Project Report
Part -1

Objective: To Understand PKI and launching a Man in the Middle Attack.

Lab Environment: Ubuntu 16.04 vm downloaded from SEED website.

Library and commands used : OpenSSL

Procedure/Tasks and Observation :

Task 1:
Becoming the Certificate Authority:

A Certificate Authority (CA) is a trusted entity that issues digital certificates.

1) I copied the config file from “/usr/lib/ssl/openssl.cnf” to the pwd.

2) I created the following sub-directories(in the pwd):

dir	./demoCA	Where everything is kept
certs	$dir/certs	Where the issued certs are kept
crl_dir	$dir/crl	Where the issued crl are kept
new_certs_dir	$dir/newcerts	default place for new certs.
database	$dir/index.txt	database index file.
serial	$dir/serial	The current serial number

3) I generated a self signed certificate for our CA by running the following command: “ openssl req -new -x509 -keyout ca.key -out ca.crt -config openssl.cnf”

Task 2:
Creating a Certificate for SEEDPKILab2018.com:

1) I generated the public/private key pairs for the website using the following command: “ openssl genrsa -aes128 -out server.key 1024”

The keys will be stored in the file server.key which is encoded text file (also encrypted) To see those, I ran the following command: “ openssl rsa -in server.key -text”

2) I generated a Certificate Signing Request (CSR) for our client website by running the following command : “ openssl req -new -key server.key -out server.csr -config openssl.cnf”

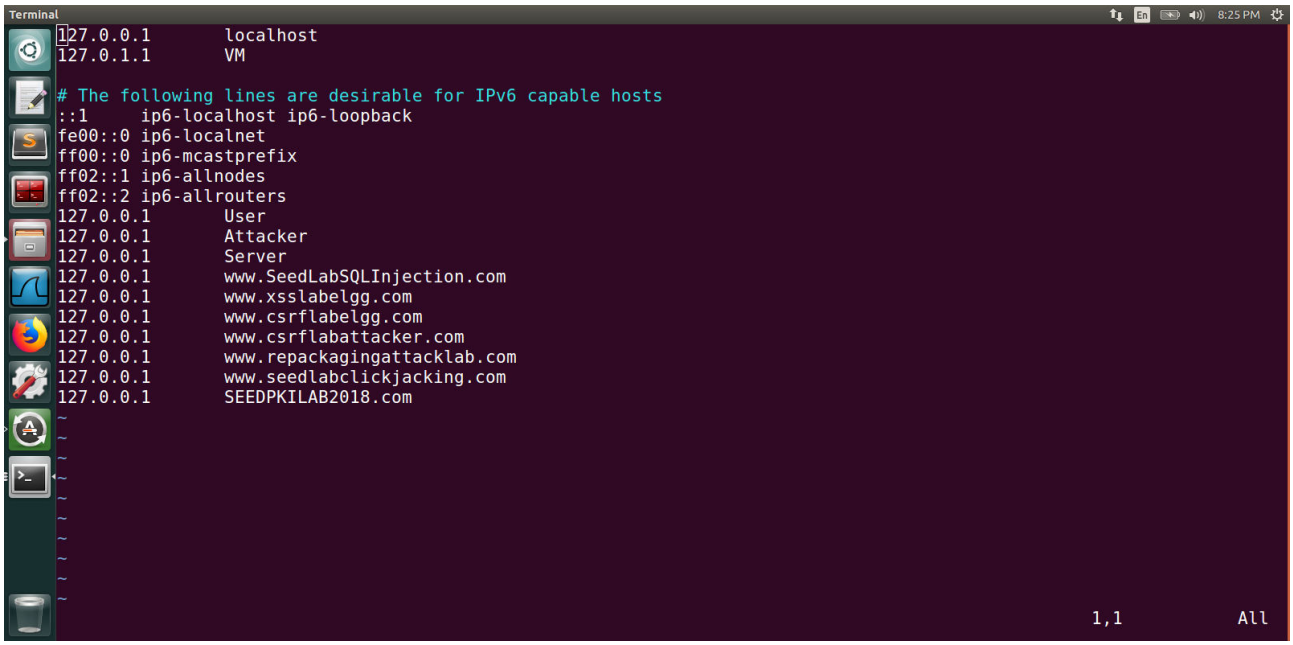
This command is very similar to the privious command with the only difference being the -x509 option which generates a self signed certificate instead of a CSR.

