Kwame Nkrumah University Of Science And Technology

FACULTY OF COMPUTER ENGINEERING

COE 456 SECURE NETWORK SYSTEM

END OF SEMESTER LAB PROJECT

Encryption & Firewalls

Student Name	Student ID
Gideon Adjei	3019520

Lecturer in charge:

Dr. E. Tchao

Submission Date : 7/08/2024

Contents

1 Introduction

In today's digital age, web servers are integral components of online communication, hosting websites and providing essential services. However, they are also prime targets for cyber attacks, ranging from data breaches to denial-of-service (DoS) attacks. Ensuring the security of web servers is crucial to protect sensitive information and maintain the integrity and availability of services.

The primary objectives of this lab are to enhance the security of a web server through the implementation of encryption, construction of robust firewalls[4], and secure remote access using SSH[6]. Encryption, specifically through SSL/TLS[3] certificates, ensures that the data transmitted between the web server and its clients remains confidential and tamper-proof. Firewalls act as a barrier, filtering incoming and outgoing traffic based on predetermined security rules, thereby protecting the server from unauthorized access and attacks. Additionally, SSH (Secure Shell) provides a secure method for remote server management, allowing encrypted communication between the client and server.

This lab report details the procedures for configuring a web server with SSL/TLS encryption, setting up firewall rules, and securely connecting to the server using SSH. By following these steps, we aim to demonstrate how these security measures can effectively protect a web server from common threats and vulnerabilities, contributing to a safer and more reliable online environment.

In the following sections, we will discuss the materials and methods used, provide a detailed procedure, present the results of our security configurations, and conclude with an analysis of the effectiveness of these measures. This lab underscores the importance of implementing comprehensive security strategies to safeguard web servers against the ever-evolving landscape of cyber threats.

2 Objectives and Learning Outcomes

1. The Objectives of this lab is to:

   (a) Connect to a remote server with SSH: Learn how to securely access and manage a remote server using SSH (Secure Shell). This involves initiating a secure connection from your local machine to the remote server, authenticating with either a password or SSH key, and executing commands or transferring files remotely. Gain practical experience in using SSH to perform administrative tasks and ensure secure communication between your client and the server.

   (b) Configure Firewalls:

   Gain hands-on experience in setting up firewall rules to control and filter incoming and outgoing traffic, providing an additional layer of security for the web server.

   (c) Implement Encryption:

   Learn how to obtain and install TLS certificates to encrypt data transmitted between the web server and its clients, ensuring secure communication.

2. The learning objectives of this lab are to understand the importance of web server security, implement SSL/TLS certificates for encrypted communication, configure firewall rules to protect the web server, connect to a remote server using SSH, verify and test these security configurations, identify and troubleshoot security issues, and effectively document and communicate the procedures and outcomes.

3 Equipments

- Personal computer with/without virtual box, and with at least 4GB of RAM on host machine if running vm and at least 1GHz dual core CPU.
- Tools Needed To be Installed on PC:

  1. vscode
  2. docker
  3. kathara
  4. wireshark

4 Methodology

4.1 Configurations of Network with Kathara

4.1.1 Emulation and Simulation

1. Kathara was used to emulate the a network shown in Figure 1.
2. The image used to create the docker containers for each device was instructed to be **rantwi/kathara:proj** except the wireshark. uses . The lab.conf is as shown below.

