Understanding the TCP Protocol Three-way Handshake: Working, Role, and Security Implications

Author: Ravipati Srinivasa Kalyan

Date: May 26, 2025

Abstract

The Transmission Control Protocol (TCP) is a cornerstone of modern networking, providing reliable, ordered, and error-checked delivery of data between applications across IP networks. This research paper explores the fundamental workings of TCP, with particular emphasis on the three-way handshake mechanism used to establish reliable connections between hosts. The handshake process consisting of SYN, SYN-ACK, and ACK messages are critical to initiating communication sessions in a stateful and secure manner. While TCP is essential for ensuring data integrity and flow control, its initial handshake also introduces potential vulnerabilities that can be exploited in various cybersecurity attacks, such as SYN flood and man-in-the-middle (MitM) attacks.

In the context of cybersecurity, understanding the intricacies of TCP's connection establishment process is vital for designing secure network architectures and for implementing defensive mechanisms like firewalls, intrusion detection systems (IDS), and rate limiting. This paper examines the role of TCP in secure communications, evaluates known vulnerabilities associated with the handshake process, and reviews current mitigation strategies to harden TCP against exploitation. By combining theoretical analysis with practical insights, this research highlights the delicate balance between network efficiency and security, and underscores the need for continued innovation in protocol design and cyber defence techniques.

Keywords: TCP, IDS, rate limiting, three-way Handshake, SYN, SYN-ACK, ACK, MITM, SYN flood attack.

1. Introduction

In the realm of computer networking, the Transmission Control Protocol (TCP) stands out as one of the most fundamental and widely used communication protocols. It operates at the transport layer of the TCP/IP protocol suite, which is the foundation of data exchange over the internet and most private networks. TCP is designed to provide reliable, connection-oriented communication between devices, ensuring that data sent from one host arrives intact and in the correct order at its destination.

Unlike simpler protocols that merely send data without confirmation, TCP establishes a reliable communication channel before any actual data transmission occurs. It handles essential tasks such as error detection, retransmission of lost packets, flow control, and congestion management, making it suitable for applications where data integrity and delivery accuracy are critical. Common applications that rely on TCP include web browsing (HTTP/HTTPS), email (SMTP, IMAP), and file transfers (FTP).

A key feature that distinguishes TCP from other protocols is its connection establishment process, commonly known as the three-way handshake. This handshake process ensures that both parties are ready for communication and have agreed on initial sequence numbers, setting the stage for a reliable data transfer session.

While the three-way handshake plays a crucial role in ensuring proper connection setup, it also presents potential vulnerabilities from a cybersecurity perspective. Attackers can exploit this mechanism to launch various types of network-based attacks, such as SYN flood attacks, which can overwhelm servers and degrade network performance. As such, understanding the principles behind TCP and its handshake mechanism is not only important from a networking standpoint but also essential for identifying and mitigating security threats.

Understanding this process is crucial for anyone working in networking, cybersecurity, or system administration, as it not only aids in performance tuning and troubleshooting but also in identifying and mitigating potential security threats that target the handshake process. This paper explores the structure and purpose of TCP and its three-way handshake, highlighting their importance in both reliable networking and cybersecurity. By examining how this foundational protocol works and the challenges it presents, we can better appreciate its role in modern digital communication and the strategies needed to protect it.

2. Transmission Control Protocol Overview

The TCP/IP model is divided into four layers:

1. Application layer
2. Transport Layer
3. Internet Layer
4. Data-Link Layer

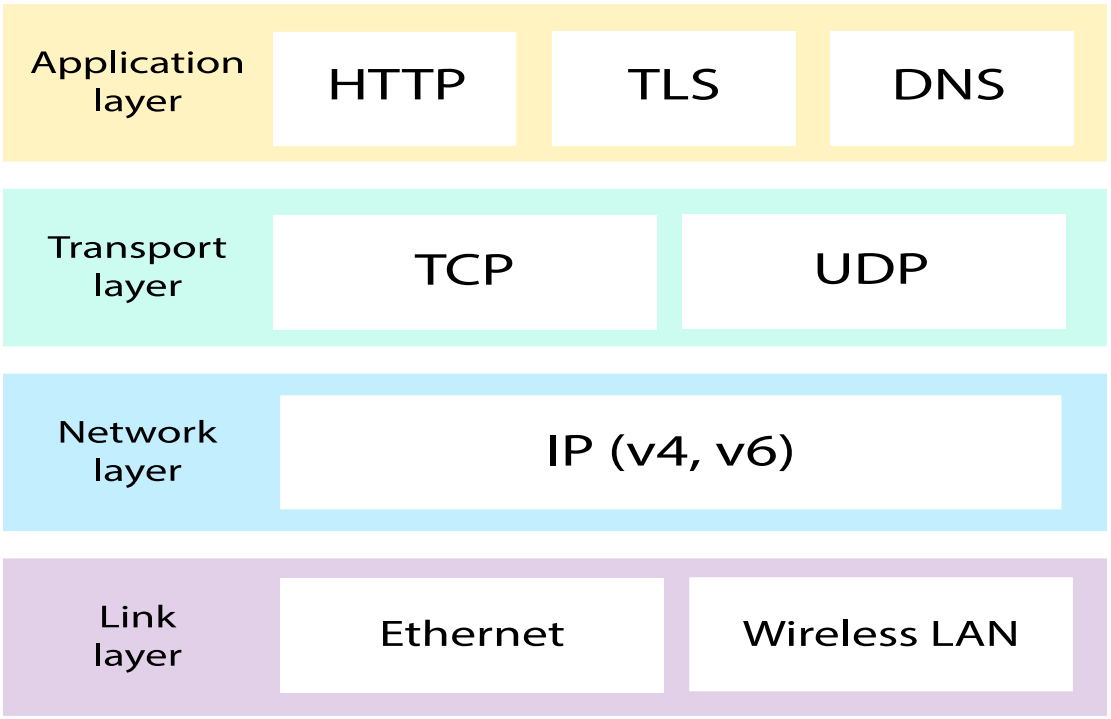

Figure-1. TCP/IP Reference Model with Protocols

The transport layer is responsible for ensuring reliable data transmission between devices. This is where TCP (Transmission Control Protocol) operates, providing connection-oriented communication that guarantees ordered and error-free delivery of data. Another protocol at this layer is UDP (User Datagram Protocol), which offers faster, connectionless transmission without reliability guarantees. The transport layer handles segmentation, flow control, and error correction.

