RUNGTA COLLEGE OF ENGINEERING & TECHNOLOGY

TECHNICAL TRAINING PROJECT (CYBER SECURITY)

Name of Project: Comparative Security Analysis of Telnet and SSH Using Network Traffic Capture

Branch: CSE (AIML) – 3rd Semester

Section: C

Student Name: Harsh Raj Singh

ERP ID: 6606931

Professor Name: Prashant Kamkar

Date of Submission: 26-12-25

Student Signature: _____

Professor Signature: _____

Index

Context

In today's networked world, remote access to computers and servers is a common requirement for system administration and management. As data travels across networks, security becomes a major concern, especially when sensitive information such as usernames, passwords, and commands are transmitted. Earlier remote access protocols were designed with functionality in mind, without considering modern security threats.

This project focuses on the study of Telnet and Secure Shell (SSH) to understand the importance of secure communication in computer networks. Telnet, being one of the earliest remote login protocols, transmits data in plaintext, making it vulnerable to network sniffing and man-in-the-middle attacks. SSH was developed to overcome these weaknesses by providing encrypted communication and secure authentication.

By configuring and using both Telnet and SSH in a controlled environment and analysing their traffic using Wireshark, this project demonstrates the clear difference between insecure and secure protocols. The comparison highlights why Telnet is considered obsolete and why SSH has become the industry-standard protocol for secure remote access.

TELNET

Telnet is one of the earliest protocols designed for remote terminal access. It allows a user sitting at one computer to log in to another computer over a network and work on as if they were locally connected. In a Telnet session, the client and server exchange characters representing keystrokes and command outputs. However, Telnet was created at a time when security was not a major concern, so no cryptographic protection was added to the protocol. Every character typed by the user and every response from the server is transmitted as plain ASCII text over the network.

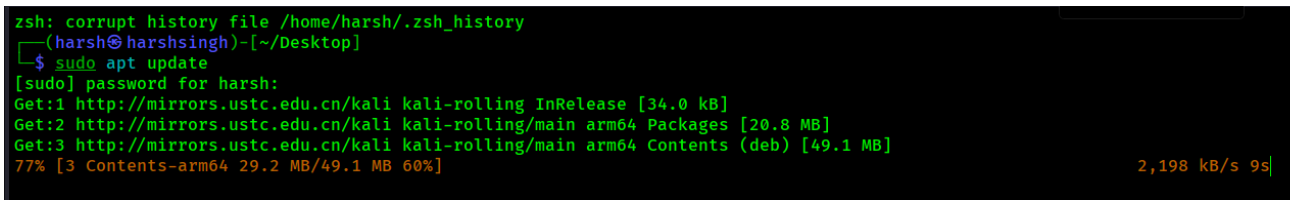
Because there is no encryption, Telnet traffic can be easily intercepted and read by anyone who has access to the same network segment. If an attacker runs a packet-capturing tool on the network, they can see the user's username, password and all commands in a human-readable form. For this reason, Telnet is now considered insecure and is almost completely replaced by secure alternatives such as SSH. In modern environments, Telnet is kept only for testing, legacy systems or educational demonstrations where the goal is to show how insecure protocols look on the wire.

How Telnet was configured and usedIn this project, the Kali Linux virtual machine was used as a Telnet server. The goal was to create a working Telnet session from the Mac host, capture the traffic and then prove that all data appears in clear text inside the capture.

(1) Installing and enabling Telnet on Kali

First, the package list on Kali was updated so that the latest software versions were available. Then, the Telnet daemon and the inetd service were installed to provide Telnet server functionality. The inetd service acts as a super-server that listens for incoming connections and starts the Telnet daemon when required.

>>>>update package list:
Command executed:



```
harsh@harshsingh: ~/Desktop
Session  Actions  Edit  View  Help
zsh: corrupt history file /home/harsh/.zsh_history
┌──(harsh㉿harshsingh)-[~/Desktop]
└─$ sudo apt update
[sudo] password for harsh:
Get:1 http://mirrors.ustc.edu.cn/kali kali-rolling InRelease [34.0 kB]
Get:2 http://mirrors.ustc.edu.cn/kali kali-rolling/main arm64 Packages [20.8 MB]
Get:3 http://mirrors.ustc.edu.cn/kali kali-rolling/main arm64 Contents (deb) [49.1 MB]
Fetched 70.0 MB in 14s (5,000 kB/s)
588 packages can be upgraded. Run 'apt list --upgradable' to see them.

┌──(harsh㉿harshsingh)-[~/Desktop]
└─$
```

>>>>install Telnet server and inted:
Command executed:

```
┌──(harsh㉿harshsingh)-[~/Desktop]
└─$ sudo apt install telnetd openbsd-inetd -y
telnetd is already the newest version (0.17+2.6-4).
openbsd-inetd is already the newest version (0.20221205-3+b3).
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 588

┌──(harsh㉿harshsingh)-[~/Desktop]
└─$
```

>>>> start and enable inted:
Command executed:

