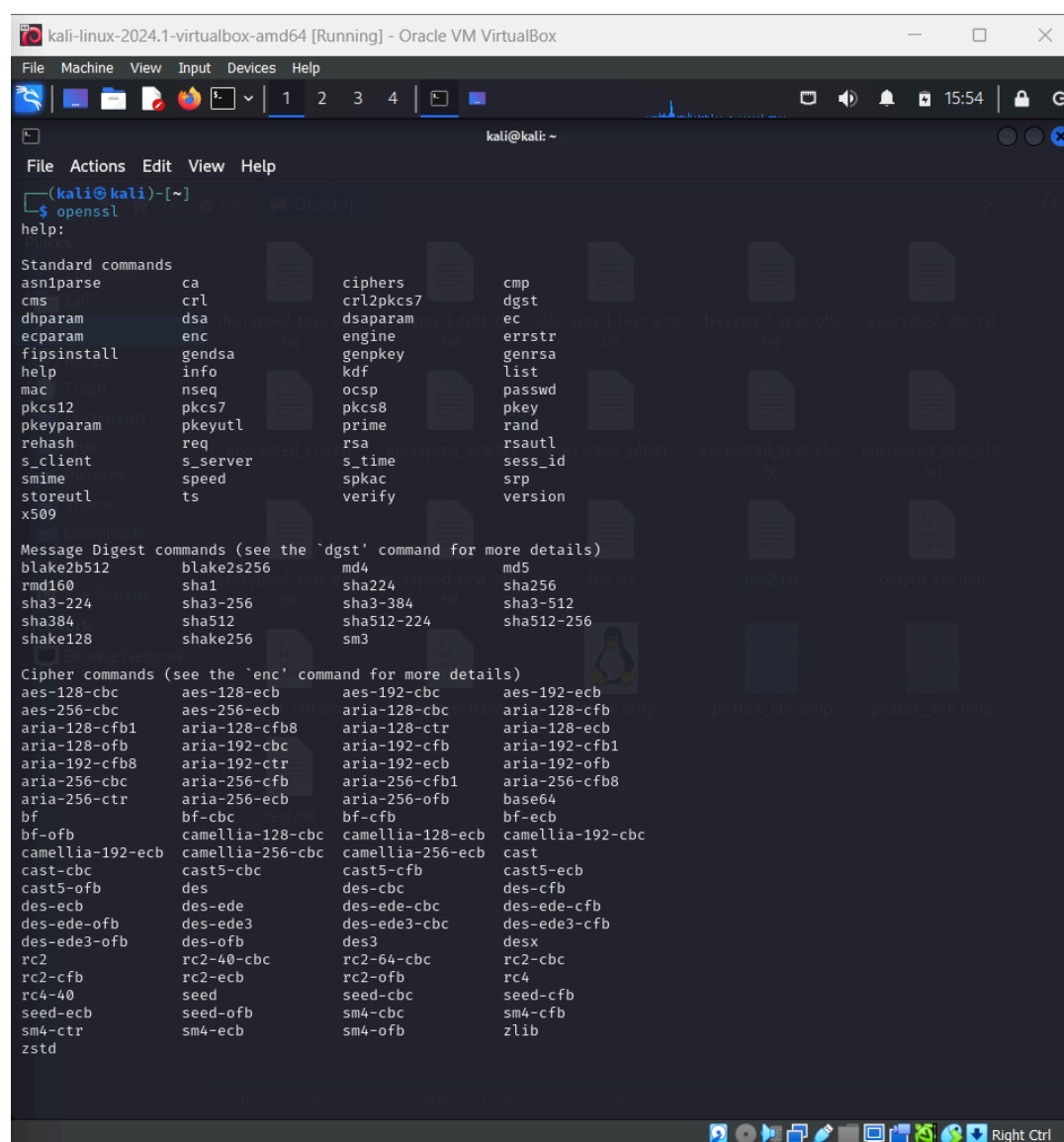


Lab Task 5

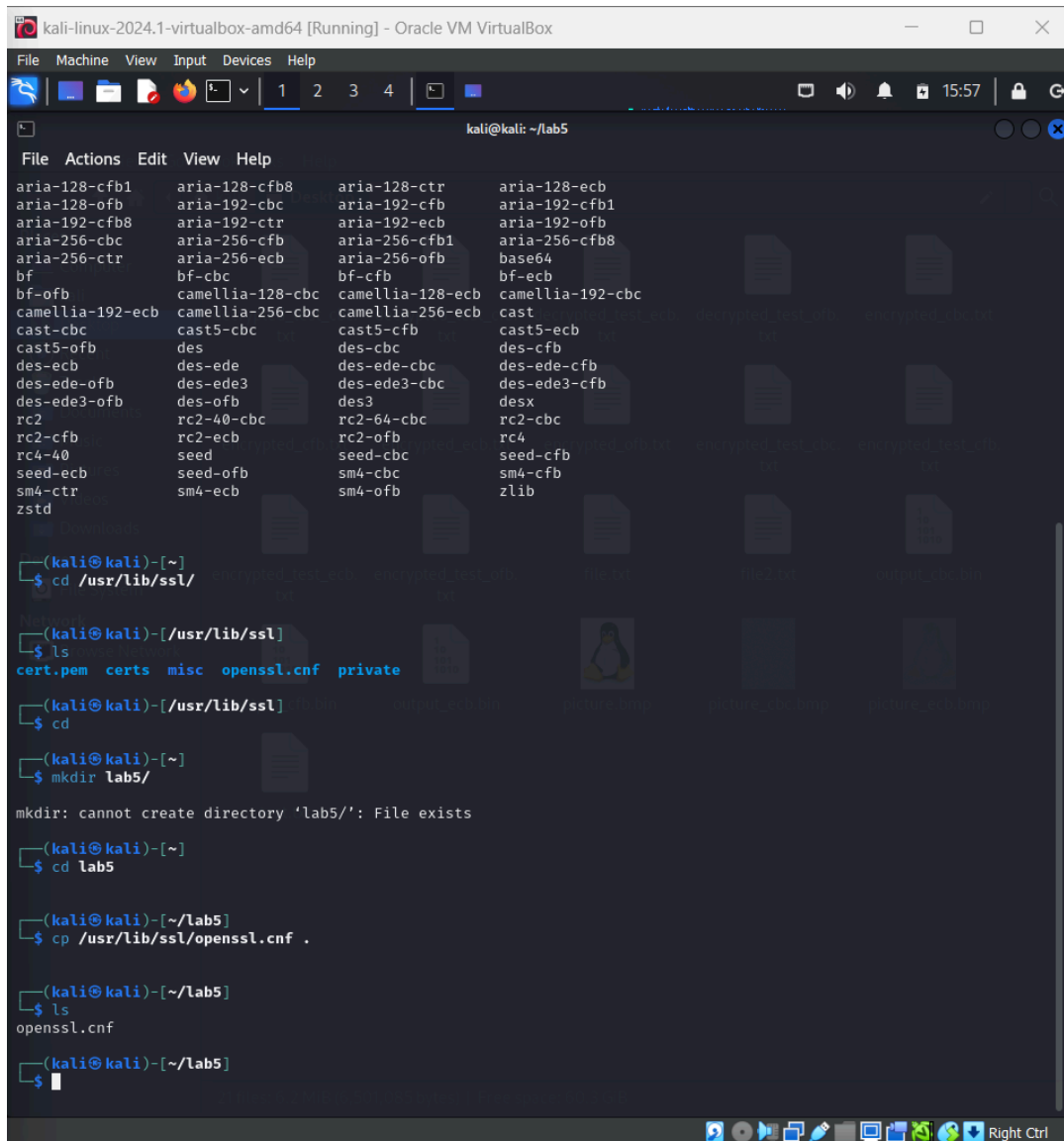
Task-1: Becoming a certificate authority

In this lab, you will need to create digital certificates, but you will not be going to pay to any commercial CA. You will become a root CA, and then use this CA to issue certificate for others (e.g. servers). In this task, you will make yourself a root CA, and generate a certificate for this CA. Unlike other certificates, which are usually signed by another CA, the root CA's certificates are self-signed. Root CA's certificates are usually pre-loaded into most operating systems, web browsers, and other software that rely on certificate-based security. Root CA's certificates are unconditionally trusted.