Within this model, TCP operates at the transport layer, playing a critical role in managing data transmission. It establishes connections between hosts using the three-way handshake, maintains reliable communication sessions, and ensures that data is delivered in the correct order and without errors. By functioning in coordination with other layers, TCP contributes to the overall goal of end-to-end communication in IP-based networks.

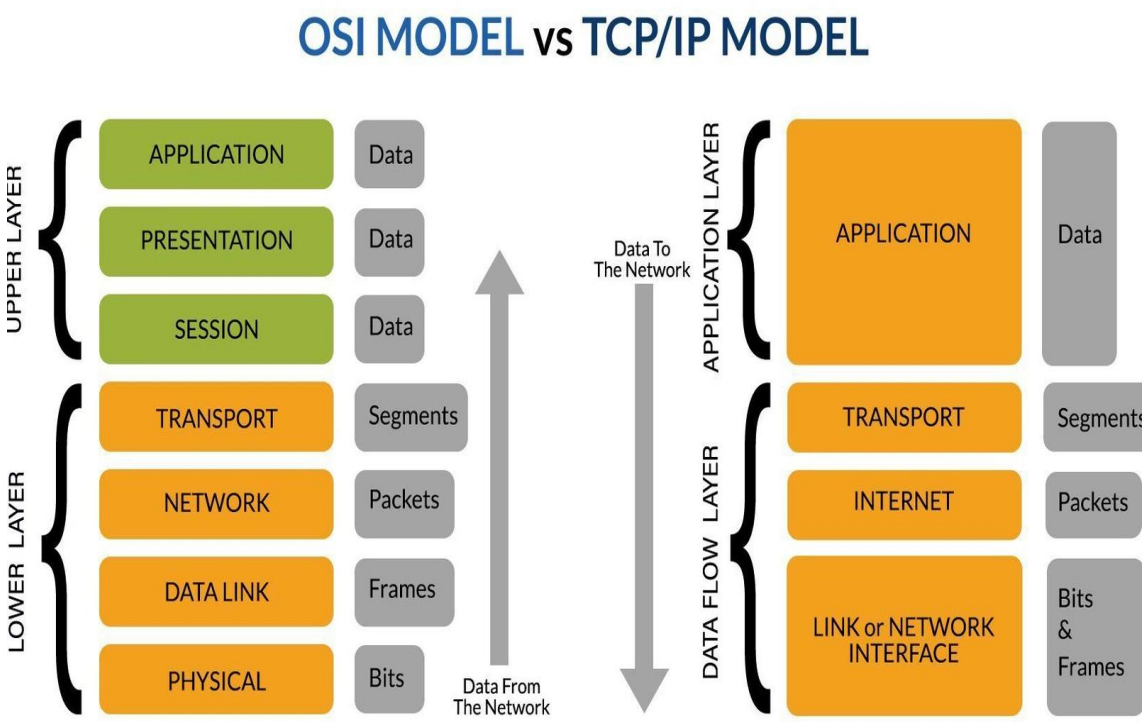

Figure-2. OSI to TCP/IP Model

The Transmission Control Protocol (TCP) operates at the transport layer of the Internet protocol suite, providing a reliable communication service between application processes and the underlying network. It serves as an intermediary between application programs and the Internet Protocol (IP), offering a host-to-host data transfer mechanism without requiring the application to handle the complexities of network transmission such as IP fragmentation or media-specific limitations like the maximum transmission unit (MTU).

TCP is responsible for managing the connection setup and teardown processes, typically through a network socket interface, and ensures that the connection appears seamless and reliable from the perspective of the application. It handles the critical tasks of initiating communication (via the three-way handshake), sequencing data, and ensuring its integrity during transmission.

At the lower layers of the networking stack, unpredictable behaviours such as packet loss, duplication, or reordering can occur due to factors like congestion or varying network paths. TCP mitigates these issues by detecting transmission errors, reordering out-of-sequence packets, requesting retransmissions for lost segments, and managing traffic flow to avoid overwhelming the network. In cases where data cannot be successfully delivered despite these efforts, TCP informs the sending host of the failure.

Once all data segments are received and correctly ordered, TCP reassembles them and delivers the complete stream to the appropriate application. This process effectively hides the complexity of unreliable network conditions from the application, allowing developers to focus on functionality without dealing with low-level transmission concerns. In this way, TCP provides a robust and reliable channel for application-level communication over inherently unreliable IP-based networks.

Advantages of TCP

1. Reliable Data Transmission

TCP ensures that all data sent from the source reaches the destination accurately and in the correct order. It uses mechanisms like acknowledgments (ACKs) and retransmissions to detect and correct errors during transmission.

2. Connection-Oriented Protocol

TCP establishes a dedicated connection between sender and receiver using the three-way handshake before transmitting data. This connection remains active throughout the session, ensuring stable communication**.**

3. Error Detection and Recovery

TCP includes built-in error-checking mechanisms using checksums and automatic retransmission of lost or corrupted packets, improving data integrity.

4. Flow Control

TCP implements flow control using the sliding window protocol, which ensures that the sender does not overwhelm the receiver with more data than it can handle at a time.

5. Congestion Control

TCP actively monitors network conditions and adjusts its data transmission rate to prevent congestion, ensuring efficient use of network resources.

6. Data Sequencing

Packets may arrive out of order due to network routing, but TCP reorders the data segments using sequence numbers before delivering them to the application.

