ACO431 Network Security Project 3 TLS

Stage 1: Build Public Key Infrastructure to Secure Web Communication
Due: November 17, 2025 before class starts
Note: This project should be done in a group of two

Description: In Stage 1 of this project, you are asked to set up a Certificate Authority (CA) to issue and manage certificates, then configure a web server and a browser to use TLS to secure their web communication. openssl can be used to carry out CA functionalities. Apache server has an embedded module called mod_ssl for TLS. A web browser automatically uses TLS to communicate with an https web server.

Network Configuration:
Server: The guest computer from project 1 with Kali installed can serve as Certificate Authority and host the secure web server.
Client: The host computer from project 1 can serve as the web client.

Step 1: Set up a simulated certificate authority

1. Set up the root CA. Study naming guidelines for the common name of CA and name yours properly.
2. Generates a key pair for the web server and issues a certificate request. The common name of your server should be www.yourgroupname.com. The server's common name should be different from CA's common name to avoid naming conflict. Make sure subject alternative name information is also included in the certificate request.
3. The root CA issues certificate for the server.
4. Study the CA's certificate and server's certificate and answer the following questions:
    a. How can you tell a certificate belongs to a root CA's. Use your certificate to explain.
    b. How can you tell a certificate belongs to a web server? Use your certificate to explain.
    c. For your web server's certificate, report the following:
        i. Its public key
        ii. key and algorithm used to sign the certificate
        iii. Does the certificate include the private key of the subject? Why yes, why not.

Note: please name the certificate file with .crt suffix and the key file with .pem suffix

Step 2: Set up a TLS-enabled web server

1. Configure apache web server to enable ssl and specify server key and certificate. Make sure the certificate and key files are under the directory you specified in the configuration file
2. Enable the ssl module (a2enmod ssl), enable SSL Virtual Host (a2ensite default-ssl), do configuration test (apache2ctl configtest), then restart the web server

(systemctl restart apache2). Issue “netstat –na | grep 443” to make sure that ssl module is loaded.

3. Configure *hosts* file in client computer so web browser can access web server with its domain name instead of ip address. The domain name has to match what was specified in the server’s certificate.
4. Load root CA’s certificate to browser in client computer
5. Start your browser in host and access the https server on guest using its domain name

Deliverables:
Please report the followings:

1. A list of openssl commands used for generating the necessary keys and certificates. Briefly explain the purpose of each command.
2. The certificate of the root CA and server in their readable format.
3. A figure to illustrate your network topology
4. Description of the configuration on both client and server side to support TLS
5. Answer to the questions in Step 1.4

Following Links might contain helpful information. Please feel free to find your own reference.

1. The manual of openssl command. req, genrsa, ca, x509 are related to certificate https://www.openssl.org/docs/man1.1.1/man1/
2. A tutorial on “How to setup your own CA with OpenSSL” https://gist.github.com/Soarez/9688998
3. How To Enable HTTPS Protocol with Apache 2 on Ubuntu 20.04 https://www.rosehosting.com/blog/how-to-enable-https-protocol-with-apache-2-on-ubuntu-20-04/
4. How To Create a Self-Signed SSL Certificate for Apache in Ubuntu 18.04 (ignore the configuration of ssl-params.conf file)
   https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-apache-in-ubuntu-18-04