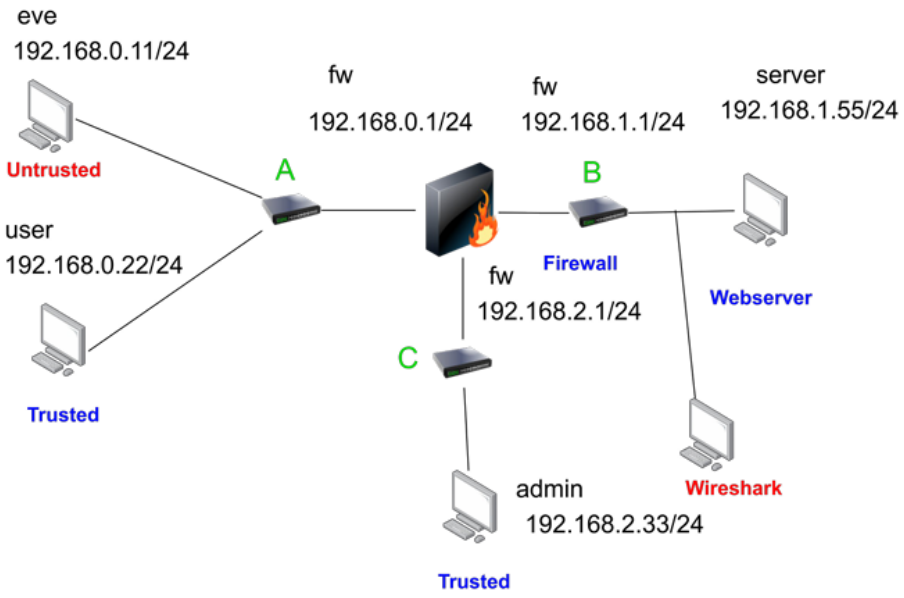

Figure 1: Network Overview

Listing 1: lab.conf

```
eve[0]=A
eve[image]="rantwi/kathara:proj"
eve[ipv6]="false"

user[0]=A
user[image]="rantwi/kathara:proj"
user[ipv6]="false"

admin[0]=C
admin[image]="rantwi/kathara:proj"
admin[ipv6]="false"

fw[0]=A
fw[1]=B
fw[2]=C
fw[image]="rantwi/kathara:proj"
fw[ipv6]="false"

server[0]=B
server[image]="rantwi/kathara:proj"
server[ipv6]="false"

wireshark[bridged]="true"
wireshark[port]="3000:3000"
wireshark[image]="lscr.io/linuxserver/wireshark"
wireshark[num_terms]=0
```

The wireshark uses lscr.io/linuxserver/wireshark image.

4.1.2 Device Configuration

1. Startup files (shell commands to be executed upon creation of containers are added to configure the network. The startup files is as shown below:

```
ip address add 192.168.0.11/24 dev eth0  #configure IP
ip route add default via 192.168.0.1     #set default gateway
```

eve.startup

```
  #Routings
ip address add 192.168.0.1/24 dev eth0
ip address add 192.168.1.1/24 dev eth1
ip address add 192.168.2.1/24 dev eth2
```

fw.startup

```
ip address add 192.168.2.33/24 dev eth0
ip route add default via 192.168.2.1
```

admin.startup

```
ip address add 192.168.0.22/24 dev eth0
ip route add default via 192.168.0.1
```

user.startup

```
ip address add 192.168.1.55/24 dev eth0
ip route add default via 192.168.1.1
systemctl start apache2 #optional
```

server.startup

2. The lab is then started with:

```
kathara lstart    #command to start kathara lab
```

3. 5 containers appear as eve, user, fw, admin and server as shown in Figure 2 below.

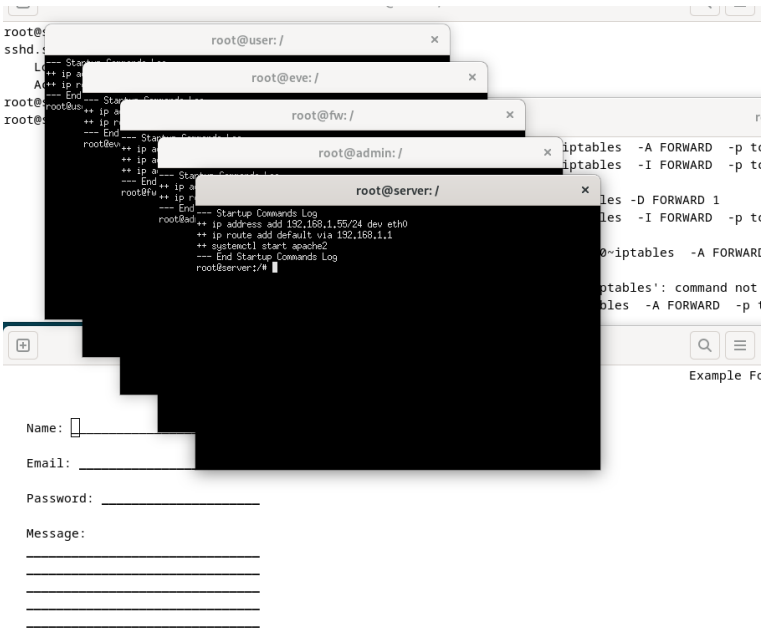

Figure 2: The started lab

4.1.3 Creating Users

1. I added new user to server, admin and fw:

```
adduser shoeisha_serverm   #on server
adduser shoeisha_admin     #on admin
adduser shoeisha_fw        #on fw
```

