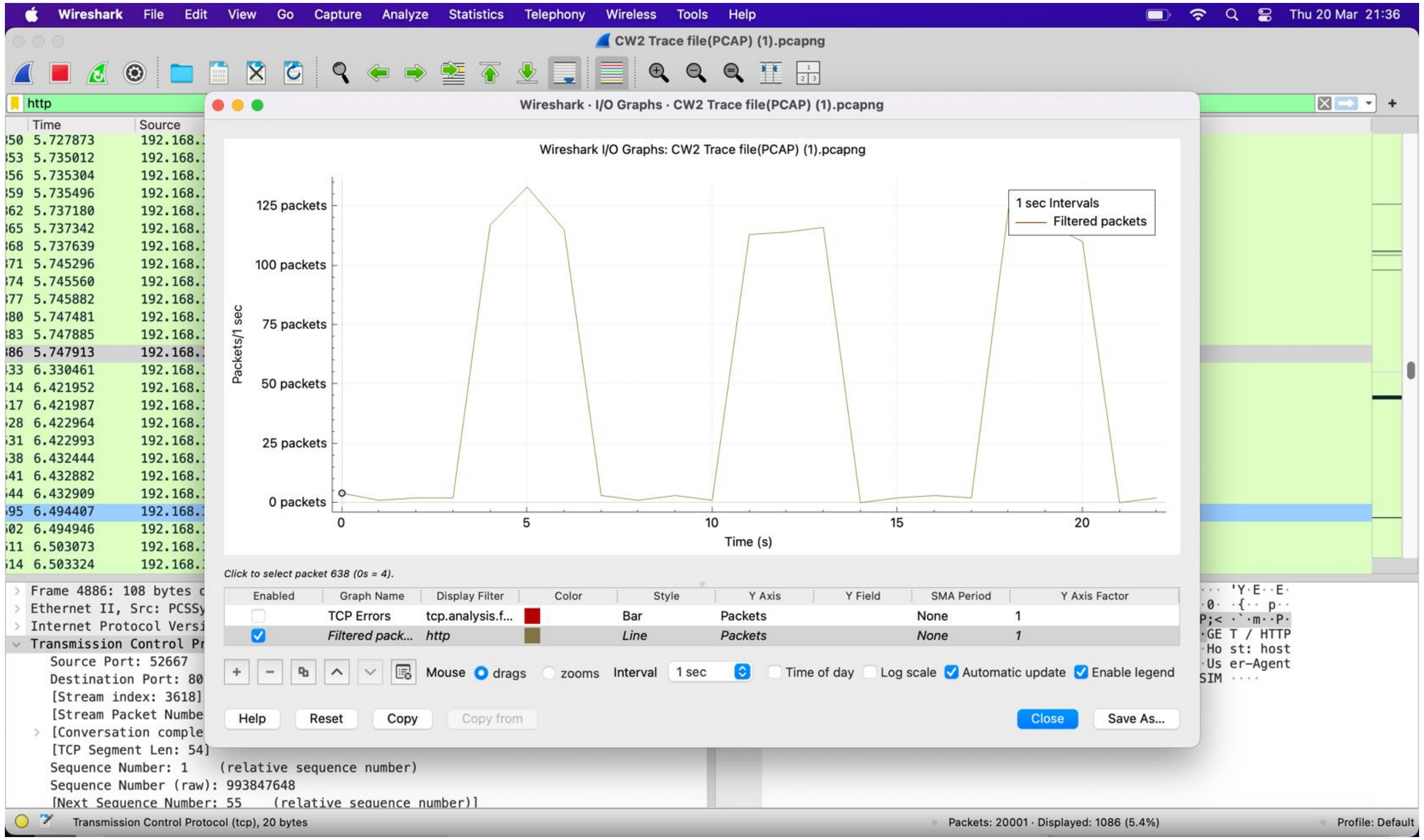
```
0000  08 00 27 40 54 b9 08 00  27 59 19 45 08 00 45 10   ··'@T··· 'Y·E··E·
0010  00 5e 00 f2 00 00 30 06  8f 7b c0 a8 20 70 1c ec   ·^····0· ·{·· p··
0020  fc 18 08 6e 00 50 18 7f  4a 22 5e 8e 11 8d 50 18   ···n·P·· J"^···P·
0030  16 d0 47 c5 00 00 47 45  54 20 2f 20 48 54 54 50   ··G···GE T / HTTP
0040  2f 31 2e 31 0d 0a 48 6f  73 74 3a 20 68 6f 73 74   /1.1··Ho st: host
0050  6e 61 6d 65 0d 0a 55 73  65 72 2d 41 67 65 6e 74   name··Us er-Agent
0060  3a 20 44 44 4f 53 49 4d  0d 0a 0d 0a               : DDOSIM ····
```

