Information Security

Assignment # 03

RSA and PKI Cryptography Lab in SEED's Lab

Mam Hina Binte Haq

Course Instructor
CS- 3002
SE- S
Due Date: April 02, 2023

Group Members:

Zeeshan Ali	20i-2465
Ans Zeeshan	20i-0543

Assignment # 03

Part 02

Task-1: Becoming a Certificate Authority

Step 1: The Configuration File openssl.conf

We use OpenSSL to create certificates and the following provides the configuration setup for creating and issuing certificates:

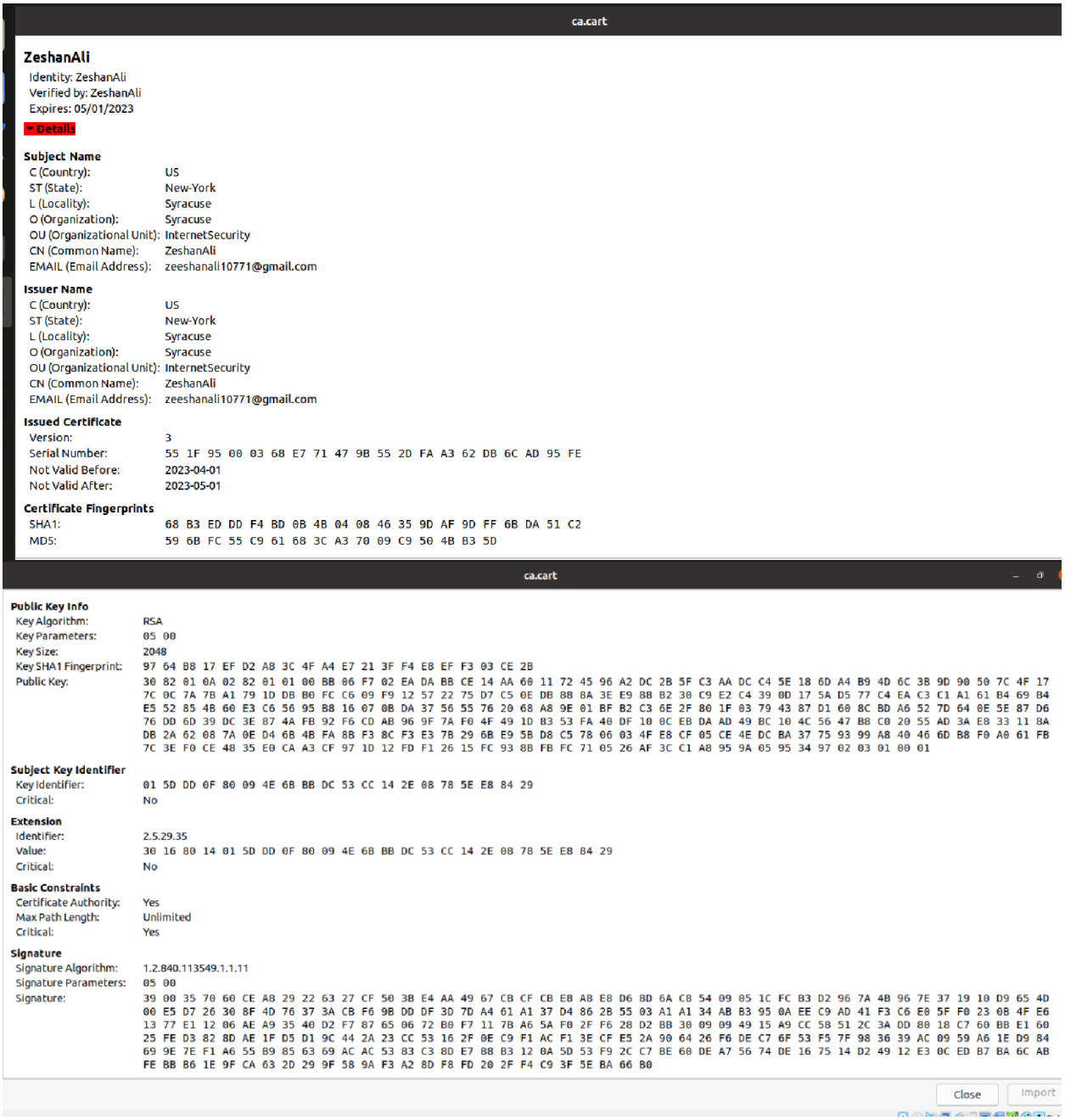
```
[04/01/23]seed@VM:~/labpart2$ cd ..
[04/01/23]seed@VM:~$ cd Desktop
[04/01/23]seed@VM:~/Desktop$ mkdir labpart2
[04/01/23]seed@VM:~/Desktop$ cd labpart2
[04/01/23]seed@VM:~/.../labpart2$ cp /usr/lib/ssl/openssl.cnf openssl.cnf
[04/01/23]seed@VM:~/.../labpart2$ ls
openssl.cnf
[04/01/23]seed@VM:~/.../labpart2$ mkdir part2
[04/01/23]seed@VM:~/.../labpart2$ cd part2/
[04/01/23]seed@VM:~/.../part2$ mkdir certs crl newcerts
[04/01/23]seed@VM:~/.../part2$ touch index.txt serial
[04/01/23]seed@VM:~/.../part2$ echo 1000 > serial
[04/01/23]seed@VM:~/.../part2$ ls
certs  crl  index.txt  newcerts  serial
[04/01/23]seed@VM:~/.../part2$ cd ..
[04/01/23]seed@VM:~/.../labpart2$ ls
openssl.cnf  part2
```

Step 2: Certificate Authority (CA)