The screenshot shows a Kali Linux terminal window titled "kali-linux-2024.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox". The terminal displays the output of the `openssl help:` command. The output is organized into several sections: Standard commands, Message Digest commands, and Cipher commands. The Standard commands section lists various tools like `ca`, `crl`, `dsa`, `enc`, `genssa`, `info`, `nseq`, `pkcs7`, `pkcs8`, `prime`, `rsa`, `s_time`, `spkac`, `verify`, `cmp`, `dgst`, `ec`, `errstr`, `genrsa`, `list`, `passwd`, `pkey`, `rand`, `rsautl`, `sess_id`, `srp`, and `version`. The Message Digest commands section lists `blake2b512`, `blake2s256`, `md4`, `md5`, `sha1`, `sha224`, `sha256`, `sha3-224`, `sha3-256`, `sha3-384`, `sha3-512`, `sha512-224`, `sha512-256`, and `sm3`. The Cipher commands section lists various encryption and decryption algorithms such as `aes-128-cbc`, `aes-128-ecb`, `aria-128-cbc`, `aria-128-ctr`, `aria-128-cfb1`, `aria-128-cfb8`, `aria-192-cbc`, `aria-192-ctr`, `aria-192-cfb1`, `aria-192-cfb8`, `aria-256-cbc`, `aria-256-ctr`, `aria-256-cfb1`, `aria-256-cfb8`, `base64`, `bf`, `bf-cbc`, `bf-cfb`, `bf-ecb`, `camellia-128-cbc`, `camellia-128-ecb`, `camellia-192-cbc`, `camellia-192-ecb`, `camellia-256-cbc`, `camellia-256-ecb`, `cast`, `cast5-cbc`, `cast5-ecb`, `cast5-ofb`, `des`, `des-cbc`, `des-cfb`, `des-ede`, `des-ede-cbc`, `des-ede-cfb`, `des-ede3`, `des-ede3-cbc`, `des-ede3-cfb`, `des-ofb`, `des3`, `desx`, `rc2`, `rc2-cbc`, `rc2-cfb`, `rc2-ecb`, `rc2-ofb`, `rc4`, `rc4-40`, `seed`, `seed-cbc`, `seed-cfb`, `seed-ofb`, `sm4-cbc`, `sm4-cfb`, `sm4-ecb`, `sm4-ofb`, `zlib`, and `zstd`.

To start this task, create a folder for this task and cd into it. In this folder, you will need to create a particular configuration file as discussed below. I created the folder advanced below for prerequisite apache setup, firewall adjustment and web server check.



The screenshot shows a Kali Linux terminal window titled "kali-linux-2024.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox". The terminal displays the following commands and output:

```
kali@kali: ~/lab5
File Actions Edit View Help
aria-128-cfb1 aria-128-cfb8 aria-128-ctr aria-128-ecb
aria-128-ofb aria-192-cbc aria-192-cfb aria-192-cfb1
aria-192-cfb8 aria-192-ctr aria-192-ecb aria-192-ofb
aria-256-cbc aria-256-cfb1 aria-256-cfb8
aria-256-ctr aria-256-ecb
bf bf-cbc bf-cfb
bf-ofb camellia-128-cbc camellia-128-ecb
camellia-192-cbc camellia-256-cbc camellia-256-ecb
cast-cbc cast5-cbc cast5-cfb
cast5-ofb des des-cbc
des-ecb des-edc des-edc-cbc
des-edc-ofb des-edc3 des-edc3-cbc
des-edc3-ofb des-ofb des3
rc2 rc2-40-cbc rc2-64-cbc rc2-cbc
rc2-cfb rc2-ecb rc2-ofb
rc4-40 seed seed-cbc seed-cfb
seed-ecb sm4-cbc sm4-cfb
sm4-ctr sm4-ecb sm4-ofb
zstd zlib

(kali@kali)-[~]
$ cd /usr/lib/ssl/

(kali@kali)-[/usr/lib/ssl]
$ ls
cert.pem certs misc openssl.cnf private

(kali@kali)-[/usr/lib/ssl]
$ cd

(kali@kali)-[~]
$ mkdir lab5/
mkdir: cannot create directory 'lab5/': File exists

(kali@kali)-[~]
$ cd lab5

(kali@kali)-[/lab5]
$ cp /usr/lib/ssl/openssl.cnf .

(kali@kali)-[/lab5]
$ ls
openssl.cnf

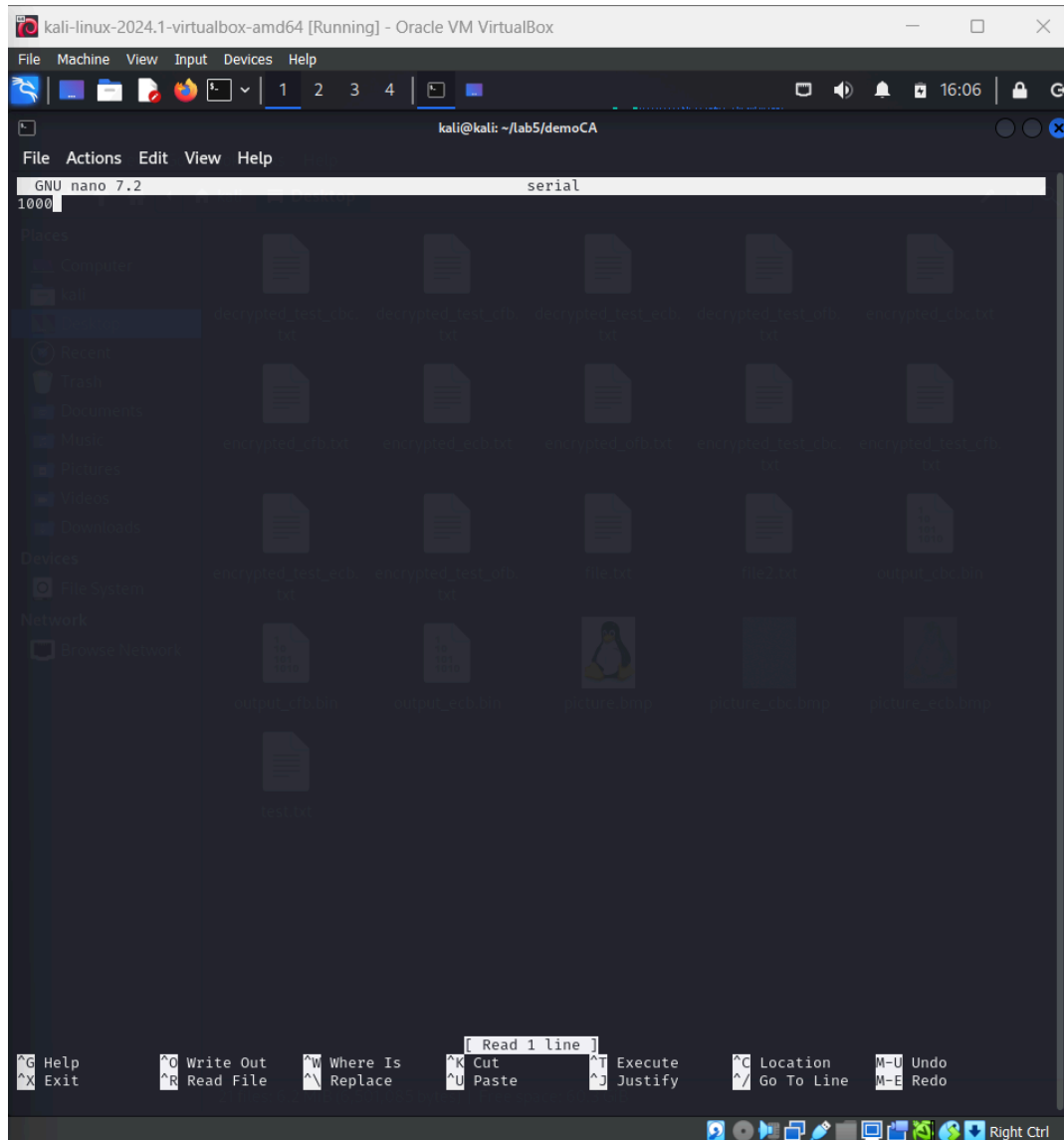
(kali@kali)-[/lab5]
$
```