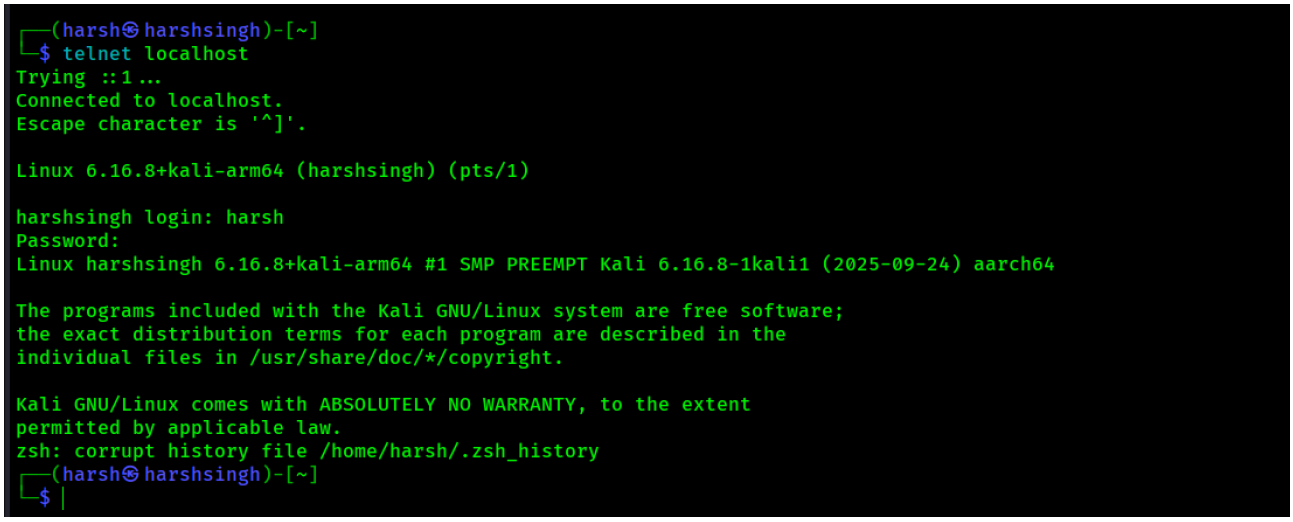
```
┌──(harsh㉿harshsingh)-[~/Desktop]
└─$ sudo systemctl start openbsd-inetd

┌──(harsh㉿harshsingh)-[~/Desktop]
└─$
```

```
┌──(harsh㉿harshsingh)-[~/Desktop]
└─$ sudo systemctl enable openbsd-inetd
Synchronizing state of openbsd-inetd.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable openbsd-inetd

┌──(harsh㉿harshsingh)-[~/Desktop]
└─$
```

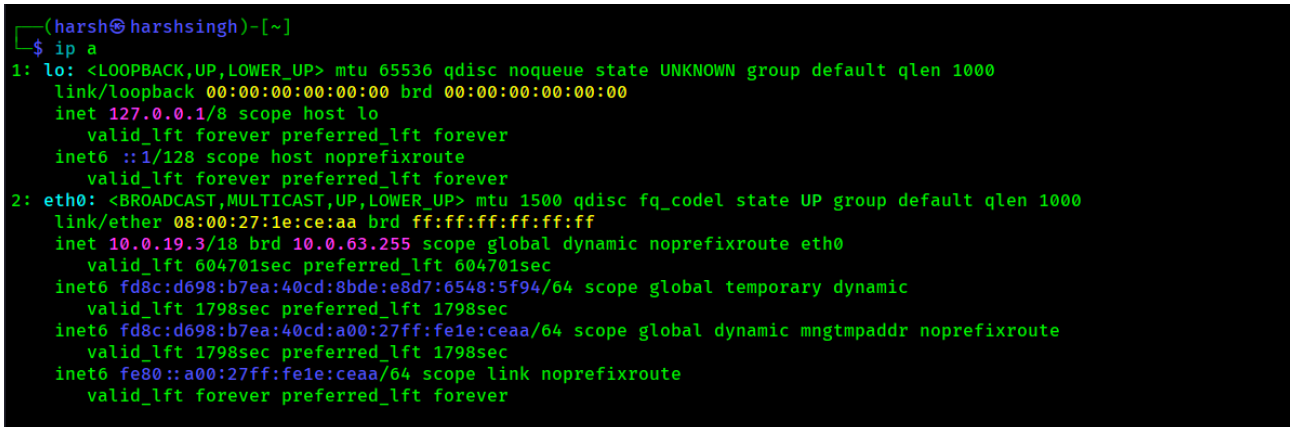
>>>>Check status:
Command executed:

```
┌──(harsh㉿harshsingh)-[~/Desktop]
└─$ sudo systemctl status openbsd-inetd
● inetd.service - Internet superserver
     Loaded: loaded (/usr/lib/systemd/system/inetd.service; enabled; preset: disabled)
     Active: active (running) since Fri 2025-12-19 19:57:57 IST; 7min ago
 Invocation: c6be338fbf4840169e1af734beda0df1
       Docs: man:inetd(8)
   Main PID: 834 (inetd)
      Tasks: 1 (limit: 8320)
     Memory: 1.5M (peak: 1.7M)
        CPU: 50ms
     CGroup: /system.slice/inetd.service
             └─834 /usr/sbin/inetd

Dec 19 19:57:57 harshsingh systemd[1]: Starting inetd.service - Internet superserver ...
Dec 19 19:57:57 harshsingh systemd[1]: Started inetd.service - Internet superserver.
```