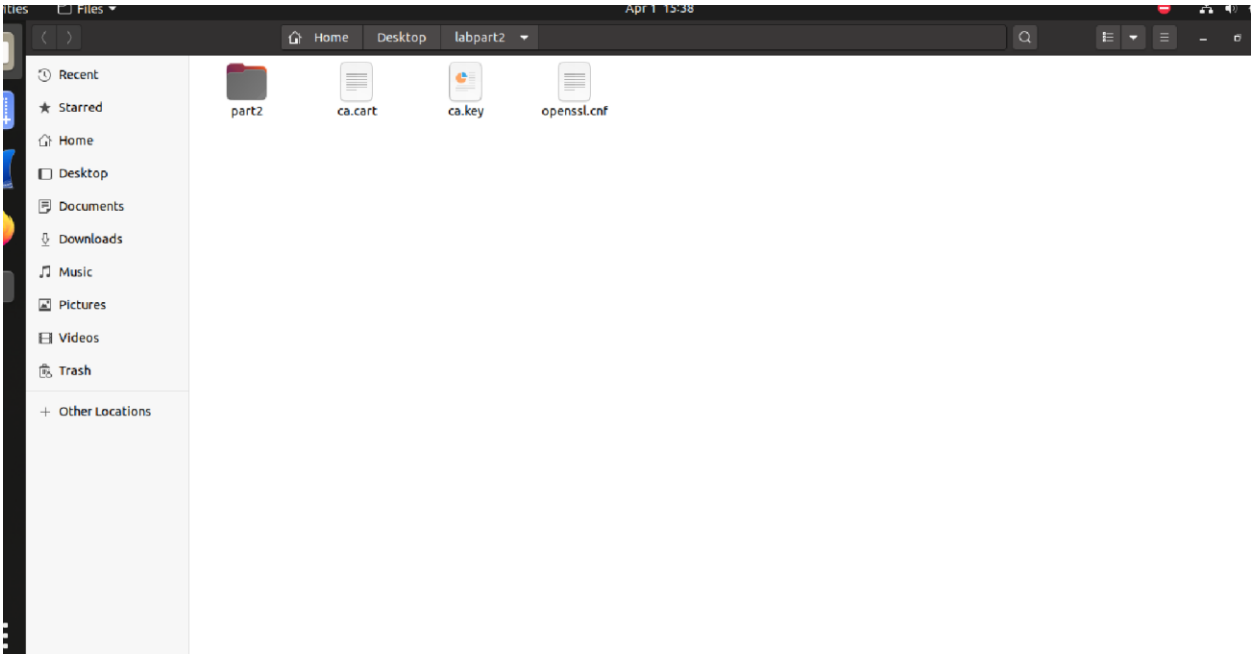
Now, we generate a self-signed certificate for our CA that will serve as the root certificate as follows:

```
[04/01/23]seed@VM:~/.../labpart2$ openssl req -new -x509 -keyout ca.key -out ca.cart -con
fig openssl.cnf
Generating a RSA private key
..............+++++
.............................................+++++
writing new private key to 'ca.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
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]:US
State or Province Name (full name) [Some-State]:New-York
Locality Name (eg, city) []:Syracuse
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Syracuse
Organizational Unit Name (eg, section) []:InternetSecurity
Common Name (e.g. server FQDN or YOUR name) []:ZeshanAli
Email Address []:zeeshanali10771@gmail.com
[04/01/23]seed@VM:~/.../labpart2$
```

The above provides a screenshot of the entered information. The output of the command is stored in two files: ca.key and ca.crt (as provided in the command). The file ca.key contains the CA's private key, while ca.crt contains the public-key certificate. The following provides the content of ca.crt:



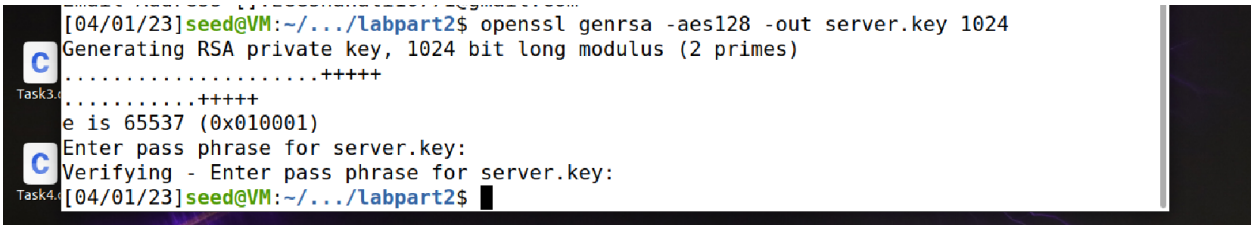
We see that the subject and issuer are the same, indicating a self-signed certificate. Also, the Certificate Authority is set to yes in Basic Constraints, indicating that this certificate can be used to issue and sign another certificate, hence becoming a certificate of a Certificate Authority (CA). This certificate will be used as the root certificate in this lab.



Task-2: Creating a Certificate for meghajak.com

Step 1: Generate public/private key pair

We first create an RSA public-private key pair using the following command:



We can use the following command to view the generated key: openssl rsa -in server.key –text

Step 2: Generate a Certificate Signing Request (CSR)

Next, we create a Certificate Signing Request (CSR) that includes the company's public key. The CSR has the following details with company's common name being meghajak.com (company's domain):

```
[04/01/23]seed@VM:~/.../labpart2$ openssl req -new -key server.key -out server.csr -confi
g openssl.cnf
Enter pass phrase for server.key:
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]:US
State or Province Name (full name) [Some-State]:New-York
Locality Name (eg, city) []:Syracuse
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Syracuse
Organizational Unit Name (eg, section) []:Labs
Common Name (e.g. server FQDN or YOUR name) []:zeeshanali
Email Address []:zeeshanali10771@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:dees
An optional company name []:zeeshanali
[04/01/23]seed@VM:~/.../labpart2$
```

The above CSR is then sent to the CA to generate a certificate for the key and company name.

