Public Key Infrastructure

**Module:** Cryptography

**Module Code:** COMP09106

**Activity No.:** LAB/09106/2425T1-W7

**Week:** 07

**Topic:** Public Key Infrastucture **Duration:** Regular Class Time **Instructor:** Dr. Althaff Mohideen **Submission:** via Aula **Assessment:** Summative (5%)

***Important:*** *These labs are designed to be completed on lab or home desktop/laptop environments.*

**\*\* Please do report any errata and spelling mistakes in the lab sheet.**

**Student Name: Student ID:**

# After completing this lab, you will be able to:

* Successfully understand public-key certificate, certificate authority and the function of public-key infrastructure (PKI)

For this lab We will use the “Server VM” as the web server and “Client VM” web client. A new VM must be created and named “CA VM” will act as our Certificate of Authority (CA). We will do most of the work in the “CA VM”.

This lab will use OpenSSL commands and libraries. If the package is not installed on CA VM, please install OpenSSL package using the below command.

sudo apt-get install openssl

# Task 01: Setting up Certificate Authority

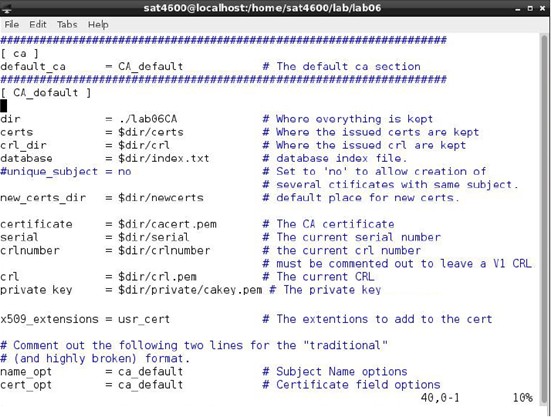
A Certificate Authority is a trusted entity that issues digital certificates. A digital certificate certifies the ownership of a public key. A number of commercial CAs is treated as root CAs. Examples include VeriSign, GlobalSign, etc.

While a digital certificate is typically signed by a CA, the certificate of a root CA is self-signed. Root CA’s certificates are usually preloaded into most operating systems, web browsers, and other applications that rely on PKI. These certificates are unconditionally trusted. You will set up a root CA in this section.

Some OpenSSL commands such as ca, req, and x509 rely on a configuration file. The default configuration file (**openssl.cnf**) is located in the following folder:

/etc/ssl/openssl.cnf

Edit the default **openssl.cnf** file to fit your own setup. You can keep the default settings in most of the sections, however you need to make changes in the **[ ca ]** section. An example is shown in the following screenshot.



Next, you need to create several sub-directories and files as specified in the configuration file in the **[ CA\_default ]** section. For the **index.txt** file, simply create an empty file

(touch index.txt). For the **serial** (no file extension) file, put a single number (*e.g.*, 0001) in the file.

TIP: mkdir is used to create directories and touch is used to create empty files

You need to create inside of lab09CA in your home/kali directory the following structure:

/certs (directory)

/crl (directory)

/index.txt (file)

/newcerts (directory)

/serial (file) [serial only]

/private (directory)

Make sure you insert a number (e.g. 0001) in the **serial** file

## Checkpoint 01: Provide screenshot of the .cnf file and the directory listings of lab09CA.

|  |  |
| --- | --- |
| ***Date:*** | In VM ubuntu : |
| ***Evidence/Snapshot:***  *(Checkpoint)* |  |
| ***Description/Detail:*** |  |
| *Note: Copy/Paste this table to provide multiple evidence or snapshots.* | |

The certificate of a root CA is self-signed. The following command will generate a self- signed certificate for the root CA:

openssl req -new -x509 -keyout private/cakey.pem -out cacert.pem - config /etc/ssl/openssl.cnf

You will be prompted for information and a password. **Please remember this password**. You will have to type in the password (passphrase) each time you use this CA to sign a certificate for the client. You will also be asked to fill in the information related to the owner of the certificate, i.e. the information about the Certificate of Authority such as Country Name, Common Name, etc. These data will be included in the certificate. The output of the command is stored in two files. The file **cakey.pem** contains the CA’s private key, while **cacert.pem** contains the public-key certificate.

