EAST WEST UNIVERSITY

Project 1 Report

Securing a networked system with Public Key Infrastructure Implementing Transport Layer Security on HTTP for https:// connection

Presentation Link

https://drive.google.com/file/d/1tRlxFwEz4eLwpKyiwaFjzH4BOItOMheX/view?usp=drive_link

Course Title: Cybersecurity, Law, and Ethics

Course Code: CSE487

Section: 04

Submitted to

Muhammed Yaseen Morshed Adib

Lecturer, Department of Computer Science & Engineering,

East West University

Submitted by

Group Member Name	ID
Zihad Khan	2022-2-60-107
Nusrat Jaben Aurnima	2022-2-60-146
Shairin Akter Hashi	2022-2-60-102
Md.Shahrukh Hossain Shihab	2022-1-60-372
Tahmina Ahmed	2022-2-60-151

Date of Submission: 29.11.2025

Table of Contents

1. Introduction

It is important to secure any communication taking place over the internet, for allowing sensitive information to stay protected and transactions to be authenticated in a digital world. Public Key Infrastructure (PKI) and Transport Layer Security (TLS) are the primary technologies that can create security for communication over the internet. PKI and TLS are used together to form a secure and trusted computing environment, allowing data to be sent securely between clients and servers that may have otherwise not place any trust in each other. By utilizing strong cryptographic methods secured to certificate-based authentication, PKI and TLS defend against eavesdropping, impersonation (man in the middle), and unauthorized access to information across modern networks.

Public Key Infrastructure (PKI)

Public Key Infrastructure (PKI) is a security framework that uses cryptographic keys to encrypt & decrypt information and provide services such as authentication, encryption, and digital signatures. PKI has two keys commonly known as public key and private key. The public key can be shared with anyone and is used for encrypting the information. The private key is a secret and is used for decrypting the information. PKI operates with Certificate Authorities (CAs) who are trusted authorities that issue digital certificates planning the assurance of an identity (server or client), also sometimes known as trust chaining.

- Certificates, Digital evidence/documents that are issued from the CA to validate a public key.
- Certificate Authority (CA). A trusted organization that delivers and manages digital certificates.
- Registration Authority (RA). The CA engages an RA to validating the identity of an entity, in advance of issuing a digital certificate.

PKI is important for establishing security, trust, and data integrity in networked systems by managing cryptographic keys and certificates. PKI is also important to prevent impersonation attacks by providing identity confirmation for users or devices, as well as data integrity by satisfying the integrity of transmitted data. Within PKI a user is provided a digital signature to confirm that the information/data received by the user is the same as the original message. PKI provides non-repudiation as the digital signature provides both accountability and a legal framework.

Transport Layer Security (TLS)

Transport Layer Security (TLS) is a new protocol that has replaced the earlier Secure Sockets Layer (SSL) protocol for securing data transferred between clients and web servers. The purpose of the SSL protocol was to protect data transferred, to some degree, from eavesdropping, tampering, or impersonation to enable a secure transfer of sensitive data like passwords, all types of personal information, or financial information. The TLS protocol is used with the Hyper Text Transfer Protocol (HTTP) to create Hyper Text Transfer Protocol Secure (HTTPS) to provide secure connections to web-based applications and is part of any online banking application or e-commerce application, or other applications that collect or manage personal information. HTTPS was created for multi-feature capability for secure connections, trust, and user privacy while also enabling PKI mechanism for certificate management as well as enabling TLS for encrypting traffic.

Requirements

- Configuration of Certification Authority AcmeCA with AcmeRootCA as the RootCA.
- Configuration of the Web Server with Apache2 on a Linux Host.
- DNS configuration for www.verysecureserver.com
- Firewall configuration to allow necessary ports (53, 80, 443) only CSR Configuration and Generation for the www.verysecureserver.com
- Transferring the CSR to AcmeCA.
- Certification process (Verification and Certificate Generation from CSR)
- Transferring the certificate from AcmeCA to www.verysecureserver.com
- Installation of the signed SSL certificate in the server of www.verysecureserver.com
- Making the system trust Acme-RootCA
- Implementation of a simple file uploading page in the server.
- Verifying the security of the connection by inspection (the padlock icon), and with Wireshark from another computer.

2. System Requirements and Setup

Download and install necessary components:

SL	Component	Install
1	Oracle Virtual Box	Version 7.1.12
2	OS Ubuntu	Version 24.04.3 LTS
3	Mozilla Firefox 145.0.1	Pre Installed
4	Gedit	Sudo apt install gedit
5	Openssl	Sudo apt install openssl
6	Bind9 DNS Server	Sudo apt install Bind9
7	UFW Firewall	Sudo apt install ufw
8	Snort IDS	Sudo apt install snort

3. Configuration

3.1 Certification Authority Setup

1. Move to root directory

```
sudo -i
```

2. Create CA Folder Structure

Create certification authority folder named ca, containing root-ca, sub-ca, server and each sub folder will contain private folder containing certs, newcerts, crl, csr folders.

```
mkdir -p ca/{root-ca,sub-
ca,server}/{private,certs,newcerts,crl,csr}
```

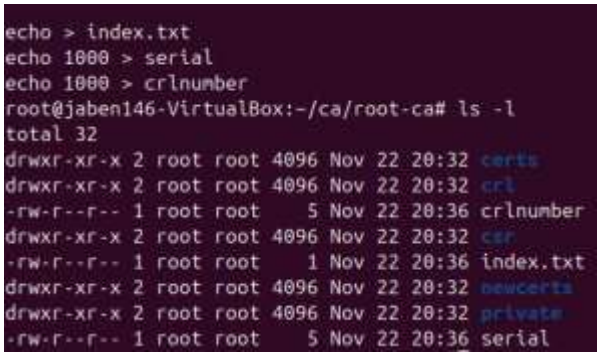
3. See the tree of files inside the root

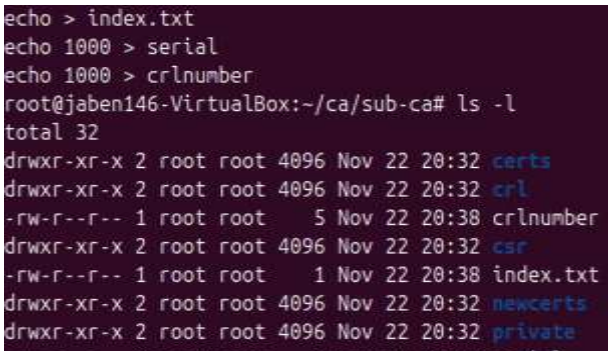
```
tree /root/ca
```

3.2 Initialize Databases (Root & Sub-CA)

```
# ROOT-CA DB
cd /root/ca/root-ca
echo > index.txt
echo 1000 > serial
echo 1000 > crlnumber
ls -l

# SUB-CA DB
cd /root/ca/sub-ca
echo > index.txt
echo 1000 > serial
echo 1000 > crlnumber
ls -l
```





3.3 Generate Private Keys

```
cd /root/ca
```

 # Root-CA key

```
openssl genrsa -aes256 -out root-ca/private/ca.key 4096
PEM PASS: root1234
```

 # Sub-CA key

```
openssl genrsa -aes256 -out sub-ca/private/sub-ca.key 4096
PEM PASS: sub1234
```

 # Server key

```
openssl genrsa -out server/private/server.key 2048
```

```
root@jaben146-VirtualBox:~/ca# openssl genrsa -aes256 -out root-ca/private/ca.key 4096
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
root@jaben146-VirtualBox:~/ca# openssl genrsa -aes256 -out sub-ca/private/sub-ca.key 40
96
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
root@jaben146-VirtualBox:~/ca# openssl genrsa -out server/private/server.key 2048
root@jaben146-VirtualBox:~/ca#
```

 # Lock down private directories

```
chmod 700 root-ca/private sub-ca/private server/private
```

 # Check

```
ls -l root-ca/private sub-ca/private server/private
```

```
root@jaben146-VirtualBox:~/ca# ls -l root-ca/private sub-ca/private server/private
root-ca/private:
total 4
-rw------- 1 root root 3434 Nov 22 20:45 ca.key

server/private:
total 4
-rw------- 1 root root 1704 Nov 22 20:45 server.key

sub-ca/private:
total 4
-rw------- 1 root root 3434 Nov 22 20:45 sub-ca.key
root@jaben146-VirtualBox:~/ca#
```