3)I generated certificates from the CSR by using the following command: “openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key -config openssl.cnf”

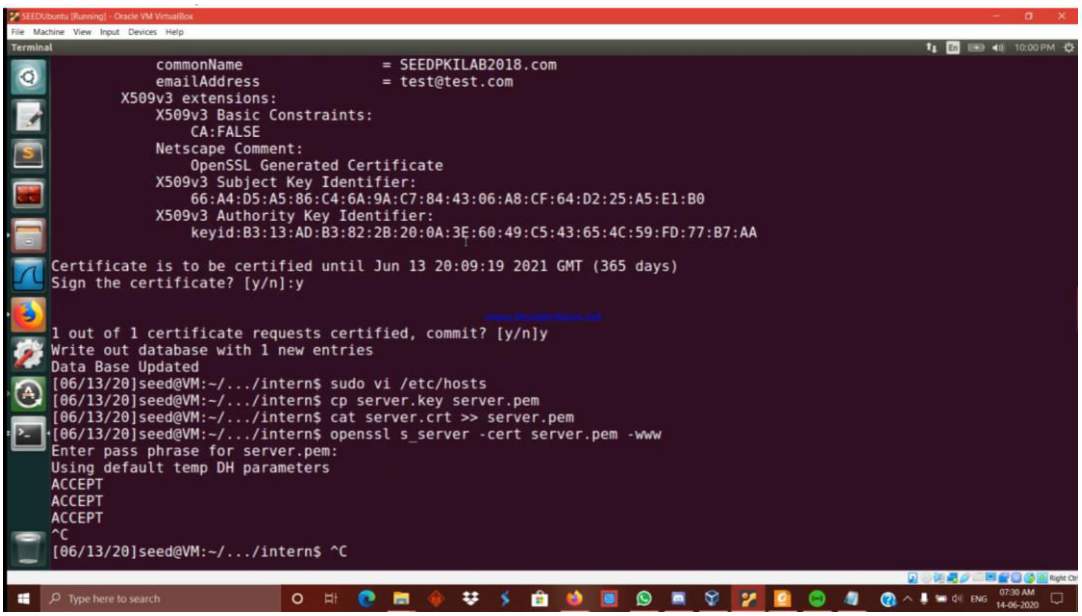
This command turns the certificate signing request (server.csr) into an X509 certificate (server.crt), using the CA’s ca.crt and ca.key

Task 3:
Deploying Certificate in an HTTPS Web Server:

1) I configured the DNS.I chose SEEDPKILab2018.com as the name of our website. To get it recognized I added the following entry to /etc/hosts:
127.0.0.1 SEEDPKILab2018.com
This entry basically maps the hostname SEEDPKILab2018.com to our localhost (i.e., 127.0.0.1).
I used the vim editor to do so using the following command:
“sudo vi /etc/host”



2) I configured the web server OpenSSL allows us to start a simple web server using the s_server command but first I need to combine the secret key and certificate into one file by using the following commands:
“cp server.key server.pem” (to copy server.key to server.pem)
“cat server.crt >> server.pem” (to concatenate server.crt to server.pem)
I now launch the web server using server.pem and use the command: “openssl s_server -cert server.pem -www”



3) I now open Firefox and the the url: https://seedpkilab2018.com:4433 and get an invalid security certificate error as our certificate is not yet trusted by the browser.