The Configuration File openssl.conf:

In order to use OpenSSL to create certificates, you have to have a configuration file. The configuration file usually has an extension .cnf. It is used by three OpenSSL commands: ca, req and x509.

For the index.txt file, simply create an empty file. For the serial file, put a single number in string format (**e.g. 1000**) in the file. Once you have set up the configuration file openssl.cnf, you can create and issue certificates

```
kali-linux-2024.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
1 2 3 4
kali@kali: ~/lab5/demoCA
File Actions Edit View Help
(kali@kali)-[~]
$ cd /usr/lib/ssl/
(kali@kali)-[/usr/lib/ssl]
$ ls
cert.pem  certs  misc  openssl.cnf  private
(kali@kali)-[/usr/lib/ssl]
$ cd
(kali@kali)-[~]
$ mkdir lab5/
mkdir: cannot create directory 'lab5/': File exists
(kali@kali)-[~]
$ cd lab5
(kali@kali)-[~/lab5]
$ cp /usr/lib/ssl/openssl.cnf .
(kali@kali)-[~/lab5]
$ ls
openssl.cnf
(kali@kali)-[~/lab5]
$ nano openssl.cnf
(kali@kali)-[~/lab5]
$ mkdir demoCA
(kali@kali)-[~/lab5]
$ mkdir demoCA/certs demoCA/crl demoCA/newcerts
(kali@kali)-[~/lab5]
$ cd demoCA
(kali@kali)-[~/lab5/demoCA]
$ touch index.txt
(kali@kali)-[~/lab5/demoCA]
$ touch serial
(kali@kali)-[~/lab5/demoCA]
$
```



Certificate Authority (CA):

As described before, you need to generate a self-signed certificate for our CA. This means that this CA is totally trusted, and its certificate will serve as the root certificate. You can run the following command to generate the self-signed certificate for the CA:

```
openssl req -new -x509 -keyout ca.key -out ca.crt -config openssl.cnf
```


Step 1: Generate public/private key pair

The company needs to first create its own public/private key pair. You can run the following command to generate an RSA key pair (both private and public keys). You will also be required to provide a password to protect the keys.