Verify if the complete information available in the CA certificate, i.e., the information stored in the certificate

openssl x509 –in cacert.pem -text -noout

***Checkpoint 02: Provide screenshot of the verification. In the description box, provide explanations about the fields that are displayed from the output of the previous command, i.e., the “certificate” fields.***

|  |  |
| --- | --- |
| ***Date:*** |  |
| ***Evidence/Snapshot:***  *(Checkpoint)* |  |
| ***Description/Detail:*** |  |
| *Note: Copy/Paste this table to provide multiple evidence or snapshots.* | |

# Task 02: Issuing Certificates

With OpenSSL tools, the following three tasks are needed to issue a certificate to a client:

* Generate public / private key pair for the client.
* Submit a Certificate Signing Request.
* Issue / sign the certificate of the public-key.

In this section, a certificate will be issued to the **uwscryptography.com** company (the client) using the root CA we just set up in the previous section.

The certificate client (Server VM) should generate its own public/private key pair by itself because the private key is not supposed to be exposed to anyone except its owner. The following OpenSSL command will generate an RSA private key.

(Run this command on Server VM – certificate client)

openssl genrsa –des3 –out server.key 2048

You will be asked to provide a password to protect the key, which will be stored in the file

**server.key**. Please remember this password. You will need it later.

Once the company has their private key, they should generate a Certificate Signing Request (CSR). The following OpenSSL command can be used to generate a CSR:

openssl req –new –key server.key –out server.csr –config

/etc/ssl/openssl.cnf

Please use **uwscryptography.com** (or your own domain name) as the common name of the CSR. Then, the CSR should be sent to the CA to request the CA generate a certificate out of the CSR.

**IMPORTANT:** You can use the scp command to send the CSR to the CA VM. At this point you must also setup the appropriate network settings to communicate between all your VMs.

## Checkpoint 03: Provide screenshot of successful creation CSR and copying the same to CA VM.

|  |  |
| --- | --- |
| ***Date:*** |  |
| ***Evidence/Snapshot:*** |  |

|  |  |
| --- | --- |
| *(Checkpoint)* |  |
| ***Description/Detail:*** | Above is the private key. |
| *Note: Copy/Paste this table to provide multiple evidence or snapshots.* | |

In reality, CSR files are usually sent to a trusted CA for their signature. Upon receiving a CSR, the CA will generate/issue a certificate after ensuring that the identity information in the CSR matches with the client’s true identity. In this lab, we will use our own CA to issue an SSL/TLS certificate to the client **uwscryptography.com**. The following OpenSSL command will generate a certificate out of the CSR:

Run the following **long** command in CA VM (in one line )

openssl ca -in server.csr –out

/home/kali/lab09CA/newcerts/server.crt -cert

/home/kali/lab09CA/cacert.pem –keyfile

/home/kali/lab09CA/private/cakey.pem -config /etc/ssl/openssl.cnf

The certificate, if generated successfully, is stored in the file **server.crt**.

If OpenSSL refuses to generate a certificate, it is very likely that the names in the CSR do not match with those of CA. The matching rules are specified in the configuration file under the **[policy\_match]** section. In this case, you can change the names in the CSR to comply with the policy, or you can change the policy. The configuration file includes another policy **(policy\_anything)**, which is less restrictive. You can choose that policy by changing the following line:

“policy = policy\_match” ===> “policy = policy\_anything”

When the certificate is generated, you should be able to display the information included in the certificate by using the OpenSSL x509 command. For example, the following command will display the certificate in text format:

openssl x509 –in lab09CA/newcerts/server.crt –noout –text

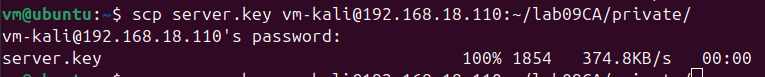
Please use man page (man x509) to find other useful options and try to retrieve the related information included in the server certificate.