3.4 ROOT-CA Config + Certificate

1. Create root-ca.conf

```
cd /root/ca/root-ca
```

Root-ca.conf

```
[ ca ]
default_ca = CA_default

[ CA_default ]
dir                     = /root/ca/root-ca

database                = $dir/index.txt
default_md              = sha256
distinguished_name      = req_distinguished_name

[ req_distinguished_name ]
countryName             = Country Name (2 letter code)
countryName_default     = BD
```

```
serial              = $dir/serial
crlnumber                       =
$dir/crlnumber

certs               = $dir/certs
crl_dir             = $dir/crl
new_certs_dir                   =
$dir/newcerts
certificate                     =
$dir/certs/ca.crt
private_key                     =
$dir/private/ca.key
crl                             =
$dir/crl/ca.crl

default_md          = sha256
default_days        = 3650
default_crl_days    = 30

unique_subject      = no
policy          = policy_strict
x509_extensions     = v3_ca
crl_extensions      = crl_ext
name_opt            = ca_default
cert_opt            = ca_default
string_mask         = utf8only

[ policy_strict ]
countryName                     =
supplied
stateOrProvinceName             =
supplied
```

```
stateOrProvinceName             =
State
stateOrProvinceName_default     =
Dhaka
localityName                    =
Locality Name
localityName_default            =
Uttara
0.organizationName              =
Organization Name
0.organizationName_default      =
Cyber_Security
organizationalUnitName          =
Organizational Unit Name
commonName                      =
Common Name (Root-CA)
commonName_default              =
RootCA
emailAddress                    =
Email Address
emailAddress_default            =
admin@example.com

[ v3_ca ]
subjectKeyIdentifier    = hash
authorityKeyIdentifier          =
keyid:always,issuer
basicConstraints                =
critical,CA:true
keyUsage                        =
critical,    digitalSignature,
cRLSign, keyCertSign

[ v3_intermediate_ca ]
```

```
organizationName                = supplied
organizationalUnitName          = optional
commonName                      = supplied
emailAddress                    = optional

[ policy_loose ]
countryName                     = optional
stateOrProvinceName             = optional
localityName                    = optional
organizationName                = optional
organizationalUnitName          = optional
commonName                      = supplied
emailAddress                    = optional
[ req ]
default_bits            = 4096
```

```
subjectKeyIdentifier    = hash
authorityKeyIdentifier          = keyid:always,issuer
basicConstraints                = critical, CA:true, pathlen:0
keyUsage                        = critical,    digitalSignature, cRLSign, keyCertSign

[ crl_ext ]
authorityKeyIdentifier          = keyid:always
```

3.5 Create ROOT-CA certificate

```
cd /root/ca/root-ca
openssl req -config root-ca.conf \
    -key private/ca.key \
    -new -x509 -days 3650 -sha256 \
    -extensions v3_ca \
```

```
-out certs/ca.crt
```

```
openssl req -config root-ca.conf \
    -key private/ca.key \
    -new -x509 -days 3650 -sha256 \
    -extensions v3_ca \
    -out certs/ca.crt
Enter pass phrase for private/ca.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [BD]:BD
State [Dhaka]:Dhaka
Locality Name [Uttara]:Uttara
Organization Name [Cyber_Security]:Cyber_Security
Organizational Unit Name []:CS
Common Name (Root-CA) [RootCA]:RootCA
Email Address [admin@example.com]:jaben@gmail.com
```

 # Check

```
openssl x509 -noout -text -in certs/ca.crt | head -n 20
```

```
root@jaben146-VirtualBox:~/ca/root-ca# openssl x509 -noout -text -in certs/ca.crt | head -n 20
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            1b:36:91:12:16:62:ab:fc:15:ec:f0:65:16:15:da:6c:09:8a:9f:b3
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = BD, ST = Dhaka, L = Uttara, O = Cyber_Security, OU = CS, CN = RootCA, emailAddress = jaben@gmail.com
        Validity
            Not Before: Nov 22 14:53:09 2025 GMT
            Not After : Nov 20 14:53:09 2035 GMT
        Subject: C = BD, ST = Dhaka, L = Uttara, O = Cyber_Security, OU = CS, CN = RootCA, emailAddress = jaben@gmail.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus:
                    00:b3:de:f3:b9:f6:47:58:d4:82:9f:61:b5:d4:09:
                    16:6f:f6:8c:db:8d:80:e9:f2:dd:c8:05:aa:50:41:
                    d5:dd:73:1d:e7:35:37:c2:f8:c8:02:e4:9b:f3:f1:
                    13:f6:a4:87:85:3f:cc:17:e3:63:13:82:1a:8b:89:
                    a0:7e:04:17:72:13:dc:c4:81:dc:2d:f9:62:d1:61:
```

3.6 SUB-CA Config + CSR + Cert

1. Create sub-ca.conf

```
cd /root/ca/sub-ca
```

Sub-ca.conf	
[ca] default_ca = CA_default [CA_default] dir = /root/ca/sub-ca	stateOrProvinceName_default = Dhaka localityName = Locality Name localityName_default = Uttara 0.organizationName = Org

```
database                        =
$dir/index.txt
serial              = $dir/serial
crlnumber                       =
$dir/crlnumber

certs               = $dir/certs
crl_dir             = $dir/crl
new_certs_dir                   =
$dir/newcerts
certificate                     =
$dir/certs/sub-ca.crt
private_key                     =
$dir/private/sub-ca.key
crl               = $dir/crl/sub-
ca.crl

default_md          = sha256
default_days        = 365
default_crl_days    = 30

unique_subject      = no
policy            = policy_loose
name_opt            = ca_default
cert_opt            = ca_default
string_mask         = utf8only

x509_extensions                 =
v3_intermediate_ca
crl_extensions      = crl_ext

[ policy_loose ]
```

```
0.organizationName_default      =
Cyber_Security
organizationalUnitName          =
Unit
organizationalUnitName_default
= AMS
commonName                      =
Common Name (Sub-CA)

[ v3_ca ]
subjectKeyIdentifier    = hash
authorityKeyIdentifier          =
keyid:always,issuer
basicConstraints                =
critical,CA:true
keyUsage                        =
critical,    digitalSignature,
cRLSign, keyCertSign

[ v3_intermediate_ca ]
subjectKeyIdentifier    = hash
authorityKeyIdentifier          =
keyid:always,issuer
basicConstraints                =
critical, CA:true, pathlen:0
keyUsage                        =
critical,    digitalSignature,
cRLSign, keyCertSign

# Profile for SERVER CERTS
(issued by Sub-CA)
[ server_cert ]
basicConstraints                =
CA:FALSE
```

```
countryName                     =
optional
stateOrProvinceName             =
optional
localityName                    =
optional
organizationName                =
optional
organizationalUnitName          =
optional
commonName                      =
supplied
emailAddress                    =
optional

[ req ]
default_bits            = 4096
default_md              = sha256
distinguished_name              =
req_distinguished_name

[ req_distinguished_name ]
countryName                     =
Country Name
countryName_default             =
BD
stateOrProvinceName             =
State

commonName_default              =
AcmeSubCA
emailAddress                    =
Email Address
```

```
subjectKeyIdentifier    = hash
authorityKeyIdentifier          =
keyid,issuer
keyUsage                        =
critical,    digitalSignature,
keyEncipherment
extendedKeyUsage                =
serverAuth
subjectAltName                  =
@alt_names

[ alt_names ]
DNS.1                           =
www.verysecureserver.com
DNS.2 = verysecureserver.com

[ crl_ext ]
authorityKeyIdentifier          =
keyid:always
```

Save like previouly

5.2 Create SUB-CA CSR

```
cd /root/ca/sub-ca
openssl   req   -config   sub-
ca.conf \
    -new   -key   private/sub-
ca.key \
    -sha256 \
    -out csr/sub-ca.csr
```

emailAddress_default = subca@example.com	

3.7 Create SUB-CA CSR

```
cd /root/ca/sub-ca

openssl req -config sub-ca.conf \

    -new -key private/sub-ca.key \

    -sha256 \

    -out csr/sub-ca.csr
```

3.8 Sign SUB-CA with ROOT-CA

```
cd /root/ca/sub-ca
openssl ca -config /root/ca/root-ca/root-ca.conf \
    -extensions v3_intermediate_ca \
    -days 3650 -notext \
    -in csr/sub-ca.csr \
    -out certs/sub-ca.crt
```

```
Using configuration from /root/ca/root-ca/root-ca.conf
Enter pass phrase for /root/ca/root-ca/private/ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4096 (0x1000)
        Validity
            Not Before: Nov 22 15:07:54 2025 GMT
            Not After : Nov 20 15:07:54 2035 GMT
        Subject:
            countryName               = BD
            stateOrProvinceName       = Dhaka
            organizationName          = Cyber_Security
            organizationalUnitName    = AMS
            commonName                = AcmeSubCA
            emailAddress              = hashi@gmail.com
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                33:CB:D5:F4:78:AF:55:5B:FC:E3:E0:20:C6:8F:A4:22:EC:17:1F:49
            X509v3 Authority Key Identifier:
                39:9A:BE:72:CE:19:AF:0D:BC:57:0B:47:F9:64:5B:41:1A:AA:1D:7B
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
Certificate is to be certified until Nov 20 15:07:54 2035 GMT (3650 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Database updated
```

 # Check

```
openssl x509 -noout -text -in certs/sub-ca.crt | head -n 20
```

```
root@jaben146-VirtualBox:~/ca/sub-ca# openssl x509 -noout -text -in certs/sub-ca.crt |
head -n 20
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4096 (0x1000)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = BD, ST = Dhaka, L = Uttara, O = Cyber_Security, OU = CS, CN = RootC
A, emailAddress = jaben@gmail.com
        Validity
            Not Before: Nov 22 15:07:54 2025 GMT
            Not After : Nov 20 15:07:54 2035 GMT
        Subject: C = BD, ST = Dhaka, O = Cyber_Security, OU = AMS, CN = AcmeSubCA, emai
lAddress = hashi@gmail.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus:
                    00:9d:ce:87:6c:d6:57:63:6f:3b:a0:c9:7c:e5:42:
                    09:ea:ae:83:42:1e:81:e6:6c:9f:16:9a:df:0b:21:
                    e0:ce:8f:2c:5a:69:a4:27:1b:8e:cb:d7:74:f6:b3:
                    7b:2f:09:f0:2f:ae:07:f8:3a:80:50:3e:15:9f:e1:
                    74:bb:c4:3f:06:b5:e1:4e:2e:67:a1:80:57:8c:54:
                    6b:15:ce:93:b8:7f:bf:ac:3b:a6:95:7a:0d:d0:71:
root@jaben146-VirtualBox:~/ca/sub-ca#
```