Transmission Control Protocol (tcp), 20 bytes   Packets: 20001 · Displayed: 635 (3.2%)   Profile: Default

In this attack, the network traffic shows a high volume of HTTP GET request ***(Filter: http.request.method == "GET")***. The attached Wireshark screenshot illustrates the repeated requests coming from similar **Attacker IP: 192.168.32.122**

Figure 03: Wireshark screenshot

IMPACT ON SERVER: HTTP FLOOD ATTACK


Wireshark File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Thu 20 Mar 21:36
CW2 Trace file(PCAP) (1).pcapng
http
Wireshark · I/O Graphs · CW2 Trace file(PCAP) (1).pcapng
Wireshark I/O Graphs: CW2 Trace file(PCAP) (1).pcapng
125 packets
100 packets
75 packets
50 packets
25 packets
0 packets
Packets/1 sec
0
5
10
15
20
Time (s)
1 sec Intervals
Filtered packets

```
 Time      Source
50 5.727873  192.168.
53 5.735012  192.168.
56 5.735304  192.168.
59 5.735496  192.168.
62 5.737180  192.168.
65 5.737342  192.168.
68 5.737639  192.168.
71 5.745296  192.168.
74 5.745560  192.168.
77 5.745882  192.168.
80 5.747481  192.168.
83 5.747885  192.168.
86 5.747913  192.168.
33 6.330461  192.168.
14 6.421952  192.168.
17 6.421987  192.168.
28 6.422964  192.168.
31 6.422993  192.168.
38 6.432444  192.168.
41 6.432882  192.168.
44 6.432909  192.168.
95 6.494407  192.168.
02 6.494946  192.168.
11 6.503073  192.168.
14 6.503324  192.168.
```

```
> Frame 4886: 108 bytes c
> Ethernet II, Src: PCSSy
> Internet Protocol Versi
v Transmission Control Pr
    Source Port: 52667
    Destination Port: 80
    [Stream index: 3618]
    [Stream Packet Numbe
  > [Conversation comple
    [TCP Segment Len: 54]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 993847648
    [Next Sequence Number: 55    (relative sequence number)]
```

Click to select packet 638 (0s = 4).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☐	TCP Errors	tcp.analysis.f...	🟥	Bar	Packets		None	1
☑	*Filtered pack...*	*http*	🟫	*Line*	*Packets*		*None*	*1*

Mouse ● drags ○ zooms Interval 1 sec ☐ Time of day ☐ Log scale ☑ Automatic update ☑ Enable legend