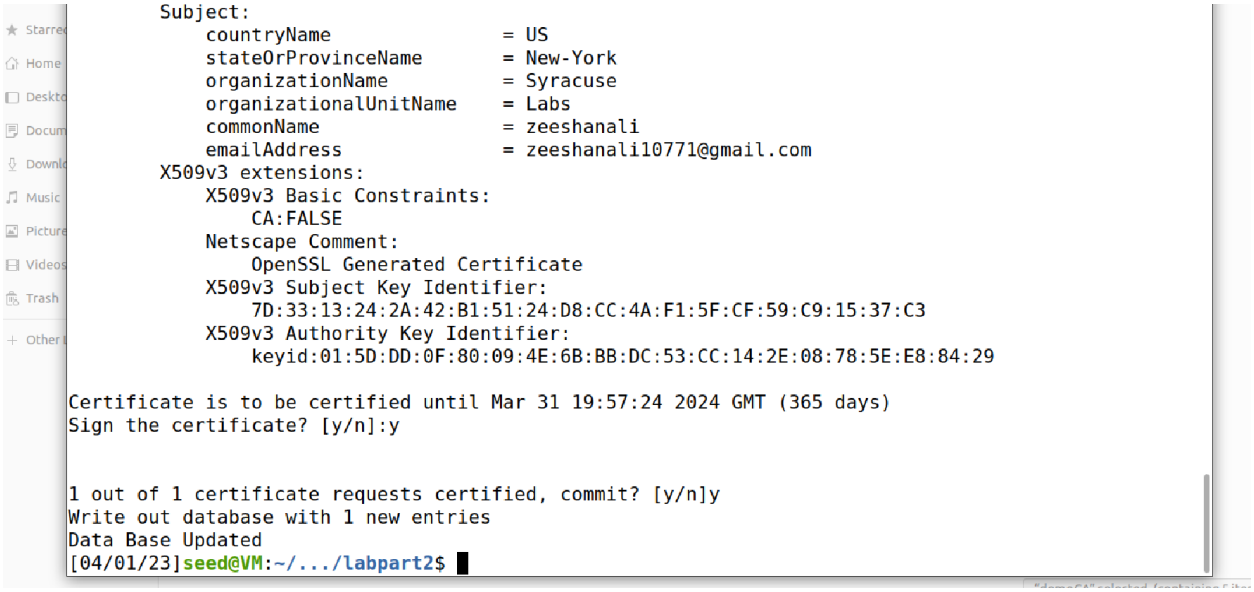
Step 3: Generating Certificates

The CSR file needs to have the CA's signature to form a certificate. We change the policy to policy_anything from policy_match to avoid any errors as seen in the following:

```
# A few difference way of specifying how similar the request should look
# For type CA, the listed attributes must be the same, and the optional
# and supplied fields are just that :-)
policy          = policy_anything
```

The following command turns the certificate signing request (server.csr) into an X509 certificate (server.crt), using the CA's ca.crt and ca.key. We see the details of the CSR being signed:

```
[04/01/23]seed@VM:~/.../labpart2$ openssl ca -in server.csr -out server.crt -cert ca.cart -keyfile
ca.key -config openssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4096 (0x1000)
        Validity
            Not Before: Apr  1 19:57:24 2023 GMT
            Not After : Mar 31 19:57:24 2024 GMT
        Subject:
            countryName               = US
            stateOrProvinceName       = New-York
            organizationName          = Syracuse
            organizationalUnitName    = Labs
            commonName                = zeeshanali
            emailAddress              = zeeshanali10771@gmail.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
```



This step achieves a certificate for meghajak.com from the root certificate.

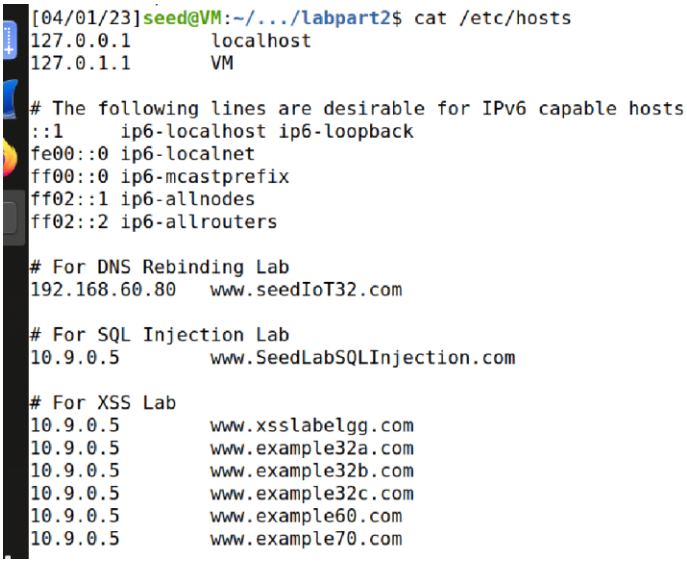
Task-3: Deploying Certificate in an HTTPS Web Server

Step 1: Configuring DNS

We add the following entry to the /etc/hosts file in order for the machine to recognize zeeshanali.com on the web browser without changing the DNS:

127.0.0.1 zeeshanali.com

This entry basically maps the hostname zeeshanali.com to the localhost:



```
[04/02/23]seed@VM:~/.../labpart2$ cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       VM

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

# For DNS Rebinding Lab
192.168.60.80   www.seedIoT32.com

# For SQL Injection Lab
10.9.0.5        www.SeedLabSQLInjection.com

# For XSS Lab
10.9.0.5        www.xsslabelgg.com
10.9.0.5        www.example32a.com
10.9.0.5        www.example32b.com
10.9.0.5        www.example32c.com
10.9.0.5        www.example60.com
10.9.0.5        www.example70.com

# For CSRF Lab
10.9.0.5        www.csrflabelgg.com
10.9.0.5        www.csrflab-defense.com
10.9.0.105      www.csrflab-attacker.com

# For Shellshock Lab
10.9.0.80       www.seedlab-shellshock.com
127.0.0.1       zeeshanali.com
10.0.2.7        github.com
[04/02/23]seed@VM:~/.../labpart2$
```