Disadvantages of TCP

1. Higher Overhead

TCP introduces additional processing overhead due to features like error checking, acknowledgments, sequencing, and connection management. This can consume more CPU and memory resources, especially in high-traffic environments.

2. Slower Transmission Speed

Due to its focus on reliability and accuracy, TCP is generally slower than connectionless protocols like UDP. It waits for acknowledgments and retransmits lost packets, which can delay overall data transfer.

3. Vulnerability to Attacks

TCP is vulnerable to specific types of cybersecurity threats, such as SYN flood attacks, where an attacker exploits the handshake process to exhaust server resources, potentially leading to denial-of-service (DoS) conditions.

4. Complex Implementation

TCP's robust feature set makes it more complex to implement and maintain compared to simpler protocols. This complexity can lead to more bugs and security issues if not handled properly.

TCP Header Format

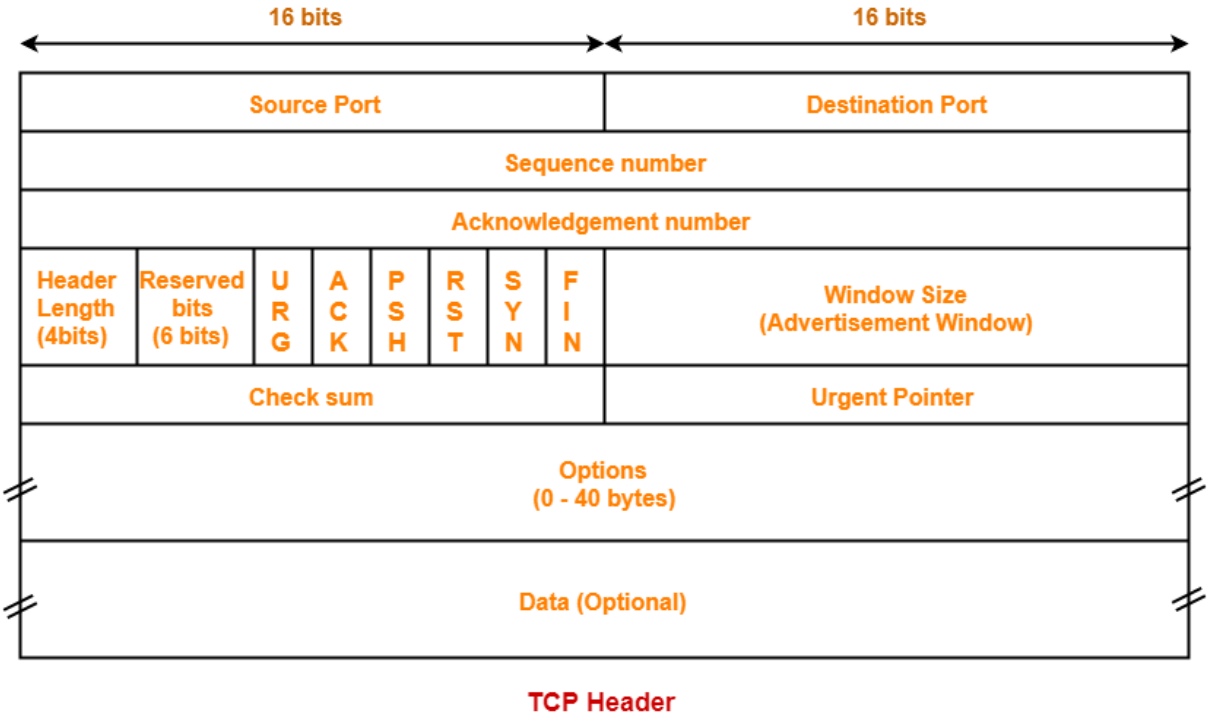

Figure-3. TCP Header Format

The TCP header is a critical part of every TCP segment. It contains various fields that enable TCP to provide reliable, ordered, and error-checked communication between hosts. The standard TCP header is 20 bytes in size but can be extended up to 60 bytes with optional fields. Below is an overview of the key fields in a TCP header:

Source Port (16 bits) Identifies the sending application's port number. It helps the receiving system know which application on the sender's side is communicating.

Destination Port (16 bits) Identifies the receiving application's port number. This helps route the data to the correct process on the destination machine.

Sequence Number (32 bits) Used to track the order of data. It represents the position of the first byte of data in this segment, helping with reordering and loss recovery.

Acknowledgment Number (32 bits) If the ACK flag is set, this field contains the next expected byte from the sender. It confirms successful receipt of data.

Data Offset (4 bits) Also known as the header length, this field specifies the size of the TCP header in 32-bit words. It tells the receiver where the data begins.

Reserved (3 bits) Reserved for future use and always set to zero.

Flags (9 bits)

Also called control bits, these are used to manage the connection:

URG – Urgent pointer field is significant

ACK – Acknowledgment field is significant

PSH – Push function

RST – Reset the connection

SYN – Synchronize sequence numbers (used in connection setup)

FIN – No more data from sender (used in connection termination)

Window Size (16 bits) Specifies the amount of data (in bytes) that the receiver is currently willing to accept, enabling flow control.

Checksum (16 bits) Used for error-checking the TCP header and data. It ensures data integrity during transmission.

Urgent Pointer (16 bits) Points to the end of urgent data if the URG flag is set. Rarely used in modern applications.

Options (variable) Provides additional functionalities, such as Maximum Segment Size (MSS), window scaling, and selective acknowledgments. This field is optional and varies in length.

Padding Extra bits are added to make the header length a multiple of 32 bits (4 bytes).

3. Working Of TCP Three-way Handshake

The TCP three-way Handshake is the process of establishing a connection Between the client and the server with the help of TCP (Transmission Control Protocol).

Client IP: 10.0.2.15

Server Ip: 45.33.32.156 (scanme.nmap.org)

Nmap TCP scan was performed to establish a TCP connection with Nmap server and using Wireshark for analyzing the packets from client to server.