The keys will be stored in the file server.key:

openssl genrsa -des3 -out server.key 2048

```
kali@kali: ~/lab5
$ openssl genrsa -des3 -out server.key 2048
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:

(kali@kali)~/lab5
$ openssl req -new -key server.key -out server.csr -config 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]:BD
State or Province Name (full name) [Some-State]:Dhaka
Locality Name (eg, city) []:Dhaka
Organization Name (eg, company) [Internet Widgits Pty Ltd]:DSi
Organizational Unit Name (eg, section) []:SQA
Common Name (e.g. server FQDN or YOUR name) []:example.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:kali
An optional company name []:

(kali@kali)~/lab5
$ openssl ca -in server.csr -out server.crt -cert ca.crt -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: Jul  7 20:11:44 2024 GMT
    Not After : Jul  7 20:11:44 2025 GMT
  Subject:
    countryName           = BD
    stateOrProvinceName   = Dhaka
    organizationName      = DSi
    organizationalUnitName = SQA
    commonName            = example.com
  X509v3 extensions:
    X509v3 Basic Constraints:
```

Step 2: Generate a Certificate Signing Request (CSR)

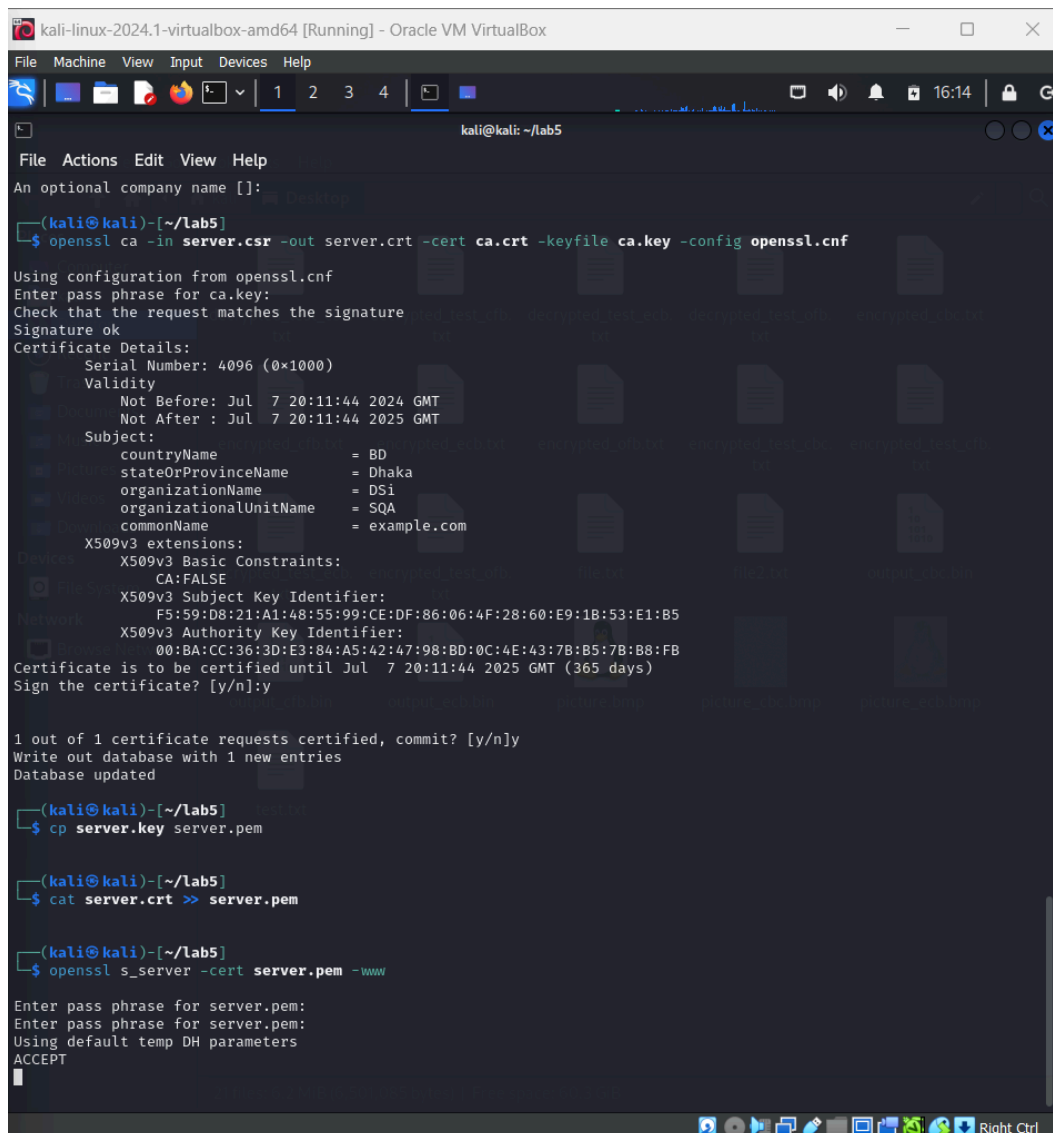
Once the company has the key file, it should generate a Certificate Signing Request (CSR). The CSR will be sent to the CA, who will generate a certificate for the key (usually after ensuring that identity information in the CSR matches with the server's true identity). Please use example.com as the common name of the certificate request.

