Roll#	Name
251683960	Timothious Gill

Assignment #2

Public Key Infrastructure (PKI) using OpenSSL

Introduction:

Public Key Infrastructure (PKI) is a crucial framework for enabling secure communication and authentication in digital systems. This report demonstrates the use of OpenSSL, a widely used open-source tool, to implement PKI tasks. By generating keys, creating a Certificate Signing Request (CSR), issuing self-signed certificates, and acting as a Certification Authority (CA), this assignment explores the foundational processes behind PKI.

Objective:

The primary objective of this assignment is to demonstrate the use of Public Key Infrastructure (PKI) using the OpenSSL tool. The tasks aim to provide hands-on experience with:

1. Generating private and public keys.
2. Creating a Certificate Signing Request (CSR).
3. Generating a self-signed certificate.
4. Acting as a Certification Authority (CA) to issue certificates.

Methodology:

This assignment was performed step-by-step using OpenSSL commands to implement the PKI tasks. Each step involved executing specific commands, verifying the results, and capturing screenshots as evidence.

Below is a detailed explanation of the tasks performed, including the commands used and their outcomes.

1. Generate a Private and Public Key

Command Used:

$ openssl genrsa -out private_key.pem 2048

This command generated a private key file named private_key.pem with 2048-bit RSA encryption.

Command Used:

$ openssl rsa -in private_key.pem -pubout -out public_key.pem

This command extracted the public key from the private key and saved it in a file named public_key.pem.

```
┌──(kali㉿kali)-[~/InfoSec/A2]
└─$ openssl genrsa -out private_key.pem 2048


┌──(kali㉿kali)-[~/InfoSec/A2]
└─$ openssl rsa -in private_key.pem -pubout -out public_key.pem

writing RSA key

┌──(kali㉿kali)-[~/InfoSec/A2]
└─$ ls private_key.pem public_key.pem

private_key.pem  public_key.pem
```

2. Generate a Certificate Signing Request (CSR):

   Command Used:

   $ openssl req -new -key private_key.pem -out certificate_request.csr

   This command created a Certificate Signing Request (CSR) file named certificate_request.csr. During execution, the OpenSSL tool prompted for Distinguished Name (DN) details such as country, state, organization, and common name.

   Example input provided during CSR generation:

   - **Country Name:** PK
   - **State:** Punjab
   - **Locality:** Lahore
   - **Organization Name:** IT
   - **Common Name:** Timothious
   - **Email Address:** timothiousgill23@gmail.com

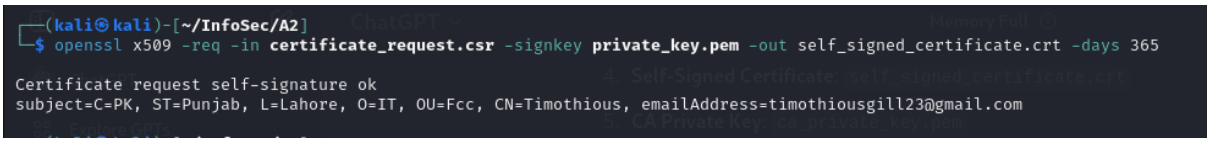
```
┌──(kali㉿kali)-[~/InfoSec/A2]
└─$ openssl req -new -key private_key.pem -out certificate_request.csr

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) [AU]:PK
State or Province Name (full name) [Some-State]:Punjab
Locality Name (eg, city) []:Lahore
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IT
Organizational Unit Name (eg, section) []:Fcc
Common Name (e.g. server FQDN or YOUR name) []:Timothious
Email Address []:timothiousgill23@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:kali
An optional company name []:kali
```

3. Create a Self-Signed Certificate:

   Command Used:

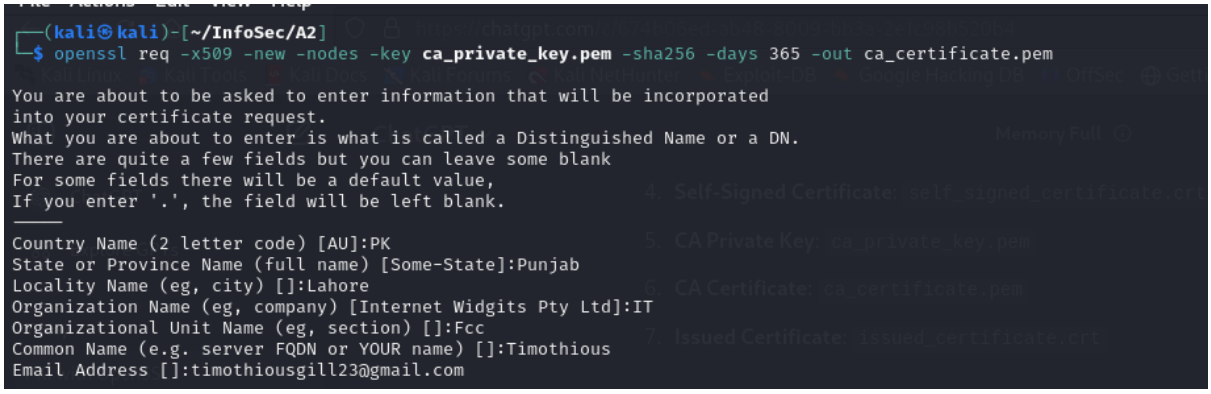
$ openssl x509 -req -in certificate_request.csr -signkey private_key.pem -out self_signed_certification.crt -days 365

This command used the CSR and private key to create a self-signed certificate named self_signed_certification.crt, valid for 365 days.



4. Act as a Certification Authority (CA) to Issue Certificates:

- Step 1: Generate a CA Private Key
  - Command Used:
    - $ penssl genrsa -out ca_private_key.pem 2048
- Step 2: Create a CA Certificate
  - Command Used:
    - $ openssl req -x509 -new -nodes -key ca_private_key.pem -sha256 -days 365 -out ca_certificate.pem



5. Verify the Issued Certificate:

Command Used:

$ openssl verify -CAfile ca_certificate.pem issued_certificate.crt

6. Conclusion:

Through this assignment, I successfully demonstrated the use of OpenSSL to implement Public Key Infrastructure (PKI). The process included generating keys, creating a CSR, issuing self-signed certificates, and acting as a Certification Authority (CA). This exercise provided valuable insights into the fundamental operations of PKI, showcasing its importance in secure communication and authentication.