4) I make the browser accept our CA certificate by importing it from the following menu:
Edit -> Preference -> Privacy & Security -> View Certificates

Here I import ca.crt and select the following option: “Trust this CA to identify web sites”.

5) I now open a new tab and type the url again and get the following text in the website:
(also when I tried accessing by typing localhost I recieved the following error : SSL_ERROR_BAD_CERT_DOMAIN)
“

s_server -cert server.pem -www
Secure Renegotiation IS supported
Ciphers supported in s_server binary
TLSv1/SSLv3:ECDHE-RSA-AES256-GCM-SHA384TLSv1/SSLv3:ECDHE-ECDSA-AES256-GCM-SHA384
TLSv1/SSLv3:ECDHE-RSA-AES256-SHA384
TLSv1/SSLv3:ECDHE-ECDSA-AES256-SHA384

TLSv1/SSLv3:ECDHE-RSA-AES256-SHA
TLSv1/SSLv3:ECDHE-ECDSA-AES256-SHA
TLSv1/SSLv3:SRP-DSS-AES-256-CBC-SHA
TLSv1/SSLv3:SRP-RSA-AES-256-CBC-SHA
TLSv1/SSLv3:SRP-AES-256-CBC-SHA TLSv1/SSLv3:DH-DSS-AES256-GCM-SHA384
TLSv1/SSLv3:DHE-DSS-AES256-GCM-SHA384TLSv1/SSLv3:DH-RSA-AES256-GCM-SHA384
TLSv1/SSLv3:DHE-RSA-AES256-GCM-SHA384TLSv1/SSLv3:DHE-RSA-AES256-SHA256
TLSv1/SSLv3:DHE-DSS-AES256-SHA256 TLSv1/SSLv3:DH-RSA-AES256-SHA256
TLSv1/SSLv3:DH-DSS-AES256-SHA256
TLSv1/SSLv3:DHE-RSA-AES256-SHA
TLSv1/SSLv3:DHE-DSS-AES256-SHA TLSv1/SSLv3:DH-RSA-AES256-SHA
TLSv1/SSLv3:DH-DSS-AES256-SHA TLSv1/SSLv3:DHE-RSA-CAMELLIA256-SHA
TLSv1/SSLv3:DHE-DSS-CAMELLIA256-SHA
TLSv1/SSLv3:DH-RSA-CAMELLIA256-SHA
TLSv1/SSLv3:DH-DSS-CAMELLIA256-SHA
TLSv1/SSLv3:ECDH-RSA-AES256-GCM-SHA384
TLSv1/SSLv3:ECDH-ECDSA-AES256-GCM-SHA384TLSv1/SSLv3:ECDH-RSA-AES256-SHA384
TLSv1/SSLv3:ECDH-ECDSA-AES256-SHA384
TLSv1/SSLv3:ECDH-RSA-AES256-SHA
TLSv1/SSLv3:ECDH-ECDSA-AES256-SHA
TLSv1/SSLv3:AES256-GCM-SHA384
TLSv1/SSLv3:AES256-SHA256 TLSv1/SSLv3:AES256-SHA
TLSv1/SSLv3:CAMELLIA256-SHA TLSv1/SSLv3:PSK-AES256-CBC-SHA
TLSv1/SSLv3:ECDHE-RSA-AES128-GCM-SHA256TLSv1/SSLv3:ECDHE-ECDSA-AES128-GCM-SHA256