Step 2: Configuring the web server

We combine the key and certificate into one file and then launch a web server using the openssl's s_server command. We enter the passphrase for the private key and see that the web server is now in the accept state – indicating it is running:

```
[04/01/23]seed@VM:~/.../labpart2$ cp server.key server.pem
[04/01/23]seed@VM:~/.../labpart2$ cat server.crt >> server.pem
[04/01/23]seed@VM:~/.../labpart2$ openssl s_server -cert server.pem -www
Enter pass phrase for server.pem:
Using default temp DH parameters
ACCEPT
```

By default, the server listens on port 4433 and we can access the server using the URL:

https://zeeshanali.com:4433/

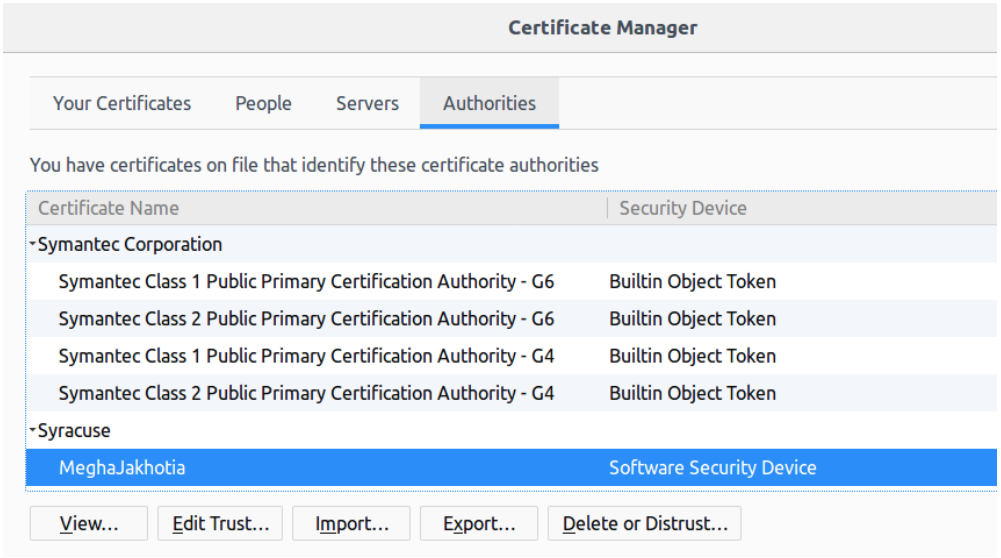
On loading the above URL, we see that the web browser displays an error message with the connection being insecure. On loading more details by clicking on Advanced, we see that it states that the issuer's certificate is unknown i.e. the root certificate is unknown to the browser. This is because, even though the root certificate is created by us, it is not known to the browser and hence it cannot validate the server's certificate i.e. zeeshanali.com's certificate.

Step 3: Getting the browser to accept our CA certificate.

In order for our root certificate to be accepted by the Firefox browser, we manually add our certificate to the browser by following the menu sequence and importing ca.crt:

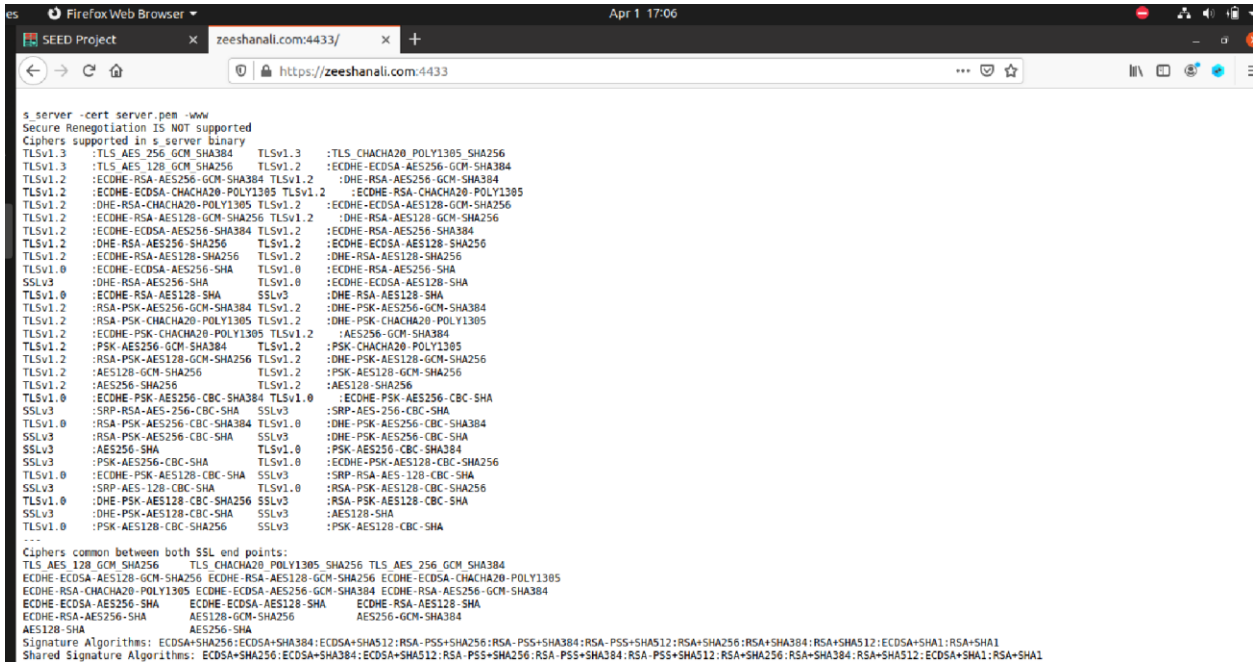
Edit -> Preference -> Privacy & Security -> View Certificates

