Title: Encryption Communication with SSH on Ubuntu Linux Virtual Machine

Prepared By : Ifeanyi Christian Edeh

Date: April 7, 2025

Table of Contents page

Executive Summary

As organizations increasingly adopt remote work and cloud-based infrastructure, secure remote access has become a critical component of enterprise cybersecurity. One of the most commonly used tools for remote system administration is the Secure Shell (SSH) protocol. While SSH is inherently more secure than older protocols like Telnet, its effectiveness relies heavily on how well it is configured and maintained. A default or misconfigured SSH server can still expose critical vulnerabilities that attackers can exploit (Mozilla, 2023; NIST, 2019).

This project focuses on transforming a basic Ubuntu Linux SSH server into a secure and hardened environment by implementing industry-standard security measures. These include disabling root login, enforcing SSH key-based authentication, changing the default port, applying session timeouts, and restricting the use of weak cryptographic algorithms. Each of these settings was applied through edits to the SSH daemon's configuration file (/etc/ssh/sshd_config) and verified through testing to ensure security and functionality.

Disabling root login and password authentication directly mitigates the risk of brute-force attacks. A common tactic used by automated bots and human adversaries to guess login credentials (TechTarget, 2022). By using key-based authentication, this setup ensures that only users with valid private keys can gain access, making credential theft or reuse virtually impossible. Port changes and idle session timeouts provide additional layers of protection by reducing exposure to opportunistic attacks and minimizing session hijacking risk (Vultr Docs, 2023).

The project also includes a comparative analysis between SSH and Telnet traffic using Wireshark. This analysis clearly shows that Telnet transmits all data in plaintext, exposing sensitive information such as usernames and passwords, while SSH encrypts every stage of the connection, ensuring confidentiality and integrity of transmitted data (Kagoshima, 2025).

Through careful configuration and validation, this project demonstrates how organizations even at small or medium scale can deploy a hardened SSH server that meets modern cybersecurity expectations. The practical steps outlined here are aligned with security guidelines published by authoritative sources such as Mozilla, NIST, and the Center for Internet Security. By adopting these best practices, administrators can significantly reduce the attack surface of their systems and strengthen the organization's overall security posture.

1. Introduction

Secure Shell (SSH) is a cryptographic network protocol designed to enable secure remote login and other network services over an unsecured network. SSH was developed as a secure alternative to older protocols like Telnet, which transmit data in plain text and are vulnerable to eavesdropping (Ganguly, 2020). In modern enterprise environments, SSH is the standard for administrator access to servers and network devices because it provides strong encryption, authentication, and integrity protection for communications (Kagoshima, 2025). This report documents the setup of an SSH server and outlines measures taken to harden its security. It covers the implementation of SSH key-based authentication, critical SSH server configuration hardening steps, and an analysis of network traffic using Wireshark to compare the legacy Telnet protocol with SSH. By highlighting differences between plaintext Telnet traffic and encrypted SSH traffic, we demonstrate why SSH is essential for secure remote management. All recommendations are aligned with industry best practices and authoritative guidelines (Mozilla, 2023; DigitalOcean, 2022).

2. SSH Server Installation and Basic Configuration

Before hardening SSH, one must ensure the OpenSSH server is installed and running on the Ubuntu system. On modern Ubuntu releases, the package is typically named openssh-server. In this setup, Ifeanyi@linux-server is the SSH server while student@linux-server is the local or client machine that initiates SSH connections.The SSH server is the machine that accepts incoming SSH connections and runs the sshd service. SSH was installed and updated to SSH server using apt: `sudo apt update && sudo apt install -y openssh-server`. Once installed, the SSH daemon (sshd) starts automatically. To verify that the SSH service is active and enabled to start on boot, we use the command: `sudo systemctl status sshd`**,** followed by `sudo systemctl enable sshd`

By default, SSH listens on TCP port 22 and allows login with any system user accounts (using password or key authentication as configured). The main configuration file for the SSH server is located at `/etc/ssh/sshd_config`. It's good practice to back up this file before making significant changes.we backup the file using this command: `sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.bak`
After each configuration change, the SSH daemon must be restarted for changes to take effect. We use this command `sudo systemctl restart sshd` to restart the SSH daemon. With the SSH service up and running, we can proceed to apply a series of hardening measures. These include disabling direct root access, turning off password logins, changing the listening port, restricting cryptographic algorithms, and setting session timeouts.

3. SSH Server Configuration Hardening

To secure the SSH service on a Linux VM, we will adjust the SSH daemon configuration (/etc/ssh/sshd_config) to enforce strict security measures. Each change enhances the security of the server and mitigates specific threats. Below are the key hardening steps and their corresponding configuration changes:

3.1 Disable Root Login via SSH

Logging in directly as root is risky. We set PermitRootLogin no to disallow root over SSH (Vultr Docs, 2023). This means even with a key, the root user cannot log in remotely. Instead, administrators should log in as a normal user and use sudo for privileged actions. Disabling root login thwarts many automated attacks that specifically target the root account with password guesses (Hackviser, 2023; TechTarget, 2022).

3.2 Enforce Public Key Authentication (Disable Passwords)

SSH key-based logins are much more secure than passwords. We disable password authentication so that only cryptographic key pairs can be used to log in. In sshd_config, set PasswordAuthentication no (and ensure PubkeyAuthentication yes). This prevents attackers from brute-forcing passwords entirely, since no password login is accepted (TechTarget, 2022).

