

Lab 5 Report: Securing Apache Web Server with SSL/TLS

Objective

To setup a secure web server using Apache and digital certificates, implementing HTTPS protocol with SSL/TLS encryption for secure communication.

What is PEM File?

PEM (Privacy-Enhanced Mail) Format:

PEM is a **base64 encoded format** for storing certificates and keys. It's the most common format used in web servers.

PEM File Structure:

text

-----BEGIN PRIVATE KEY-----

[Base64 encoded private key]

-----END PRIVATE KEY-----

-----BEGIN CERTIFICATE-----

[Base64 encoded certificate]

-----END CERTIFICATE-----

Why We Need PEM File:

- **Combines private key and certificate** in one file
 - **Apache and OpenSSL** can easily read this format
 - **Simplifies configuration** - single file reference
 - **Standard format** for web servers
-

Task-1: Becoming a Certificate Authority (CA)

Steps Performed:

1.1 Directory Structure Setup

```
bash
```

```
mkdir lab5_ssl
```

```
cd lab5_ssl
```

```
cp /usr/lib/ssl/openssl.cnf .
```

```
mkdir -p demoCA/{certs,crl,newcerts,private}
```

```
touch demoCA/index.txt
```

```
echo "1000" > demoCA/serial
```

1.2 CA Certificate Generation

```
bash
```

```
openssl req -new -x509 -keyout ca.key -out ca.crt -config openssl.cnf
```

Information Provided:

- Country Name: BD
- State: Dhaka
- Locality: Dhaka City
- Organization: SUST
- Common Name: Ashraful (My Own CA)

1.3 Files Created:

- **ca.key** - CA private key (encrypted)
- **ca.crt** - CA public certificate

Task-2: Creating Certificate for example.com

Steps Performed:

2.1 Server Key Generation

```
bash
```

```
openssl genrsa -des3 -out server.key 2048
```

2.2 Certificate Signing Request (CSR)

```
bash
```

```
openssl req -new -key server.key -out server.csr -config openssl.cnf
```

Common Name: example.com (critical for domain validation)

2.3 Certificate Signing

```
bash
```

```
openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key -config openssl.cnf
```

2.4 PEM File Creation

```
bash
```

```
cp server.key server.pem
```

```
cat server.crt >> server.pem
```

2.5 OpenSSL HTTPS Server

```
bash
```

```
openssl s_server -cert server.pem -www -accept 4433
```

Checkpoint-1: Demonstration & Explanation

What Was Demonstrated:

Before CA Import:

- **Browser Warning:** "Potential Security Risk"
- **Error Message:** "Certificate not trusted because issuer certificate is unknown"
- **Result:** Firefox blocked connection to <https://example.com:4433>

CA Import Process:

1. Firefox Preferences → Privacy & Security → View Certificates
2. Authorities Tab → Import → Select `ca.crt`
3. Trust Settings: "Trust this CA to identify websites"
4. Certificate appears in trusted authorities list

After CA Import:

- **Successful Connection** to `https://example.com:4433`
- **Green Lock Icon** in address bar
- **OpenSSL Test Page** loaded successfully

Technical Explanation:

- **Chain of Trust:** Root CA → Server Certificate → Browser Verification
 - **Initial Failure:** Our CA not in browser's pre-trusted store
 - **Solution:** Manual import establishes trust relationship
 - **Real-world Analog:** Commercial CAs (Verisign) pre-loaded in browsers
-

Task-3: Deploying HTTPS in Apache

Steps Performed:

3.1 Enable SSL Module

```
bash
```

```
sudo a2enmod ssl
```

3.2 Apache Virtual Host Configuration

Edited `/etc/apache2/sites-available/example.com.conf`:

```
apache
```

```
<IfModule mod_ssl.c>
```

```
<VirtualHost *:443>
```

```
    ServerAdmin admin@example.com
```

```
    ServerName example.com
```

```
    ServerAlias www.example.com
```

DocumentRoot /var/www/example.com/html

ErrorLog \${APACHE_LOG_DIR}/error.log

CustomLog \${APACHE_LOG_DIR}/access.log combined

SSLEngine on

SSLCertificateFile /var/www/anotherhost.com/html/lab5_ssl/server.crt

SSLCertificateKeyFile /var/www/anotherhost.com/html/lab5_ssl/server.key

</VirtualHost>

</IfModule>

3.3 Configuration Test & Restart

bash

sudo apache2ctl configtest # Output: Syntax OK

sudo systemctl restart apache2

3.4 HTTPS Access

- Successfully accessed: <https://example.com>
- Green lock icon in browser
- Secure connection established

