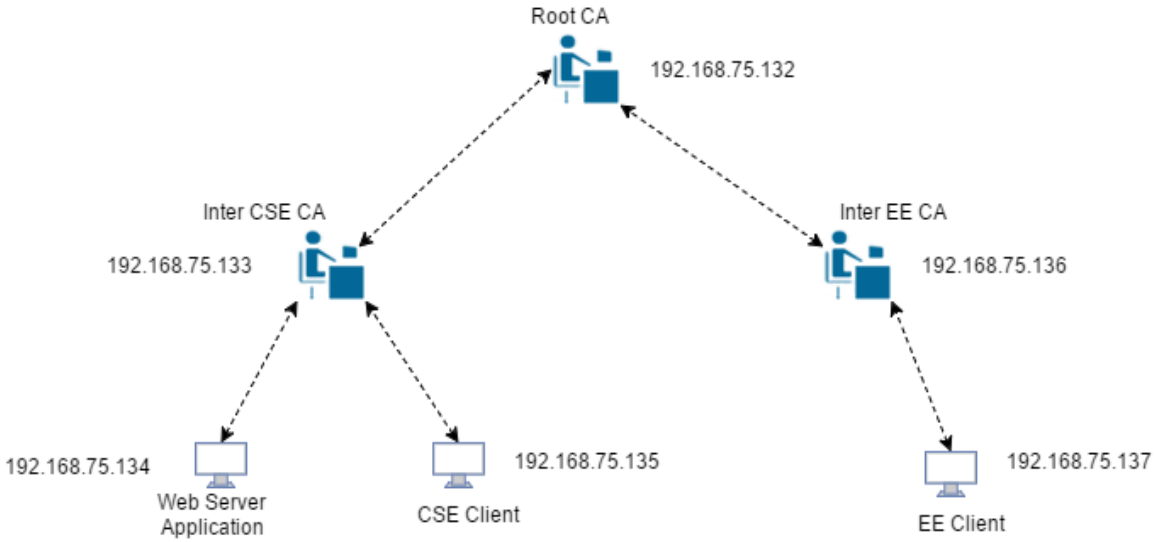
Report

Mrinal Aich (CS16MTECH11009)

Architecture:



Part – 1: **Creating a Public Key Infrastructure (PKI) for IITH:**

Tasks w.r.t. **Client**:
The following points are useful to create the certificate at the client side:

1.To generate the keys for the Certificate Signing Request (CSR):
openssl genrsa -des3 -out <key_name>.key 2048

2. To create the CSR, run the following command at a terminal prompt:
openssl req -new -key <key_name>.key -out <request_csr>.csr

3. The csr request is sent to the Certificate Authority in a secured way.

4. On receiving the new certificate from the CA, configure the appropriate applications to use it. The default location to install certificates is **/etc/ssl/certs**.

Tasks w.r.t. **Certification Authority**:

1 .Create the directories to hold the CA certificate and related files:

mkdir /etc/ssl/CA
mkdir /etc/ssl/newcerts

2. The CA needs a few additional files to operate,

2.1 To keep track of the last serial number used by the CA, each certificate must have a unique serial number.
2.2 To record which certificates have been issued.

echo '01' > /etc/ssl/CA/serial
touch /etc/ssl/CA/index.txt

3. The CA configuration file. Convenient when issuing multiple certificates.
Edit **/etc/ssl/openssl.cnf**, and in the [CA_default] change:

```
dir             = /etc/ssl                  # Where everything is kept
database        = $dir/CA/index.txt         # database index file.
certificate     = $dir/certs/<CA_cert>.pem  # The CA certificate
serial          = $dir/CA/serial            # The current serial number
private_key     = $dir/private/<ca_key>.pem         # The private key
```

4. Create self-signed ROOT certificate:

openssl req -new -x509 -extensions v3_ca -keyout cakey.pem -out cacert.pem -days 3650