TLSv1/SSLv3:ECDHE-RSA-AES128-SHA256
TLSv1/SSLv3:ECDHE-ECDSA-AES128-SHA256
TLSv1/SSLv3:ECDHE-RSA-AES128-SHA
TLSv1/SSLv3:ECDHE-ECDSA-AES128-SHA
TLSv1/SSLv3:SRP-DSS-AES-128-CBC-SHA
TLSv1/SSLv3:SRP-RSA-AES-128-CBC-SHA
TLSv1/SSLv3:SRP-AES-128-CBC-SHA TLSv1/SSLv3:DH-DSS-AES128-GCM-SHA256
TLSv1/SSLv3:DHE-DSS-AES128-GCM-SHA256TLSv1/SSLv3:DH-RSA-AES128-GCM-SHA256
TLSv1/SSLv3:DHE-RSA-AES128-GCM-SHA256TLSv1/SSLv3:DHE-RSA-AES128-SHA256
TLSv1/SSLv3:DHE-DSS-AES128-SHA256 TLSv1/SSLv3:DH-RSA-AES128-SHA256
TLSv1/SSLv3:DH-DSS-AES128-SHA256
TLSv1/SSLv3:DHE-RSA-AES128-SHA
TLSv1/SSLv3:DHE-DSS-AES128-SHA TLSv1/SSLv3:DH-RSA-AES128-SHA
TLSv1/SSLv3:DH-DSS-AES128-SHA TLSv1/SSLv3:DHE-RSA-SEED-SHA
TLSv1/SSLv3:DHE-DSS-SEED-SHA TLSv1/SSLv3:DH-RSA-SEED-SHA
TLSv1/SSLv3:DH-DSS-SEED-SHA TLSv1/SSLv3:DHE-RSA-CAMELLIA128-SHA
TLSv1/SSLv3:DHE-DSS-CAMELLIA128-SHA
TLSv1/SSLv3:DH-RSA-CAMELLIA128-SHA
TLSv1/SSLv3:DH-DSS-CAMELLIA128-SHA
TLSv1/SSLv3:ECDH-RSA-AES128-GCM-SHA256
TLSv1/SSLv3:ECDH-ECDSA-AES128-GCM-SHA256TLSv1/SSLv3:ECDH-RSA-AES128-SHA256
TLSv1/SSLv3:ECDH-ECDSA-AES128-SHA256
TLSv1/SSLv3:ECDH-RSA-AES128-SHA
TLSv1/SSLv3:ECDH-ECDSA-AES128-SHA
TLSv1/SSLv3:AES128-GCM-SHA256

TLSv1/SSLv3:AES128-SHA256 TLSv1/SSLv3:AES128-SHA
TLSv1/SSLv3:SEED-SHA
TLSv1/SSLv3:CAMELLIA128-SHA
TLSv1/SSLv3:PSK-AES128-CBC-SHA
TLSv1/SSLv3:ECDHE-RSA-RC4-SHA
TLSv1/SSLv3:ECDHE-ECDSA-RC4-SHA
TLSv1/SSLv3:ECDH-RSA-RC4-SHA
TLSv1/SSLv3:ECDH-ECDSA-RC4-SHA
TLSv1/SSLv3:RC4-SHA
TLSv1/SSLv3:RC4-MD5 TLSv1/SSLv3:PSK-RC4-SHA
TLSv1/SSLv3:ECDHE-RSA-DES-CBC3-SHA
TLSv1/SSLv3:ECDHE-ECDSA-DES-CBC3-SHA
TLSv1/SSLv3:SRP-DSS-3DES-EDE-CBC-SHA
TLSv1/SSLv3:SRP-RSA-3DES-EDE-CBC-SHA
TLSv1/SSLv3:SRP-3DES-EDE-CBC-SHA
TLSv1/SSLv3:EDH-RSA-DES-CBC3-SHA
TLSv1/SSLv3:EDH-DSS-DES-CBC3-SHA TLSv1/SSLv3:DH-RSA-DES-CBC3-SHA
TLSv1/SSLv3:DH-DSS-DES-CBC3-SHA
TLSv1/SSLv3:ECDH-RSA-DES-CBC3-SHA
TLSv1/SSLv3:ECDH-ECDSA-DES-CBC3-SHA
TLSv1/SSLv3:DES-CBC3-SHA
TLSv1/SSLv3:PSK-3DES-EDE-CBC-SHA
“