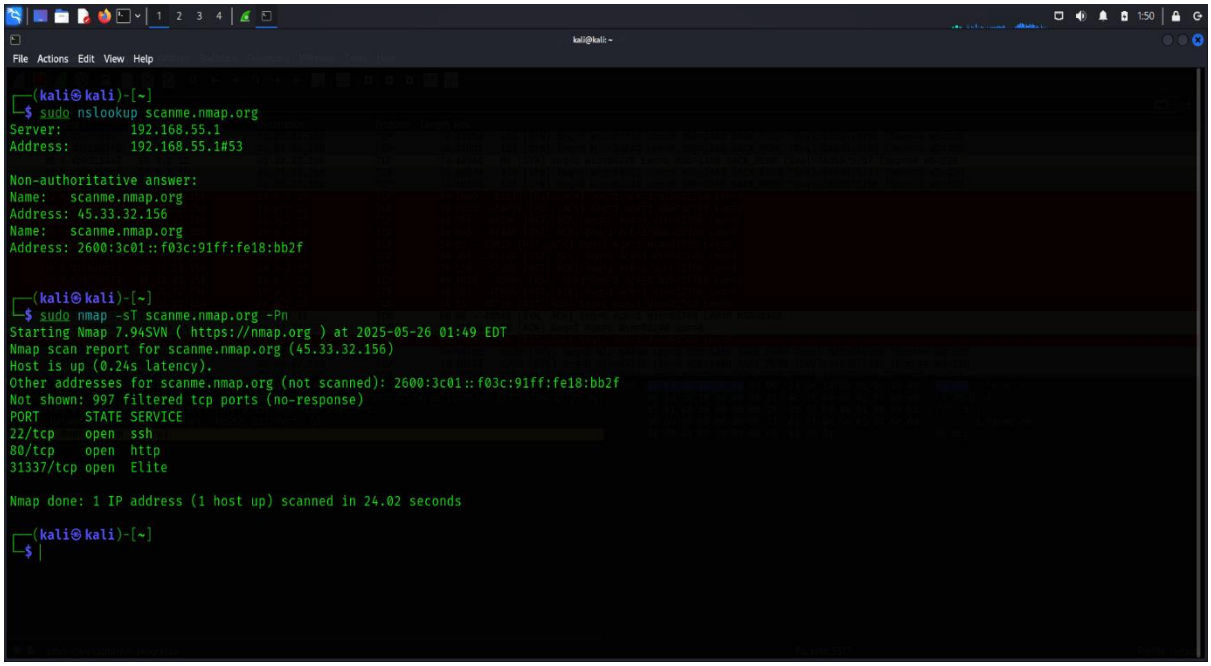

Figure-4: Nmap TCP scan on nmap.org

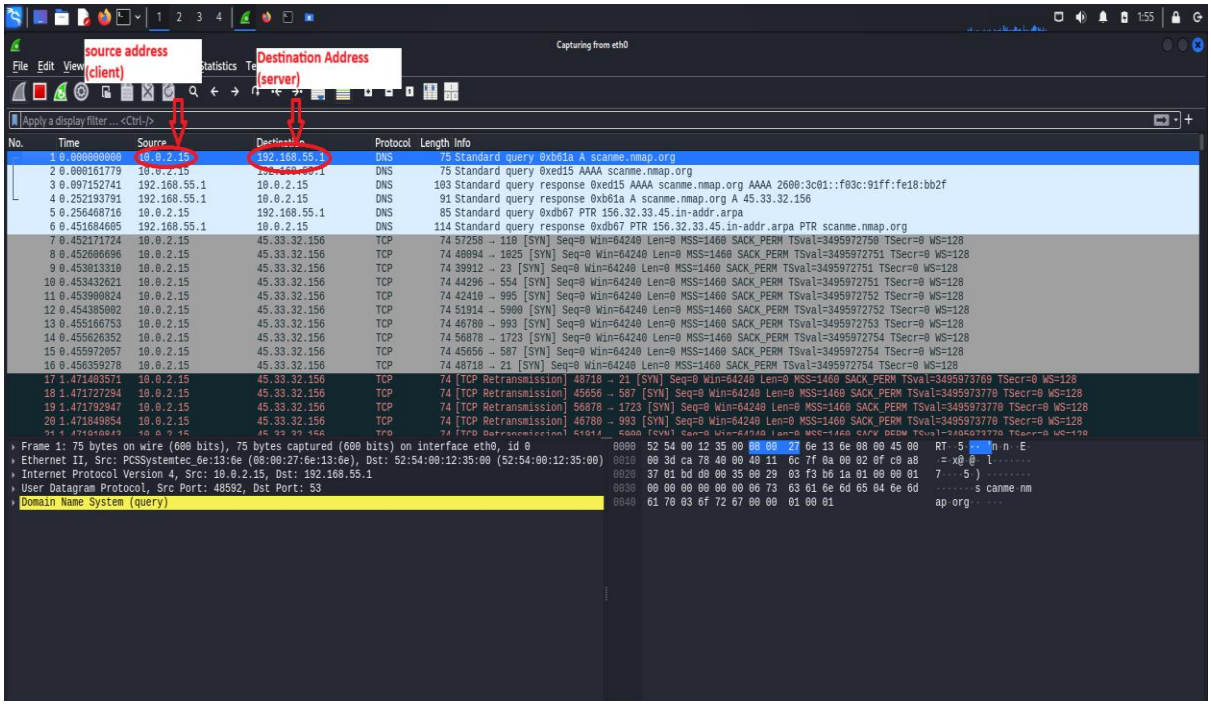

Figure-5. Using Wireshark to analyse Packet

Step-1:SYN(Synchronize)

The client initiates the connection by sending a SYN packet and ISN to the server. This packet includes an initial sequence number, which marks the beginning of the data stream the client will send.

