Name : Omkar Gurav
Roll no : 8048

Assignment No: 4

Title: Configure and Demonstrate web security with Open SSL tool kit.

Objective: To study SSL TOOL Digital Certificate.

Theory:

1. Terminology

Digital Certificate: A Digital Certificate, or Digital ID, is the electronic counterpart to a driver.s license, or passport. It can be presented electronically to prove your identity, or to validate your right to access private information or services online.

Certification Authority (CA): An entity that issues digital (X.509) certificates and vouches for the data contained in such certificates. A CA may be thought of as a trusted third party who "signs" certificates, making them valid. Eg. *Verisign,Thawte*
Terminology _ **CRL:**Certificate Revocation List
PEM: Privacy-Enhanced Mail format
DER: Distinguished encoding rules
X.509 Certificate: The standard format for digital certificates.

2. Setting up HTTPS server using Digital Certificate

To create a key for CA

/usr/bin/openssl genrsa -des3 1024 > ca.key

This will ask for a pass phrase.

To create certificate request

_
/usr/bin/openssl req -new -key ca.key -out ca.csr

This will ask for the pass phrase of the key. Enter the one you gave in the previous step.
Along with that all the information of the CA, like the country code, State, etc. need to be supplied.

To generate a self signed certificate
/usr/bin/openssl req -new -key ca.key -x509 -days 365 -out ca.crt

This will generate a self-signed CA certificate.
Thus now we have the self-signed CA certificate which can be used to
sign other certificates.

To generate ca certificate in pem format

/usr/bin/openssl req -newkey rsa:1024 -keyout $PEM1 -nodes -x509 -days
365 -out $PEM2 ; \
cat $PEM1 > ca.pem ; \
echo "" >> ca.pem ; \
cat $PEM2 >> ca.pem ; \

This certificate is required to sign the user certificate.
The CA maintains a text database of the certificates issued. A pre-defined directory structure is expected for the signing process which is defined in */usr/share/ssl/openssl.cnf*
file. You can change the required directory structure.
So create the following directory structure:

demoCA
|-private
|-newcerts
mkdir demoCA
cd demoCA
mkdir private
mkdir newcerts
Also, create files which are required for the database
vi serial
(put "00" in the file).
touch index.txt (create empty file).

Please go through the various parameter in this file. Some of the parameters are
"optional" or "match". So for example a parameter Organization Unit (OU) is match then the CA can sign a certificate with only the same OU entry for the request certificate. An "optional" parameter has no restriction on the field.

Similarly create a key and csr for the server

(we will be using aryan as our server name. Please replace the same with some
other name or your server name).

/usr/bin/openssl genrsa -des3 1024 > aryan.key
/usr/bin/openssl req -new -key aryan.key -out aryan.csr

This will ask all the information. In the common name field, give the server url
or the IP address. This csr(certificate request) now needs to be signed by the CA. Hence we submit it to our CA for signature. The public key of the server and all the other information is provided with the csr. This is signed by the CA.

Sign the certificate
openssl ca -infiles aryan.csr > aryan.crt
The CA signs the certificate with his private key. Thus the certificate contains the public key and the general information of the server signed by the private key of CA. In no case the public key of the server is made available to any one, even the CA. The aryan.crt file is our server certificate. We need to install this key on the web server. The server key, csr, and crt files are copied into the appropriate dierctories.

Copy the files files in appropriate directory of apache
(You need to be root(administrator) for this).

cp aryan.key /etc/httpd/conf/ssl.key/server.key
cp aryan.crt /etc/httpd/conf/ssl.crt/server.crt
cp aryan.csr /etc/httpd/conf/ssl.csr/server.csr

Restart the apache server.
httpd -k stop
httpd -k start
You will be asked the passphrase for the server key.
(You need to be root(administrator) for this).

The ***make*** file provided also does the same thing. The steps required are
make ca.key --> Gen. CA key
make ca.csr --> Gen. CA csr
make ca.crt --> Gen. self signed CA certificate
make dirstruct --> Create the directory structure and files required.
make aryan.key --> Gen server key.
make aryan.csr --> Gen server csr.
make sign --> Sign the server certificate by CA. .
make install --> Copy the server keys at proper locations
make restart --> Restart the apache web server.
(Needs root perms for last two operations).

3. Testing in Browser-Mozilla

Open your browser (we will use mozilla here)
Accesss the site:
We will be using *10.12.14.10* for the web server address. Replace the same with your server url or .Common Name. used in to create the certificate.
Type in the url
http://10.12.14.10/
You can access this normally. Now try to access the same site with https protocol.
https://10.12.14.10/

And this time you are asked about the site being untrusted and some reasons give out. Try to analyze the results.
Accept the certificate only for the session. Go to some other site and then come back again. you will not be asked any thing.
Now close the browser and then again visit the same URL, you are again asked for the certificate verification. This time accept the certificate permanently. Browse the site. Then close the browser.
Start the browser again. This time browser is not asking for any verification. This is because we have accepted the certificate permanently. So where is it stored? To find out go to Edit -> Preferences -> Privacy and Security -> Certificates -> Manage Certificates. Open the "web sites" tab. You will find one entry about the certificate that we accepted permanently. View the certificate. It is the same certificate that we saw before accepting. Delete the certificate from the "web sites" tab. Close the browser and open again to browse our site. As we have deleted the site, this time we are again asked about the site as in the first case.

Conclusion: We have implemented web security with open SSL toolkit Successfully.

Certificate Screenshot :