**Mini Project Report**

**Course Code :** CSE487

**Course Title :** Computer and Cyber Security

**Section :** 02

**Semester :** Summer 2024

**Group : 14**

**Project Title : “Securing a Networked System with Public Key Infrastructure”**

**Presentation Link :**

[**https://drive.google.com/file/d/15PGRJV9LSsVLmB3ewJVf68vEoWadN2su/view?usp=sharing**](https://drive.google.com/file/d/15PGRJV9LSsVLmB3ewJVf68vEoWadN2su/view?usp=sharing)

**Submitted By :**

**Prinom Mojumder (2021-2-60-098)**

**Umme Atika Borsha (2021-2-60-076)**

**Tasnim Israk Synthia (2021-2-60-097)**

**Rokeya Jahan Chowdhury Ettifa (2020-1-60-232)**

**Submitted To :**

**Dr. Md. Hasanaul Ferdaus**

Assistant Professor

Department of Computer Science & Engineering

**Date of Report Submission :** 11 January, 2025

**Introduction**

Public Key Infrastructure (PKI) security for networked systems uses digital certificates and cryptographic keys to create and preserve a secure communication environment. The Certificate Authority (CA), which is responsible for issuing digital certificates, and other essential elements such as certificate repositories and registration authorities make up PKI. Digital certificates provide data integrity, secrecy, and non-repudiation by authenticating people and devices. Creating keys, issuing certificates, distributing information securely, and integrating with apps are all part of implementation. In order to maintain a reliable and secure network infrastructure, ongoing maintenance entails critical lifecycle management, monitoring, and auditing. PKI is necessary to secure data in transit, prevent unwanted access, and build confidence in online transactions.

**Objectives**

* Authentication: Verify the identities of users, devices, and services through digital certificates and cryptographic keys.
* Data Integrity: Ensure the integrity of data by using digital signatures to detect tampering during transmission.
* Confidentiality: Employ encryption to secure communication and protect sensitive information from unauthorized access.
* Secure Key Management: Implement secure practices for key generation, distribution, renewal, and revocation.
* Resilience Against Cyber Threats: Enhance network resilience against cyber threats such as eavesdropping and identity spoofing.
* Access Control: Access policies based on verified identities and permissions.

**Requirements**

● Configuration of Certification Authority AcmeCA with AcmeRootCA as the RootCA.

● Configuration of the Web Server with Apache2 on a Linux Host.

● 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 the SSL certificate in the server www.verysecureserver.com

● Making the system trust Acme-RootCA

**Project Statement**

This project focuses on strengthening network security through the deployment of a robust Public Key Infrastructure (PKI). By incorporating digital certificates and cryptographic keys, the initiative aims to enhance authentication, shield data integrity, ensure confidentiality, and establish trust within the networked system.

**Components**

1. Ubuntu Operating System
2. Ubuntu Terminal to execute command
3. Xampp Server
4. Notepad

**Implementation Details:**

***1. Preparing the environment***

**Moving to the root using**

sudo -i

**Creating directory:**

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

**Changing the root of ca and sub ca private folder**

chmod -v 700 ca/{root-ca,sub-ca,server}/private

**Creating file index in both root ca and sub ca**

touch ca/{root-ca,sub-ca}/index

**Generating hexadecimal random number of 16 character**

openssl rand -hex 16

**writing serial number of root ca**

openssl rand -hex 16 > ca/root-ca/serial

**writing serial number of sub ca**

openssl rand -hex 16 > ca/sub-ca/serial

tree ca

**moving to ca directory**

cd ca

A screenshot of a computer program

Description automatically generated

***2. Generating private key for root ca, sub ca and server***

**Public key for rootCA**

openssl genrsa -aes256 -out root-ca/private/ca.key 4096

**Public key for subCA**

openssl genrsa -aes256 -out sub-ca/private/sub-ca.key 4096

**Public key for server**

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

A screen shot of a computer

Description automatically generated

***3. Generating certificates***

**Root-CA**

**Creating root ca.config**

gedit root-ca/root-ca.conf

[ca]