3.3 Change the Default SSH Port

By default, SSH listens on TCP port 22. We change this to a non-standard port (for example, Port 2222) (Hackviser, 2023). This is a form of security by obscurity. It doesn't replace other defenses, but it can reduce noise from automated bots targeting port 22 (Vultr Docs, 2023). Casual attackers often scan the default port, so moving SSH to a different port might avoid opportunistic attacks (TechTarget, 2022). After changing the port, update firewall rules to allow the new port. This step by itself won't stop a determined attacker who port-scans your system, but it cuts down on random login attempts (TechTarget, 2022).

3.4 Enforce Strong Ciphers and Algorithms

We restrict the cipher algorithms to strong modern ciphers. We set the ciphers: Ciphers aes256-ctr,aes192-ctr,aes128-ctr. This limits SSH to use AES in CTR mode with 128-bit or higher keys, which are considered strong encryption algorithms (Hackviser, 2023). Weak ciphers (like old 3DES or RC4) and MAC algorithms should be disabled to prevent cryptographic attacks that target older, vulnerable algorithms. By using only strong ciphers, we ensure the confidentiality of the SSH session cannot be easily broken.

3.5 Configure Idle Session Timeout

It’s important to automatically close SSH sessions that are idle, in case a user forgets to log out or an attacker gains access to an unattended session. We configure the ClientAlive parameters in sshd_config. We configure the ClientAliveInterval 300 and the ClientAliveCountMax 0. This means the server will send an alive message every 300 seconds and, with count 0, will terminate the session if the client doesn’t respond (i.e. if it’s inactive) (Hackviser, 2023). In effect, any SSH session idle for 5 minutes will be disconnected. This timeout helps reduce the risk of someone hijacking an abandoned session and also conserves server resources (Vultr Docs, 2023; Hackviser, 2023).

Each of the above changes should be made by editing /etc/ssh/sshd_config as root (e.g., using sudo nano /etc/ssh/sshd_config). After making these changes, save the file and restart the SSH service (using sudo systemctl restart sshd) for them to take effect (Vultr Docs, 2023).

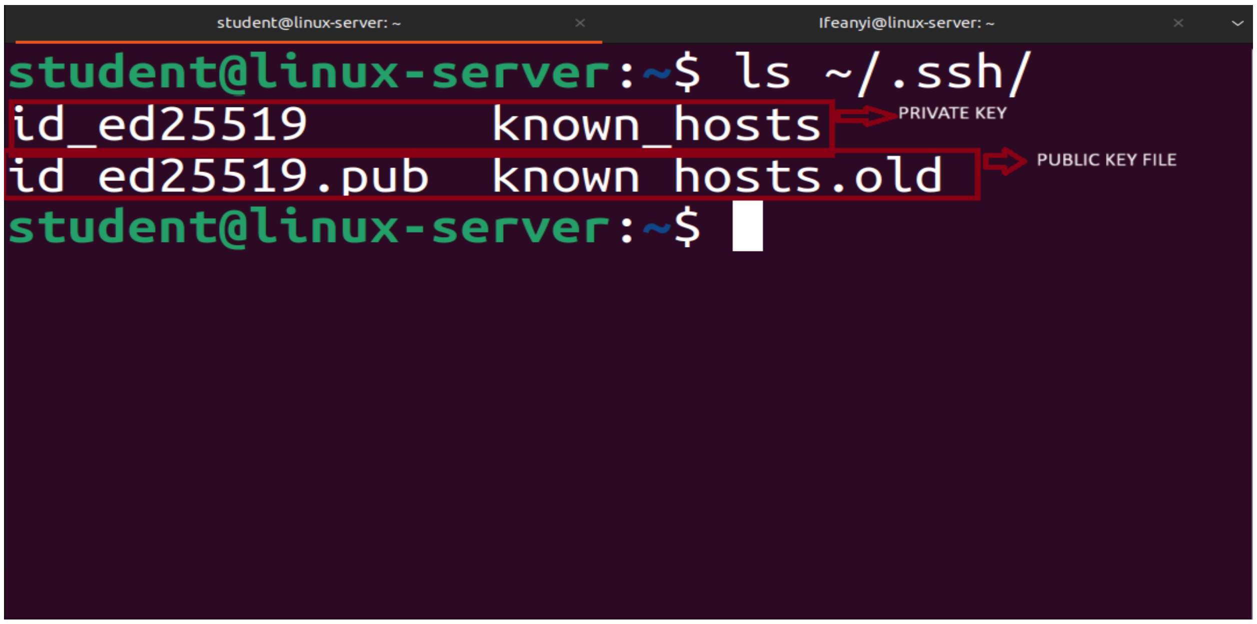
4. SSH Key-Based Authentication

4.1 Step 1: Generate SSH Key Pair

On the Client (student@linux_server Machine). Open a terminal and run: ssh-keygen -t ed25519 -C "student@email.com". Press Enter to accept the default save location: ~/.ssh/id_ed25519. This creates two files:

- ~/.ssh/id_ed25519 (private key — not to be shared)
- ~/.ssh/id_ed25519.pub (public key — to be shared with admin)

See image below:

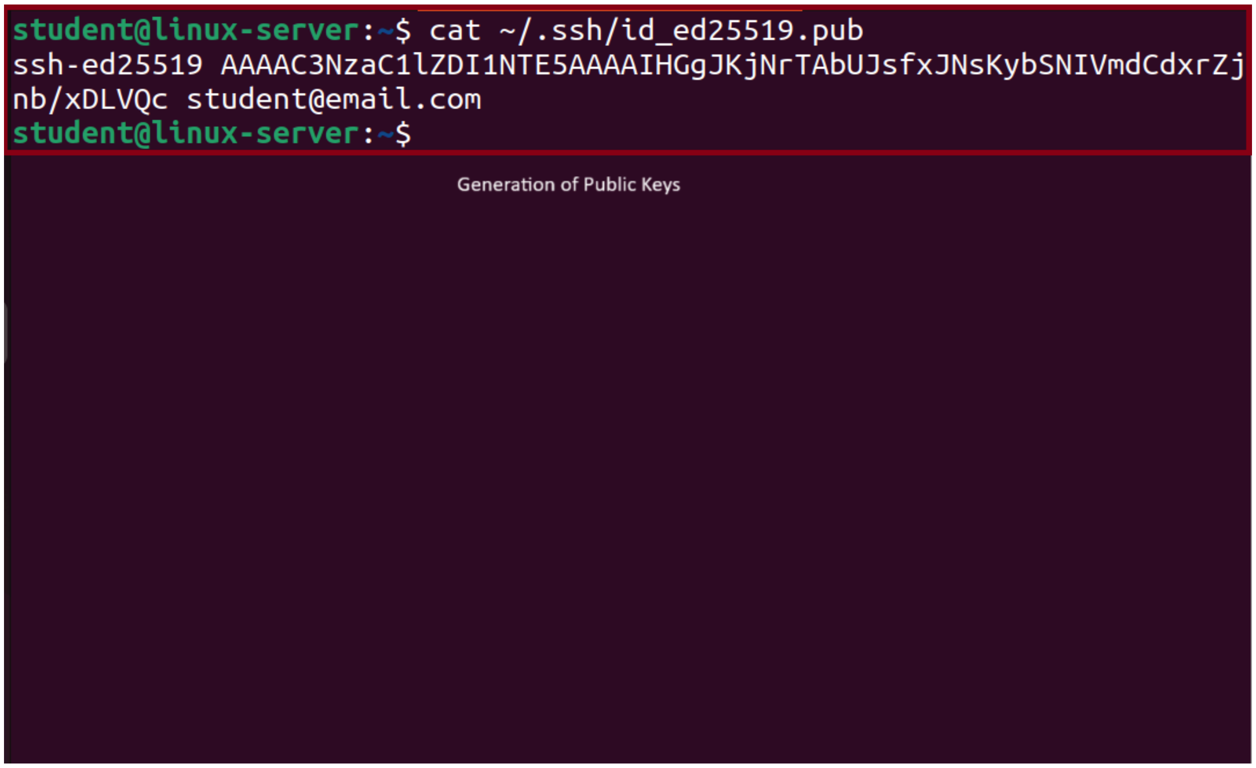


```
student@linux-server:~$ ls ~/.ssh/
id_ed25519          known_hosts
id_ed25519.pub      known_hosts.old
student@linux-server:~$
```

Image: showing the private key and public key files.

4.2 Step 2: View Your Public Key

- Use cat to print the public key: cat ~/.ssh/id_ed25519.pub
- Copy the entire output (starting with ssh-ed25519 ...)



```
student@linux-server:~$ cat ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIHGgJKjNrTAbUJsfxJNsKybSNIVmdCdxrZj
nb/xDLVQc student@email.com
student@linux-server:~$
```

Image: showing the Public key.

4.3 Step 3: Send the Public Key to Server Admin

- Create a new Admin user, Ifeanyi@linux_server with the command: sudo adduser Ifeanyi
- Switch to that user with this command: sudo su - Ifeanyi

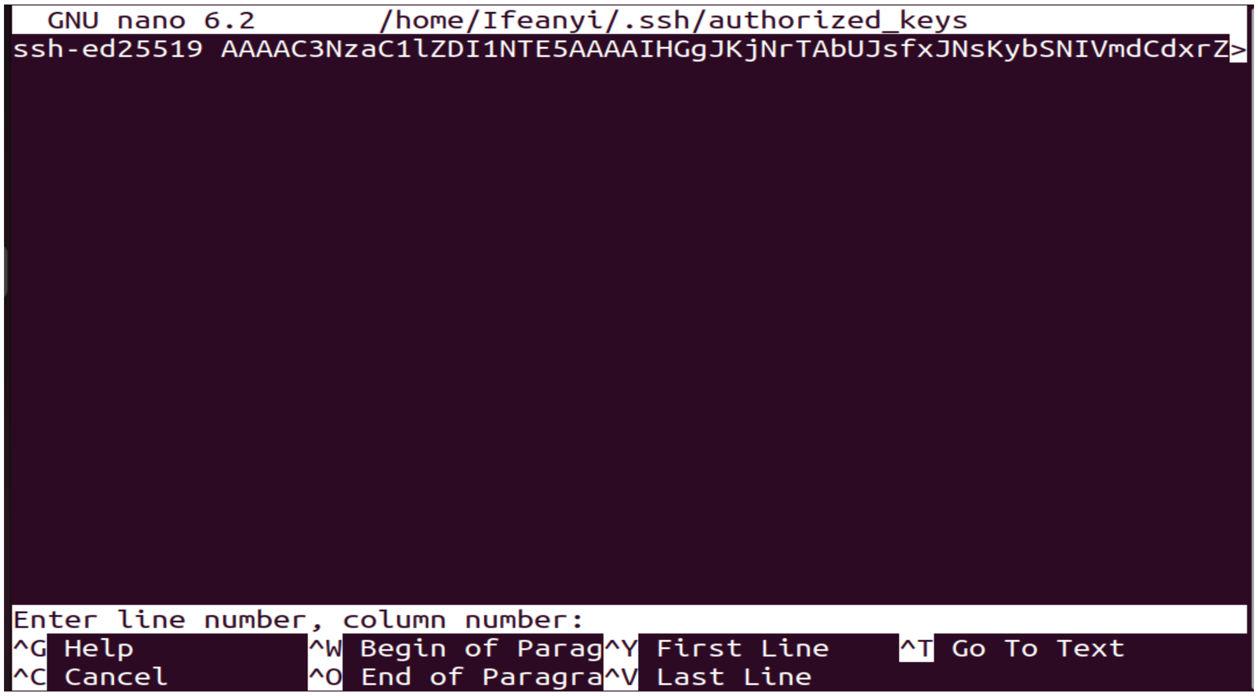
4.4 Step 4: Create the .ssh Directory with following commands on the terminal