Transmission Control Protocol (tcp), 20 bytes

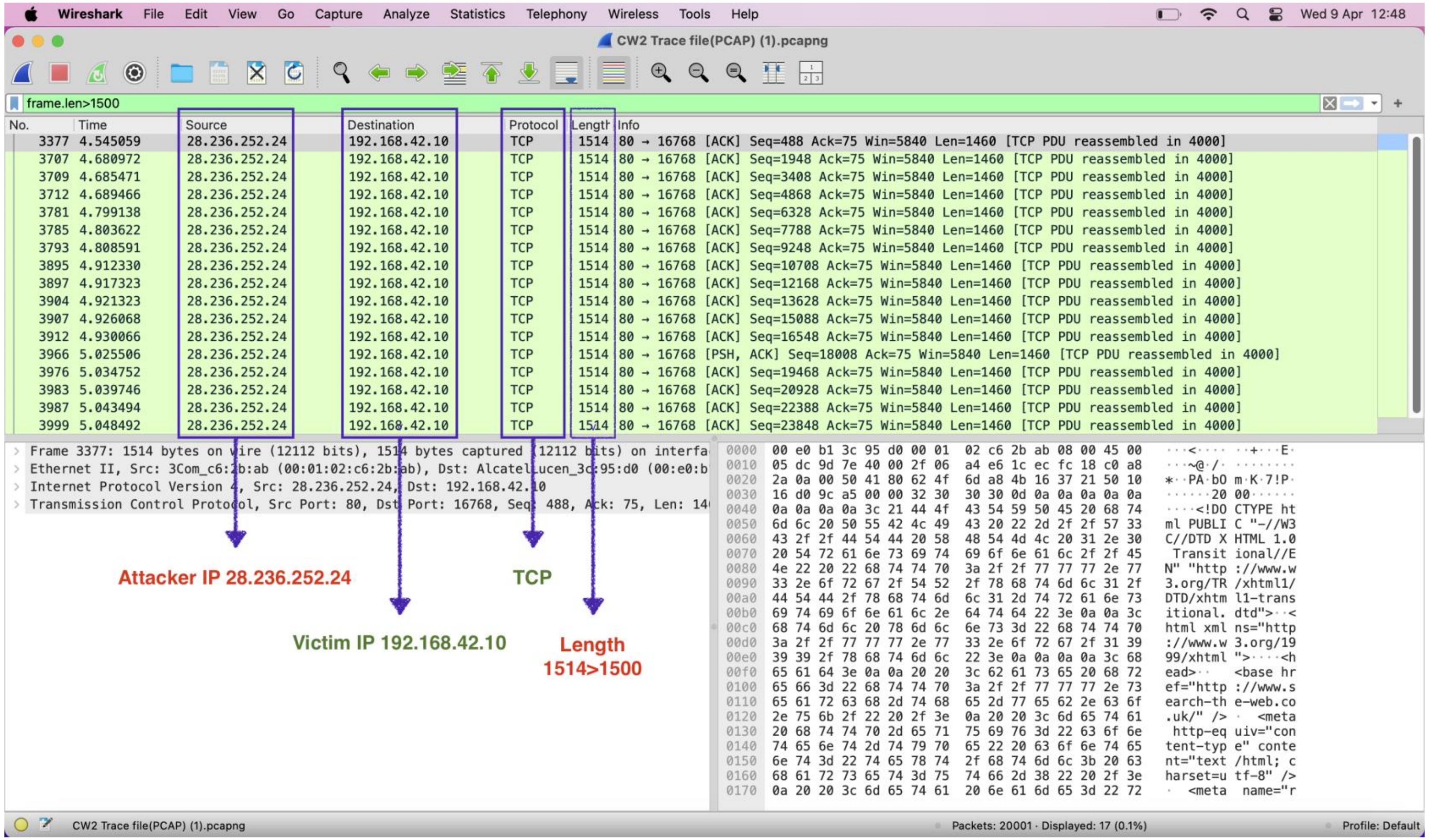
Packets: 20001 · Displayed: 1086 (5.4%)

Profile: Default

When an HTTP Flood attack occurs, the web server’s CPU and memory are consumed by handling the malicious requests. As a result, the server becomes slow and unresponsive to legitimate traffic, potentially leading to an outage or degraded service performance.

Figure 04: Wireshark I/O Graph (http)

OVERVIEW AND WIRESHARK TRAFFIC ANALYSIS: FRAGMENTATION ATTACK


Wireshark File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Wed 9 Apr 12:48
CW2 Trace file(PCAP) (1).pcapng
Attacker IP 28.236.252.24
TCP
Victim IP 192.168.42.10
Length
1514>1500
CW2 Trace file(PCAP) (1).pcapng
Packets: 20001 · Displayed: 17 (0.1%)
Profile: Default