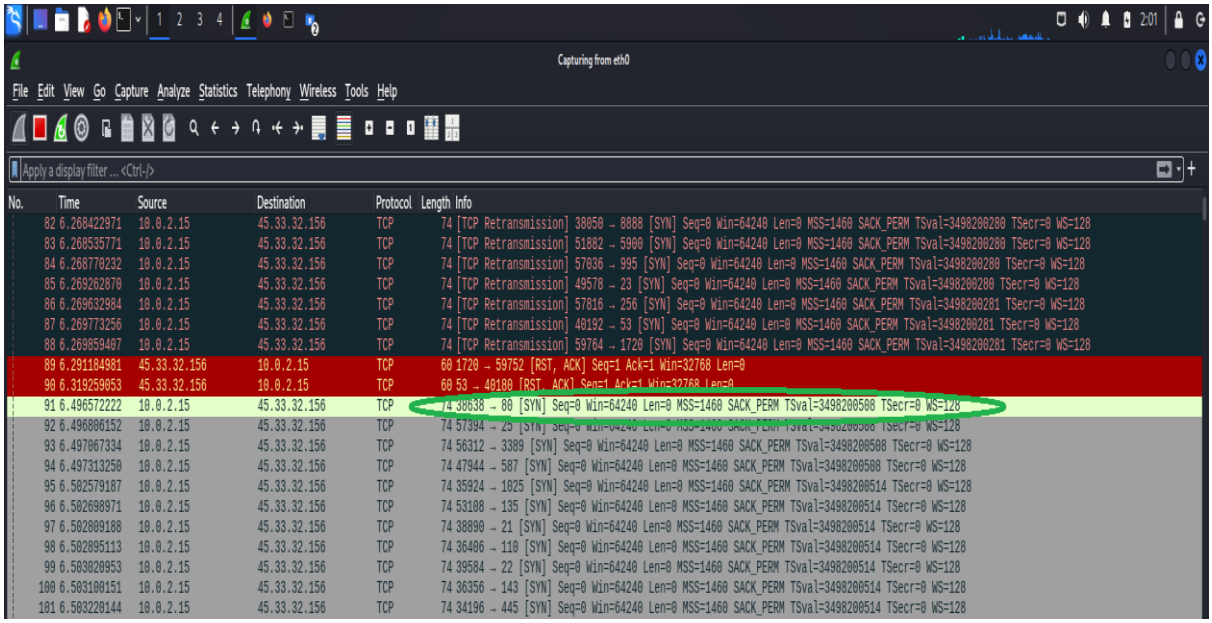

Figure-6. Tracing SYN Packet from Client to Destination

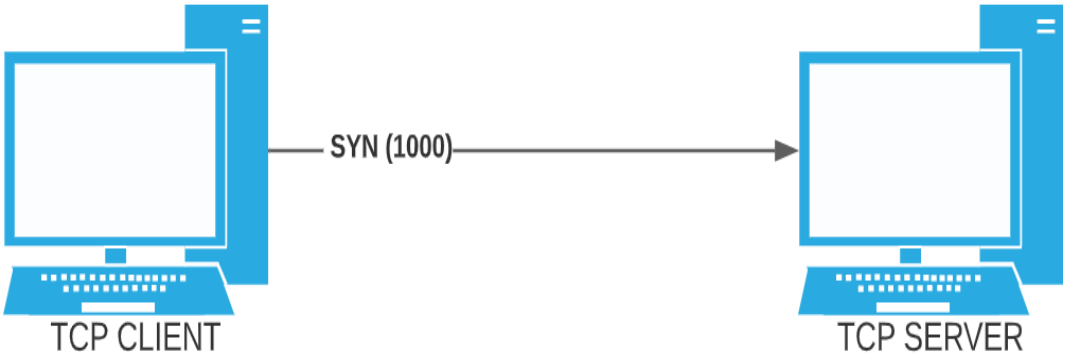

Figure-7. Client sending SYN packet to Server

Step-2: SYN-ACK (Synchronize-Acknowledge)

After receiving the SYN (synchronize) packet from the client in Step 1, the server processes the request to establish a TCP connection. If the server is ready to accept the connection, it responds with a SYN-ACK packet. This packet serves two purposes:

SYN (synchronize) – The server generates its own initial sequence number (ISN) and includes it in the SYN-ACK. This ISN will be used to track the bytes the server sends during the session.

ACK (acknowledgment) – The server also acknowledges the client's initial SYN by setting the ACK flag and including an acknowledgment number, which is the client's sequence

number plus one. This tells the client: “I received your request, and here’s my sequence number so we can begin communication.”