#/root/ca/root-ca/root-ca.conf

#see man ca

default\_ca = CA\_default

[CA\_default]

dir = /root/ca/root-ca

certs = $dir/certs

crl\_dir = $dir/crl

new\_certs\_dir = $dir/newcerts

database = $dir/index

serial = $dir/serial

RANDFILE = $dir/private/.rand

private\_key = $dir/private/ca.key

certificate = $dir/certs/ca.crt

crlnumber = $dir/crlnumber

crl = $dir/crl/ca.crl

crl\_extensions = crl\_ext

default\_crl\_days = 30

default\_md = sha256

name\_opt = ca\_default

cert\_opt = ca\_default

default\_days = 365

preserve = no

policy = policy\_strict

[ policy\_strict ]

countryName = supplied

stateOrProvinceName = supplied

organizationName = match

organizationalUnitName = optional

commonName = supplied

emailAddress = optional

[ policy\_loose ]

countryName = optional

stateOrProvinceName = optional

localityName = optional

organizationName = optional

organizationalUnitName = optional

commonName = supplied

emailAddress = optional

[ req ]

# Options for the req tool, man req.

default\_bits = 2048

distinguished\_name = req\_distinguished\_name

string\_mask = utf8only

default\_md = sha256

# Extension to add when the -x509 option is used.

x509\_extensions = v3\_ca

[ req\_distinguished\_name ]

countryName = Country Name (2 letter code)

stateOrProvinceName = State or Province Name

localityName = Locality Name

0.organizationName = Organization Name

organizationalUnitName = Organizational Unit Name

commonName = Common Name

emailAddress = Email Address

countryName\_default = BD

stateOrProvinceName\_default = Dhaka

localityName\_default = Demra

0.organizationName\_default = EWU

organizationalUnitName\_default = Cyber\_Security

commonName\_default = AcmeRootCA

emailAddress\_default = prinom@acmeroot\_ca.com

[ v3\_ca ]

# Extensions to apply when createing root ca

# Extensions for a typical CA, man x509v3\_config

subjectKeyIdentifier = hash

authorityKeyIdentifier = keyid:always,issuer

basicConstraints = critical, CA:true

keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ v3\_intermediate\_ca ]

# Extensions to apply when creating intermediate or sub-ca

# Extensions for a typical intermediate CA, same man as above

subjectKeyIdentifier = hash

authorityKeyIdentifier = keyid:always,issuer

#pathlen:0 ensures no more sub-ca can be created below an intermediate

basicConstraints = critical, CA:true, pathlen:0

keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ server\_cert ]

# Extensions for server certificates

basicConstraints = CA:FALSE

nsCertType = server

nsComment = "OpenSSL Generated Server Certificate"

subjectKeyIdentifier = hash

authorityKeyIdentifier = keyid,issuer:always

keyUsage = critical, digitalSignature, keyEncipherment

extendedKeyUsage = serverAuth

***3.1.Generating root ca certificate***

openssl req -config root-ca.conf -key private/ca.key -new -x509 -days 7305 -sha256 -extensions v3\_ca -out certs/ca.crt

A computer screen shot of white text

Description automatically generated

***3.2.Ensuring that the certificate has been created properly***

openssl x509 -noout -in certs/ca.crt -text

A screenshot of a computer program

Description automatically generated

***3.3.Generating Sub ca certificate***

**Moving a step back and then to sub-ca**

cd ../sub-ca

**Sub-CA**

**Creating sub-ca.config**

gedit sub-ca.conf

**Inserting the code into sub-ca.config file**

[ca]

#/root/ca/sub-ca/sub-ca.conf

#see man ca

default\_ca = CA\_default

[CA\_default]

dir = /root/ca/sub-ca

certs = $dir/certs

crl\_dir = $dir/crl

new\_certs\_dir = $dir/newcerts

database = $dir/index

serial = $dir/serial

RANDFILE = $dir/private/.rand

private\_key = $dir/private/sub-ca.key

certificate = $dir/certs/sub-ca.crt

crlnumber = $dir/crlnumber

crl = $dir/crl/ca.crl

crl\_extensions = crl\_ext

default\_crl\_days = 30

default\_md = sha256

name\_opt = ca\_default

cert\_opt = ca\_default

default\_days = 365

preserve = no

policy = policy\_loose

[ policy\_strict ]

countryName = supplied

stateOrProvinceName = supplied

organizationName = match

organizationalUnitName = optional

commonName = supplied

emailAddress = optional

[ policy\_loose ]

countryName = optional

stateOrProvinceName = optional

localityName = optional

organizationName = optional

organizationalUnitName = optional

commonName = supplied

emailAddress = optional

[ req ]