- mkdir -p ~/.ssh
- chmod 700 ~/.ssh

4.5 Step 5: Add the Public Key

- Create the authorized_keys file using the nano command : nano ~/.ssh/authorized_keys
- Paste the entire public key received (see step 2). Save and exit (CTRL+X, Y, Enter).
- Set correct permissions: chmod 600 ~/.ssh/authorized_keys

See image below:



```
  GNU nano 6.2          /home/Ifeanyi/.ssh/authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIHGgJKjNrTAbUJsfxJNsKybSNIVmdCdxrZ>

Enter line number, column number:
^G Help              ^W Begin of Parag^Y First Line      ^T Go To Text
^C Cancel            ^O End of Paragra^V Last Line
```

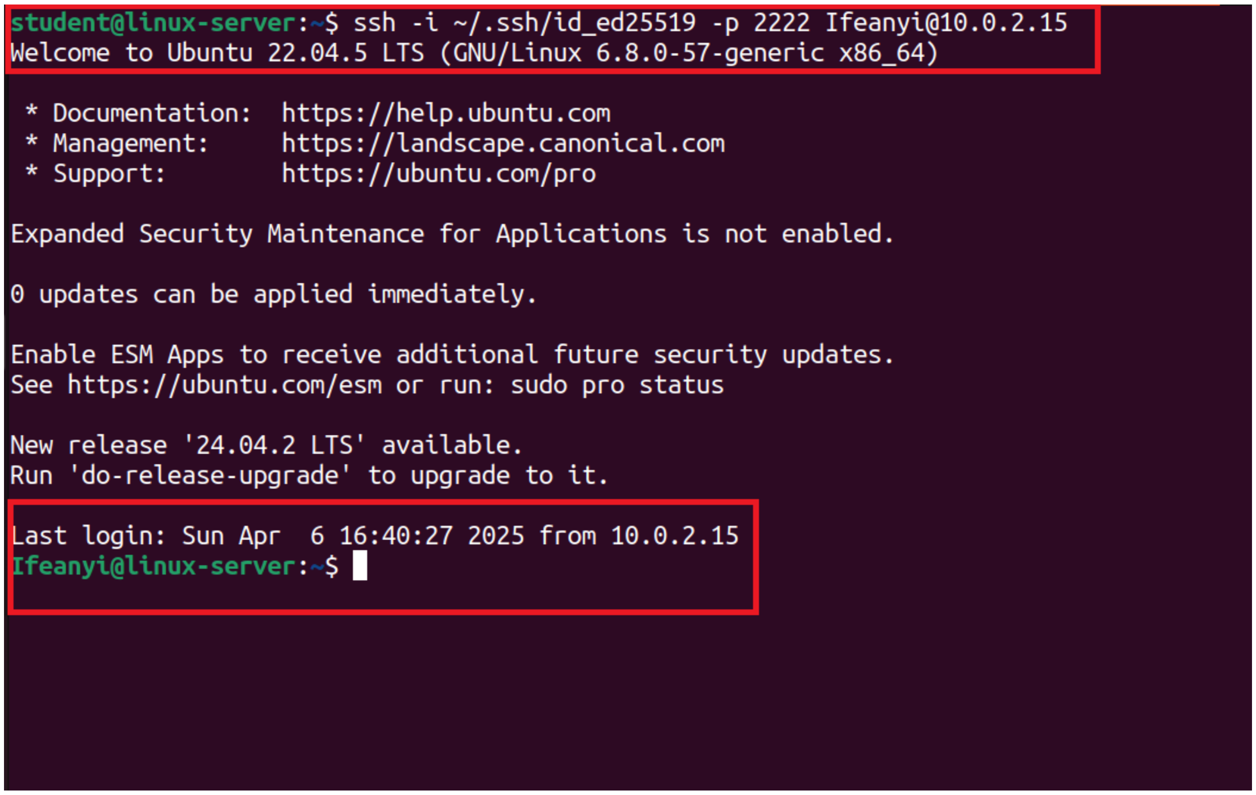
Image: showing the Authorized_keys

4.6 Step 6: Ensure Correct Ownership

- To ensure correct ownership, this command was used: sudo chown -R ifeanyi:Ifeanyi ~/.ssh

4.7 Step 7: Test SSH Login from Client

- On the client machine: student@linux_server
- Type this command: ssh -i ~/.ssh/id_ed25519 -p 2222 Ifeanyi@10.0.2.15 (see image below)
- This command tells the SSH client to use your private key (**-i**) file for authentication and to connect to port 2222 (instead of the default 22).



```
student@linux-server:~$ ssh -i ~/.ssh/id_ed25519 -p 2222 Ifeanyi@10.0.2.15
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-57-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '24.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Apr  6 16:40:27 2025 from 10.0.2.15
Ifeanyi@linux-server:~$ 
```

Image:showing the successful SSH connection.The Ubuntu client – student@linux-server successfully connecting to the Admin user – Ifeanyi@linux-server

These steps ensure that users create strong authentication keys and use them properly. The server admin only needs to install the public key once. From then on, the user can log in with the private key. This setup eliminates the need to transmit passwords over the network at all (Mozilla, 2023).