3.9 Sign Server CSR with SUB-CA

```
openssl ca -config sub-ca.conf \
    -extensions server_cert \
    -days 365 \
    -notext \
    -in ../server/csr/server.csr \
    -out ../server/certs/server.crt
```

```
root@jaben146-VirtualBox:~/ca/sub-ca# openssl ca -config sub-ca.conf \
  -extensions server_cert \
  -days 365 \
  -notext \
  -in ../server/csr/server.csr \
  -out ../server/certs/server.crt
Using configuration from sub-ca.conf
Enter pass phrase for /root/ca/sub-ca/private/sub-ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4096 (0x1000)
        Validity
            Not Before: Nov 22 15:26:04 2025 GMT
            Not After : Nov 22 15:26:04 2026 GMT
        Subject:
            countryName               = BD
            stateOrProvinceName       = Dhaka
            localityName              = Uttara
            organizationName          = Cyber_Security
            organizationalUnitName    = AMS
            commonName                = www.verysecureserver.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Subject Key Identifier:
                E2:CF:BB:77:28:0F:FD:0A:02:F0:7A:37:D4:28:ED:71:98:5B:6E:C0
            X509v3 Authority Key Identifier:
                33:CB:D5:F4:78:AF:55:5B:FC:E3:E0:20:C6:8F:A4:22:EC:17:1F:49
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Subject Alternative Name:
                DNS:www.verysecureserver.com, DNS:verysecureserver.com
Certificate is to be certified until Nov 22 15:26:04 2026 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Database updated
```

3.10 Build CHAIN File

```
cd /root/ca
cat sub-ca/certs/sub-ca.crt root-ca/certs/ca.crt \
  > server/certs/ca-chain.crt
ls server/certs
```

```
root@jaben146-VirtualBox:~/ca/sub-ca# cd /root/ca

cat sub-ca/certs/sub-ca.crt root-ca/certs/ca.crt \
  > server/certs/ca-chain.crt
root@jaben146-VirtualBox:~/ca# ls server/certs
ca-chain.crt  server.crt
```

3.11 Prepare Certificates for Apache (XAMPP)

1. Map the host

```
echo "127.0.0.1  www.verysecureserver.com" >> /etc/hosts
ping -c 2 www.verysecureserver.com
```

3.12 COPY CERTIFICATES FOR XAMPP

1. Create a folder for your certificates inside your user home

```
mkdir -p /home/jaben146/certificate
```

2. Copy the certs from CA to XAMPP

```
cp /root/ca/root-ca/certs/ca.crt
/home/jaben146/certificate/root-ca.crt
cp /root/ca/sub-ca/certs/sub-ca.crt
/home/jaben146/certificate/sub-ca.crt
cp /root/ca/server/certs/server.crt
/home/jaben146/certificate/server.crt
cp /root/ca/server/certs/ca-chain.crt
/home/jaben146/certificate/ca-chain.crt
cp /root/ca/server/private/server.key
/home/jaben146/certificate/server.key
```

3. Fix permissions for XAMPP

```
chown -R jaben146:jaben146 /home/jaben146/certificate
chmod 600 /home/jaben146/certificate/server.key
chmod 644 /home/jaben146/certificate/*.crt
/home/jaben146/certificate/ca-chain.crt
```

```
root@jaben146-VirtualBox:~# mkdir -p /home/jaben146/certificate
root@jaben146-VirtualBox:~# cp /root/ca/root-ca/certs/ca.crt              /home/jaben14
6/certificate/root-ca.crt
cp /root/ca/sub-ca/certs/sub-ca.crt          /home/jaben146/certificate/sub-ca.crt
cp /root/ca/server/certs/server.crt          /home/jaben146/certificate/server.crt
cp /root/ca/server/certs/ca-chain.crt        /home/jaben146/certificate/ca-chain.crt
cp /root/ca/server/private/server.key        /home/jaben146/certificate/server.key
root@jaben146-VirtualBox:~# chown -R jaben146:jaben146 /home/jaben146/certificate
chmod 600 /home/jaben146/certificate/server.key
chmod 644 /home/jaben146/certificate/*.crt /home/jaben146/certificate/ca-chain.crt
root@jaben146-VirtualBox:~#
```

3.13 CONFIGURE XAMPP SSL

```
nano /opt/lampp/etc/extra/httpd-ssl.conf
```

1. Replace SSLCertificateFile line

```
SSLCertificateFile"/home/jaben146/certificate/server.crt"
```

2. Replace SSLCertificateKeyFile line

```
SSLCertificateKeyFile "/home/jaben146/certificate/server.key"
```

3. Replace the CA chain file

```
SSLCertificateChainFile "/home/jaben146/certificate/ca-
chain.crt"
```

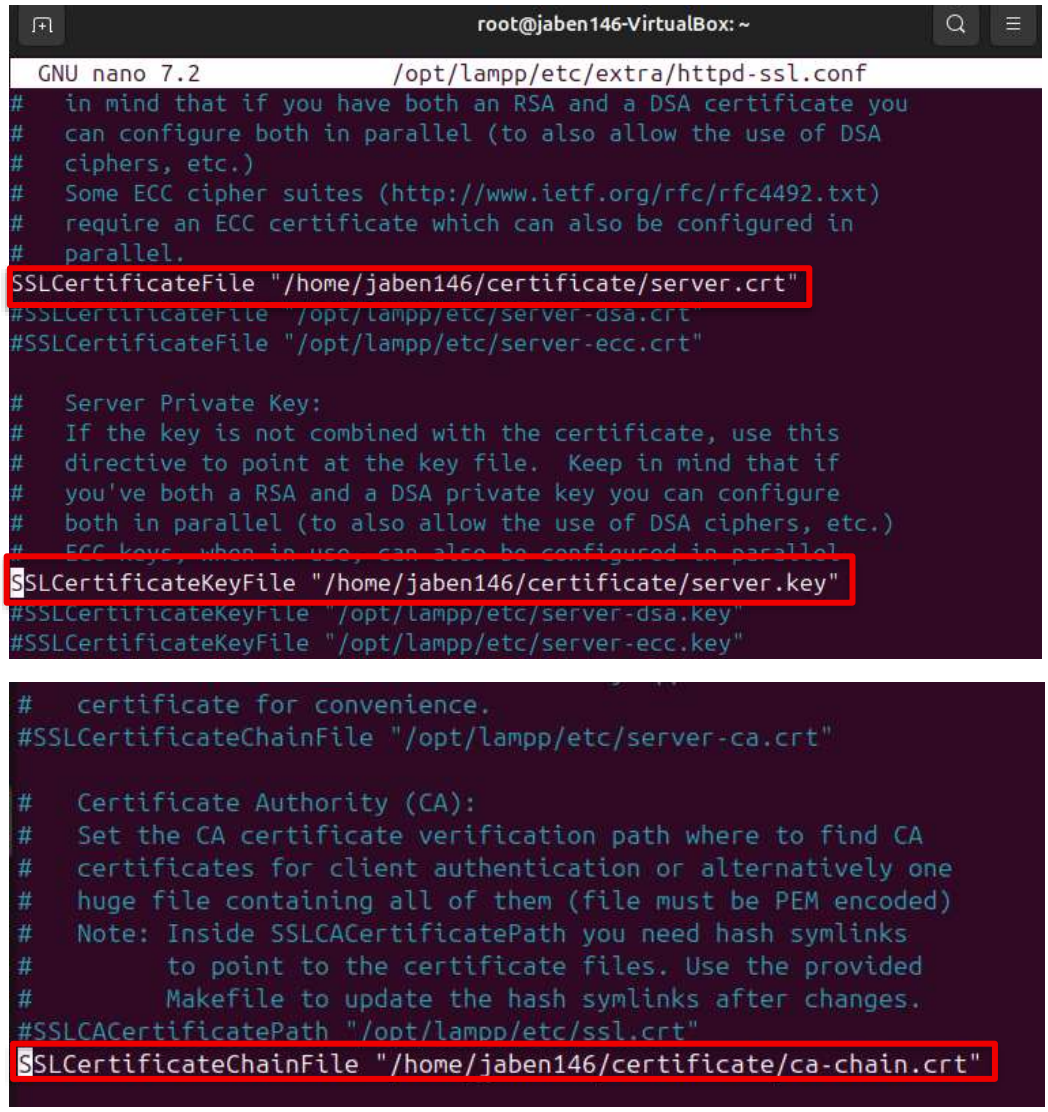
Before

```
#   SSL Engine Switch:
#   Enable/Disable SSL for this virtual host.
SSLEngine on

#   Server Certificate:
#   Point SSLCertificateFile at a PEM encoded certificate.  If
#   the certificate is encrypted, then you will be prompted for a
#   pass phrase.  Note that a kill -HUP will prompt again.  Keep
#   in mind that if you have both an RSA and a DSA certificate you
#   can configure both in parallel (to also allow the use of DSA
#   ciphers, etc.)
#   Some ECC cipher suites (http://www.ietf.org/rfc/rfc4492.txt)
#   require an ECC certificate which can also be configured in
#   parallel.
SSLCertificateFile "/opt/lampp/etc/ssl.crt/server.crt"
#SSLCertificateFile "/opt/lampp/etc/server-dsa.crt"
#SSLCertificateFile "/opt/lampp/etc/server-ecc.crt"

#   Server Private Key:
#   If the key is not combined with the certificate, use this
#   directive to point at the key file.  Keep in mind that if
#   you've both a RSA and a DSA private key you can configure
#   both in parallel (to also allow the use of DSA ciphers, etc.)
#   ECC keys, when in use, can also be configured in parallel
SSLCertificateKeyFile "/opt/lampp/etc/ssl.key/server.key"
#SSLCertificateKeyFile "/opt/lampp/etc/server-dsa.key"
#SSLCertificateKeyFile "/opt/lampp/etc/server-ecc.key"
```