openssl req -new -key server.key -out server.csr -config openssl.cnf

Step 3: Generating Certificates

The CSR file needs to have the CA's signature to form a certificate. In the real world, the CSR files are usually sent to a trusted CA for their signature. In this lab, you will use our own trusted CA to generate certificates.

openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key -config openssl.cnf



```
kali-linux-2024.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
kali@kali: ~/lab5
File Actions Edit View Help
An optional company name []:
(kali@kali)~/lab5
$ openssl ca -in server.csr -out server.crt -cert ca.crt -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: Jul  7 20:11:44 2024 GMT
    Not After : Jul  7 20:11:44 2025 GMT
  Subject:
    countryName           = BD
    stateOrProvinceName   = Dhaka
    organizationName      = DSI
    organizationalUnitName = SQA
    commonName            = example.com
  X509v3 extensions:
    X509v3 Basic Constraints:
      CA: FALSE
    X509v3 Subject Key Identifier:
      F5:59:D8:21:A1:48:55:99:CE:DF:86:06:4F:28:60:E9:1B:53:E1:B5
    X509v3 Authority Key Identifier:
      00:BA:CC:36:3D:E3:84:A5:42:47:98:BD:0C:4E:43:7B:B5:7B:B8:FB
Certificate is to be certified until Jul  7 20:11:44 2025 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
Database updated
(kali@kali)~/lab5
$ cp server.key server.pem
(kali@kali)~/lab5
$ cat server.crt >> server.pem
(kali@kali)~/lab5
$ openssl s_server -cert server.pem -www
Enter pass phrase for server.pem:
Enter pass phrase for server.pem:
Using default temp DH parameters
ACCEPT
```


If OpenSSL refuses to generate certificates, it is very likely that the names in your requests do not match with those of CA. Fix this and re-issue the above command.

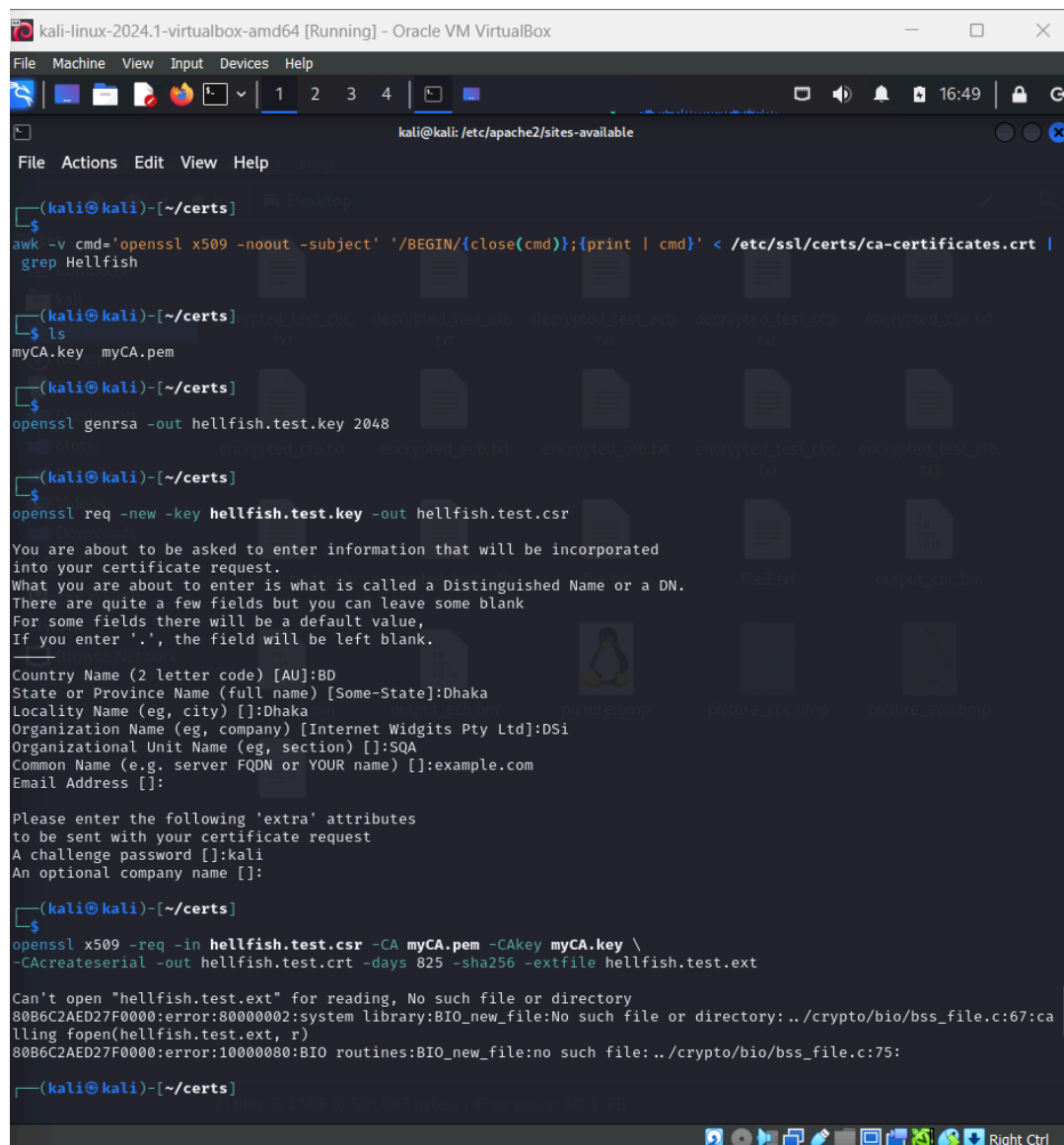
Next, let us launch a simple web server with the certificate generated in the previous task.