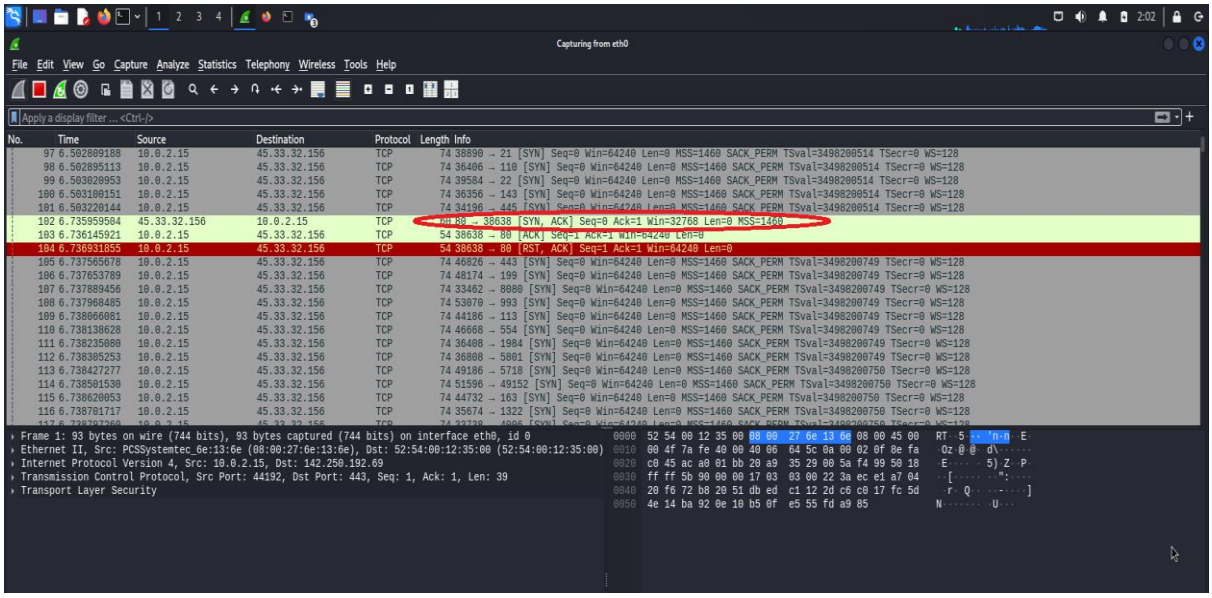

Figure-8. Using Wireshark to analyse TCP SYN/ACK sending to Client

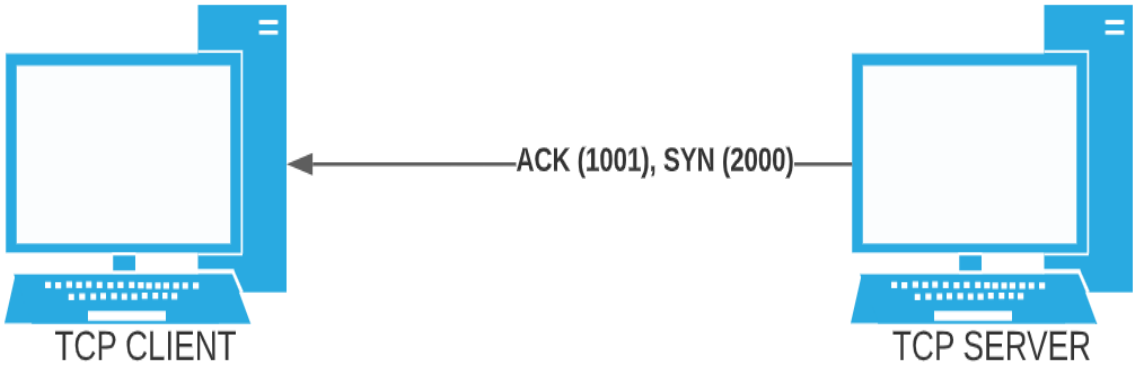

Figure-9. Client sending ACK packet to Server

Step-3: ACK (Acknowledge)

In the final step of the TCP three-way handshake, the client responds to the server's SYN-ACK by sending an ACK packet. This step completes the handshake and officially establishes a reliable, bidirectional TCP connection between the client and the server. Once the server receives this ACK from the client, the connection is fully established. Both client and server can now begin sending and receiving data over this reliable TCP channel.