After



3.14 CREATE CUSTOM HOMEPAGE (INSIDE XAMPP)

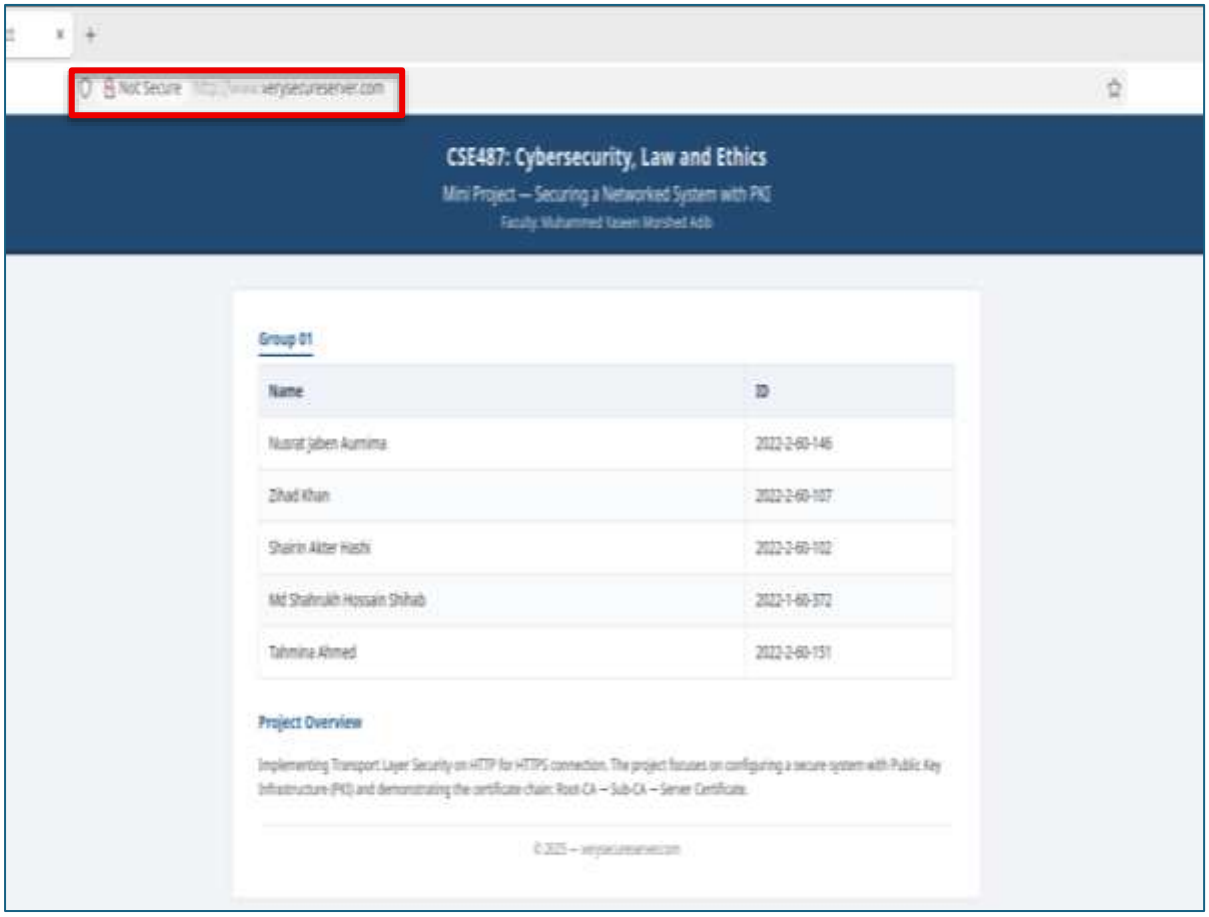
1. Create a new homepage

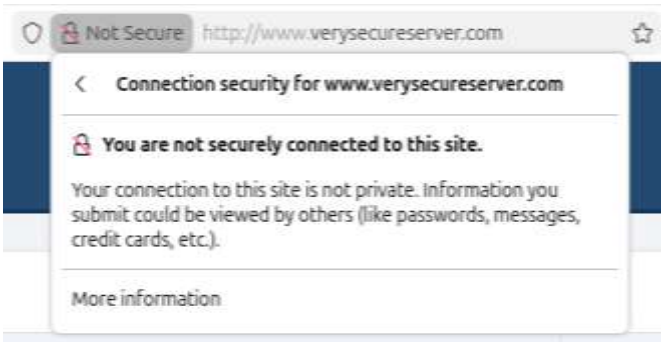
An index.html (custom HTML homepage) has been created by going to the XAMPP web directory, so that after enabling HTTPS, it can be verified whether the secure connection is working properly in the browser.

```
cd /opt/lampp/htdocs
nano index.html
ls
```

3.15 Test HTTP and HTTPS

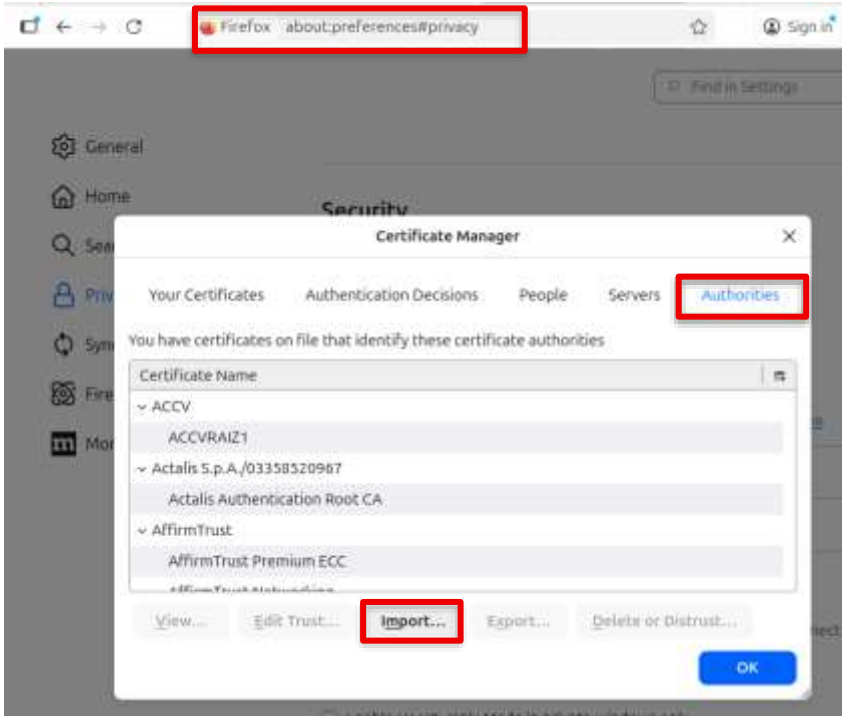
1.Unencrypted (HTTP): http://www.verysecureserver.com