4.1.4 Testing

1. The various containers were pinged from one to another to make sure the network is correctly configured. For instance from the user the server is pinged with:

```
ping 192.168.1.55     #pinging server from user
```

   The result is is shown in :

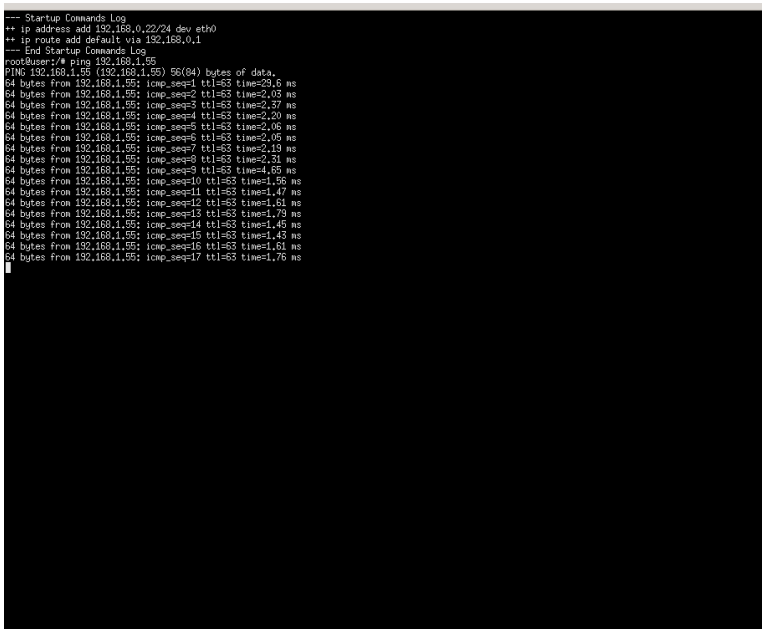

Figure 3: The pinging result

4.2 FIREWALL RULES

4.2.1 SSH

1. I configure a firewall on fw to reject all ssh packets going to the server and observed the traffic in the network.

```
#block all ssh packects to server.
iptables -A FORWARD -p tcp -d 192.168.1.55 --dport 22 -j REJECT
```

2. I repeated 1 to allow only admin to access server through ssh

```
#block all ssh packects to server.
iptables -I FORWARD -p tcp -s 192.168.2.33 -d 192.168.1.55 --dport
22 -j ACCEPT
```

3. I tried to achieve an ssh connection to the server from the various devices and observed the traffic. I made sure I run the ssh daemon on the server to listen at port 22(common port for SSH) for such requests.

```
ssh shoesisha_server@192.168.1.55 # connect to server
systemctl start sshd # start an ssh daemon
systemctl status sshd # check the status of a ssh daemon
```

4.2.2 HTTP

1. SSH was used to connect to the server remotely from admin to configure the server to web-server.

```
su - shoesisha # switch user
               #(- indicates to home directory)
touch index.html # create index.html file
```

```
touch submit.php
scp * shoesisha_server@192.168.1.55:/var/www/html
```

2. I started web server daemon(apache)[2] on server to serve clients with index.html and submit.php.

```
systemctl start apache2 # starts the apache webserver
systemctl status apache2 # confirm
kathara lconfig -n wireshark --add B
```

The wireshark was configured as shown on figure 1. The wireshark was used to analyse traffic to the server.

The index.html and submit.php was filled as shown below:

```
<!DOCTYPE html>
<html>
<head>
   <title>Example Form</title>
</head>
<body>
   <h1>Example Form</h1>
   <form action="submit.php" method="post">
       <label for="name">Name:</label>
       <input type="text" id="name" name="name" required><br><br>
       <label for="email">Email:</label>
       <input type="email" id="email" name="email" required><br><br>
       <label for="email">Password:</label>
       <input type="password" id="password"
       name="password" required><br><br>
       <label for="message">Message:</label>
       <textarea id="message" name="message" rows="5"
       cols="30" required></textarea><br><br>
       <input type="submit" value="Submit">
   </form>
</body>
</html>
```

index.html

```php
<?php
 if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Retrieve form data
    $name = $_POST["name"];
    $email = $_POST["email"];
    $message = $_POST["message"];
    $password = $_POST["password"];
    // Display the submitted data
    echo "Name: " . $name . "<br>";
    echo "Email: " . $email . "<br>";
    echo "Password: " . $password . "<br>";
    echo "Message: " . $message . "<br>";
 }
 ?>
```

submit.php

3. I configured a firewall on fw to reject all http packets going to the server from network A from figure 1 and observed the traffic in the network.

```
#reject all A
iptables  -A FORWARD  -p tcp  -s 192.168.0.0/24  --dport 80  -j REJECT
```

4. Step 1 is repeated to allow only user to access website from network A

```
#allow only user from network A
iptables  -I FORWARD  -p tcp  -s 192.168.0.22  -d 192.168.1.55  --dport
80  -j ACCEPT
```

The resulting traffic for all configurations were observed.