5. Attacks Mitigated by a Hardened SSH Configuration

Implementing the above SSH hardening measures significantly improves security and helps protect against several common attacks:

5.1 Brute-Force Password Attacks

With password authentication disabled, attackers cannot perform password guessing or dictionary attacks against the SSH server (TechTarget, 2022). Even if they scan the SSH port, there is no password prompt to abuse. Only key authentication is allowed, and guessing a 256-bit key is computationally infeasible. Additionally, disabling root login means attackers can’t target the superuser account at all, further reducing the attack surface (Hackviser, 2023).

5.2 Man-in-the-Middle (MITM) Attacks

SSH is designed to prevent MITM attacks through the use of host key verification and encryption. When configured with strong, up-to-date ciphers and protocols, SSH ensures that an attacker cannot easily insert themselves between the client and server to decrypt or alter traffic (Kagoshima, 2025). The client is warned if the server's host key changes unexpectedly (potential MITM), and all session data is encrypted (preventing a MITM from reading it). By contrast, Telnet has no defense against impostors or tampering. Our hardened setup, combined with users verifying host key fingerprints on first connection, thwarts most MITM attempts. An attacker would not only have to break the encryption (which is practically impossible with strong ciphers) but also fool the client into trusting a fraudulent host key, an extremely difficult task under SSH's design. In short, the hardened SSH ensures authenticity and confidentiality that protect against active network interception.

5.3 Credential Interception (Eavesdropping)

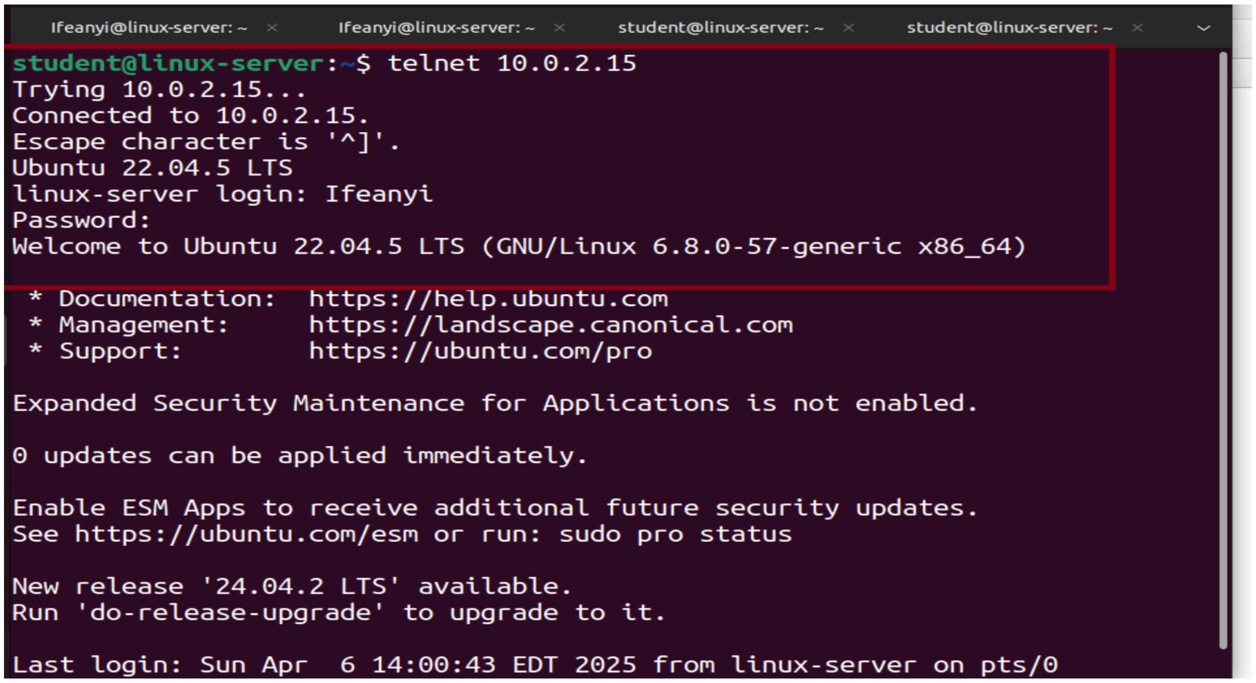
The use of strong encryption in SSH (and elimination of passwords in transit) protects against credential theft via network sniffing. An attacker spying on network traffic cannot read usernames, passwords, or session data from an SSH connection (Kagoshima, 2025). This counters any eavesdropping or "sniffing" attacks. Even a man-in-the-middle who records the encrypted packets can't extract the login credentials or commands due to the encryption. By using public key auth, even the act of authentication doesn't reveal a reusable secret (it's a challenge-response with keys), so credentials are safe from interception.

6. Network Traffic Analysis: Telnet vs. SSH (Wireshark)

To underscore the importance of using SSH over insecure protocols, we conducted a network traffic analysis comparing Telnet and SSH sessions. Wireshark, a network protocol analyzer, was used to capture and inspect the packets for each session. The findings vividly illustrate the fundamental security improvement SSH provides by encrypting traffic, as opposed to Telnet's plaintext communication.