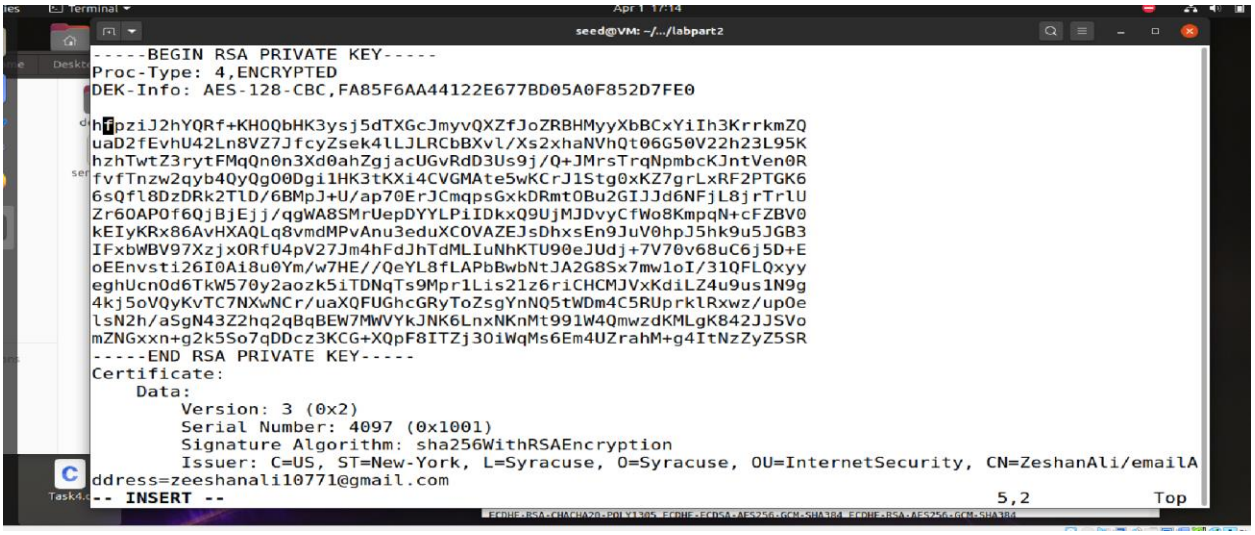
The following shows that the root CA is now in the Firefox's trusted certificates:



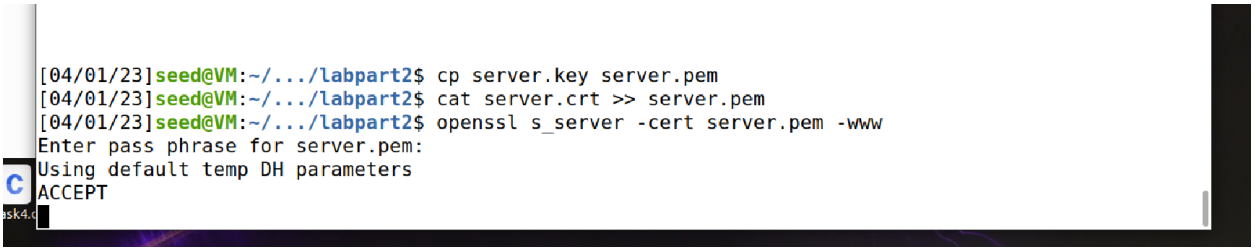
Step 4. Testing our HTTPS website

Now on loading the URL, we see the web page successfully:

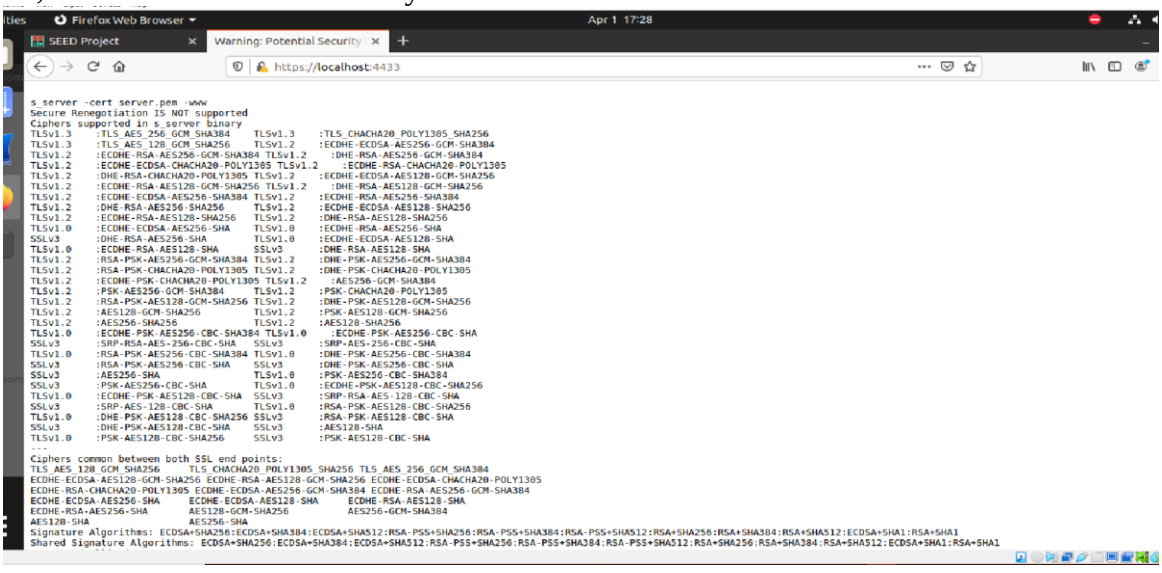




Now,



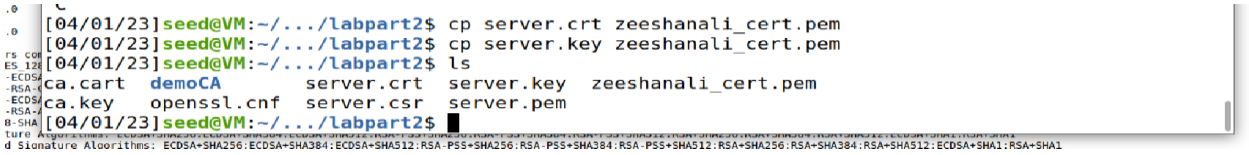
Now, on restarting the server and reloading the URL, we see that a single change in server.pem file caused the browser to throw an error of secure connection failed due to an invalid signature, This proves that the certificate's integrity is a must and any change in the certificate will break the connection.
Now, we load localhost:4433 instead of zeeshanali.com:4433. We see that it gives us an error with the connection being insecure. However, since zeeshanali.com is linked with 127.0.0.1, essentially the localhost, the website should not really be insecure.



We see that in the additional details, the certificate is invalid because it is not meant for name localhost. On clicking on Add Exception in order to see more details, we see that the certificate is for zeeshanali.com and the URL being loaded is localhost. This basically causes the error to pop up. This proves that the browser also checks for the correctness of the common name and the URL requested. The certificate is valid but for zeeshanali.com and not localhost, and even though both of them point to the same server, the names are different and hence browser does not allow this. If we add the exception, the result will be same as that of zeeshanali.com.

Task-4: Deploying Certificate in an Apache-Based HTTPS Website

We create the pem file for the server's certificate and key to be used in the Apache website configuration:



We add a VirtualHost entry to the default-ssl.conf file as follows:

```
<VirtualHost *:443>
    ServerName zeeshanali.com
    DocumentRoot /var/www/zeeshanali
    DirectoryIndex index.html

    SSLEngine On
    SSLCertificateFile  /home/seed/Desktop/labpart2/zeeshanali_cert.pem
    SSLCertificateKeyFile  /home/seed/Desktop/labpart2/zeeshanali_key.pem
</VirtualHost>
```

Plain Text Tab Width

The ServerName entry specifies the name of the website, while the DocumentRoot entry specifies where the files for the website are stored. The following show the website's files:

```
Terminal
[04/07/20]seed@VM:.../sites-available$ sudo apachectl configtest
AH00558: apache2: Could not reliably determine the server's fully qualified domain
name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this mes
sage
Syntax OK
[04/07/20]seed@VM:.../sites-available$ sudo a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Module socache_shmcb already enabled
Module ssl already enabled
[04/07/20]seed@VM:.../sites-available$ sudo a2ensite default-ssl
Site default-ssl already enabled
[04/07/20]seed@VM:.../sites-available$ sudo service apache2 restart
```

Now, we open the browser and load the URL and see that we can successfully browse the HTTPS site:

There is an example simple HTML page in the Internal Network.

Task-5: Launching a Man-In-The-Middle Attack

Step 1: Setting up the malicious website

Now, we try to impersonate github.com and change the VirtualHost entry in the Apache's SSL configuration to have the ServerName as github.com, and everything else remains the same:

```
<VirtualHost *:443>
    ServerName zeeshanali.com
    DocumentRoot /var/www/zeeshanali
    DirectoryIndex index.html

    SSLEngine On
    SSLCertificateFile  /home/seed/Desktop/labpart2/zeeshanali_cert.pem
    SSLCertificateKeyFile  /home/seed/Desktop/labpart2/zeeshanali_key.pem
</VirtualHost>
```

Plain Text Tab Width

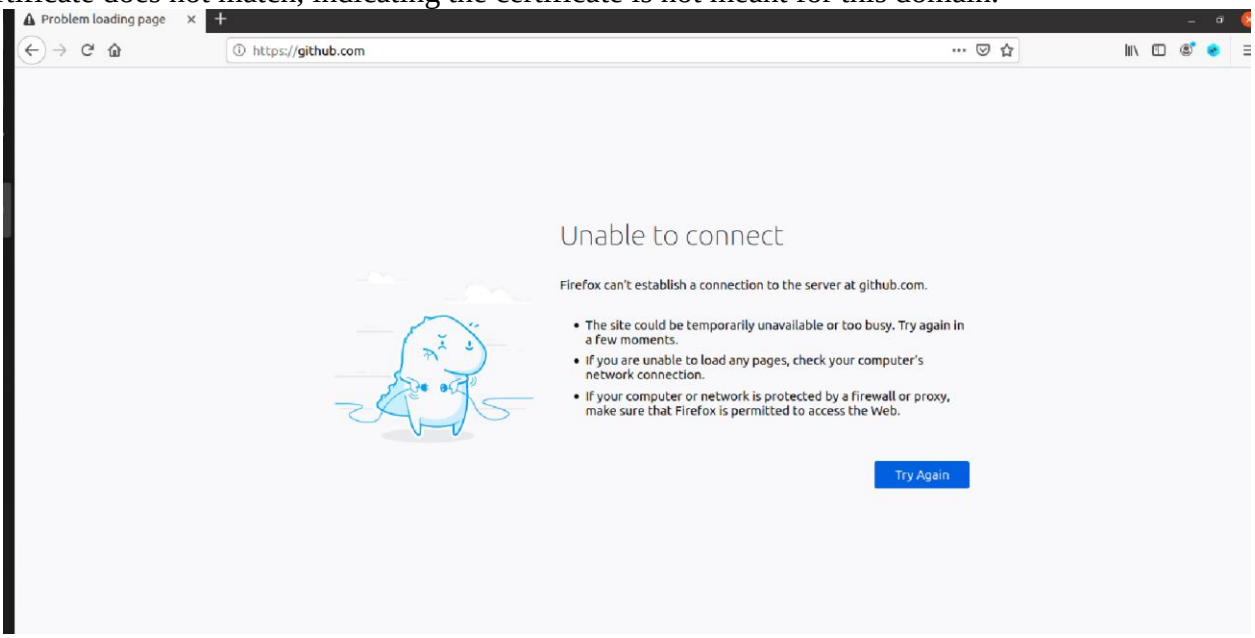
This will launch a fake github.com website. We restart the apache web server to save the changes.