# Options for the req tool, man req.

default\_bits = 2048

distinguished\_name = req\_distinguished\_name

string\_mask = utf8only

default\_md = sha256

# Extension to add when the -x509 option is used.

x509\_extensions = v3\_ca

[ req\_distinguished\_name ]

countryName = Country Name (2 letter code)

stateOrProvinceName = State or Province Name

localityName = Locality Name

0.organizationName = Organization Name

organizationalUnitName = Organizational Unit Name

commonName = Common Name

emailAddress = Email Address

countryName\_default = BD

stateOrProvinceName\_default = Dhaka

localityName\_default = Demra

0.organizationName\_default = EWU

organizationalUnitName\_default = Cyber\_Security

commonName\_default = AcmeCA

emailAddress\_default = prinom @acmesub\_ca.com

[ v3\_ca ]

# Extensions to apply when createing root ca

# Extensions for a typical CA, man x509v3\_config

subjectKeyIdentifier = hash

authorityKeyIdentifier = keyid:always,issuer

basicConstraints = critical, CA:true

keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ v3\_intermediate\_ca ]

# Extensions to apply when creating intermediate or sub-ca

# Extensions for a typical intermediate CA, same man as above

subjectKeyIdentifier = hash

authorityKeyIdentifier = keyid:always,issuer

#pathlen:0 ensures no more sub-ca can be created below an intermediate

basicConstraints = critical, CA:true, pathlen:0

keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ server\_cert ]

# Extensions for server certificates

basicConstraints = CA:FALSE

nsCertType = server

nsComment = "OpenSSL Generated Server Certificate"

subjectKeyIdentifier = hash

authorityKeyIdentifier = keyid,issuer:always

keyUsage = critical, digitalSignature, keyEncipherment

extendedKeyUsage = serverAuth

[ca]

#/root/ca/sub-ca/sub-ca.conf

#see man ca

default\_ca = CA\_default

[CA\_default]

dir = /root/ca/sub-ca

certs = $dir/certs

crl\_dir = $dir/crl

new\_certs\_dir = $dir/newcerts

database = $dir/index

serial = $dir/serial

RANDFILE = $dir/private/.rand

private\_key = $dir/private/sub-ca.key

certificate = $dir/certs/sub-ca.crt

crlnumber = $dir/crlnumber

crl = $dir/crl/ca.crl

crl\_extensions = crl\_ext

default\_crl\_days = 30

default\_md = sha256

name\_opt = ca\_default

cert\_opt = ca\_default

default\_days = 365

preserve = no

policy = policy\_loose

[ policy\_strict ]

countryName = supplied

stateOrProvinceName = supplied

organizationName = match

organizationalUnitName = optional

commonName = supplied

emailAddress = optional

[ policy\_loose ]

countryName = optional

stateOrProvinceName = optional

localityName = optional

organizationName = optional

organizationalUnitName = optional

commonName = supplied

emailAddress = optional

[ req ]

# Options for the req tool, man req.

default\_bits = 2048

distinguished\_name = req\_distinguished\_name

string\_mask = utf8only

default\_md = sha256

# Extension to add when the -x509 option is used.

x509\_extensions = v3\_ca

[ req\_distinguished\_name ]

countryName = Country Name (2 letter code)

stateOrProvinceName = State or Province Name

localityName = Locality Name

0.organizationName = Organization Name

organizationalUnitName = Organizational Unit Name

commonName = Common Name

emailAddress = Email Address

countryName\_default = BD

stateOrProvinceName\_default = Dhaka

localityName\_default = Demra

0.organizationName\_default = EWU

organizationalUnitName\_default = Cyber\_Security

commonName\_default = AcmeCA

emailAddress\_default = riad@acmesub\_ca.com

[ v3\_ca ]