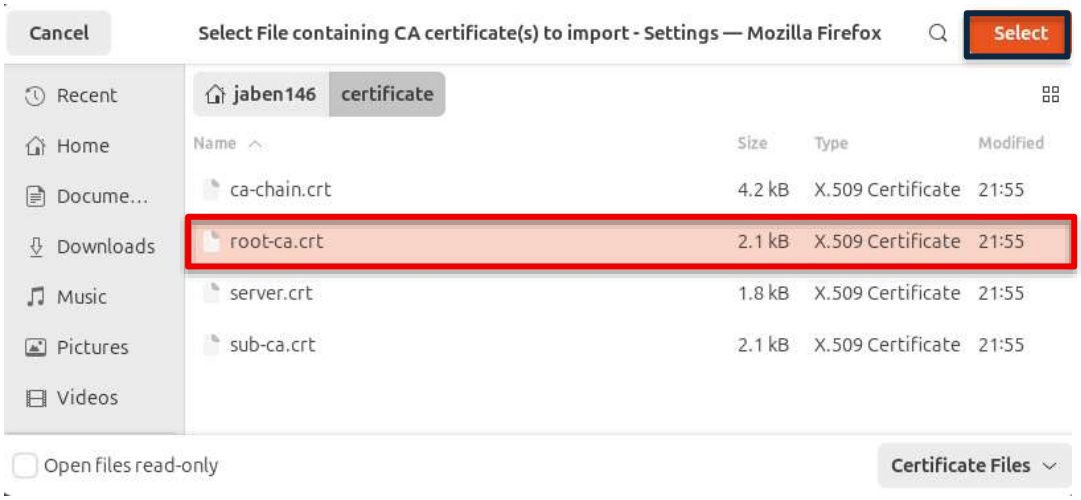


3.16 Import Root-CA Certificate into Browser

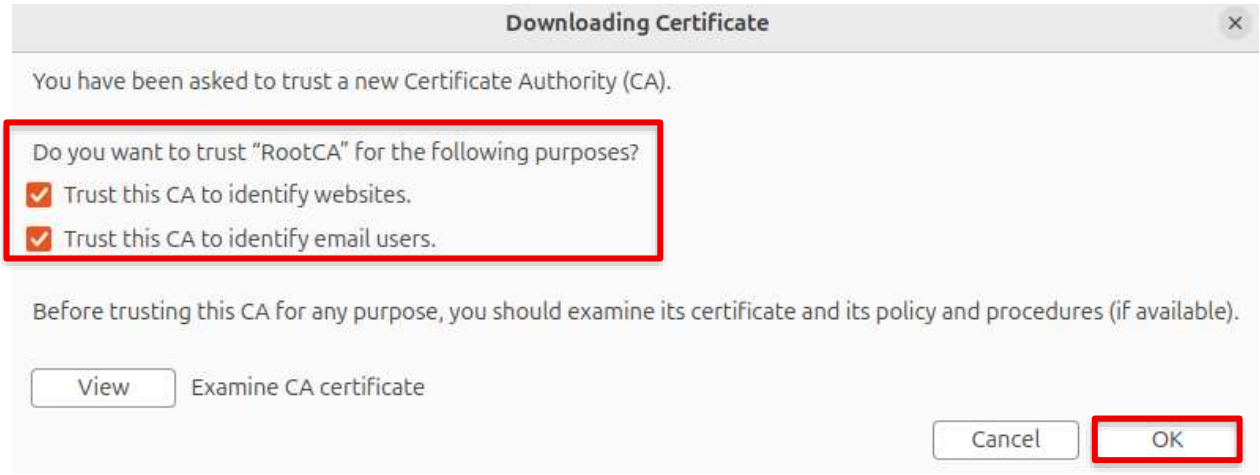
1. Open Settings → Security → Manage certificates → Authorities, then select Import to add the certificate.



2. Only import: **`/home/jaben146/certificate/root-ca.crt`**

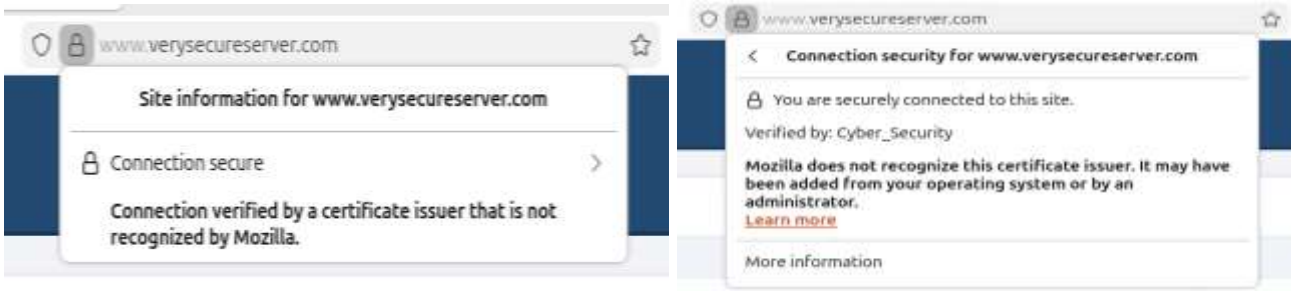


3. Check trust this certificate

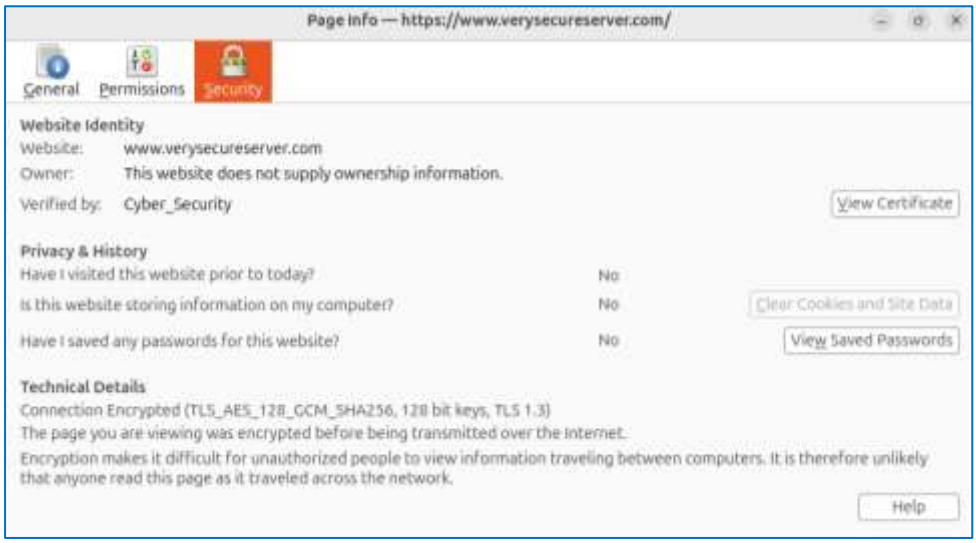


4. This time there is a padlock, indicating the site is secured.

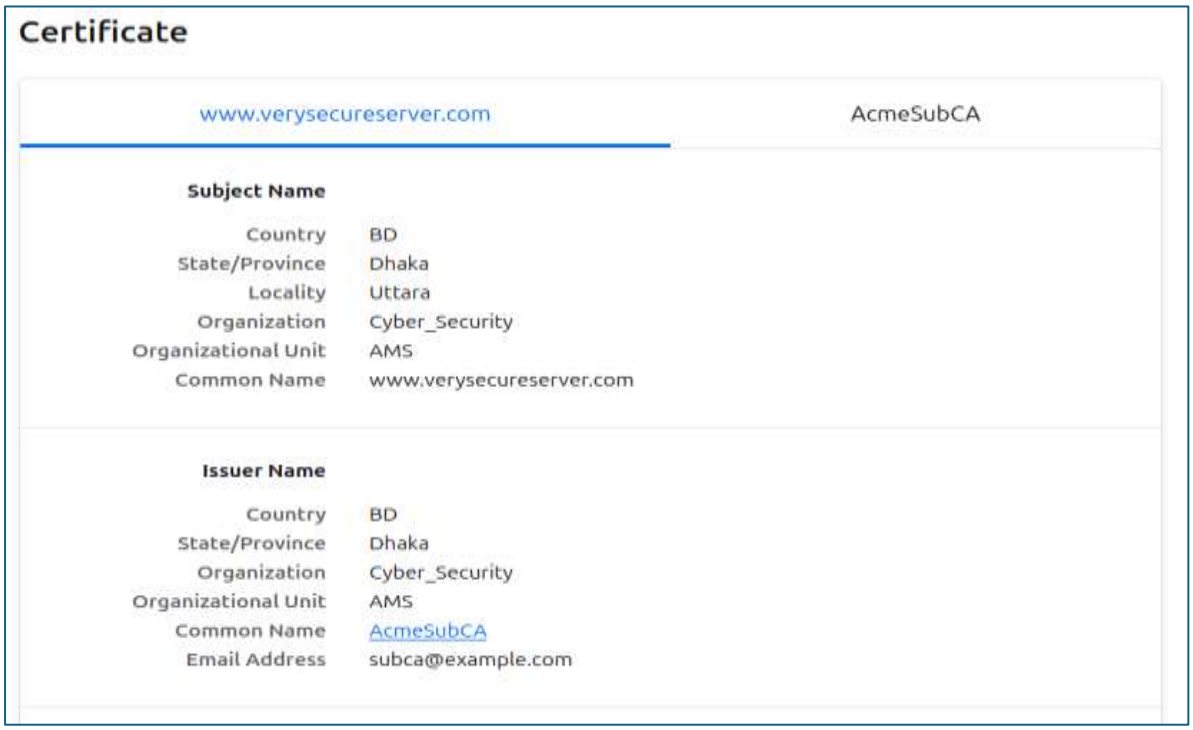
5. The browser is ensuring a secure connection, verified by a custom CA installed on te system.