Step 2: Becoming the man in the middle

Now, to perform MITM, we change the DNS of the client machine to have an entry that will redirect the requests going to the original github.com to the fake web server set by us. We change the /etc/hosts file:

Step 3: Browse the target website

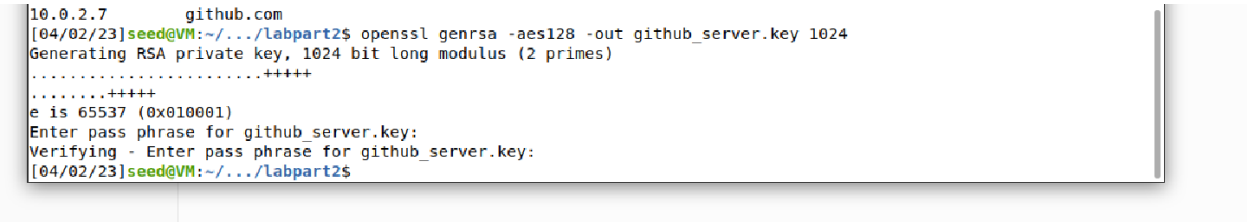
Now we browse the target website – github.com and see that we get an error saying that the connection is not secure. On looking at more details, we see that the common name of the requested URL and the certificate does not match, indicating the certificate is not meant for this domain.



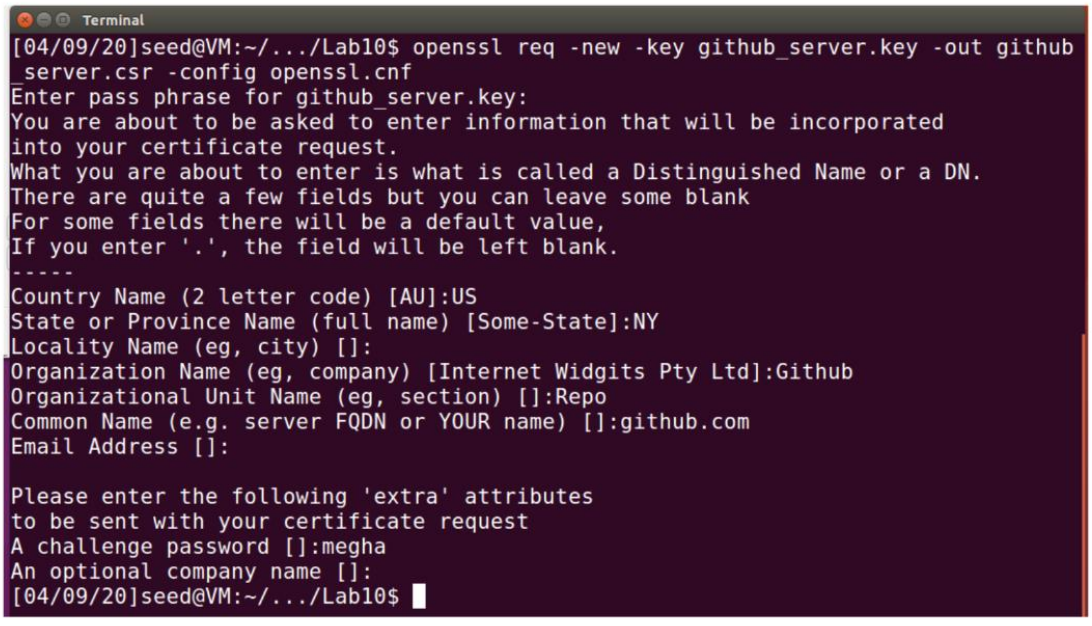
This proves that the MITM attack is defeated in the use of public key infrastructure. If the common name matched, then the website would have loaded. However, the common name will match only if the server had a valid certificate from the CA for its domain, which it didn't, in this case.

Task-6: Launching a Man-In-The-Middle Attack with a Compromised CA

Considering the same setting as in Task 5 and the root CA's private key being compromised. We first create a private-public key for github.com, the target website. We could have used the same keys generated before but just for simplicity, we create new keys.



Now, we create a CSR for github.com using OpenSSL in the same way as before, to be signed by a CA:



Now, since we have the root CA's private key and the certificate is publicly available, we can create a certificate ourselves from the CSR and do not require the CA to do it for us. We use the same command as before to create a certificate.