# Extensions to apply when createing root ca

# Extensions for a typical CA, man x509v3\_config

subjectKeyIdentifier = hash

authorityKeyIdentifier = keyid:always,issuer

basicConstraints = critical, CA:true

keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ v3\_intermediate\_ca ]

# Extensions to apply when creating intermediate or sub-ca

# Extensions for a typical intermediate CA, same man as above

subjectKeyIdentifier = hash

authorityKeyIdentifier = keyid:always,issuer

#pathlen:0 ensures no more sub-ca can be created below an intermediate

basicConstraints = critical, CA:true, pathlen:0

keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ server\_cert ]

# Extensions for server certificates

basicConstraints = CA:FALSE

nsCertType = server

nsComment = "OpenSSL Generated Server Certificate"

subjectKeyIdentifier = hash

authorityKeyIdentifier = keyid,issuer:always

keyUsage = critical, digitalSignature, keyEncipherment

extendedKeyUsage = serverAuth

***3.4. Requesting for sub ca certificate signing request.***

openssl req -config sub-ca.conf -new -key private/sub-ca.key -sha256 -out csr/sub-ca.csr

**moving to the previous folder**

cd -

**Signing the request of sub ca by root ca**

openssl ca -config root-ca.conf -extensions v3\_intermediate\_ca -days 3652 -notext -in ../sub-ca/csr/sub-ca.csr -out ../sub-ca/certs/sub-ca.crt

to confirm insert “y”

→.pem file has been generated

**See the signing**

cat index

→Root ca signed sub ca

**Seeing detail**

openssl x509 -noout -text -in ../sub-ca/certs/sub-ca.crt

A close up of a screen

Description automatically generated

***4. Configuring server***

**Moving to server**

cd ../server

**Generating certificate signing request from server**

openssl req -key private/server.key -new -sha256 -out csr/server.csr

**moving to sub ca to sign the server’s certificate**

cd ../sub-ca

**Sub ca signing certificate request of server**

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

**seeing detail**

cat index

**moving to certs folder to see certificate of server**

cd ../server/certs/

**See the directory by using command:**

Ls

→ We can see that the server.crt file has been generated

**Now, concating sub-ca.crt and server.crt and naming the new file chained.crt**

cat server.crt ../../sub-ca/certs/sub-ca.crt > chained.crt

**Seeing the change**

ls

**moving back to server directory**

cd ..

A computer screen shot of a computer error

Description automatically generated

echo "127.0.0.2 www.verysecureserver.com" >> /etc/hosts

ping [www.verysecureserver.com](http://www.verysecureserver.com/)

A computer screen with white text

Description automatically generated

***4.1.Turning on the ssl port***

**Turning on the ssl port**

openssl s\_server -accept 443 -www -key private/server.key -cert certs/server.crt -CAfile ../sub-ca/certs/sub-ca.crt

→Opening new terminal

**Again to root**

sudo -i

**See the port number used by different Ip addresses**

ss -ntl

sudo apt update

**To download or transfer files/data from or to a server using FTP, HTTP, HTTPS, SCP, SFTP, SMB and other supported protocols, installing curl:**