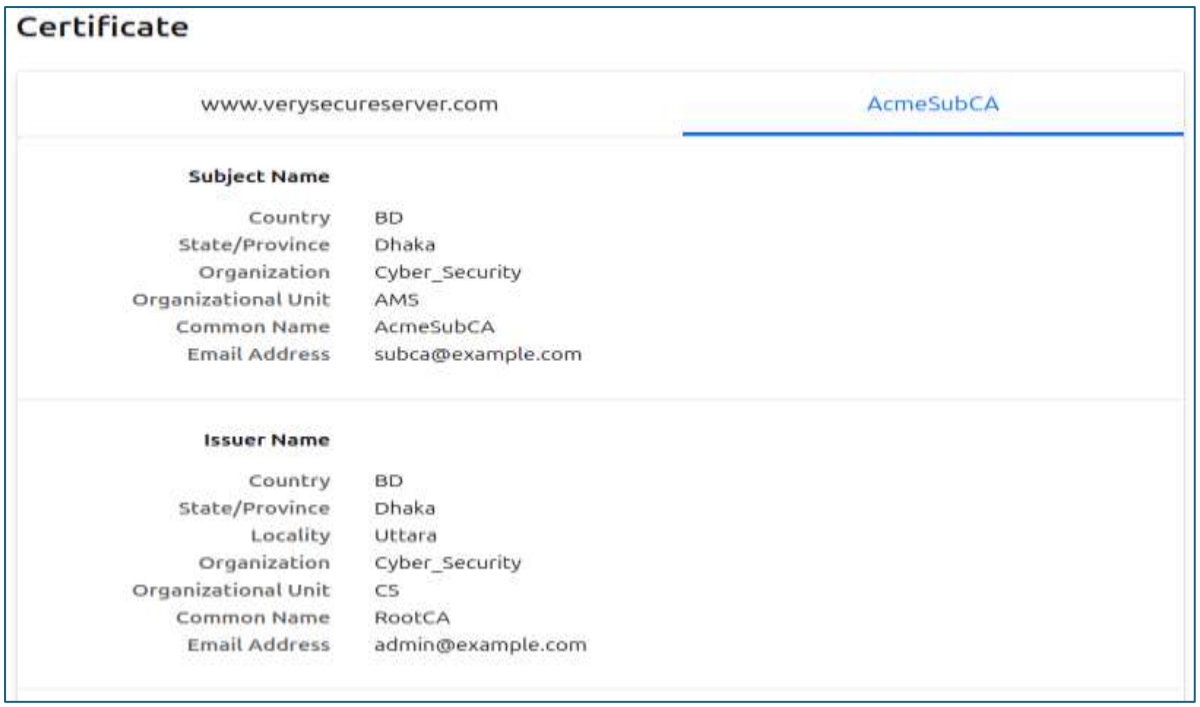


6. The Security tab shows details about the site's encryption.



7. Certificate Details





4. DNS Configuration

4.1 Install bind9

```
sudo apt-get update
sudo apt-get install -y bind9
```

4.2 Go to the bind configuration folder

```
cd /etc/bind
pwd
ls
```

4.3. Edit /etc/hosts and add the server IP

```
sudo nano /etc/hosts
```

```
  GNU nano 7.2                        /etc/hosts *
127.0.0.1       localhost
192.168.0.147   host.verysecureserver.com   host
127.0.0.2       www.verysecureserver.com

# The following lines are desirable for IPv6 capable hosts
::1             ip6-localhost ip6-loopback
fe00::0         ip6-localnet
ff00::0         ip6-mcastprefix
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters
```

4.4. Verify hostname settings

```
hostname
dnsdomainname
hostname --fqdn
```

```
root@host:/etc/bind# hostname
host
root@host:/etc/bind# dnsdomainname
verysecureserver.com
root@host:/etc/bind# hostname --fqdn
host.verysecureserver.com
```

4.5. Setup configuration files

1. Install gedit

```
sudo apt update
sudo apt install gedit -y
```

2. Backup original files

```
cp named.conf.options named.conf.options.orig
cp named.conf.local named.conf.local.orig
```

3. Edit named.conf.options

```
gedit named.conf.options
```

named.conf.options

```
options {
    directory "/var/cache/bind";

    // If there is a firewall between you and nameservers you want
    // to talk to, you may need to fix the firewall to allow multiple
    // ports to talk.

    // If your ISP provided IP addresses for stable nameservers,
    // you probably want to use them as forwarders.
    // Uncomment the following block and insert the addresses.

    forwarders {
        192.168.0.1;
    };

    // DNSSEC settings
    dnssec-validation auto;

    // Listen only on our local DNS server IP
    listen-on { 192.168.0.147; };

    // Allow IPv6 anywhere
    listen-on-v6 { any; };

    // Do NOT allow zone transfer
    allow-transfer { none; };
};
```

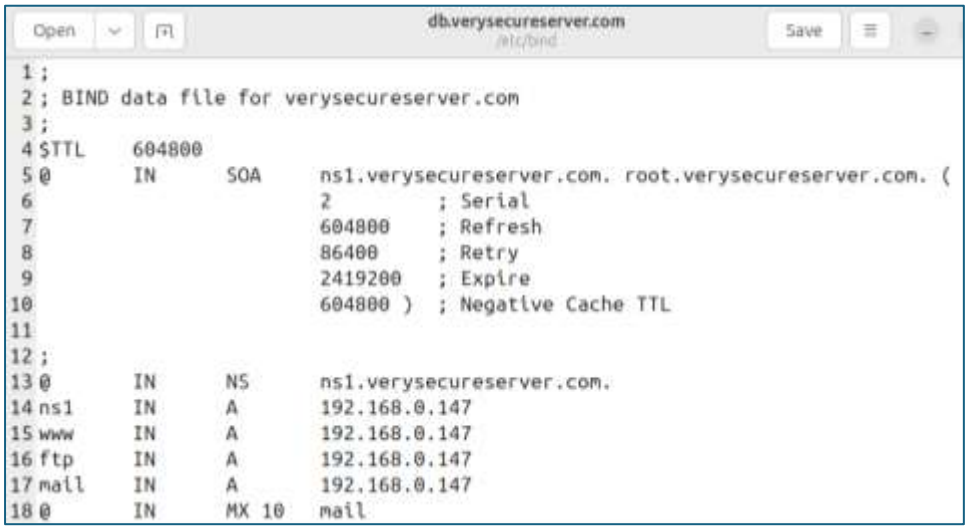
4. Edit named.conf.local

```
gedit /etc/bind/named.conf.local
```

4.6 Setup DB Files

1. Create forward lookup file

```
cp db.local db.verysecureserver.com
gedit db.verysecureserver.com
```



4.7 Create reverse lookup file

```
cp db.127 db.0.168.192
gedit db.0.168.192
```

```
Open    db.0.168.192 /etc/bind    Save
1 ;
2 ; BIND reverse data file for 192.168.0.x network
3 ;
4 $TTL    604800
5 @       IN      SOA     ns1.verysecureserver.com. root.verysecureserver.com. (
6                         1         ; Serial
7                         86400     ; Refresh
8                         86400     ; Retry
9                         2419200   ; Expire
10                        604800 )  ; Negative Cache TTL
11 ;
12 @      IN      NS      ns1.verysecureserver.com.
13
14 147    IN      PTR     ns1.verysecureserver.com.
15 147    IN      PTR     www.verysecureserver.com.
16 147    IN      PTR     ftp.verysecureserver.com.
17 147    IN      PTR     mail.verysecureserver.com.
```

4.8 Verify zones

```
named-checkzone verysecureserver.com db.verysecureserver.com
named-checkzone 0.168.192.in-addr.arpa db.0.168.192
named-checkconf
```

```
root@host:/etc/bind# named-checkzone verysecureserver.com db.verysecureserver.com
named-checkzone 0.168.192.in-addr.arpa db.0.168.192
named-checkconf
zone verysecureserver.com/IN: loaded serial 2
OK
zone 0.168.192.in-addr.arpa/IN: loaded serial 1
OK
```

4.9 Restart DNS

```
service bind9 restart
service bind9 status
```

```
root@host:/etc/bind# service bind9 restart
service bind9 status
● named.service - BIND Domain Name Server
     Loaded: loaded (/usr/lib/systemd/system/named.service; enabled; preset: enabled)
     Active: active (running) since Sun 2025-11-23 04:01:22 +06; 82ms ago
       Docs: man:named(8)
   Main PID: 3451 (named)
     Status: "running"
      Tasks: 10 (limit: 5533)
     Memory: 23.7M (peak: 24.3M)
        CPU: 293ms
     CGroup: /system.slice/named.service
             └─3451 /usr/sbin/named -f -u bind

Nov 23 04:01:22 host named[3451]: zone 255.in-addr.arpa/IN: loaded serial 1
Nov 23 04:01:22 host named[3451]: zone 0.in-addr.arpa/IN: loaded serial 1
Nov 23 04:01:22 host named[3451]: zone 127.in-addr.arpa/IN: loaded serial 1
Nov 23 04:01:22 host named[3451]: zone localhost/IN: loaded serial 2
Nov 23 04:01:22 host named[3451]: zone verysecureserver.com/IN: loaded serial 2
Nov 23 04:01:22 host named[3451]: all zones loaded
Nov 23 04:01:22 host named[3451]: running
Nov 23 04:01:22 host systemd[1]: Started named.service - BIND Domain Name Server.
Nov 23 04:01:22 host named[3451]: managed-keys-zone: Key 20326 for zone . is now truste
Nov 23 04:01:22 host named[3451]: managed-keys-zone: Key 38696 for zone . is now truste
lines 1-22/22 (END)
```

4.10 Test DNS resolution:

`Nslookup www.verysecureserver.com`

```
root@host:/etc/bind# nslookup www.verysecureserver.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Name:   www.verysecureserver.com
Address: 127.0.0.2
```