4.3 TLS ENCRYPTION

In practice, certificates are acquired from trusted authorities but for the purpose of this lab, a self-assigned certificate was generated with openssl[3].

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048
keyout /etc/ssl/private/apache-selfsigned.key -out
/etc/ssl/certs/apache-selfsigned.crt

#the following was answers to the prompts
 Country Name (2 letter code) [XX]:GH
 State or Province Name (full name) []:Ashanti Region
 Locality Name (eg, city) [Default City]:Kumasi
 Organization Name (eg, company) [Default Company Ltd]:KNUST
 Organizational Unit Name (eg, section) []:Computer Engineering
```

```
Common Name (eg, your name or your servers hostname)
[]:192.168.1.55
Email Address []:shoesisha@example.com
```

After creating a self assigned certificate on the server, I configured the sever to use the .key and .cert to encrypt every traffic.

```
#create and edit sns.conf
 nano /etc/apache2/sites-available/sns.conf
```

The sns.conf is filled with:

```
 <VirtualHost *:443>
ServerName 192.168.1.55
DocumentRoot /var/www/html
SSLEngine on
SSLCertificateFile /etc/ssl/certs/apache-selfsigned.cr
SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key
</VirtualHost
```

sns.conf

The configuration is enabled by the following commands:

```
a2enmod ssl
a2ensite sns.conf
apache2ctl configtest
systemctl reload apache2
```

Traffic is then analysed using wireshark by making https requests from user eve and admin using:

```
links https://192.168.1.55
```

5 Result

5.1 FIREWALL RULES

5.1.1 SSH

The following sections shows the results of configuring each rule on the firewall.

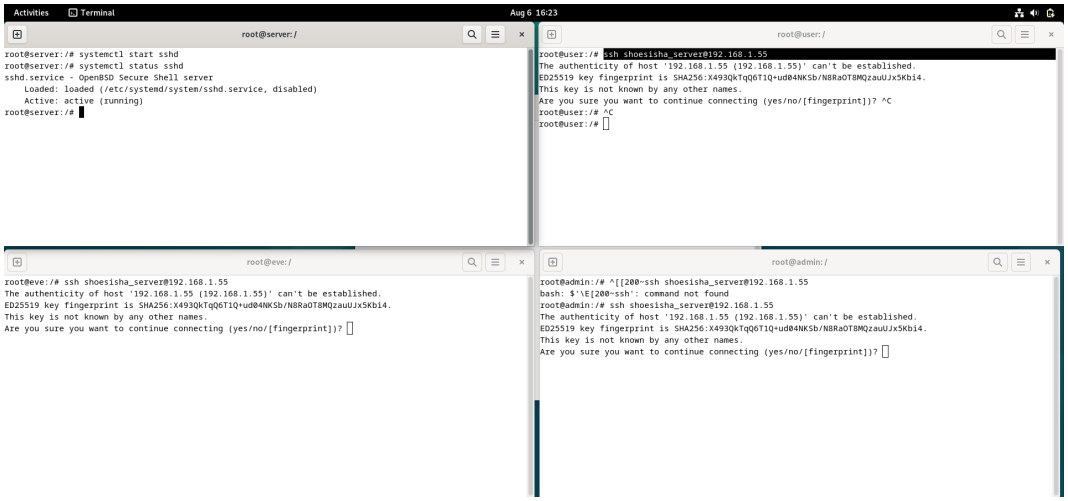

Figure 4: No firewall Rule

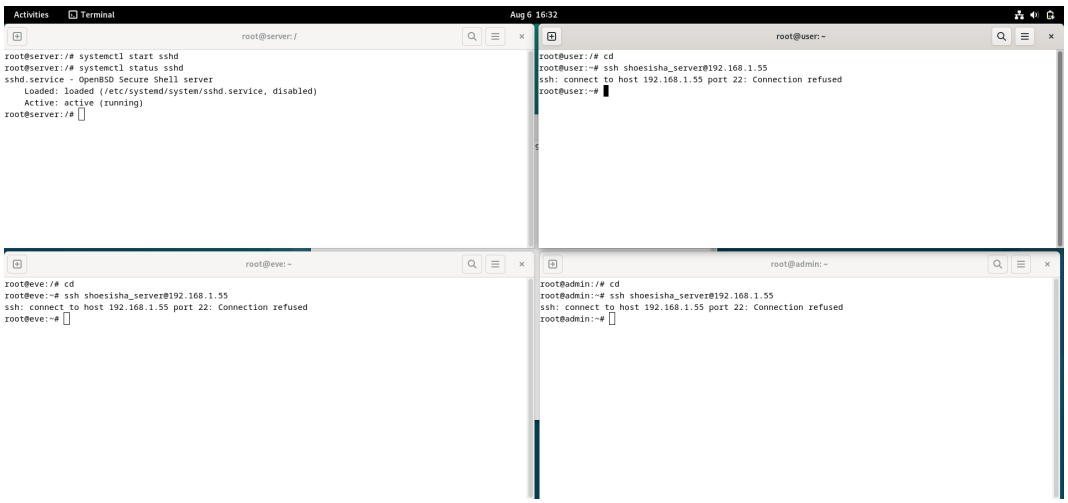

Figure 5: Reject every ssh packet rule to server

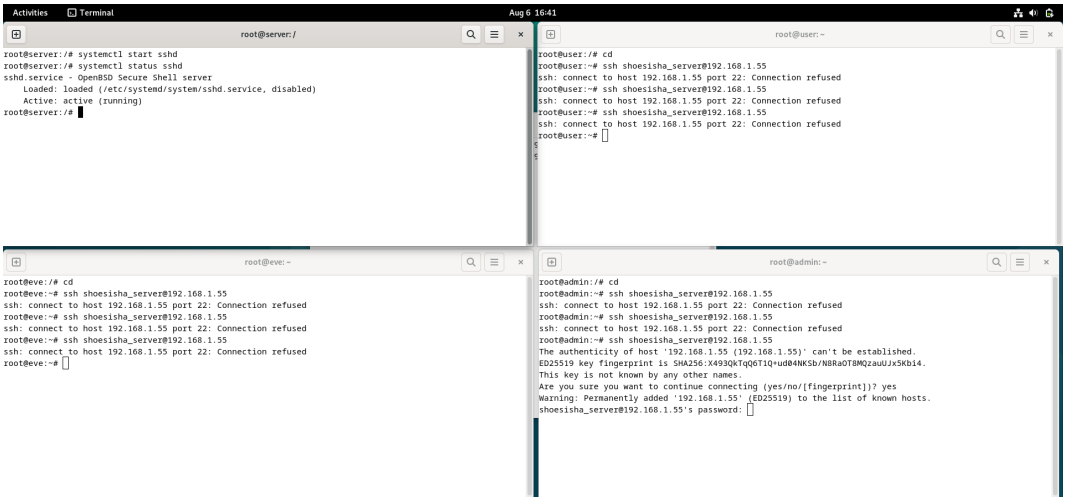

Figure 6: Forward packets from admin

5.1.2 HTTP

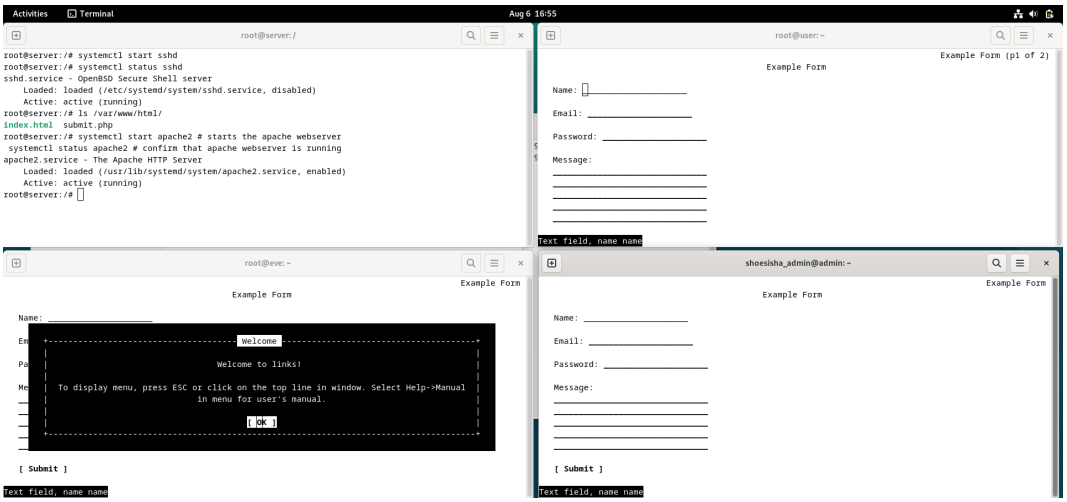

Figure 7: No http firewall rule

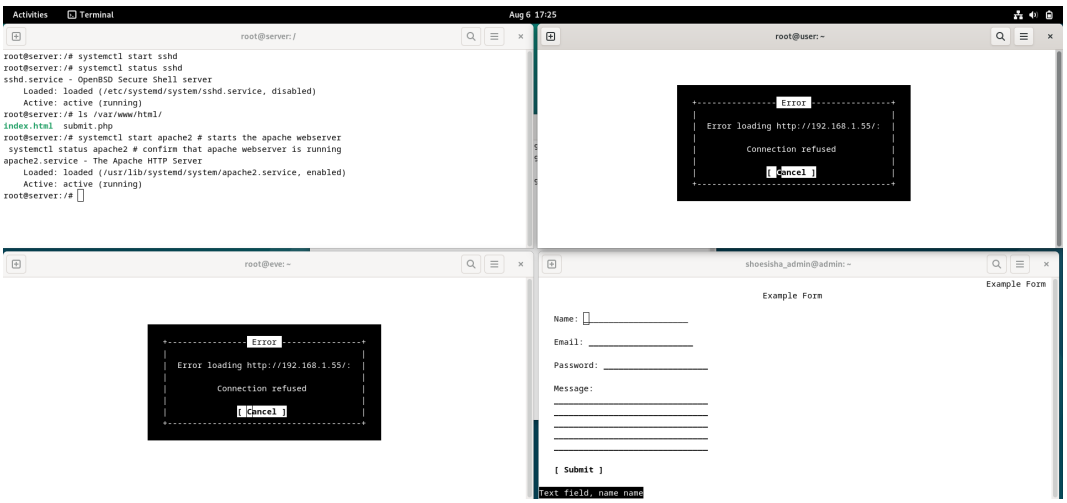

Figure 8: Reject http requests from network A

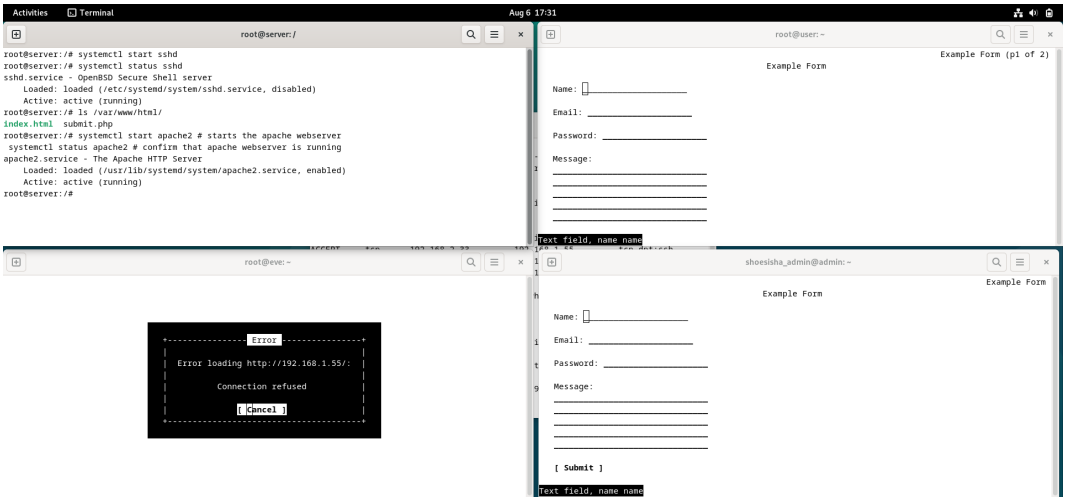

Figure 9: Forward http packets from user

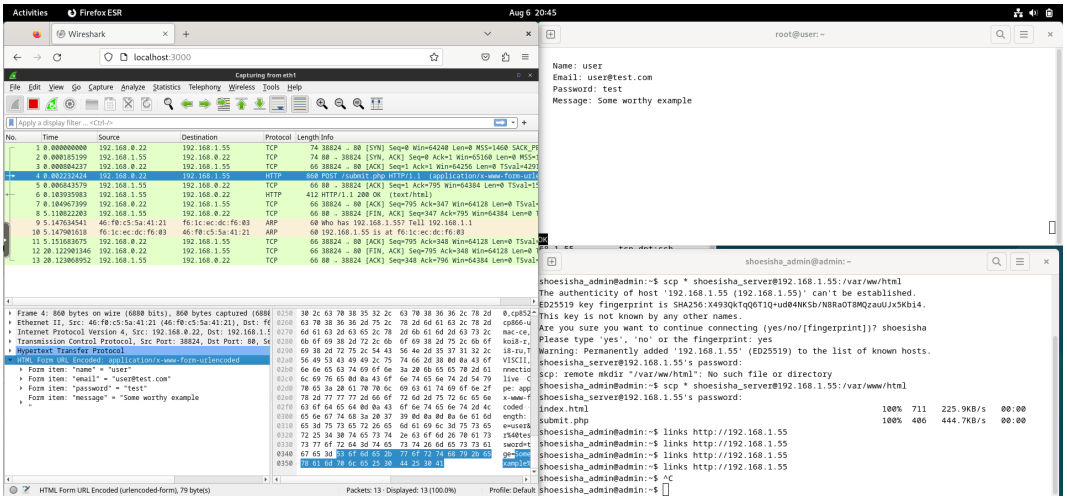

Figure 10: Http packets are not encrypted

5.2 TLS ENCRYPTION

5.2.1 Results of Acquiring Certificate

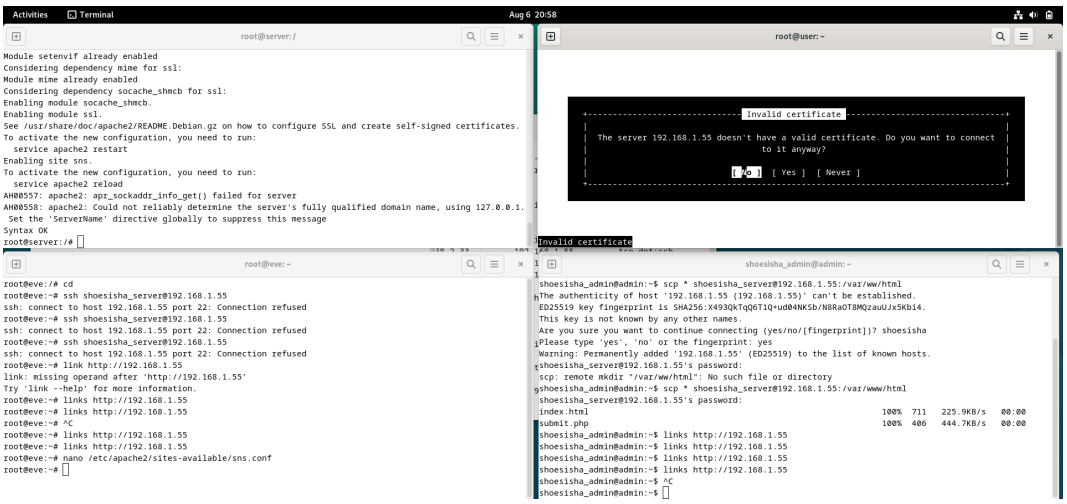

Figure 11: With a secured http

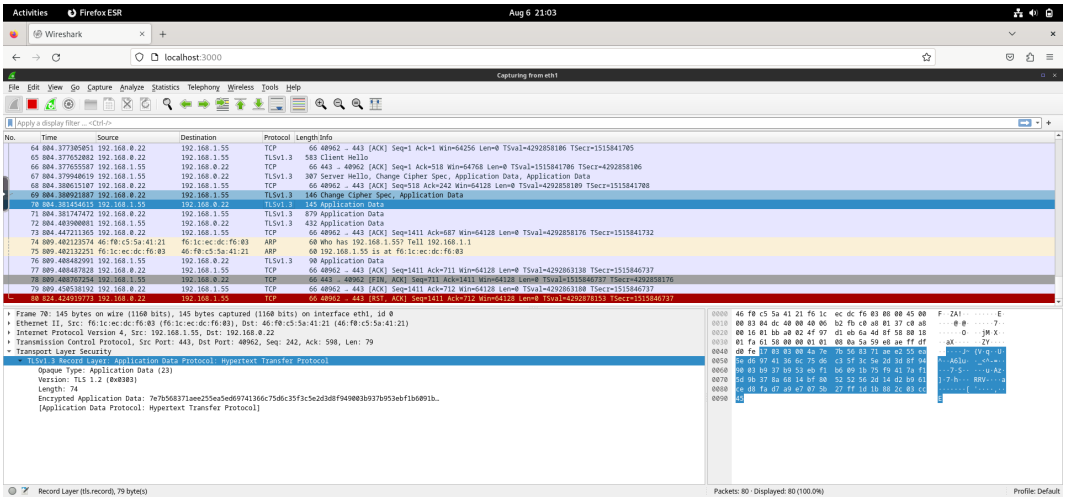

Figure 12: Traffic with TLS Encryption

6 Discussion

The primary objectives of this lab were to secure a web server by implementing SSL/TLS encryption, configuring firewall rules, and establishing a secure SSH connection. The aim was to ensure data confidentiality, integrity, and secure remote access.

The successful implementation of SSL/TLS encryption was verified by the ability to access the web server via HTTPS. Firewall rules were correctly configured to restrict access to essential services only, thereby enhancing the server's security. Additionally, SSH access was established using key-based authentication, providing a secure method for remote management.