sudo apt install curl

**copying the certificate to ca certificate folder**

cp ca/root-ca/certs/ca.crt /usr/local/share/ca-certificates/

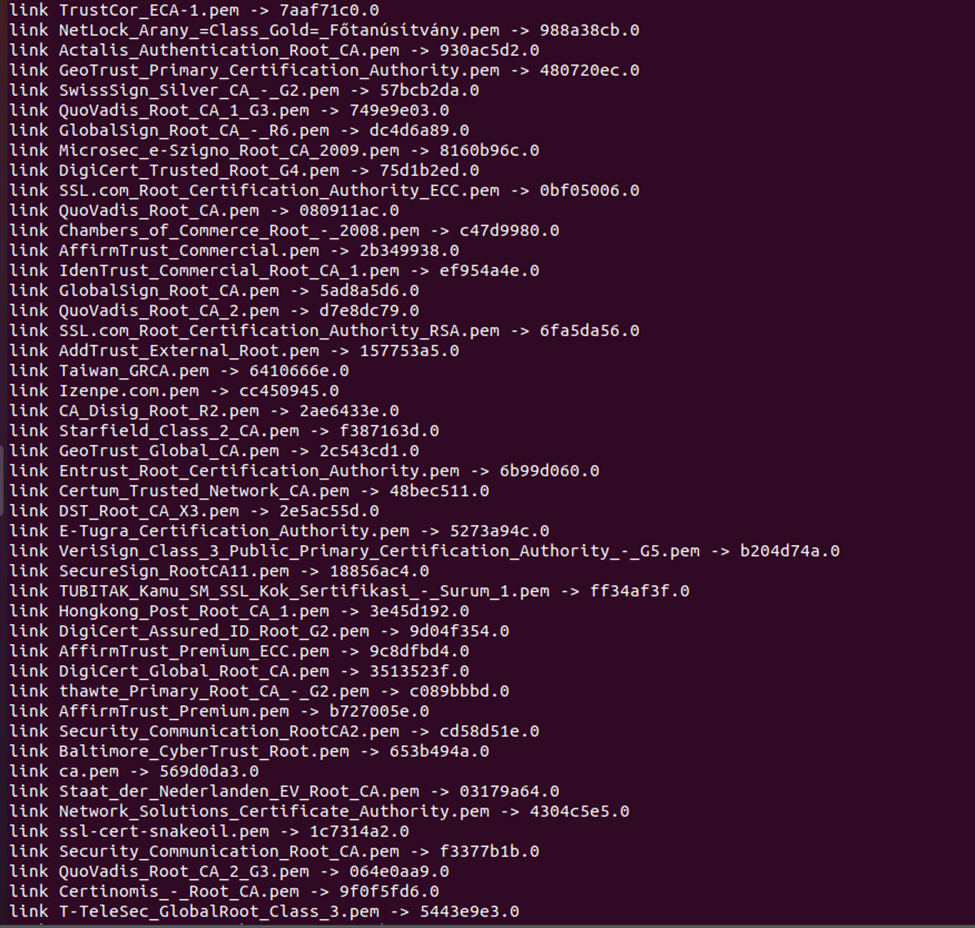
**Updating ca certificate folder**

update-ca-certificates –v

A screenshot of a computer program

Description automatically generated

***4.2.Update the file***



***4.3. Copy to newcerts directory***

sudo -i

tree ca

**finally copy to newcerts directory**

cp /root/ca/root-ca/newcerts/30F7CF5C522B5AE3ECC631190F31C3B2.pem ~ prinom /

cp /root/ca/sub-ca/newcerts/ 0ΕΑΕ96AF2467BE64D77862677E2E8710.pem ~ prinom /

cp /root/ca/root-ca/certs/ca.crt /home/ prinom /certificatea

cp /root/ca/sub-ca/certs/sub-ca.crt /home/ prinom /certificate/

cp /root/ca/server/certs/chained.crt /home/ prinom /certificate/

cp /root/ca/server/certs/server.crt /home/ prinom /certificate/

cp /root/ca/server/private/server.key /home/ prinom /certificate/



A computer screen shot of a computer code

Description automatically generated

***4.4.Open Httpd-ssl conf***

sudo -i

cd /opt/lampp/etc/extra

chmod 777 httpd-ssl.conf

gedit httpd-ssl.conf

**line 106**

change server.crt location with your server.crt file location