4.11 Verifying DNS Server & Browser Access

1. Ping Test

`ping -c 3 www.verysecureserver.com`

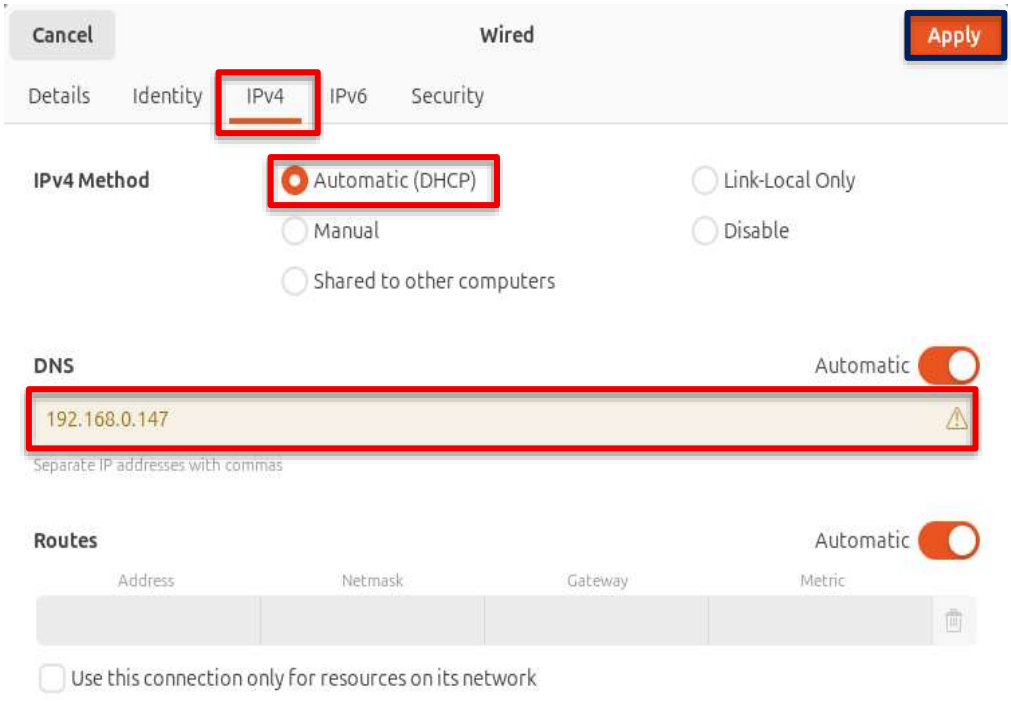
```
root@host:/etc/bind# ping -c 3 www.verysecureserver.com
PING www.verysecureserver.com (127.0.0.2) 56(84) bytes of data.
64 bytes from www.verysecureserver.com (127.0.0.2): icmp_seq=1 ttl=64 time=4.46 ms
64 bytes from www.verysecureserver.com (127.0.0.2): icmp_seq=2 ttl=64 time=0.169 ms
64 bytes from www.verysecureserver.com (127.0.0.2): icmp_seq=3 ttl=64 time=0.047 ms

--- www.verysecureserver.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2056ms
rtt min/avg/max/mdev = 0.047/1.558/4.458/2.051 ms
```

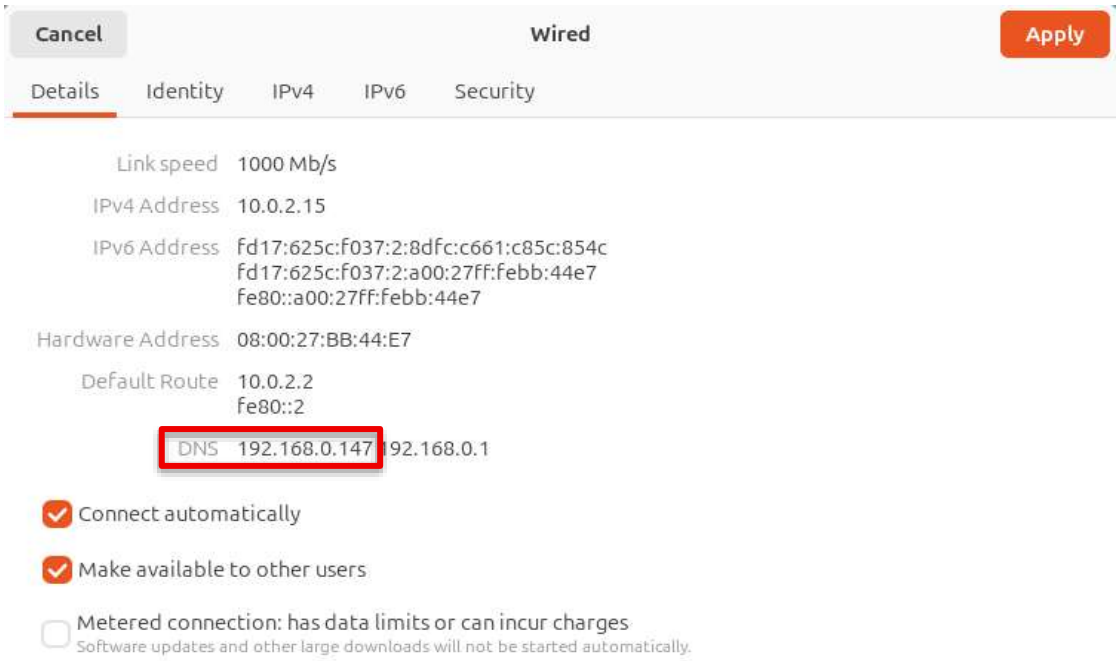
4.12 Verifying DNS Server Working Properly

1. After ping test, go to the client VM

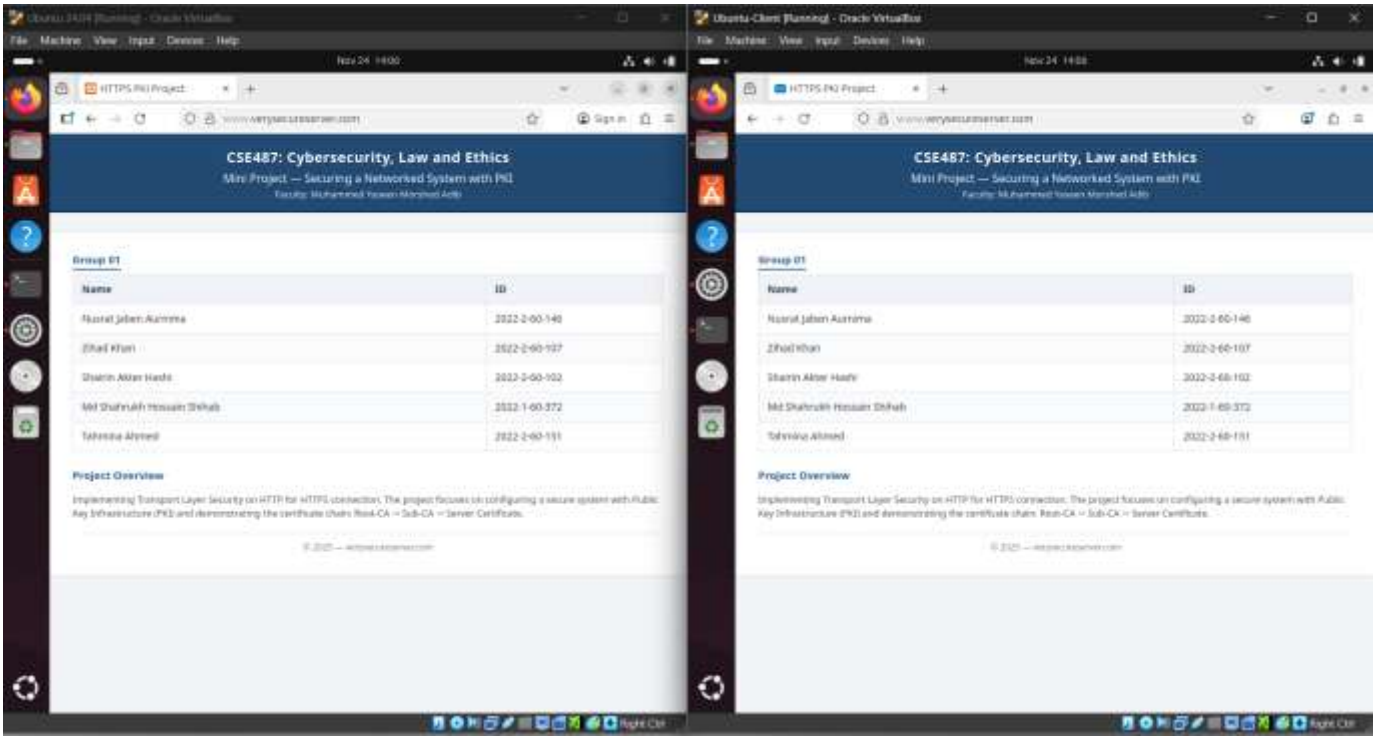
- Open Settings → Network → Adapters
- Set DNS to the DNS server IP



4.13 After Restarting, the host IP will be added



4.14 Checking Through Browser



5. WireShark

This section explains the basic concepts of network monitoring using Wireshark to verify the HTTPS traffic encryption of our server.