OpenSSL allows us to start a simple web server using the `s_server` command. Use the following steps:

Step 1: Combine the secret key and certificate into one file

```
cp server.key server.pem
```

```
cat server.crt >> server.pem
```

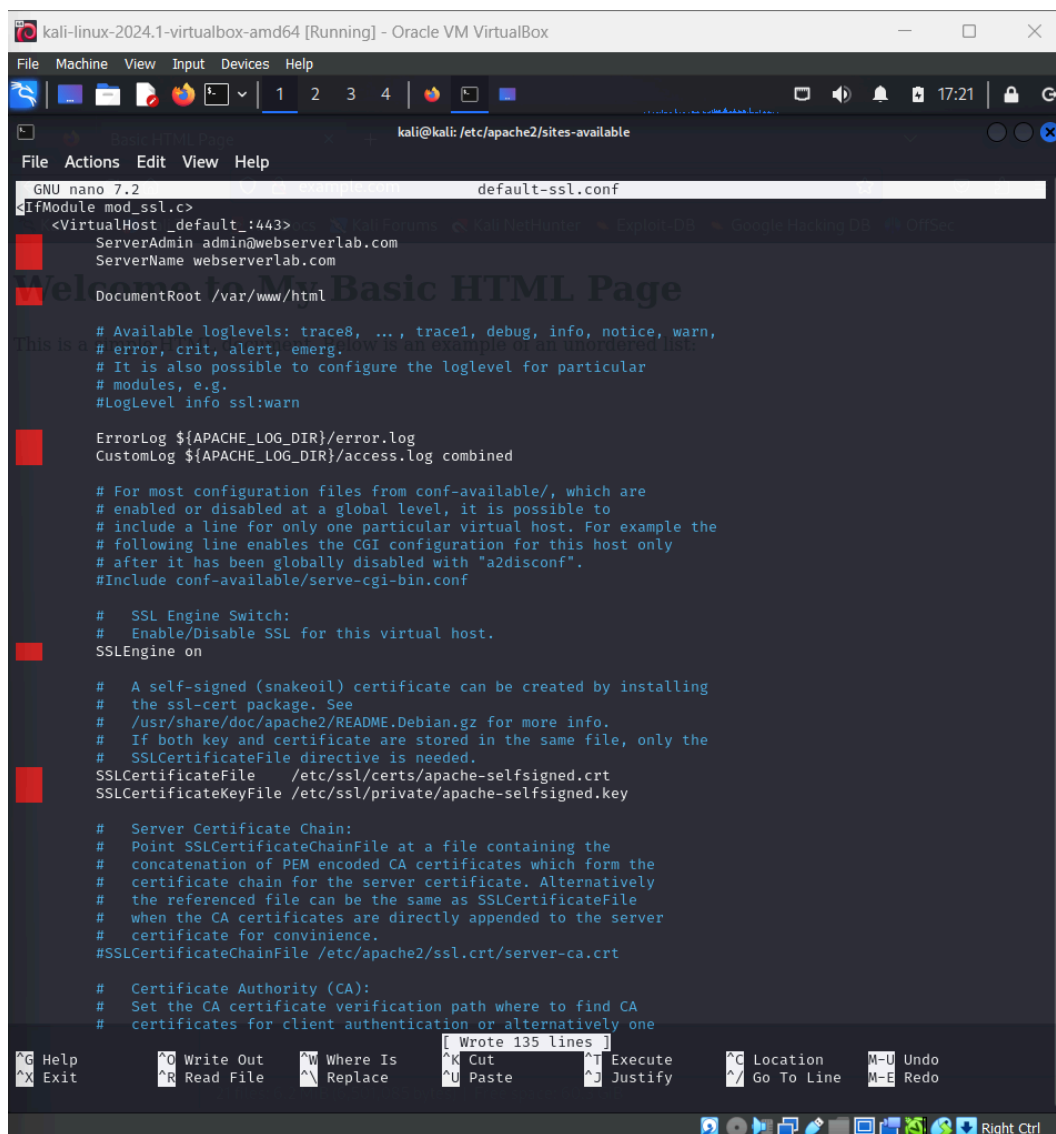


```
kali-linux-2024.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
1 2 3 4
kali@kali: /etc/apache2/sites-available
File Actions Edit View Help
(kali@kali)~[/certs]
$
awk -v cmd='openssl x509 -noout -subject' '/BEGIN/{close(cmd)};{print | cmd}' < /etc/ssl/certs/ca-certificates.crt |
grep Hellfish
(kali@kali)~[/certs]
$ ls
myCA.key  myCA.pem
(kali@kali)~[/certs]
$
openssl genrsa -out hellfish.test.key 2048
(kali@kali)~[/certs]
$
openssl req -new -key hellfish.test.key -out hellfish.test.csr
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]:BD
State or Province Name (full name) [Some-State]:Dhaka
Locality Name (eg, city) []:Dhaka
Organization Name (eg, company) [Internet Widgits Pty Ltd]:DSi
Organizational Unit Name (eg, section) []:SQA
Common Name (e.g. server FQDN or YOUR name) []:example.com
Email Address []:
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:kali
An optional company name []:
(kali@kali)~[/certs]
$
openssl x509 -req -in hellfish.test.csr -CA myCA.pem -CAkey myCA.key \
-CAcreateserial -out hellfish.test.crt -days 825 -sha256 -extfile hellfish.test.ext
Can't open "hellfish.test.ext" for reading, No such file or directory
80B6C2AED27F0000:error:80000002:system library:BIO_new_file:No such file or directory:../crypto/bio/bss_file.c:67:ca
lling fopen(hellfish.test.ext, r)
80B6C2AED27F0000:error:10000080:BIO routines:BIO_new_file:no such file:../crypto/bio/bss_file.c:75:
(kali@kali)~[/certs]
```

