Report: Lab 5 - Securing Apache Web Server Using Digital Certificates and HTTPS

Objective:
The lab focused on setting up a secure Apache web server by implementing HTTPS using SSL/TLS and digital certificates. This involved creating a Root Certificate Authority (CA), generating certificates for servers, and configuring Apache to support secure communication.

Overview:
The HTTP protocol is inherently insecure, as it lacks encryption. SSL and its successor TLS overlay cryptographic security onto HTTP to provide HTTPS, which ensures confidentiality, integrity, and authentication between browsers and servers. The critical element in HTTPS is the use of digital certificates, which validate the identity of the server and enable encrypted communication using asymmetric and symmetric cryptography.

Lab Tasks and Steps:

1. Becoming a Certificate Authority (CA):
- Created a self-signed Root CA certificate (ca.crt) and private key (ca.key) using OpenSSL and a custom configuration file (openssl.cnf).
- Root CA certificates are trusted entities that issue certificates for servers, and their own certificates are self-signed.
- Set up required directory structure, index files, and serial number files for the CA.
2. Creating Server Certificates:
- Generated RSA public/private key pairs for server domains (example.com, webserverlab.com) with password protection.
- Created Certificate Signing Requests (CSRs) for each server key.
- Signed CSRs using the Root CA key and certificate, thus issuing server certificates trusted by the custom CA.
3. Launching OpenSSL Test Server:
- Combined the server key and certificate into a PEM file.
- Launched a simple OpenSSL web server listening on port 4433 with the generated PEM file.
- Accessed the server via HTTPS; browsers initially showed certificate warnings because the custom CA was unknown.
- Added the Root CA certificate to browser trusted certificate authorities manually to resolve warnings.
4. Deploying HTTPS on Apache:
- Enabled SSL module in Apache (`sudo a2enmod ssl`).

- Configured Apache virtual hosts to use SSL and point to the proper certificate and key files.
- Verified Apache configuration with `apache2ctl configtest` and restarted Apache.
- Accessed configured sites (example.com, webserverlab.com) via HTTPS and confirmed secure connections using the custom certificates.

Observations:

- The lab demonstrated the creation and use of digital certificates for HTTPS.
- Browsers trust only pre-installed or user-added CAs, which is why initial warnings appear for self-signed or custom CA certificates.
- The process of issuing digital certificates involves careful coordination between CA and server key management.
- HTTPS setup enhances web server security by encrypting the web traffic and authenticating the server to clients.

Conclusion:
This lab provided practical experience in securing web servers with SSL/TLS by becoming a certificate authority, issuing trusted server certificates, and configuring Apache for HTTPS support. It emphasized the importance of certificate chains and trust in public key infrastructure for secure internet communication.