>>>>Telnet configuration:
Command executed:



>>>>Discovering the server ip address:
Command executed:



>>>>Establishing a Telnet Connection from the Client:

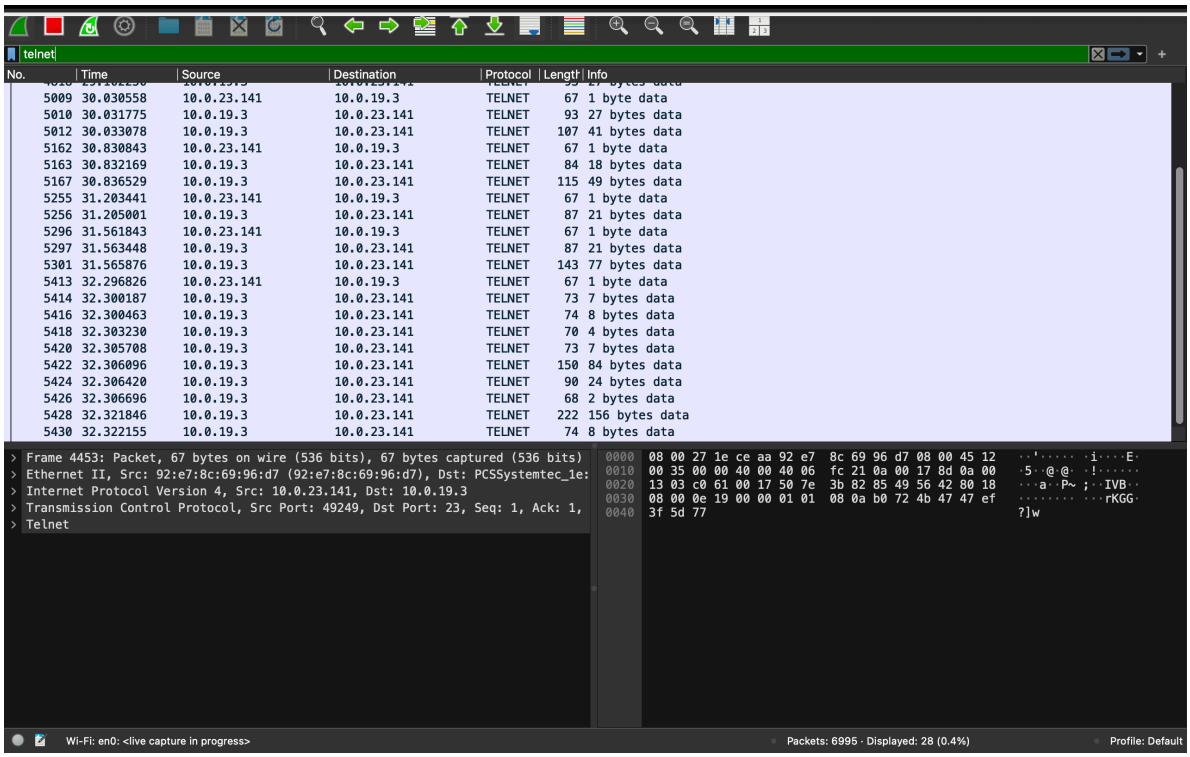
Telnet client connection was established from the user terminal to the Kali Linux server using its IP address. The client machine acted as the remote user system, while the Kali Linux machine functioned as the Telnet server. After entering the command telnet <KALI_IP>, the client successfully connected to the Telnet service running on the Kali system and displayed the remote login prompt.