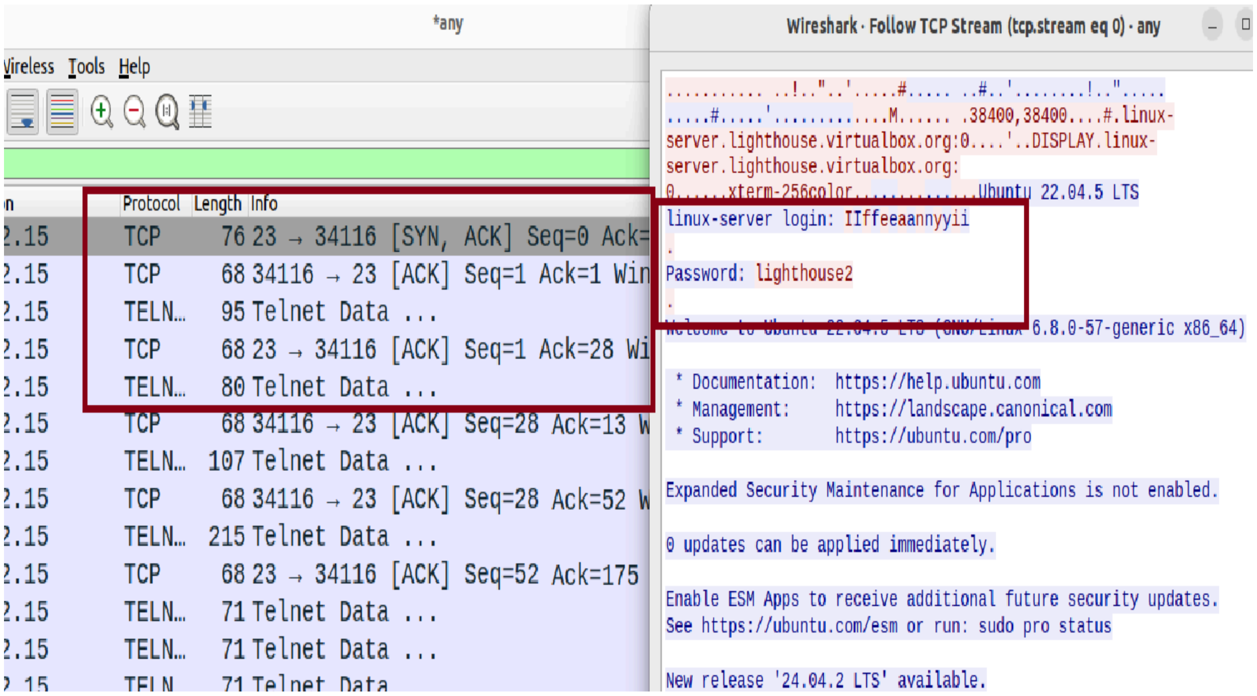
6.1 Telnet (Plaintext) Traffic

Telnet is a legacy protocol that operates on TCP port 23 and provides remote shell access. It does not encrypt any traffic. We initiated a telnet login using this command `telnet 10.0.2.15` as shown in the image below:



```
student@linux-server:~$ telnet 10.0.2.15
Trying 10.0.2.15...
Connected to 10.0.2.15.
Escape character is '^]'.
Ubuntu 22.04.5 LTS
linux-server login: Ifeanyi
Password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-57-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '24.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Apr  6 14:00:43 EDT 2025 from linux-server on pts/0
```

We also captured the Telnet login session in Wireshark, the results showed that everything – including the username and password – was transmitted in clear text across the network.



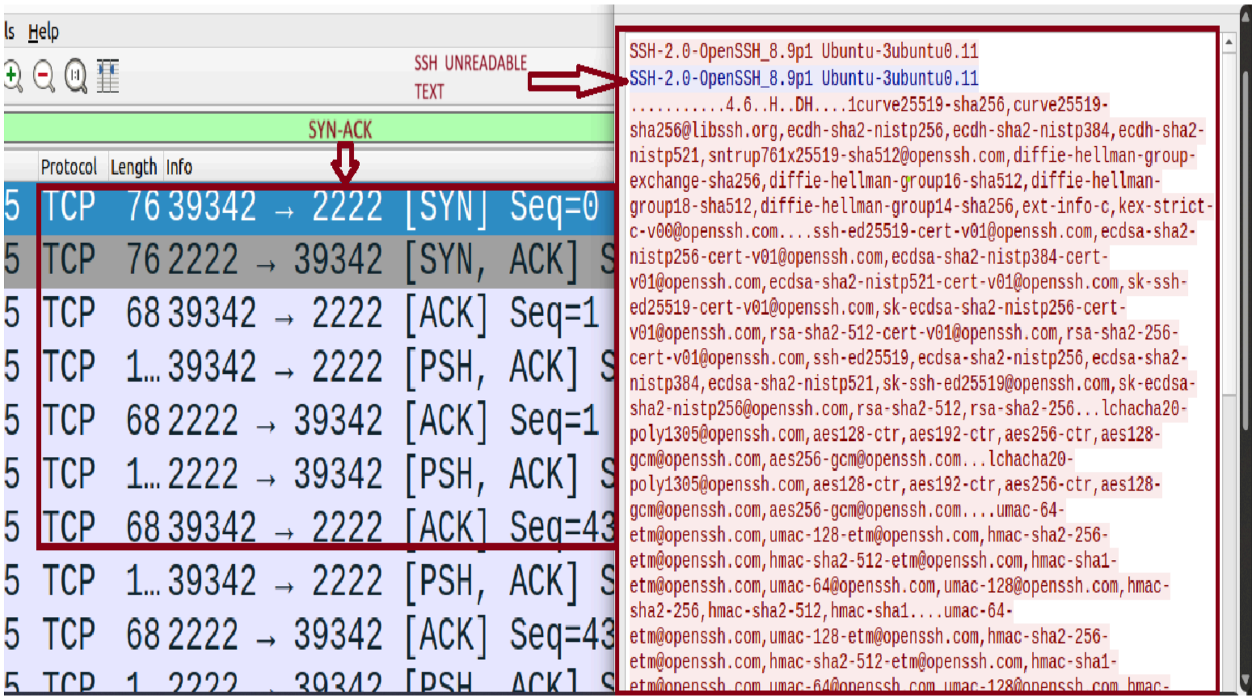
```
n     Protocol  Length Info
2.15  TCP       76 23 → 34116 [SYN, ACK] Seq=0 Ack=
2.15  TCP       68 34116 → 23 [ACK] Seq=1 Ack=1 Win
2.15  TELN…     95 Telnet Data ...
2.15  TCP       68 23 → 34116 [ACK] Seq=1 Ack=28 Wi
2.15  TELN…     80 Telnet Data ...
2.15  TCP       68 34116 → 23 [ACK] Seq=28 Ack=13 W
2.15  TELN…    107 Telnet Data ...
2.15  TCP       68 34116 → 23 [ACK] Seq=28 Ack=52 W
2.15  TELN…    215 Telnet Data ...
2.15  TCP       68 23 → 34116 [ACK] Seq=52 Ack=175
2.15  TELN…     71 Telnet Data ...
2.15  TELN…     71 Telnet Data ...
2.15  TELN…     71 Telnet Data
```

```
........... ..!.."..'.....#..... ..#..'........!..".....
.....#.....'...............M...... .38400,38400....#.linux-
server.lighthouse.virtualbox.org:0....'..DISPLAY.linux-
server.lighthouse.virtualbox.org:
0......xterm-256color...............Ubuntu 22.04.5 LTS
linux-server login: IIffeeaannyyii
.
Password: lighthouse2
.
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-57-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '24.04.2 LTS' available.
```