frame.len>1500

```
No.   Time      Source          Destination     Protocol Length Info
 3377 4.545059  28.236.252.24   192.168.42.10   TCP      1514   80 → 16768 [ACK] Seq=488 Ack=75 Win=5840 Len=1460 [TCP PDU reassembled in 4000]
 3707 4.680972  28.236.252.24   192.168.42.10   TCP      1514   80 → 16768 [ACK] Seq=1948 Ack=75 Win=5840 Len=1460 [TCP PDU reassembled in 4000]
 3709 4.685471  28.236.252.24   192.168.42.10   TCP      1514   80 → 16768 [ACK] Seq=3408 Ack=75 Win=5840 Len=1460 [TCP PDU reassembled in 4000]
 3712 4.689466  28.236.252.24   192.168.42.10   TCP      1514   80 → 16768 [ACK] Seq=4868 Ack=75 Win=5840 Len=1460 [TCP PDU reassembled in 4000]
 3781 4.799138  28.236.252.24   192.168.42.10   TCP      1514   80 → 16768 [ACK] Seq=6328 Ack=75 Win=5840 Len=1460 [TCP PDU reassembled in 4000]
 3785 4.803622  28.236.252.24   192.168.42.10   TCP      1514   80 → 16768 [ACK] Seq=7788 Ack=75 Win=5840 Len=1460 [TCP PDU reassembled in 4000]
 3793 4.808591  28.236.252.24   192.168.42.10   TCP      1514   80 → 16768 [ACK] Seq=9248 Ack=75 Win=5840 Len=1460 [TCP PDU reassembled in 4000]
 3895 4.912330  28.236.252.24   192.168.42.10   TCP      1514   80 → 16768 [ACK] Seq=10708 Ack=75 Win=5840 Len=1460 [TCP PDU reassembled in 4000]
 3897 4.917323  28.236.252.24   192.168.42.10   TCP      1514   80 → 16768 [ACK] Seq=12168 Ack=75 Win=5840 Len=1460 [TCP PDU reassembled in 4000]
 3904 4.921323  28.236.252.24   192.168.42.10   TCP      1514   80 → 16768 [ACK] Seq=13628 Ack=75 Win=5840 Len=1460 [TCP PDU reassembled in 4000]
 3907 4.926068  28.236.252.24   192.168.42.10   TCP      1514   80 → 16768 [ACK] Seq=15088 Ack=75 Win=5840 Len=1460 [TCP PDU reassembled in 4000]
 3912 4.930066  28.236.252.24   192.168.42.10   TCP      1514   80 → 16768 [ACK] Seq=16548 Ack=75 Win=5840 Len=1460 [TCP PDU reassembled in 4000]
 3966 5.025506  28.236.252.24   192.168.42.10   TCP      1514   80 → 16768 [PSH, ACK] Seq=18008 Ack=75 Win=5840 Len=1460 [TCP PDU reassembled in 4000]
 3976 5.034752  28.236.252.24   192.168.42.10   TCP      1514   80 → 16768 [ACK] Seq=19468 Ack=75 Win=5840 Len=1460 [TCP PDU reassembled in 4000]
 3983 5.039746  28.236.252.24   192.168.42.10   TCP      1514   80 → 16768 [ACK] Seq=20928 Ack=75 Win=5840 Len=1460 [TCP PDU reassembled in 4000]
 3987 5.043494  28.236.252.24   192.168.42.10   TCP      1514   80 → 16768 [ACK] Seq=22388 Ack=75 Win=5840 Len=1460 [TCP PDU reassembled in 4000]
 3999 5.048492  28.236.252.24   192.168.42.10   TCP      1514   80 → 16768 [ACK] Seq=23848 Ack=75 Win=5840 Len=1460 [TCP PDU reassembled in 4000]
```

```
> Frame 3377: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interfa
> Ethernet II, Src: 3Com_c6:2b:ab (00:01:02:c6:2b:ab), Dst: AlcatelLucen_3c:95:d0 (00:e0:b
> Internet Protocol Version 4, Src: 28.236.252.24, Dst: 192.168.42.10
> Transmission Control Protocol, Src Port: 80, Dst Port: 16768, Seq: 488, Ack: 75, Len: 14
```

```
0000  00 e0 b1 3c 95 d0 00 01  02 c6 2b ab 08 00 45 00   ···<···· ··+···E·
0010  05 dc 9d 7e 40 00 2f 06  a4 e6 1c ec fc 18 c0 a8   ···~@·/· ········
0020  2a 0a 00 50 41 80 62 4f  6d a8 4b 16 37 21 50 10   *··PA·bO m·K·7!P·
0030  16 d0 9c a5 00 00 32 30  30 30 0d 0a 0a 0a 0a 0a   ······20 00······
0040  0a 0a 0a 0a 3c 21 44 4f  43 54 59 50 45 20 68 74   ····<!DO CTYPE ht
0050  6d 6c 20 50 55 42 4c 49  43 20 22 2d 2f 2f 57 33   ml PUBLI C "-//W3
0060  43 2f 2f 44 54 44 20 58  48 54 4d 4c 20 31 2e 30   C//DTD X HTML 1.0
0070  20 54 72 61 6e 73 69 74  69 6f 6e 61 6c 2f 2f 45    Transit ional//E
0080  4e 22 20 22 68 74 74 70  3a 2f 2f 77 77 77 2e 77   N" "http ://www.w
0090  33 2e 6f 72 67 2f 54 52  2f 78 68 74 6d 6c 31 2f   3.org/TR /xhtml1/
00a0  44 54 44 2f 78 68 74 6d  6c 31 2d 74 72 61 6e 73   DTD/xhtm l1-trans
00b0  69 74 69 6f 6e 61 6c 2e  64 74 64 22 3e 0a 0a 3c   itional. dtd">··<
00c0  68 74 6d 6c 20 78 6d 6c  6e 73 3d 22 68 74 74 70   html xml ns="http
00d0  3a 2f 2f 77 77 77 2e 77  33 2e 6f 72 67 2f 31 39   ://www.w 3.org/19
00e0  39 39 2f 78 68 74 6d 6c  22 3e 0a 0a 0a 0a 3c 68   99/xhtml ">····<h
00f0  65 61 64 3e 0a 0a 20 20  3c 62 61 73 65 20 68 72   ead>··   <base hr
0100  65 66 3d 22 68 74 74 70  3a 2f 2f 77 77 77 2e 73   ef="http ://www.s
0110  65 61 72 63 68 2d 74 68  65 2d 77 65 62 2e 63 6f   earch-th e-web.co
0120  2e 75 6b 2f 22 20 2f 3e  0a 20 20 3c 6d 65 74 61   .uk/" /> ·  <meta
0130  20 68 74 74 70 2d 65 71  75 69 76 3d 22 63 6f 6e    http-eq uiv="con
0140  74 65 6e 74 2d 74 79 70  65 22 20 63 6f 6e 74 65   tent-typ e" conte
0150  6e 74 3d 22 74 65 78 74  2f 68 74 6d 6c 3b 20 63   nt="text /html; c
0160  68 61 72 73 65 74 3d 75  74 66 2d 38 22 20 2f 3e   harset=u tf-8" />
0170  0a 20 20 3c 6d 65 74 61  20 6e 61 6d 65 3d 22 72   ·  <meta  name="r
```