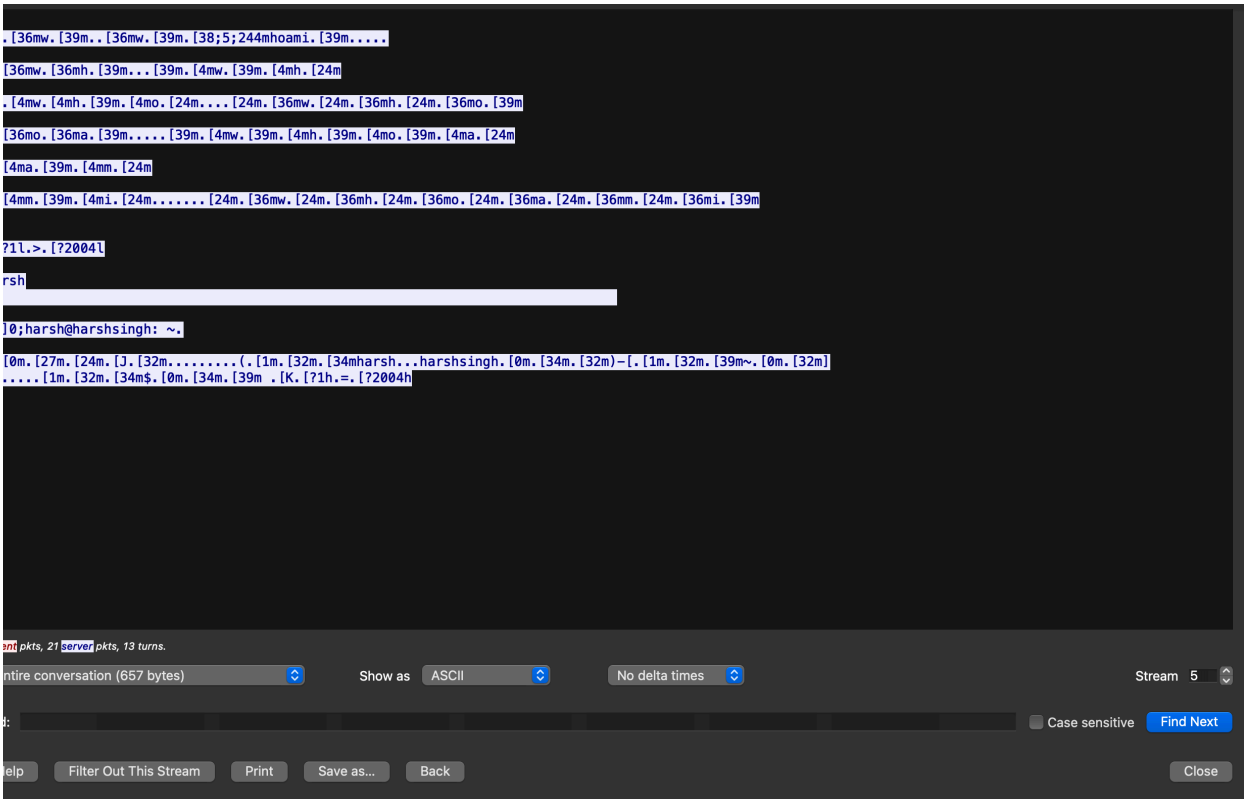
Command executed over telnet

```
Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
┌──(harsh㉿ harshsingh)-[~]
└─$ whoami
harsh

┌──(harsh㉿ harshsingh)-[~]
└─$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos

┌──(harsh㉿ harshsingh)-[~]
└─$ pwd
/home/harsh

┌──(harsh㉿ harshsingh)-[~]
└─$
```

>>>>Analysing Telnet traffic in Wireshark

In this step, the Telnet traffic generated in the previous section is analysed using Wireshark. After stopping the capture, one Telnet packet from the session is selected and the TCP stream is followed. The stream window clearly shows the username, password and all commands in readable plain text. This demonstrates that Telnet does not provide any encryption and that an attacker who captures the traffic can easily see the login credentials and command history.




.[36mw.[39m..[36mw.[39m.[38;5;244mhoami.[39m.....
[36mw.[36mh.[39m...[39m.[4mw.[39m.[4mh.[24m
.[4mw.[4mh.[39m.[4mo.[24m....[24m.[36mw.[24m.[36mh.[24m.[36mo.[39m
[36mo.[36ma.[39m.....[39m.[4mw.[39m.[4mh.[39m.[4mo.[39m.[4ma.[24m
[4ma.[39m.[4mm.[24m
[4mm.[39m.[4mi.[24m.......[24m.[36mw.[24m.[36mh.[24m.[36mo.[24m.[36ma.[24m.[36mm.[24m.[36mi.[39m
?1l.>.[?2004l
rsh
]0;harsh@harshsingh: ~.
[0m.[27m.[24m.[J.[32m.........(.[1m.[32m.[34mharsh...harshsingh.[0m.[34m.[32m)-[.[1m.[32m.[39m~.[0m.[32m]
.....[1m.[32m.[34m$.[0m.[34m.[39m .[K.[?1h.=.[?2004h
ent pkts, 21 server pkts, 13 turns.
ntire conversation (657 bytes)
Show as
ASCII
No delta times
Stream
5
Case sensitive
Find Next
elp
Filter Out This Stream
Print
Save as...
Back
Close

SSH

SSH (Secure Shell) is a cryptographic network protocol that provides secure remote access to computers and servers over an unsecured network. It was developed as a secure replacement for insecure protocols like Telnet and rlogin, which transmit data including usernames, passwords, and commands in plaintext, making them vulnerable to eavesdropping attacks.

What SSH DoesSSH

establishes a secure channel between a client (your computer) and a server (remote machine) using strong encryption algorithms. All communication—login credentials, file transfers, port forwarding, and command execution—travels encrypted through this tunnel, protecting against packet sniffing, man-in-the-middle attacks, and unauthorised interception.

Core Components of SSH

SSH Client: Software on your local machine (like OpenSSH client on Linux/ Windows) that initiates connections.