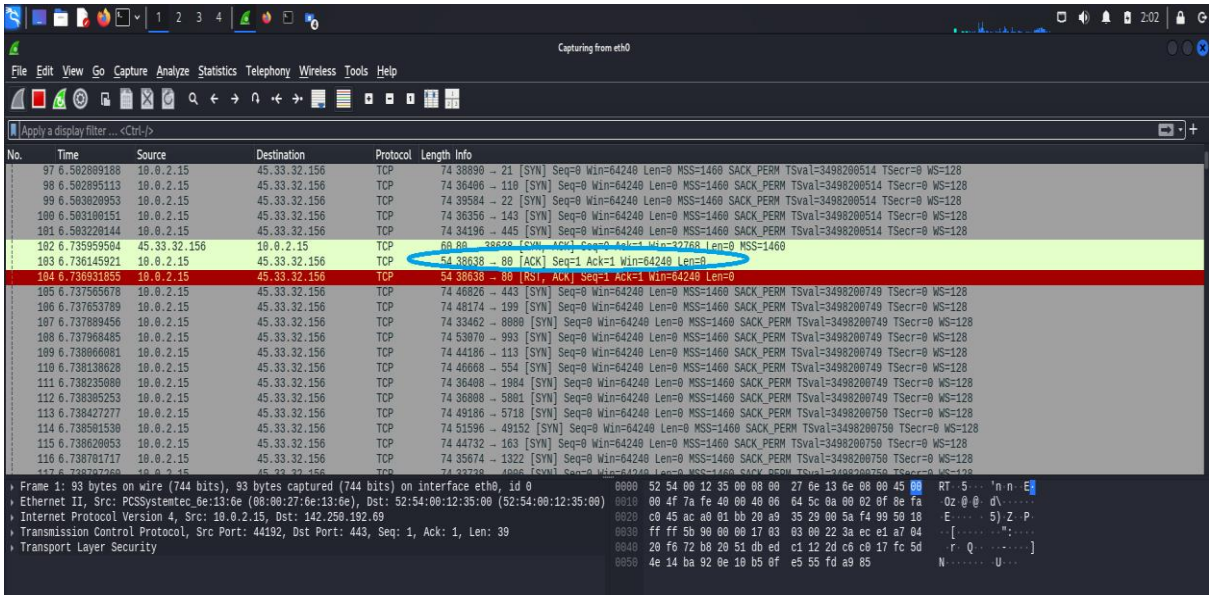

Figure-10. Using Wireshark to analyse TCP ACK sending to Server

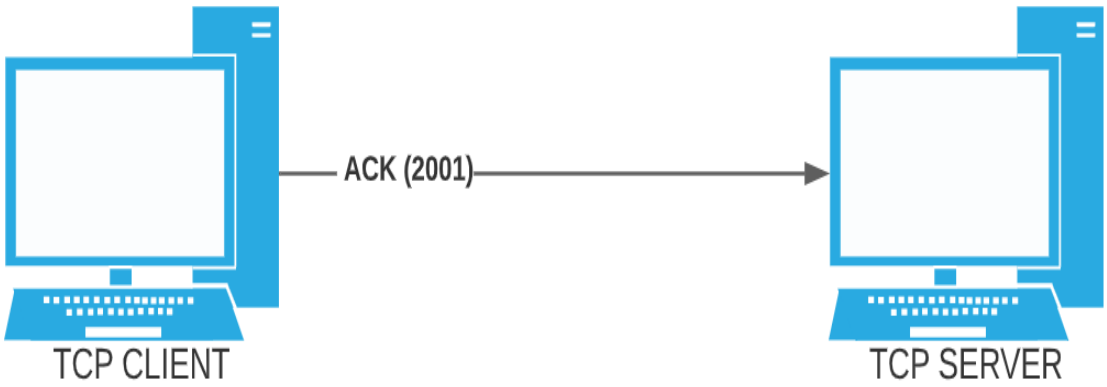

Figure-11. Client sending ACK packet to Server

4. Example Connection

- Client sent (Step 1): SYN, Seq = 1000
- Server replied (Step 2): SYN-ACK, Seq = 5000, Ack = 1001
- Client replies (Step 3): ACK, Seq = 1001, Ack = 5001

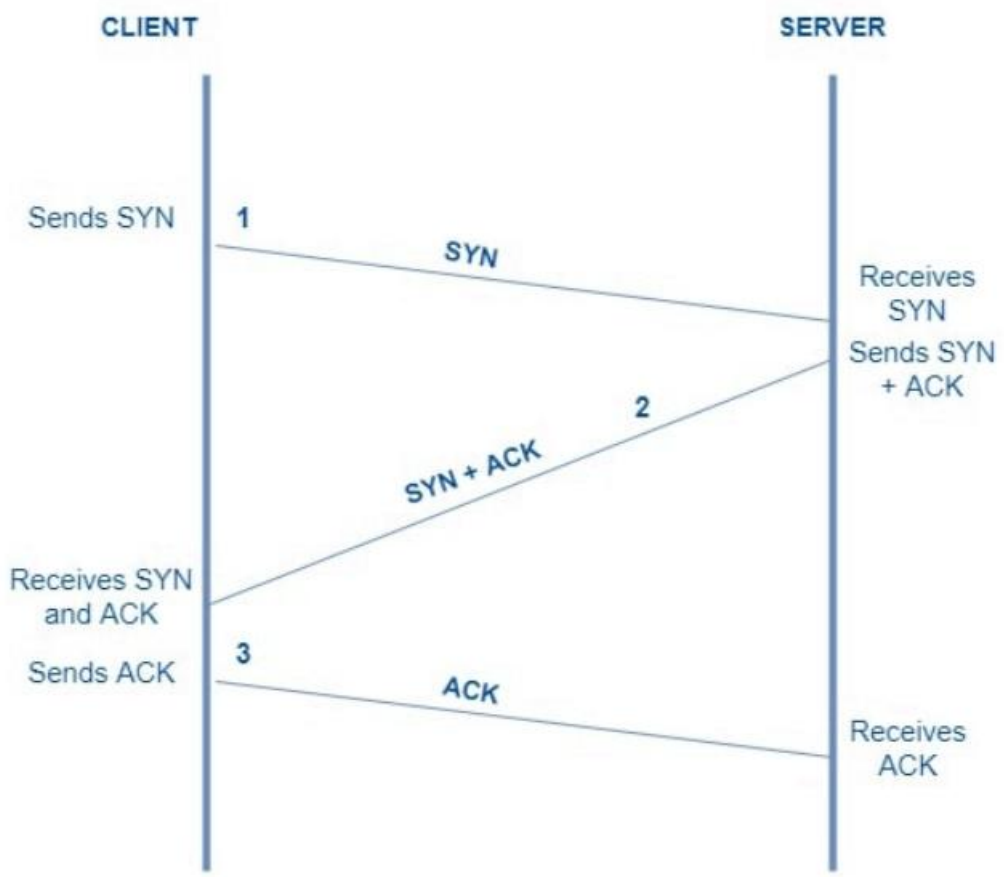

Figure-12. TCP Three-way Handshake

5. Purpose and Importance

Connection Establishment: Ensures that both client and server are ready to communicate. **Sequence Synchronization:** Sets up sequence numbers to ensure ordered data delivery. **Reliability:** Confirms that both ends have successfully received each other's messages before starting communication.

6. Cyber Threat

The most dangerous threat in the TCP connection is the SYN-flood Attack.

SYN Flood Attack in TCP

A SYN flood attack is a type of Denial-of-Service (DoS) attack that targets the TCP three-way handshake process. It exploits the way a TCP connection is established to overload a server's resources, making it unable to handle legitimate traffic.

Working:

The attacker sends a large number of SYN (synchronize) packets to the target server, each appearing to request a new TCP connection.

For each SYN received, the server responds with a SYN-ACK packet and reserves system resources (like memory and connection slots) to keep track of the partially open connection.

Instead of completing the handshake by sending the final ACK, the attacker either:

- Doesn't respond at all, or
- Uses spoofed IP addresses (fake sender addresses), ensuring that no ACK ever arrives.

These half-open connections (called TFO – TCP half-open) accumulate quickly, consuming the server's connection queue and memory.

As a result, the server is overwhelmed and unable to accept new, legitimate connection requests, effectively denying service to real users.

The SYN flood attack is a classic example of abusing TCP's reliability mechanism to cause service disruption. By overwhelming the server with fake connection requests, attackers can exhaust system resources and block legitimate users. Understanding and defending against this kind of attack is a key part of network security.