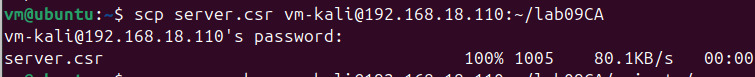
***Checkpoint 04: Provide screenshot of successful creation certificate.***

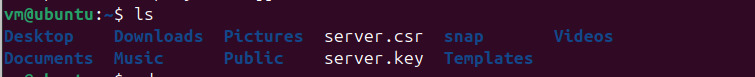
|  |  |
| --- | --- |
| ***Date:*** |  |
| ***Evidence/Snapshot:***  *(Checkpoint)* |  |
|  |  |

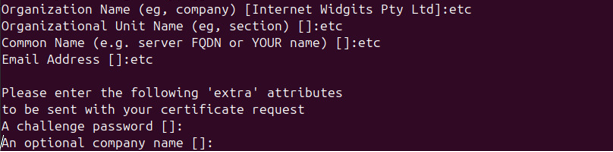
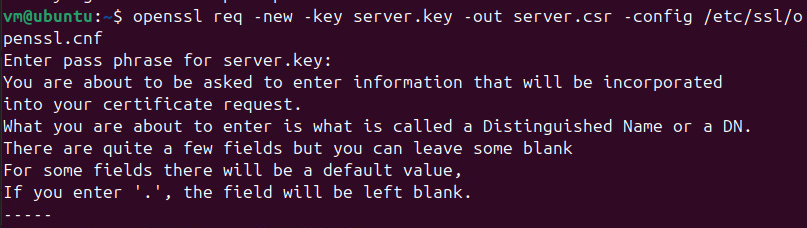
|  |  |
| --- | --- |
|  | Below ss is here |
| ***Description/Detail:*** |  |
| *Note: Copy/Paste this table to provide multiple evidence or snapshots.* | |

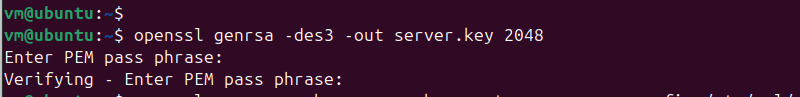
***Here is the some commands that I have run on VM ubuntu :***

******

******

******

******

******

# Task 03: Setting up the Certificate

In this section, we will explore how the PKI and certificates can be used by a web server to secure web browsing. To make it simpler, let´s go to the Client VM and add the following entry to the **/etc/hosts** file on the Linux computer:

**<your server IP> uwscryptography.com**

This will allow an application to resolve the IP address of the **uwscryptography.com**

domain.

Next, we need to launch a simple web server in the Server VM. We will use simple web server shipped with OpenSSL package or you can use already existing Apache web server.

In the Server VM, we are going to install and start SSH to allow remote access to the VM from another VM and remote copy of files. You must have accomplished this task in previous labs.

sudo apt-get install ssh sudo service sshd start

Please perform the following activities in the **CA VM**:

* + Combine the secret key and certificate into one file:

cp server.key server.pem

mv server.pem ~/lab09CA/newcerts

**change directory to lab09CA/newcerts**

cat server.crt >> server.pem

* + Copy server credentials to the server machine.

scp server.pem kali@server\_ip:~

In the Server VM, now, let´s launch a simple web server (HTTPS server) with the **server.pem**

certificate by using the OpenSSL s\_server command:

cd ~

openssl s\_server –cert server.pem -www

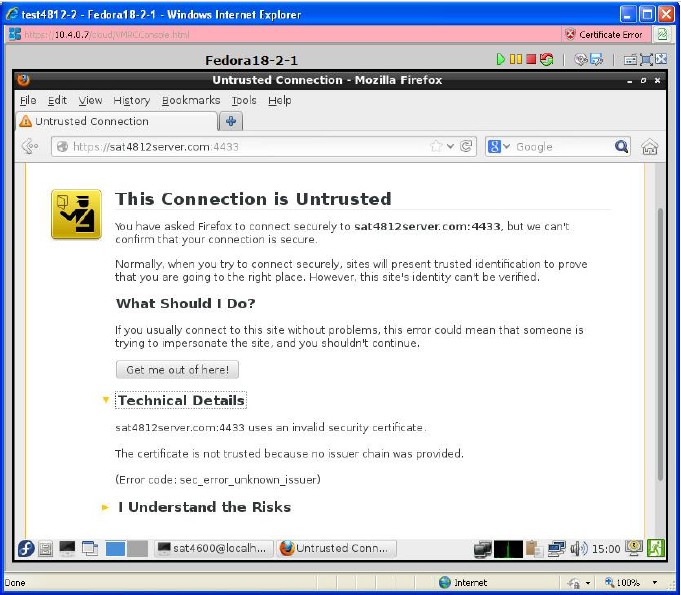
By default, the server will listen on port 4433. The default port number can be altered by using the **-accept** option. Now, please access the HTTPS server using a web browser in the **Client VM.** Type in the URL:

https:// **uwscryptography.com**:4433/.

You may need to modify the file /etc/hosts with sudo adding the following line:

# “127.0.0.1 uwscryptography.com” on Server VM

Most likely, an error similar to the following screenshot will appear on the web browser. **At this point, please do not add an exception for this web site.** The secure connection failed because “The certificate is not trusted because no issuer chain was provided.”



Had this certificate been assigned by VeriSign, we would not have this failure, because VeriSign’s certificate is very likely preloaded into Firefox’s certificate repository already. Unfortunately, the certificate of **uwscryptography.com** is signed by our own CA, which is not recognized by Firefox. There are typically two ways to make Firefox accept our CA’s self- signed certificate.

* + We can request Mozilla to include our CA’s certificate in Firefox. Therefore, the certificate is known by Firefox and “trusted” by the user. This is how it works in reality. Unfortunately, the market of our own CA is not large enough for Mozilla to include its certificate in the official distribution of Mozilla.
  + Instead, we can manually load our CA’s certificate into Firefox because we “trust” our own CA. To do this, please click on the following menu sequence on Firefox browser:
    - First let us copy the certificate from the CA to the Client (remember, it is the PUBLIC certificate).
    - In the **CA VM** from within the /lab09CA directory, run

scp cacert.pem klai@client\_ip:~

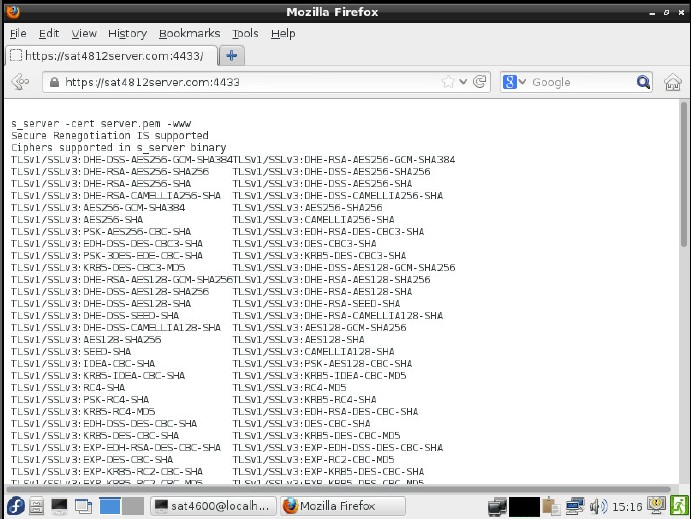
* + - Now, we do go to the Mozilla browser previous opened in the client and then

Edit => Preferences => Advanced => Certificated => **View Certificates** or search **View Certificates**

From there, you will see a list of certificates that are already included in Firefox. Do you trust them? The Firefox does and you are supposed to.

Please click on **import** to load the **cacert.pem** into Firefox’s certificate repository in the **Authorities** category, and select the option: “Trust this CA to identify web sites”. This is to tell Firefox that you trust our own CA.

Now, try to access https:// **uwscryptography.com**:4433/. This access should be successful without any error. A web page similar to the one in the following screenshot should show up. Otherwise, try to fix what you have done wrong and make the access successful.



## Checkpoint 05: Provide screenshot of successful listing of cipher suites in the client browser.

|  |  |
| --- | --- |
| ***Date:*** | http://ubuntu-ip :4433 |
| ***Evidence/Snapshot:***  *(Checkpoint)* |  |
| ***Description/Detail:*** |  |
| *Note: Copy/Paste this table to provide multiple evidence or snapshots.* | |