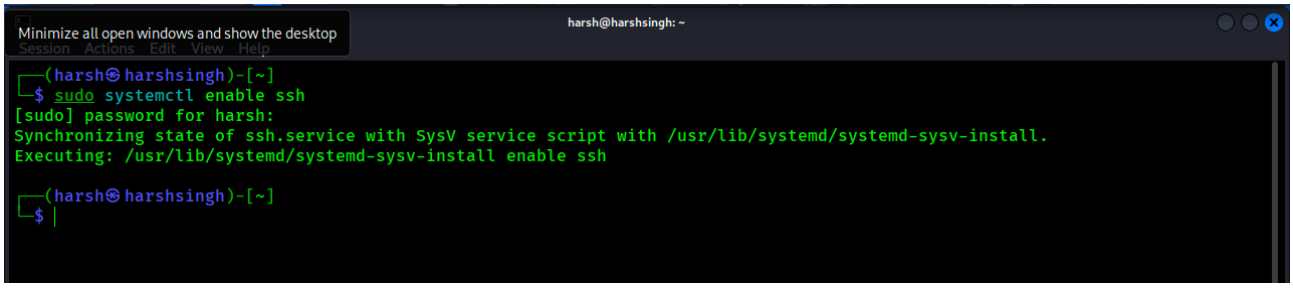
SSH Server (sshd): Daemon running on the remote machine listening on TCP port 22, handling incoming connections.

Encryption: Uses symmetric ciphers (AES-256) for bulk data, asymmetric (RSA/ ECDSA) for key exchange and authentication.

Authentication Methods: Password-based, public-key (preferred), host-based, or multi-factor.

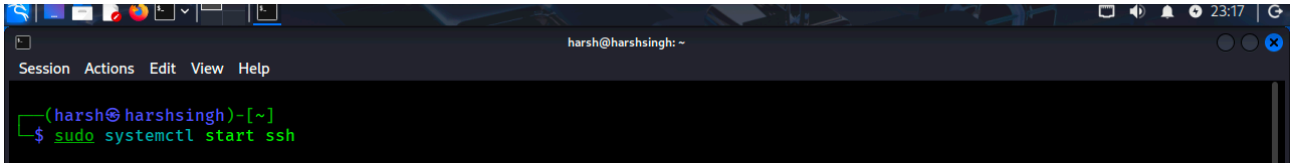
>>>>Enabling SSH Service:

The SSH service was enabled to start automatically on boot using systemd.
Command executed:



>>>>Starting SSH Service:

The SSH daemon was manually started to begin listening on port 22 immediately.
Command executed:



>>>>Verifying SSH Service Status:

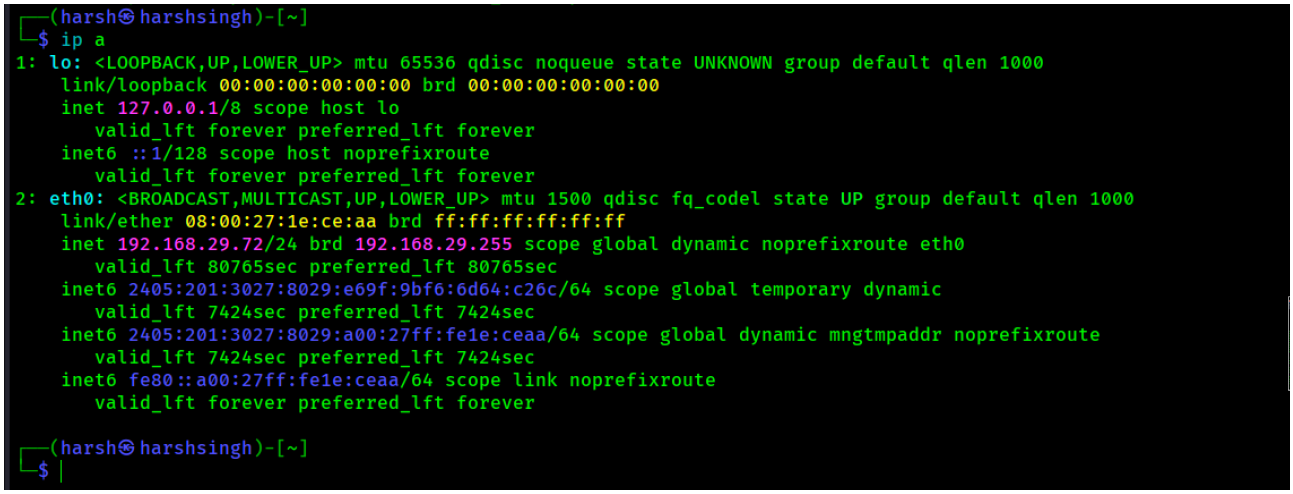
The status of the SSH service was checked to confirm it was active and running properly.
Command executed:

```
(harsh㉿harshsingh)-[~]
$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/usr/lib/systemd/system/ssh.service; enabled; preset: disabled)
     Active: active (running) since Fri 2025-12-19 22:48:16 IST; 29min ago
 Invocation: f8dc686a7e8141acb3f015d3bae28f18
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 3592 (sshd)
      Tasks: 1 (limit: 8320)
     Memory: 7.6M (peak: 26.9M)
        CPU: 191ms
     CGroup: /system.slice/ssh.service
             └─3592 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Dec 19 22:50:44 harshsingh sshd-session[4443]: pam_unix(sshd:auth): check pass; user unknown
Dec 19 22:50:46 harshsingh sshd-session[4443]: Failed password for invalid user harshsingh from 192.168.29.52 port 49240 ss>
Dec 19 22:50:52 harshsingh sshd-session[4443]: pam_unix(sshd:auth): check pass; user unknown
Dec 19 22:50:54 harshsingh sshd-session[4443]: Failed password for invalid user harshsingh from 192.168.29.52 port 49240 ss>
Dec 19 22:50:56 harshsingh sshd-session[4443]: Connection closed by invalid user harshsingh 192.168.29.52 port 49240 [preau>
Dec 19 22:50:56 harshsingh sshd-session[4443]: PAM 2 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rho>
Dec 19 22:55:37 harshsingh sshd-session[7540]: Accepted password for harsh from ::1 port 33200 ssh2
Dec 19 22:55:37 harshsingh sshd-session[7540]: pam_unix(sshd:session): session opened for user harsh(uid=1000) by harsh(uid>
Dec 19 22:57:25 harshsingh sshd-session[8320]: Accepted password for harsh from 192.168.29.52 port 49243 ssh2
Dec 19 22:57:25 harshsingh sshd-session[8320]: pam_unix(sshd:session): session opened for user harsh(uid=1000) by harsh(uid>

(harsh㉿harshsingh)-[~]
$
```