*Figure: Wireshark capture of a Telnet login. The packet bytes (followed TCP stream) reveal the login: **Ifeanyi** and Password: **Lighthouse2** in plain text. All keystrokes and responses are human-readable. An attacker eavesdropping on this traffic can easily intercept credentials and commands (Chandel, 2021; Kagoshima, 2025).*

6.2 SSH (Encrypted) Traffic

SSH runs on TCP port 22 and, unlike Telnet, encrypts all traffic between the client and server. We captured an SSH session (using this command ssh -i ~/.ssh/id_rsa Ifeanyi@10.0.2.15 -p 2222). In contrast to Telnet, the SSH session packets did not reveal any readable information about the contents of the communication. When following the TCP stream in Wireshark for the SSH session, instead of plaintext usernames or commands, one sees sequences of bytes that are not intelligible (see image below).



```
5 TCP  76 39342 → 2222 [SYN] Seq=0
5 TCP  76 2222 → 39342 [SYN, ACK] S
5 TCP  68 39342 → 2222 [ACK] Seq=1
5 TCP  1… 39342 → 2222 [PSH, ACK] S
5 TCP  68 2222 → 39342 [ACK] Seq=1
5 TCP  1… 2222 → 39342 [PSH, ACK] S
5 TCP  68 39342 → 2222 [ACK] Seq=43
5 TCP  1… 39342 → 2222 [PSH, ACK] S
5 TCP  68 2222 → 39342 [ACK] Seq=43
```

```
SSH-2.0-OpenSSH_8.9p1 Ubuntu-3ubuntu0.11
SSH-2.0-OpenSSH_8.9p1 Ubuntu-3ubuntu0.11
...........4.6..H..DH....1curve25519-sha256,curve25519-
sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-
nistp521,sntrup761x25519-sha512@openssh.com,diffie-hellman-group-
exchange-sha256,diffie-hellman-group16-sha512,diffie-hellman-
group18-sha512,diffie-hellman-group14-sha256,ext-info-c,kex-strict-
c-v00@openssh.com....ssh-ed25519-cert-v01@openssh.com,ecdsa-sha2-
nistp256-cert-v01@openssh.com,ecdsa-sha2-nistp384-cert-
v01@openssh.com,ecdsa-sha2-nistp521-cert-v01@openssh.com,sk-ssh-
ed25519-cert-v01@openssh.com,sk-ecdsa-sha2-nistp256-cert-
v01@openssh.com,rsa-sha2-512-cert-v01@openssh.com,rsa-sha2-256-
cert-v01@openssh.com,ssh-ed25519,ecdsa-sha2-nistp256,ecdsa-sha2-
nistp384,ecdsa-sha2-nistp521,sk-ssh-ed25519@openssh.com,sk-ecdsa-
sha2-nistp256@openssh.com,rsa-sha2-512,rsa-sha2-256...lchacha20-
poly1305@openssh.com,aes128-ctr,aes192-ctr,aes256-ctr,aes128-
gcm@openssh.com,aes256-gcm@openssh.com...lchacha20-
poly1305@openssh.com,aes128-ctr,aes192-ctr,aes256-ctr,aes128-
gcm@openssh.com,aes256-gcm@openssh.com....umac-64-
etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-256-
etm@openssh.com,hmac-sha2-512-etm@openssh.com,hmac-sha1-
etm@openssh.com,umac-64@openssh.com,umac-128@openssh.com,hmac-
sha2-256,hmac-sha2-512,hmac-sha1....umac-64-
etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-256-
etm@openssh.com,hmac-sha2-512-etm@openssh.com,hmac-sha1-
etm@openssh.com,umac-64@openssh.com,umac-128@openssh.com,hmac-
```