5.1 Download Wireshark

Npca We downloaded by official website for live netwrok capture.

```
winget install -e --id WiresharkFoundation.Wireshark
```

5.2 Capture packet

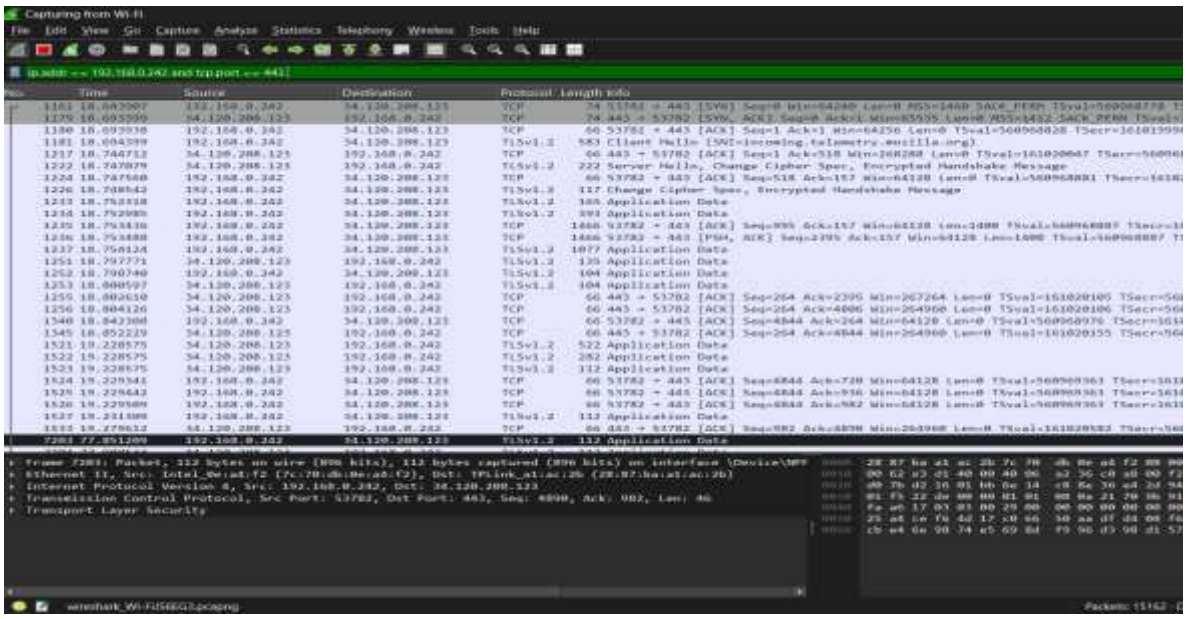
To verify whether HTTPS traffic is encrypted, we capture packets coming from Ubuntu VM (IP: 192.168.0.242) using Wireshark on Windows host. We select Wi-Fi interface and apply display filter:

```
ip.addr == 192.168.0.242 and tcp.port == 443
```

5.3 Display

Now enter this address from a Windows browser: https://192.168.0.242

Then Wireshark shows the TLS handshake (Client Hello, Server Hello, etc.) and all packets after that are displayed as TLS Application Data. They do not contain any HTTP header or body. If we follow the TCP Stream, the data is displayed as fully encrypted bytes.This proves that HTTPS traffic is actually fully encrypted by TLS and that a network observer cannot read any HTTP data.



If we follow the TCP Stream, we will only see encrypted bytes with no HTTP header or body.

6. Firewall Configuration

This section explains the basic firewall settings used to control what network traffic enters our server.

6.1 Install UFW

```
sudo apt install ufw
```

6.2 Set default rules

```
sudo ufw default allow outgoing
sudo ufw default deny incoming
```

6.3 Allow only necessary service ports

```
Allow DNS (53), HTTP (80), and HTTPS (443):
sudo ufw allow 53
sudo ufw allow 80
sudo ufw allow 443
```

6.4 Enable the firewall

```
sudo ufw enable
```

6.5 Verify the firewall rules

```
sudo ufw status numbered
```

6.6 Output

- **53/tcp, udp ALLOW IN**
- **80/tcp ALLOW IN**
- **443/tcp ALLOW IN**

All other ports remain blocked by default.

```
jaben146@host:~$ sudo apt install ufw
[sudo] password for jaben146:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ufw is already the newest version (0.36.2-6).
ufw set to manually installed.
The following package was automatically installed and is no longer required:
  libllvm19
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 30 not upgraded.
jaben146@host:~$ sudo ufw default allow outgoing
sudo ufw default deny incoming
Default outgoing policy changed to 'allow'
(be sure to update your rules accordingly)
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
jaben146@host:~$ sudo ufw allow 53
sudo ufw allow 80
sudo ufw allow 443
Rules updated
Rules updated (v6)
Rules updated
Rules updated (v6)
Rules updated
Rules updated (v6)
```

The output below shows that UFW has only allowed ports 53, 80, and 443.

```
jaben146@host:~$ sudo ufw enable
Firewall is active and enabled on system startup
jaben146@host:~$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 53                         ALLOW IN    Anywhere
[ 2] 80                         ALLOW IN    Anywhere
[ 3] 443                        ALLOW IN    Anywhere
[ 4] 53 (v6)                    ALLOW IN    Anywhere (v6)
[ 5] 80 (v6)                    ALLOW IN    Anywhere (v6)
[ 6] 443 (v6)                   ALLOW IN    Anywhere (v6)

jaben146@host:~$
```

7. Intrusion Detection System (IDS)

IDS is used for monitoring the server's network traffic and detect any suspicious activity.

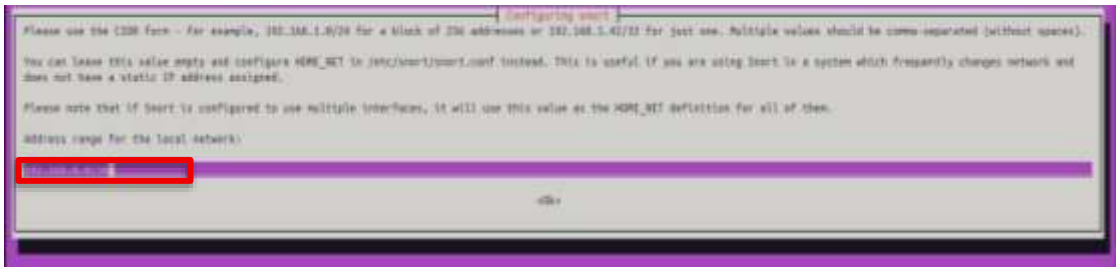
7.1 IDS Setup

- Ensure root access.
- Detect the active network interface and local IP address.
- Install **Snort** using apt-get: **`sudo apt install snort -y`**
- Update the **HOME_NET** variable in the Snort configuration file.
- Add a simple rule to detect suspicious traffic on the server.
- Test the configuration and fix any errors.
- Start Snort in the background and log output to **/var/log/snort.log**

7.2 Snort Setup

- Install Snort and set it up with the correct network interface and IP.
- Edit the Snort config file: **`sudo nano /etc/snort/snort.conf`**

7.3 Update HOME_NET



`var HOME_NET 192.168.0.0/24`

7.4 Add a simple custom rule in

`sudo nano /etc/snort/rules/local.rules`

7.5 Test the Snort configuration

```
sudo snort -T -c /etc/snort/snort.conf
```

7.6 Start Snort in the background

```
sudo snort -q -c /etc/snort/snort.conf -i enp0s3 -D
```

7.7 Real-Time Monitoring

- Snort starts with live monitoring.
- Snort watches incoming traffic in real-time.
- Logs alerts for any suspicious packets.
- Alerts are stored in /var/log/snort/alert.
- Administrators can review the alert file to identify abnormal activity.

8. Certificate Revocation

8.1 Navigate to the Sub-CA directory

```
cd /root/ca/sub-ca
```

8.2 Revoke the server certificate

```
openssl ca -config sub-ca.conf -revoke
../server/certs/server.crt
```

8.3 Update the system's trusted certificates

```
sudo update-ca-certificates
```

Check the domain again in the browser to confirm that the revoked certificate is no longer trusted.

9. Observation

- Learned how to set up and secure a networked system using PKI and TLS.
- Practically implemented the full PKI workflow: key generation, Root CA, Intermediate CA, and server certificate.
- Gained understanding of trust hierarchies and how browsers verify certificate chains.
- Implemented TLS on a web server, improving knowledge of encrypted communication and certificate-based authentication.
- Configured DNS and firewall rules, learning how different network components work together to maintain a secure environment.
- Used Snort to create IDS rules, gaining practical insight into real-time intrusion detection.
- Simulated a DoS attack to understand how abnormal traffic is identified and analyzed.
- Worked with certificate revocation and CRLs, learning the importance of full certificate lifecycle management.

10. Future Work

Although the current solution is secure and works adequately, the introduction of several improvements would significantly improve the ability of the system to scale and to be reliable and secure. Automating PKI operations for issuance, renewal, and revocation of certificates via shell scripts or tools such as Ansible would eliminate many workloads and manual configuration errors. To improve flexibility, fault tolerance, and improvement in deployment to prod, deployment on cloud infrastructure such as AWS or Azure would be a part of even more improvement. The addition of an IPS such as Suricata would provide a more proactive approach to security, blocking malicious traffic instead of looking for it. Installation of a SIEM solution would present a single pane of glass for log analysis and help quickly locate security incidents. In summary, that automating HTTPS management (for example, Certbot or alternate ACME clients) will greatly simplify maintenance and the life cycle of the certificates. Hardening of web applications, scanning for vulnerabilities, and rate limiting are all additional steps that will improve the overall security posture. All of these enhancements will go a long way to improving the resilience and security of the architecture.

11. Conclusion

The mini project of building and securing a networked system with PKI and TLS provided great hands-on learning. We learned about PKI in practice, key generation, and the building of a full certificate chain, which included a Root CA, an Intermediate CA, and a server certificate. This drew us closer to understanding how trust hierarchies are built and how a browser validates the certificate. TLS configuration on the web server solidified the understanding of encrypted communication and authentication using certificates. Implementing DNS and firewall rules contributed to the learning process by demonstrating how different parts of the network choreography interact and provide security for the network. Creating IDS rules with Snort allowed for insight into practical intrusion detection in real time. Also, simulating a DoS attack illustrated how abnormal traffic could be identified and analyzed. Lastly, revoking a certificate and working with CRLs demonstrated the importance of managing the full life cycle of a certificate.