6.1 FIREWALL CONFIGURATION

6.1.1 SSH

In this part of the lab, after setting up the lab and making sure everything is working and starting the sshd daemon on the server, I tried achieve an ssh connection to the server from each device on the network. The result as show in figure 4, indicate that the devices can access the remote server using ssh. This is desired, as it poses serious security threat because accessing the terminal of a devices grants a the user a lot of privilages. [1].

A firewall is configured to reject all packets that are going to the server on the router(fw). The result as shown in figure 5, indicate that no devices on other networks could access the server. However We would to give remote access to the server to the admin.

In order to grant admin access, another firewall rule is inserted into the ACL(access control list) from the firewall rules to forward packets coming from admin. From the result, as shown in figure 6, only admin can access the server.

6.1.2 HTTP

In this part of the lab, I made use of the ssh remote access to the server from the admin to configure server to process http request. This is done in practice to help administrators to remotely access servers securely and then manage these servers.

After configuring the server and starting apache2 web-server on the server, all devices could access the website including eve as shown in the result in figure 7.

However, because there bad actors on untrusted networks, we need to secure the web-server. I configured a firewall rule to reject all packets coming from the untrusted network(network A). The results, as shown in figure 8, indicates only people outside network A can access the website. This rule was appended to the list.

Not all devices on network A are not trusted, so I added a rule to the firewall ACL to allow only trusted devices(user in this case) to access the website. The restult is shown in figure 9.

6.2 TLS ENCRYPTIONS

Although, firewall secures the website, it is not enough to prevent a bad actor from for example eavesdropping. The wireshark in figure 1 is for instance eavesdropping all packets going to the server. In figure 10, it could be seen that the data sent from user could easily be seen from the http connection.

In practice, certificates are obtained from trusted authorities like let's encrypt, digicert[1] and many others. In this lab however, I created a self assigned certificate using openssl [3].

After securing the website with the certificate created, I made request to the secured website through https(usually on port 443) rather than http(port 80). The result as shown in figure 11. The result shows a warning about the certificate not been trusted. This is because I assigned it myself and of course I am not a well known trusted certificate granter and web browsers know the trusted ones.

Encryption prevents eavesdroppers from getting the data sent over the network. As shown in the wireshark capture in figure 12, although it can still get the data sent, the data is in ciphertext form.

7 Conclusion

In this lab, we successfully implemented key security measures for a web server, including SSL/TLS encryption, firewall configuration, and secure SSH access. These actions ensured data encryption, robust traffic control, and secure remote management, significantly enhancing the server's security. The results demonstrated the importance of a comprehensive security strategy combining encryption, access control, and authentication. Despite the lab's limitations, such as its simplified environment, it provided valuable hands-on experience and underscored the need for multi-layered security approaches. Future research should explore more complex environments and additional security layers to further improve web server defenses.

8 References

[1] digicert CA https://www.digicert.com/blog/what-is-a-certificate-authority.

[2] Apache HTTP Server Version 2.4 Documentation https://httpd.apache.org/docs/2.4/.

[3] OpenSSL Documentation https://www.openssl.org/docs/

[4] Iiptables - Administration tool for IPv4 packet filtering and NAT https://netfilter.org/projects/iptables/index.html

[5] Iiptables - latinsud https://www.latinsud.com/pub/iptables/

[6] OpenSSH 8.8 Release Notes https://www.openssh.com/releasenotes.html