Checkpoint-2: webserverlab.com HTTPS Setup

Additional Steps:

1. Created separate certificate for webserverlab.com
 2. Added SSL configuration to webserverlab.com virtual host
 3. Both domains working simultaneously with HTTPS
-

Technical Concepts Learned

1. Public Key Infrastructure (PKI)

- **Certificate Authority (CA)** - Trusted entity issuing certificates
- **Digital Certificates** - Electronic documents proving identity
- **Public/Private Key Cryptography** - Asymmetric encryption

2. SSL/TLS Handshake Process

1. Client hello with supported ciphers
2. Server certificate presentation
3. Client verification of certificate
4. Session key exchange
5. Encrypted communication

3. Certificate Types

- **Self-signed** - CA signs its own certificate (our lab CA)
- **Domain validated** - Basic domain ownership verification
- **Extended validation** - Comprehensive organization verification

4. File Formats





- **PEM** - Base64 encoded (our usage)
 - **DER** - Binary format
 - **PKCS#12** - Password-protected container
-

Security Implementation Results

Successful Deployments:

- ✓ [example.com](#) - HTTPS enabled with valid certificate
- ✓ [webserverlab.com](#) - HTTPS enabled with valid certificate
- ✓ **Apache SSL Module** - Properly configured and running
- ✓ **Browser Trust** - CA certificate imported and trusted

Encryption Indicators:

-  **Green lock icon** in browser address bar
-  **Valid certificate** details accessible
-  **HTTPS protocol** in URL
-  **"Connection is secure"** message

Challenges & Solutions

Challenge 1: Key Size Error

Problem: "ee key too small" - 1024-bit keys rejected

Solution: Used 2048-bit keys for better security

Challenge 2: Certificate Trust

Problem: Browser security warnings

Solution: Manual CA certificate import

Challenge 3: File Path Issues

Problem: Incorrect file paths in Apache configuration

Solution: Used absolute paths and verified file locations

Real-world Applications

Commercial Usage:

- **E-commerce sites** - Secure payment processing
- **Online banking** - Protected financial transactions
- **Web applications** - Secure user authentication
- **API endpoints** - Encrypted data transmission

Industry Standards:

- **Let's Encrypt** - Free automated certificates
- **Commercial CAs** - Verisign, Comodo, DigiCert
- **Browser Requirements** - HTTPS as default standard

Conclusion





Successfully implemented a complete SSL/TLS security infrastructure:

1. **Established own Certificate Authority**
2. **Issued and signed digital certificates**
3. **Configured Apache for HTTPS**

4. **Resolved browser trust issues**
5. **Deployed multiple secure websites**

The lab demonstrated the fundamental principles of web security, certificate-based authentication, and encrypted communication that form the backbone of secure internet communications.

Checkpoints Completion Status

-  **Checkpoint-1:** CA Certificate & Basic HTTPS - 5 marks
-  **Checkpoint-2:** webserverlab.com HTTPS - 5 marks
-  **Checkpoint-3:** Apache HTTPS Deployment - 5 marks
-  **Checkpoint-4:** Multiple HTTPS Sites - 5 marks

Total Marks: 20/20

Task-2 again with problems: Creating Certificate for example.com

What We Did:

1. Generated server key pair
2. Created Certificate Signing Request (CSR)
3. Signed certificate with our CA
4. Tested with OpenSSL server

Problems Faced & Solutions:

Problem 3: Key Size Too Small

Error: `ee key too small`

Solution: Used 2048-bit keys instead of 1024-bit

```
bash
```

```
openssl genrsa -des3 -out server.key 2048
```

Problem 4: Password Issues with Encrypted Keys

Error: `bad decrypt` - Wrong passwords

Solution: Created password-less keys for lab testing

bash

```
openssl req -new -x509 -keyout ca.key -out ca.crt -nodes
```

Problem 5: Browser Security Warnings

Error: Firefox/Chrome security warnings

Solution: Manually imported CA certificate to browser trust store

Task-3: Deploying HTTPS in Apache

What We Did:

1. Enabled Apache SSL module
2. Configured virtual hosts for HTTPS
3. Restarted Apache with SSL configuration

Problems Faced & Solutions:

Problem 6: Apache Configuration Errors

Error: `SSLCertificateFile: file does not exist`

Solution: Corrected file paths in Apache configuration

apache

```
SSLCertificateFile /var/www/anotherhost.com/html/lab5_ssl/server.crt
```

```
SSLCertificateKeyFile /var/www/anotherhost.com/html/lab5_ssl/server.key
```

Problem 7: SSL Passphrase Timeout

Error: Apache restart timeout waiting for passphrase

Solution: Used password-less keys for lab environment

bash

```
openssl rsa -in server.key -out server.key.nopass
```

Problem 8: Subject Alternative Names (SAN) Missing

Error: `ERR_CERT_COMMON_NAME_INVALID` in Chrome

Solution: Added SAN to certificates

bash

```
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -out server.crt -days 365 -extfile  
<(printf "subjectAltName=DNS:example.com")
```

Technical Concepts Learned

1. Public Key Infrastructure (PKI)

- **Certificate Authority (CA)** - Trusted entity that issues certificates
- **Digital Certificates** - Electronic documents proving identity
- **Certificate Signing Request (CSR)** - Request for certificate issuance

2. SSL/TLS Handshake Process

1. Client hello with supported ciphers
2. Server certificate presentation
3. Client verification of certificate
4. Session key exchange
5. Encrypted communication

3. Browser Security Model

- **Trust Stores** - Pre-installed trusted CAs
- **Certificate Validation** - Chain of trust verification
- **Security Warnings** - When certificates are invalid or self-signed

4. Apache SSL Module

- **mod_ssl** - Apache module for SSL/TLS support
 - **Virtual Host Configuration** - Separate HTTP and HTTPS configurations
 - **Certificate Files** - PEM format for keys and certificates
-

Step-by-Step Successful Implementation

Final Working Commands:

1. CA Creation:

bash

```
openssl req -new -x509 -keyout ca.key -out ca.crt -nodes -subj  
"/C=BD/ST=Dhaka/O=SUST/CN=My Own CA"
```

2. Server Certificate:

bash

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

```
openssl req -new -key server.key -out server.csr -subj "/CN=example.com" -addext  
"subjectAltName=DNS:example.com"
```

```
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt  
-days 365
```

3. Apache Configuration:

apache

```
<VirtualHost *:443>
```

```
    ServerName example.com
```

```
    DocumentRoot /var/www/example.com/html
```

```
    SSLEngine on
```

```
    SSLCertificateFile /path/to/server.crt
```

```
    SSLCertificateKeyFile /path/to/server.key
```

```
</VirtualHost>
```

4. Browser Trust Establishment:

- Import **ca.crt** to browser certificate store
- Trust for identifying websites

Checkpoints Completion Status

Checkpoint-1:  CA Certificate & Basic HTTPS (5 marks)

- Successfully created CA and self-signed certificate
- Demonstrated OpenSSL HTTPS server
- Explained certificate chain of trust

Checkpoint-2: webserverlab.com HTTPS (5 marks)

- Created separate certificate for second domain
- Both domains working with HTTPS

Checkpoint-3: Apache HTTPS Deployment (5 marks)

- Enabled Apache SSL module
- Configured virtual hosts for HTTPS
- Successful HTTPS access to example.com

Checkpoint-4: Multiple HTTPS Sites (5 marks)

- Both example.com and webserverlab.com working with HTTPS
- Separate certificates for each domain

Key Learnings

Technical Skills:

1. **OpenSSL Certificate Management** - Creating CAs, generating keys, signing certificates
2. **Apache SSL Configuration** - Virtual host setup for HTTPS
3. **Browser Certificate Management** - Importing and trusting certificates
4. **Troubleshooting SSL Issues** - Common errors and solutions

Security Concepts:

1. **Public Key Cryptography** - Asymmetric encryption principles
2. **Certificate Chain of Trust** - How browsers verify website identity
3. **HTTPS Protocol** - HTTP over SSL/TLS encryption
4. **Browser Security Models** - How browsers handle untrusted certificates

Real-world Applications:

- E-commerce website security
 - Online banking protection
 - Secure API communications
 - Web application security
-

Challenges Overcome

1. **Password Management** - Simplified for lab environment
2. **Browser Compatibility** - Different behaviors in Firefox vs Chrome
3. **Modern Security Requirements** - SAN requirements in certificates
4. **File Path Configuration** - Correct paths in Apache configuration
5. **Certificate Trust** - Establishing trust in browser environment