>>>> SSH configuration:
Command executed:

```
(harsh㉿harshsingh)-[~]
$ ssh harsh@localhost
harsh@localhost's password:
Linux harshsingh 6.16.8+kali-arm64 #1 SMP PREEMPT Kali 6.16.8-1kali1 (2025-09-24) aarch64

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Dec 19 22:57:25 2025 from 192.168.29.52
zsh: corrupt history file /home/harsh/.zsh_history
(harsh㉿harshsingh)-[~]
$
```

>>>>Discovering the server ip address:
Command executed:

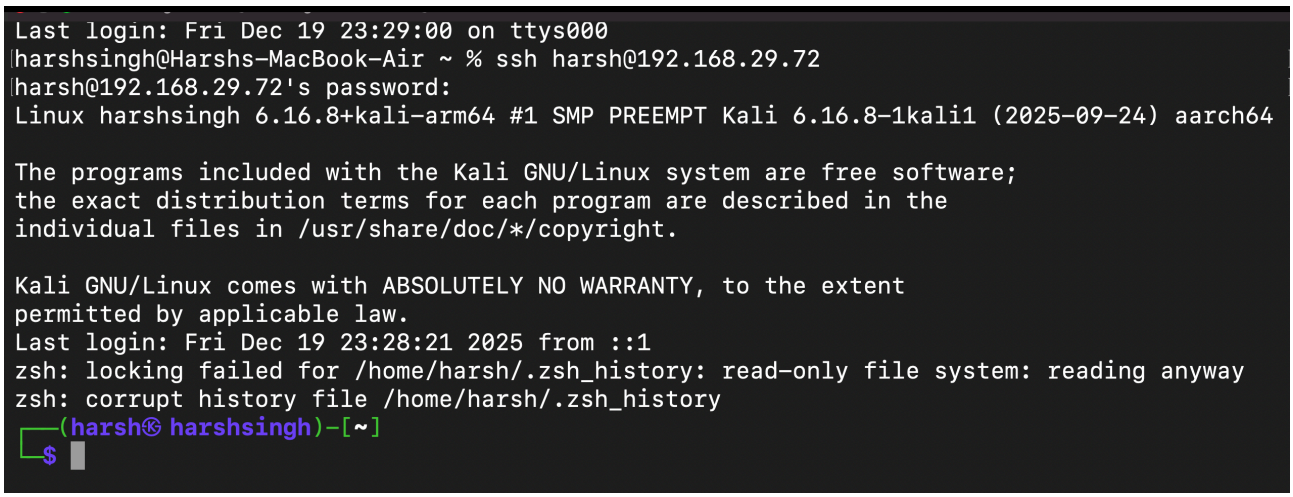


>>>>Establishing SSH Connection from Client

A secure SSH client connection was established from the user terminal to the Kali Linux server using the identified IP address and username. The client accepted the server's host key fingerprint (first-time connection), entered the user password, and gained access to an encrypted shell session on the remote system.

Commands executed over SSH:

```
ssh <username>@<KALI_IP>
whoami
ls -la
pwd
id
exit
```



```
┌──(harsh㉿harshsingh)-[~]
└─$ whoami
harsh

┌──(harsh㉿harshsingh)-[~]
└─$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos

┌──(harsh㉿harshsingh)-[~]
└─$
```

>>>>Capturing SSH Traffic in Wireshark:

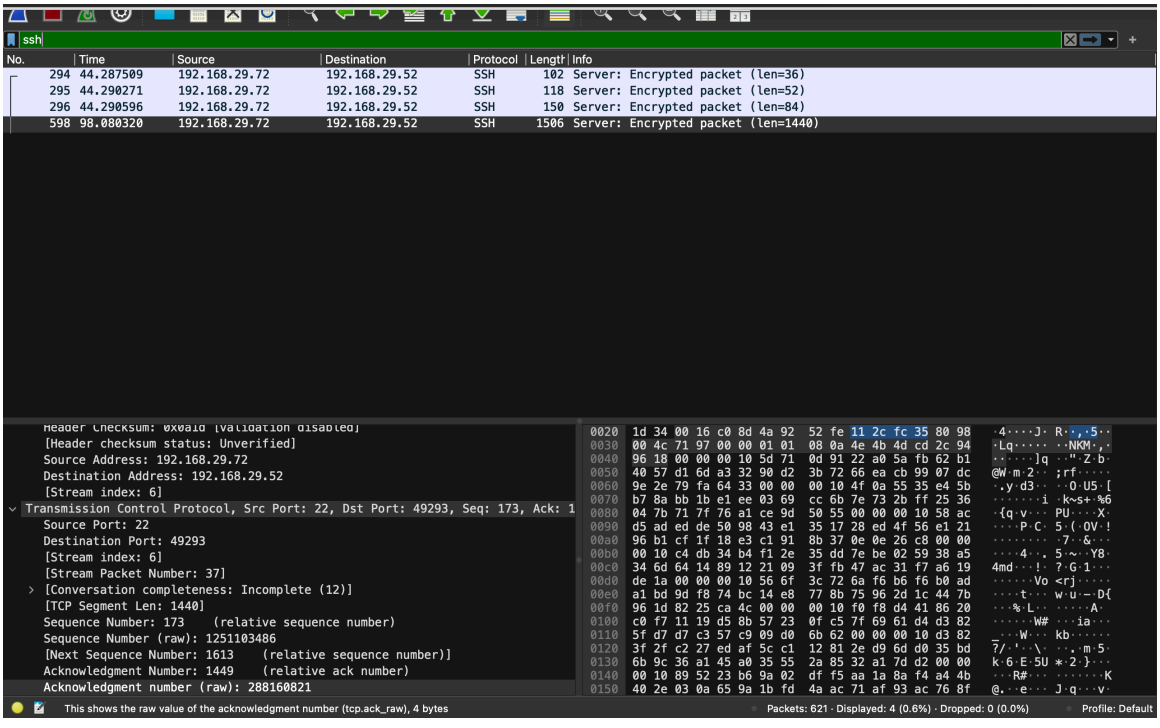
Wireshark was started on the client machine to capture network traffic on the active interface during the SSH connection. The capture began before executing the SSH command and continued through login and command execution to record the complete session.

Alternative filter:

```
ssh
```

Analysis performed:

- Right-click packet → Follow → TCP Stream
- Result: Random unreadable bytes (encrypted payload)



This analysis confirms SSH transmits all sensitive information securely encrypted, unlike Telnet's plaintext exposure.

Comparative Analysis: Telnet vs SSH

Both protocols enable remote command-line access but differ fundamentally in security implementation. Telnet transmits all data unencrypted while SSH provides comprehensive cryptographic protection.why telnet is insecure and deprecated

Feature	SSH (Secure Shell)	Telnet (Telecommunication Network)
Security	SSH encrypts the data transmitted, ensuring confidentiality and integrity.	Telnet transmits data in plain text, making it vulnerable to eavesdropping and man-in-the-middle attacks.
Authentication	SSH uses strong authentication methods like password-based or key-based authentication.	Telnet uses basic authentication methods, often without encryption, making it insecure.
Default Port	22	23
Data Encryption	Data is encrypted, providing secure communication over the network.	No encryption, data is transmitted in plain text.
Usage	Commonly used for secure remote administration of network devices and systems.	Typically used for remote management of devices but is now largely obsolete due to security risks.

- why telnet is insecure and deprecated

Plaintext Transmission: The fundamental flaw of Telnet is its lack of encryption. Anyone with access to the network path between the client and the server (e.g., on a shared switch, router, or gateway) can use a packet analyzer (like Wireshark) to intercept and read the entire communication, including login credentials and commands.

- **Vulnerability to Eavesdropping:** This plaintext communication makes sessions extremely susceptible to unauthorized surveillance (eavesdropping) and session hijacking.
- **Man-in-the-Middle (MitM) Attacks:** Attackers can position themselves between the client and server to intercept, read, and even modify the data stream without either party being aware.
- **Weak Authentication:** Telnet typically uses basic username/password authentication, which is transmitted insecurely and lacks support for stronger methods like public key cryptography, used in modern protocols.

Telnet was developed in 1969, at a time when networks were smaller, isolated, and generally considered secure, with less concern for the kind of broad security threats that exist on the modern internet.

Its deprecation is due to the availability of a superior, secure alternative: Secure Shell (SSH).

- SSH provides strong encryption for the entire session, safeguarding all data in transit.
- SSH offers robust authentication mechanisms to verify the identity of both parties and protect against identity spoofing.

- How network sniffing attacks work

Network sniffing attacks work by using specialized tools (packet sniffers) to intercept and read data packets flowing across a network, capturing sensitive info like passwords, emails, or banking details, especially on unencrypted traffic or wireless networks, by placing the sniffer in a position (often a Man-in-the-Middle role) to see traffic not meant for it, either passively (hub/wireless) or actively (ARP spoofing on switches).