Overview: A fragmentation attack involves breaking malicious packets into small fragments to evade detection by firewalls or intrusion detection systems. These fragments are reassembled at the target, allowing the attack to succeed unnoticed. It's often used in evasion and DoS attacks.

The Wireshark capture indicates a potential fragmentation attack. The **attacker IP 28.236.252.24** is sending multiple TCP packets to **victim 192.168.42.10**, each with a **size of 1514 bytes exceeding the typical MTU of 1500**. This suggests an attempt to evade detection or overload the victim's system by using oversized packets, a common tactic in evasion or DoS attacks.

Figure 05: Wireshark screenshot

MITIGATION FOR DOS (HTTP FLOOD, SYN FLOOD, AND FRAGMENTATION ATTACKS) ATTACKS:

To mitigate HTTP Flood, SYN Flood, and Fragmentation Attacks, implement robust defences. For HTTP Floods, use rate-limiting, IP filtering, and Web Application Firewalls (WAFs) to control traffic. For SYN Floods, deploy SYN cookies, delayed binding, and connection limits to manage half-open connections. To prevent Fragmentation Attacks, utilize deep packet inspection, IDS/IPS systems, and firewall rules. A layered security approach with regular updates, anomaly detection, and monitoring is key to preventing these attacks.

HTTP Flood Attacks:

- Use WAFs to filter malicious traffic.
- Apply rate-limiting to prevent overload.
- Implement CAPTCHA to distinguish between users and bots.

SYN Flood Attacks:

- Enable SYN cookies to handle half-open connections.
- Set connection limits and timeouts to prevent resource exhaustion.
- Use IDS/IPS to detect and block SYN flood activity.

Fragmentation Attacks:

- Use firewalls and IDS/IPS for packet inspection and reassembly.
- Apply deep packet inspection (DPI) to detect malicious fragments.
- Implement rate-limiting on fragmented packets.

REFERENCES

Books:

- Forouzan, B.A. (2021) *Data Communications and Networking*. 6th edn. New York: McGraw-Hill.
- Kurose, J.F. and Ross, K.W. (2021) *Computer Networking: A Top-Down Approach*. 8th edn. Boston: Pearson.
- Stallings, W. (2022) *Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud*. Boston: Pearson.
- Tanenbaum, A.S. and Wetherall, D.J. (2021) *Computer Networks*. 6th edn. Boston: Pearson.
- **Journal Articles:**
- Mirkovic, J. and Reiher, P. (2021) 'A taxonomy of DDoS attack and DDoS defence mechanisms', *ACM Computing Surveys*, 53(4), pp. 1-35.

Web Sources:

- Cisco (2024) *Understanding IP fragmentation attacks*. Available at: https://www.cisco.com/c/en/us/about/security-center/ip-fragmentation-attacks.html [Accessed 11 April 2025].
- Firewall.cx (n.d.) *How to Perform TCP SYN Flood DoS Attack & Detect it with Wireshark*. Available at: https://www.firewall.cx/tools-tips-reviews/network-protocol-analyzers/performing-tcp-syn-flood-attack-and-detecting-it-with-wireshark.html [Accessed 20 March 2025].
- IBM (n.d.) *TCP/IP Protocols*. Available at: https://www.ibm.com/docs/fi/ssw_aix_72/network/tcpip_protocols.html[Accessed 21 March 2025].
- Oracle (n.d.) *Introducing the TCP/IP Protocol Suite - System Administration Guide*. Available at: https://docs.oracle.com/cd/E18752_01/html/816-4554/ipov-6.html [Accessed 23 March 2025].