{106 SSLCertificateFile "/home/ prinom /certificate/server.crt"}

**line 116**

change server.key location with your server.key file location

{116 SSLCertificateKeyFile "/home/ prinom /certificate/server.key"}

**line 136**

change full line with your location

{136 SSLCACertificatePath "/home/ prinom /certificate"}

**For auto redirect to https**

<VirtualHost \_default\_:80>

ServerName www.example.com:80

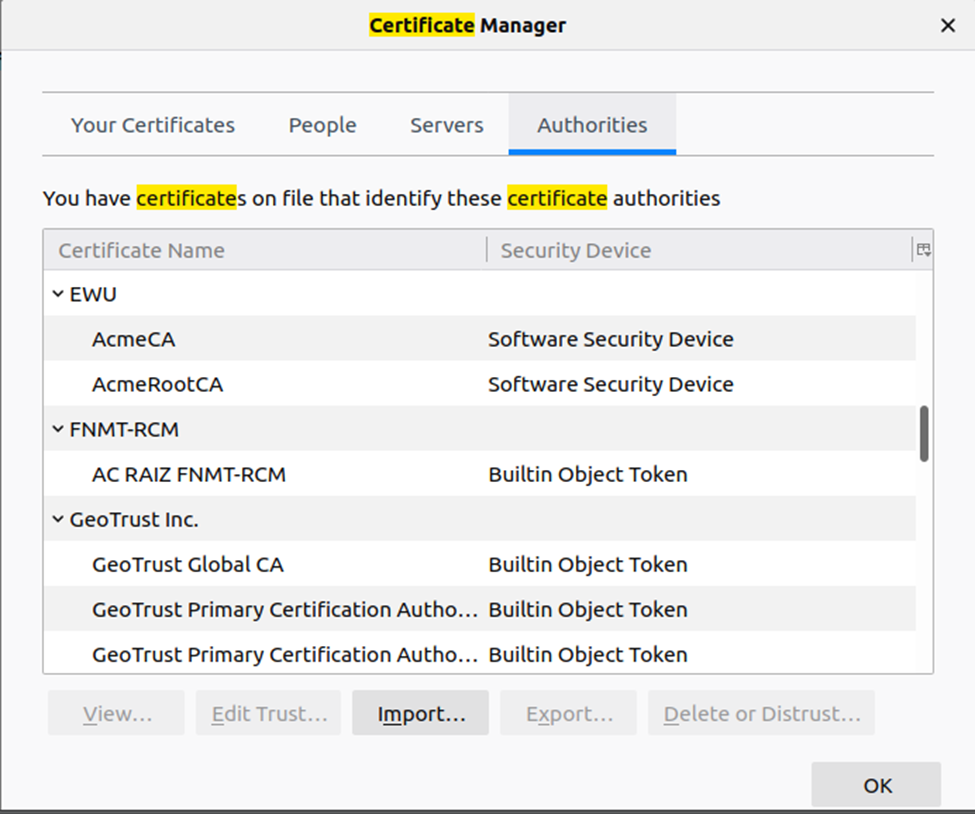
ServerAdmin you@example.com

Redirect permanent / https://www.verysecureserver.com

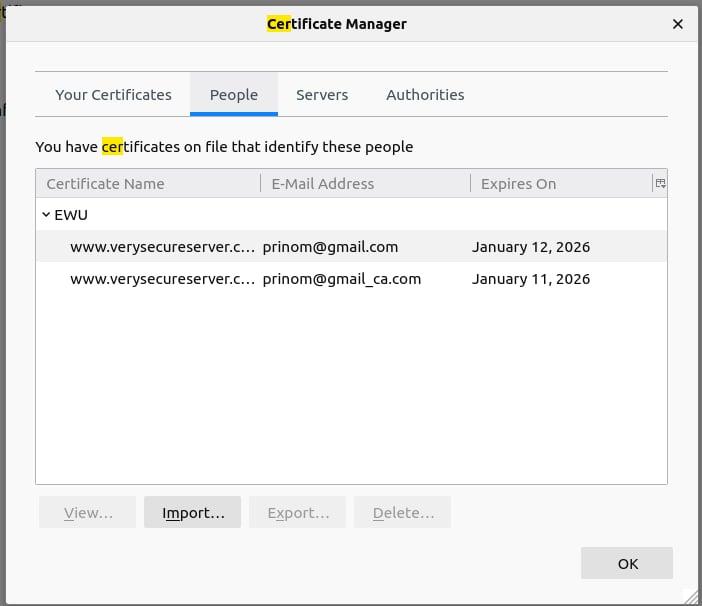
</VirtualHost>

***5.1. Upload pem file***

From preference open the certificate manager and import the pem file of the certificate. Then click on save. In the authorities section under Cyber\_Security common name the two files of the server will be shown named AcmeRootCA, AcmeCA



***5.2. Upload other certs file in people section***



***5.3. Show the secure server***

After all work is done xampp must restart the server. Then open it with [https://verysecureserver.com](https://verysecureserver.com/) It will show the server is secure.

A screenshot of a computer

Description automatically generated

**Figure: Server is secure**