How it works (Step-by-Step):

1. **Data in Packets**: All network communication (websites, emails, files) is broken into small chunks called data packets, each with source/destination info, as shown in this graphic from ScienceDirect.

2. **Sniffer Placement**: An attacker places their device (with the sniffer software) to intercept these packets.

    - **Passive Sniffing**: On shared networks (hubs, Wi-Fi), devices see all traffic. The sniffer just listens in, like wiretapping.

    - **Active Sniffing**: On switched networks (most modern LANs), traffic is directed. Attackers use techniques like ARP spoofing (tricking devices into sending traffic to the attacker's machine first) to redirect packets.

3. **Packet Capture**: The sniffer software (e.g., Wireshark, tcpdump) puts the network interface card (NIC) into "promiscuous mode" to grab all packets, not just those addressed to it.

4. **Data Analysis**: The captured packets are analyzed. If unencrypted (HTTP, old Wi-Fi), the attacker can read the content directly to find usernames, passwords, messages, or other sensitive data.

- why ssh is the industry-standard secure remote access protocol

SSH (Secure Shell) became the **industry-standard secure remote access protocol** because it combines **strong security**, **practical usability**, and **broad ecosystem support** better than any alternative. Here's a clear breakdown of *why* it earned that position:

1. Strong Security by Design 🔐

SSH was created specifically to fix the weaknesses of older protocols like **Telnet, rlogin, and FTP**, which sent data in plaintext.

SSH provides:

- **Encryption** of all traffic (protects passwords, commands, and data)
- **Integrity checking** (prevents tampering and man-in-the-middle attacks)
- **Server authentication** (you know you're connecting to the correct machine)
- **Optional client authentication** (passwords, public/private keys, certificates)

These protections make SSH safe even on untrusted networks like the public internet.

2. Public Key Authentication (Passwordless & Secure)

SSH popularised **public-key authentication**, which is:

- More secure than passwords
- Resistant to brute-force attacks
- Easy to automate securely (CI/CD, servers, scripts)

This made SSH ideal for:

- System administrators
- Cloud infrastructure
- Automated deployments

3. Secure by Default

Unlike many protocols that add security later, SSH is **secure out of the box**:

- Encryption is mandatory
- Modern implementations disable weak algorithms
- Defaults favour safety over convenience

This drastically reduces misconfiguration risks.

4. Firewall & Network Friendly

SSH typically uses **TCP port 22**, which:

- Is easy to allow through firewalls
- Can be moved to other ports if needed
- Works well over high-latency networks

This practicality helped adoption at scale.

- The Importance of Encryption in Network Security

Encryption is one of the most critical foundations of modern network security. It protects data as it travels across networks, ensuring that information remains private, accurate, and trustworthy even when transmitted over insecure or public networks such as the internet. Without encryption, network communication would be vulnerable to interception, manipulation, and abuse.

---

1. Protecting Data Confidentiality

The primary purpose of encryption is to protect **confidentiality**. When data is encrypted, it is converted into an unreadable format called cipher text. Only authorised users with the correct decryption key can convert it back into readable information.

In a network environment, data constantly moves between devices, servers, and applications. If this data is not encrypted:

- Attackers can capture it using packet sniffing tools
- Sensitive information such as passwords, credit card numbers, medical records, and personal messages can be exposed

Encryption ensures that even if attackers intercept network traffic, the data remains useless to them.

---

2. Preventing Eavesdropping on Network Communications

Networks—especially public or shared networks—are vulnerable to eavesdropping. Anyone connected to the same network can potentially monitor traffic.
Encryption protects against:

- Unauthorised monitoring
- Network spying
- Data leakage

For example, HTTPS encrypts web traffic so that attackers cannot see the content of webpages, login credentials, or form submissions.

3. Defending Against Man-in-the-Middle (MITM) Attacks
In a man-in-the-middle attack, an attacker secretly intercepts communication between two parties and may alter the data being transmitted.
Encryption helps prevent MITM attacks by:

- Encrypting the communication channel

- Authenticating servers and clients using digital certificates
- Detecting tampering through cryptographic integrity checks

Protocols such as **TLS, SSH, and VPNs** ensure that users are communicating with legitimate systems and not attackers.

---

4. Ensuring Data Integrity

Encryption systems also protect **data integrity**, which means ensuring that data has not been changed during transmission.

Using cryptographic techniques such as:

- Hash functions
- Message Authentication Codes (MACs)
- Digital signatures

Encryption allows systems to detect even the smallest alteration in data. If data is modified by an attacker, the integrity check fails and the data is rejected.

---

5. Protecting Authentication Credentials

Network authentication depends on sensitive information such as:

- Usernames and passwords
- Session cookies
- API keys
- Access tokens

Encryption ensures that these credentials:

- Are not exposed during transmission
- Cannot be reused by attackers
- Remain confidential even if traffic is intercepted

Without encryption, attackers could steal login credentials and gain unauthorised access to systems and networks.