It's worth noting what happens during an SSH session setup, as seen in the packet capture. The SSH protocol first performs a cryptographic handshake: the client and server exchange protocol versions, then perform a key exchange (e.g., Diffie-Hellman) to establish a shared secret, and negotiate encryption algorithms. In Wireshark, one can see packets labeled like Client Key Exchange or Server Key Exchange, but the details of the keys are not revealed, they are exchanged in a secure manner. After this setup, the SSH connection enters an encrypted state. From that point on, all content (including any login password if we had used one, or the verification of a public key, and all terminal traffic) is encrypted using symmetric ciphers established during the handshake. Wireshark will show the packet lengths and timing, but the payload is opaque. Essentially, SSH provides both confidentiality and integrity: not only is the content encrypted, but SSH also uses message authentication codes to ensure that the data isn't tampered with in transit without detection (Kagoshima, 2025).

7. Final SSH Configuration (sshd_config)

- Link to the sshd_config file sshd_config_copy

The table below is the consolidated **/etc/ssh/sshd_config** reflecting the settings we implemented on the Ubuntu SSH server.

Configuration Parameter	**Value**	**Security Benefit**
PermitRootLogin	no	Prevents root-level breaches
PasswordAuthentication	no	Mitigates brute force attacks
PubkeyAuthentication	yes	Secures authentication with cryptographic keys
Port	2222	Prevents automated scans targeting port 22
Ciphers	AES256-GCM, ChaCha20, AES256-CTR	Strong encryption protecting traffic data
ClientAliveInterval	300	Automatically disconnect idle sessions

Conclusion

In today’s connected world, securing remote access isn’t just a best practice, it's a necessity. This project walked through the transformation of a default Ubuntu SSH server into a secure, well-defended entry point for remote system management. We didn’t just stop at getting SSH up and running; we followed industry-recognized guidelines from Mozilla (2023), DigitalOcean (2022), and NIST (2019) to harden it against real-world threats.

One of the first and most impactful changes was turning off root login and disabling password-based authentication. It is a simple step that dramatically reduces the risk of brute-force attacks. We leaned into modern encryption by using Ed25519 key pairs and AES-256 ciphers (RFC 4253, 2006), and we added extra protections like custom ports and idle session timeouts to limit exposure and keep access tight.

Through Wireshark analysis, the difference between Telnet and SSH became crystal clear. While Telnet exposed everything (usernames, passwords, and commands), SSH kept it all encrypted and secure. It’s a powerful visual reminder of why Telnet has no place in today’s networks (Chandel, 2021; Kagoshima, 2025).

Ultimately, hardening SSH is about more than just editing a config file. It’s about thinking like an attacker, building layered defenses, and making smart choices that protect systems and people. By applying what we’ve outlined in this project, anyone managing Ubuntu systems can create a safer, more trustworthy environment for remote work and administration.

References

1. Barrett, D. J., & Silverman, R. E. (2018). *SSH, The secure shell: The definitive guide* (2nd ed.). O'Reilly Media.

2. Chandel, R. (2021, April 28). Wireshark for Pentester: Password sniffing. *HackingArticles.in*. https://www.hackingarticles.in

3. Colombier, C. (2024, April 6). How to generate a secure and robust SSH key in 2024. *Dev.to*. https://dev.to

4. DigitalOcean. (2022, April 26). Alex Garnett – How to set up SSH keys on Ubuntu 22.04. *DigitalOcean Tutorials*. https://www.digitalocean.com/community/tutorials

5. Ganguly, H. (2020, December 29). Configure SSH server with key-based and two factor authentication. *Dev.to*. https://dev.to

6. GlobalKnowledge. (2018). Telnet vs SSH. *Global Knowledge Resource Library*. https://www.globalknowledge.com

7. Hackviser. (2023). *SSH Server Hardening Guidelines*. https://hackviser.com

8. Kagoshima, A. (2025, February 8). Why you should always use SSH over Telnet to access your network components securely. *Medium.com*. https://medium.com

9. Medium (A. Kagoshima). (n.d.). Why use SSH over Telnet? *Medium.com*. https://medium.com

10. Mozilla. (2023). Mozilla Infosec – OpenSSH security guidelines. https://infosec.mozilla.org

11. NIST. (2019). *NIST SP 800-123: Guide to general server security*. https://csrc.nist.gov/publications/detail/sp/800-123/final

12. NIST SSH security guidelines. (n.d.). *National Institute of Standards and Technology*. https://csrc.nist.gov/publications

13. OWASP. (2023). Brute force attacks. *OWASP Foundation*. https://owasp.org/www-community/attacks/Brute_force_attack

14. Psychz Networks. (2018, February 20). Raviteja – Telnet vs SSH. *Psychz.net*. https://www.psychz.net

15. Superuser Q&A. (n.d.). How does host key checking prevent man-in-the-middle attacks? https://superuser.com

16. TecMint – Saive, R. (2023, August 7). How to setup SSH passwordless login in Linux (3 easy steps). *TecMint.com*. https://www.tecmint.com

17. TechTarget. (2022). *SSH Best Practices to Protect from Attacks*. https://www.techtarget.com

18. Ubuntu Server Guide. (n.d.). https://ubuntu.com/server/docs

19. Vultr Docs. (2023). *How to Harden Server SSH Access. Vultr Docs*. https://docs.vultr.com

20. Wireshark Foundation. (2023). *Wireshark user guide*. https://www.wireshark.org/docs/wsug_html_chunked