5. Install the root certificate and key:

sudo mv cakey.pem /etc/ssl/private/
sudo mv cacert.pem /etc/ssl/certs/

6. Ready to start signing certificates. The first item needed is a Certificate Signing Request (CSR). Once CA have a CSR, enter the following to generate a certificate signed by the CA:

sudo openssl ca -in <cert_request>.csr -config /etc/ssl/openssl.cnf

7. The certificate is generated at /etc/ssl/newcerts/01.pem. Copy and paste everything beginning with the line:

----BEGIN CERTIFICATE----
and continuing through the line
----END CERTIFICATE-----

lines to a file named after the hostname of the server where the certificate will be installed. Subsequent certificates will be named 02.pem, 03.pem, etc.

NOTE: Impactful change in root CA's openssl.conf file.
In sslv3_extension, Basic Constraint - the root CA authority sign's certificate for clients who can themselves act as Intermediate CA authorities. This change can be seen in the generated .pem as:

X509v3 extensions:
    X509v3 Basic Constraints:
        CA:TRUE

Procedure:
Same tasks needs to be performed by the intermediate-CAs,

1. Generate private-key and csr-request and send it to the immediate upper CA.
2. Receive the certificate and store the it in /etc/ssl/certs.
3. Now, interCA can itself behave as a Certificate Authority by following the tasks mentioned for a CA above.

Debugging:

1. Analyzing CSR Request:

openssl req -in server.csr -out cst.txt -text

2. Convert key file/crt into PEM-format:

openssl x509 -in hostname.crt -inform DER -out hostname.crt.pem -outform PEM
openssl rsa -in hostname.key -out hostname.key.pem -outform PEM

Part – 2: Creating a webserver with HTTPS support – Works!!!

1. Host the webserver application at /var/www/html/index.php

2. Configure apache2 to use ssl certificates.

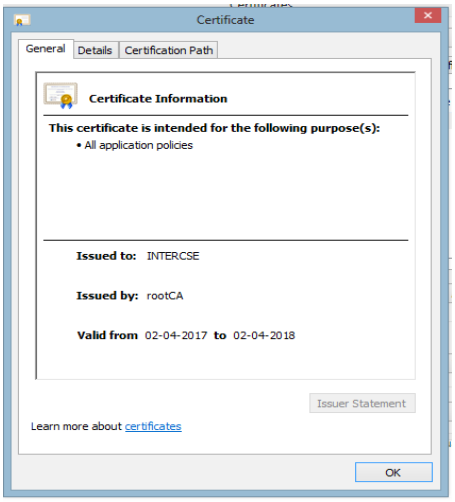
The following changes are needed at /etc/apache2/sites-available/default-ssl.conf

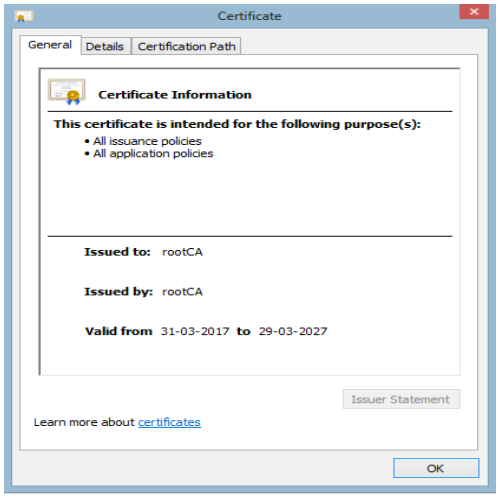
<VirtualHost 192.168.75.134:443>
ServerAdmin admin@pms
ServerName pms
DocumentRoot /var/www/html/

SSLCertificateFile <CertificateFile>
SSLCertificateKeyFile <KeyFile>
SSLCertificateChainFile <CertificateChainFile>

3. Install Certificates in the Browser.
(i). Add Root Certificate as Trusted Root Certification Authority.

(ii). Add Intermediate Certificate as Intermediate Certificate Authority which in turn is certified by the root Certificate.