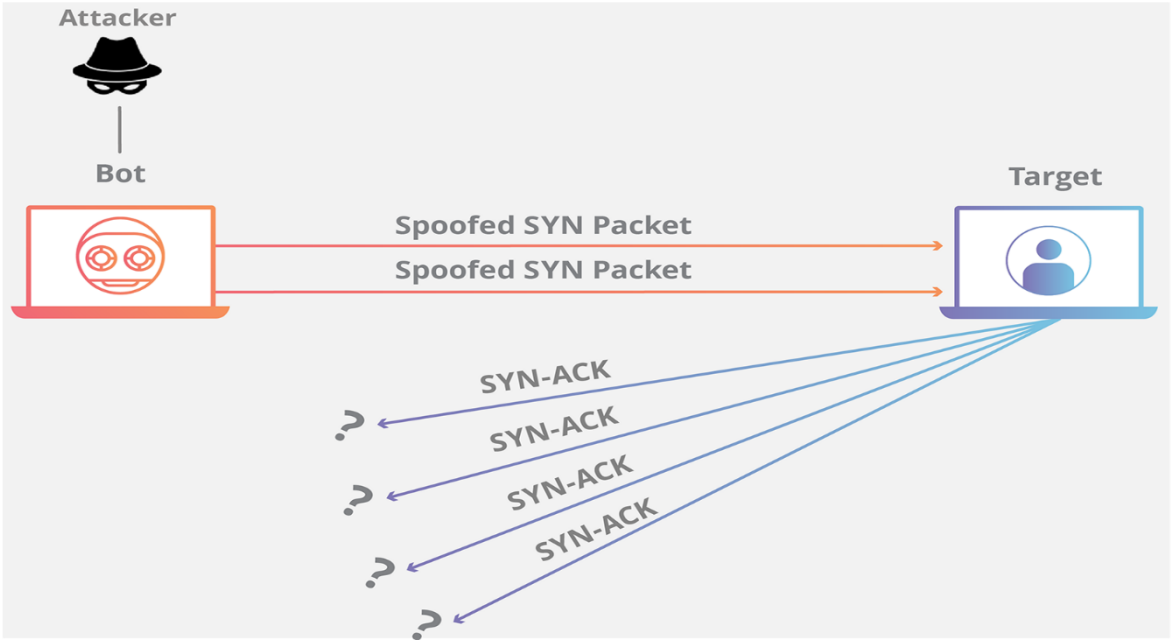

Figure-13. SYN-Flood Attack

7. Conclusion

The Transmission Control Protocol (TCP) plays a pivotal role in ensuring reliable, ordered, and error-free communication across modern computer networks. As a cornerstone of the TCP/IP protocol suite, TCP offers vital transport layer services that allow applications to transmit data over the internet with confidence in delivery integrity and sequence. It

abstracts the complexities of underlying network behaviour and provides a robust communication channel between systems, regardless of the physical medium or intermediate routing.

A key component of TCP's reliability is its three-way handshake, a structured process that establishes a connection between a client and server before data transmission begins. This handshake involves the exchange of SYN, SYN-ACK, and ACK packets. In three simple yet effective steps, both parties agree upon initial sequence numbers and confirm their readiness to transmit and receive data. This process ensures that connections are initialized securely, with proper synchronization and acknowledgement mechanisms in place.

Moreover, the handshake mechanism facilitates TCP's essential features such as flow control, congestion avoidance, and data sequencing, which together maintain the integrity and performance of the connection. However, the same handshake process can be exploited by malicious actors through attacks like SYN flooding, where numerous fake requests deplete server resources and lead to denial of service. Such vulnerabilities emphasize the importance of implementing security measures like SYN cookies, rate-limiting, and intrusion detection systems.

In conclusion, TCP and its connection establishment protocol exemplify a balance between functionality, reliability, and security in network communication. The three-way handshake not only initiates the transfer of data but also embodies the principles of trust and coordination that underpin reliable networking. Understanding the intricacies of TCP and its connection management processes is essential for network professionals, developers, and cybersecurity practitioners alike, as it forms the foundation of most internet-based communication today.

8. References

[1] J. Postel, Transmission Control Protocol, document RFC-793, IETF, 1981.

[2] J. Wu, C. Yuen, M. Wang, J. Chen, and C. W. Chen, ''TCP-oriented raptor coding for high-frame-rate video transmission over wireless networks,'' IEEE J. Sel. Areas Commun., vol. 34, no. 8, pp. 2231–2246, Aug. 2016.

[3] J. Wu, C. Yuen, and J. Chen, ''Leveraging the delay-friendliness of TCP with FEC coding in real-time video communication,'' IEEE Trans. Commun., vol. 63, no. 10, pp. 3584–3599, Oct. 2015.

[4] J. Wu, B. Cheng, M. Wang, and J. Chen, ''Priority-aware FEC coding for high-definition mobile video delivery using TCP,'' IEEE Trans. Mobile Comput., vol. 16, no. 4, pp. 1090–1106, Apr. 2017.

[5] F. R. Kevin and W. R. Stevens, ''TCP: The transmission control protocol (Preliminaries),'' in TCP/IP Illustrated: The Protocols, vol. 1, 2nd ed. Reading, MA, USA: Addison-Wesley, 2011, pp. 586–587.

[6] J. Zhang, F. Ren, and C. Lin, ''Survey on transport control in data center networks,'' IEEE Netw., vol. 27, no. 4, pp. 22–26, Jul./Aug. 2013.

[7] J. C. Mogul, ‘‘The case for persistent-connection HTTP,’’ in Proc. ACM SIGCOMM Symp., Stockholm, Sweden, 1995, pp. 299–313

[8] V. N. Padmanabhan and J. C. Mogul, ‘‘Improving HTTP latency,’’ Comput. Netw. ISDN Syst., vol. 28, nos. 1–2, pp. 25–35, 1995.

[9] E. Casilari, F. J. Gonzblez, and F. Sandoval, ‘‘Modeling of HTTP traffic,’’ IEEE Commun. Lett., vol. 5, no. 6, pp. 272–274, Jun. 2001.

[10] R. Corbel, E. Stephan, and N. Omnes, ‘‘HTTP/1.1 pipelining vs HTTP2 inthe-clear: Performance comparison,’’ in Proc. 13th Int. Conf. New Technol. Distrib. Syst. (NOTERE), Jul. 2016, pp. 1–6.