Now, since this certificate is signed by the root CA that is trusted by the Firefox browser, while authenticating the created certificate, the browser will confirm the validity of this certificate since a trusted root certificate vouched for it. The following shows the creation of a certificate from the CSR using the root CA's credentials:

```
[04/01/23]seed@VM:~/.../labpart2$ openssl ca -in server.csr -out server.crt -cert ca.cart -keyfile
ca.key -config openssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4096 (0x1000)
        Validity
            Not Before: Apr  1 19:57:24 2023 GMT
            Not After : Mar 31 19:57:24 2024 GMT
        Subject:
            countryName               = US
            stateOrProvinceName       = New-York
            organizationName          = Syracuse
            organizationalUnitName    = Labs
            commonName                = zeeshanali
            emailAddress              = zeeshanali10771@gmail.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
```

```
        Subject:
            countryName               = US
            stateOrProvinceName       = New-York
            organizationName          = Syracuse
            organizationalUnitName    = Labs
            commonName                = zeeshanali
            emailAddress              = zeeshanali10771@gmail.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                7D:33:13:24:2A:42:B1:51:24:D8:CC:4A:F1:5F:CF:59:C9:15:37:C3
            X509v3 Authority Key Identifier:
                keyid:01:5D:DD:0F:80:09:4E:6B:BB:DC:53:CC:14:2E:08:78:5E:E8:84:29

Certificate is to be certified until Mar 31 19:57:24 2024 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
[04/01/23]seed@VM:~/.../labpart2$
```

We save the created key and certificate as pem files with names used by the Apache server:

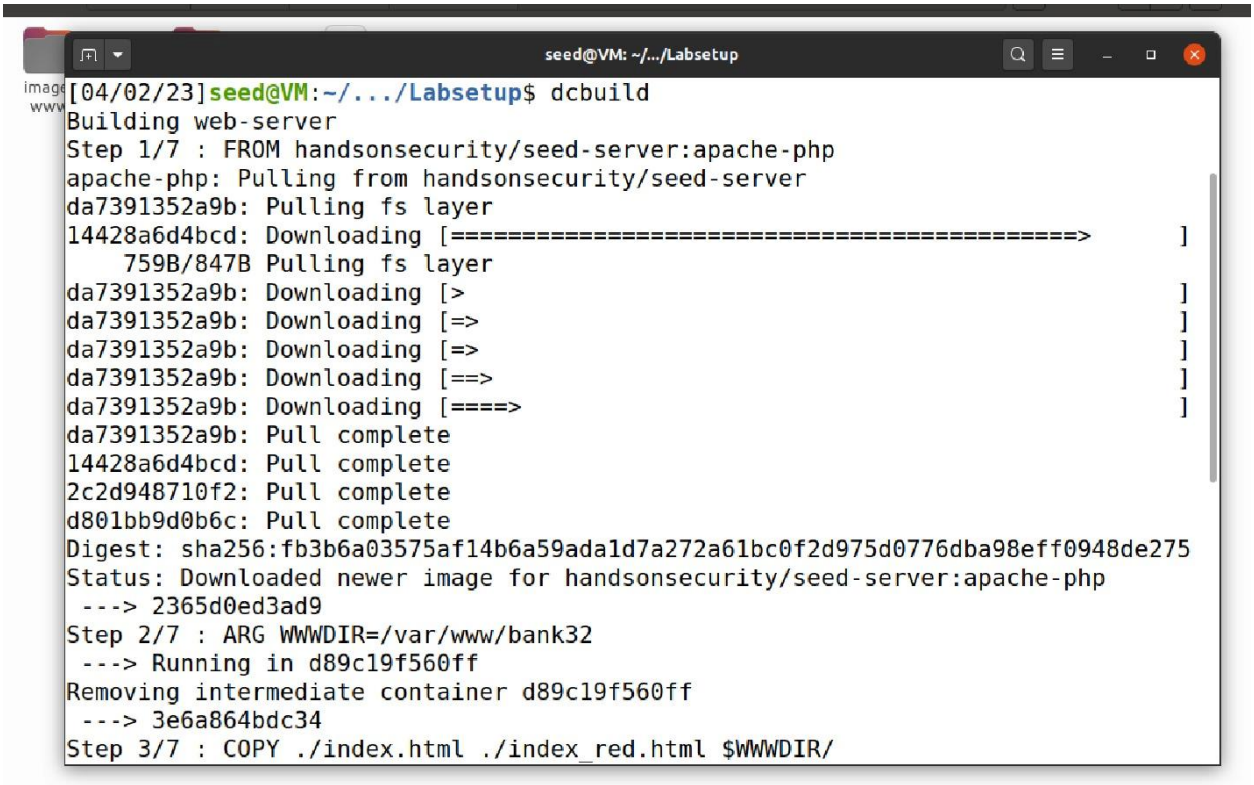
In order to incorporate the changes, we restart the Apache service:

Now, on loading the URL of the target website, we see that the browser does not throw any more errors and display the expected website.

There is an example simple HTML page in the Internal Network.

This is because the certificate is valid due to being signed by the root CA and its common name is github.com. If the root CA's key is compromised, then anyone can create a certificate for themselves and impersonate any other website.

Pre Lab Setup with Docker File




```
Removing intermediate container 653b0ebc6a44
 ---> 607776223cd5
Step 7/7 : CMD  tail -f /dev/null
 ---> Running in 6fbe66f26070
Removing intermediate container 6fbe66f26070
 ---> ac8b5ae0493c

Successfully built ac8b5ae0493c
Successfully tagged seed-image-www-pki:latest
[04/02/23]seed@VM:~/.../Labsetup$ dockerps
dockerps: command not found
[04/02/23]seed@VM:~/.../Labsetup$ dockps
[04/02/23]seed@VM:~/.../Labsetup$ dcup
Creating network "net-10.9.0.0" with the default driver
Creating www-10.9.0.80 ... done
Attaching to www-10.9.0.80
```

Work Division:

Name	Roll No	Work Division
Zeeshan Ali	20i-2465	RSA → pre-Task & Task 1,2, 3,4,5,6
Ans Zeeshan	20i-0543	PKI → pre-Setup & Task 1,2, 3,4,5,6