4. Connect to the web application.
(i). On https://<serverName | serverIP>, ssl connection is created.
(ii). Enter details at the respective fields and enter submit.
(iii). Wireshark capture shows the TLSv1.2 packets being exchanged.

Part – 3: Secure Peer-to-Peer application using PKI:

Chat application in python – **Works!!!**

Client – Server side code:

The client sends its certificate and its certificate chain in Client-Hello to the server.
Code:

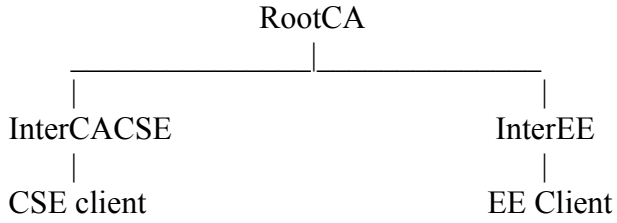
```
context = ssl.create_default_context()

context = ssl.SSLContext(ssl.PROTOCOL_SSLv23)
context.verify_mode = ssl.CERT_REQUIRED
context.load_cert_chain(certfile=CERT_FILE, keyfile=KEY_FILE, password="1234")
context.load_verify_locations(TRUSTED_FILE)
```

CERT_FILE – Own certificate file concatenated with upper layer Certificate Authorities.
KEY_FILE – Own private key file.
TRUSTED_FILE – File containing all trusted CA's certificate.

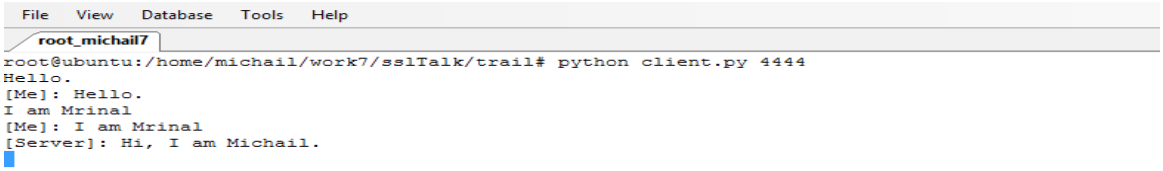
Similar code for Server-side aswell, with a basic changes indicating the socket to act as a server.

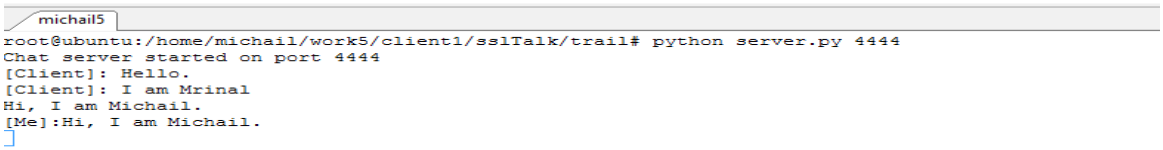
```
connstream = context.wrap_socket(newsocket, server_side=True)
```

Architectue:



Rest of the files alongwith python scripts, certificates, wireshark captures and screenshots are enclosed in the zip file.





References:

1. https://wiki.openssl.org/index.php/Manual:Openssl(1)
2. https://docs.python.org/2/library/ssl.html
3. http://simplestcodings.blogspot.in/2010/08/secure-server-client-using-openssl-in-c.html
4. https://help.ubuntu.com/lts/serverguide/certificates-and-security.html
5. https://www.digitalocean.com/community/tutorials/how-to-create-a-ssl-certificate-on-apache-for-ubuntu-14-04
6. https://www.digicert.com/ssl-support/pem-ssl-creation.html
7. https://blogs.msdn.microsoft.com/kaushal/2013/08/02/ssl-handshake-and-https-bindings-on-iis/