Step 2: Launch the web server using server.pem

openssl s_server -cert server.pem -www

By default, the server will listen on port 4433. You can alter that using the -accept option. Now, you can access the server using the following URL: <https://example.com:4433/>. Most likely, you will get an error message from the browser. In Firefox, you will see a message like the following: “example.com:4433 uses an invalid security certificate. The certificate is not trusted because the issuer certificate is unknown”.



```
kali-linux-2024.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
kali@kali: /etc/apache2/sites-available
File Actions Edit View Help
GNU nano 7.2 default-ssl.conf
IfModule mod_ssl.c>
<VirtualHost _default_:443>
  ServerAdmin admin@webserverlab.com
  ServerName webserverlab.com
  DocumentRoot /var/www/html
  # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
  # error, crit, alert, emerg.  It is also possible to configure the loglevel for particular
  # modules, e.g.
  #LogLevel info ssl:warn
  ErrorLog ${APACHE_LOG_DIR}/error.log
  CustomLog ${APACHE_LOG_DIR}/access.log combined
  # For most configuration files from conf-available/, which are
  # enabled or disabled at a global level, it is possible to
  # include a line for only one particular virtual host. For example the
  # following line enables the CGI configuration for this host only
  # after it has been globally disabled with "a2disconf".
  #Include conf-available/serve-cgi-bin.conf
  # SSL Engine Switch:
  # Enable/Disable SSL for this virtual host.
  SSLEngine on
  # A self-signed (snakeoil) certificate can be created by installing
  # the ssl-cert package. See
  # /usr/share/doc/apache2/README.Debian.gz for more info.
  # If both key and certificate are stored in the same file, only the
  # SSLCertificateFile directive is needed.
  SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt
  SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key
  # Server Certificate Chain:
  # Point SSLCertificateChainFile at a file containing the
  # concatenation of PEM encoded CA certificates which form the
  # certificate chain for the server certificate. Alternatively
  # the referenced file can be the same as SSLCertificateFile
  # when the CA certificates are directly appended to the server
  # certificate for convinience.
  #SSLCertificateChainFile /etc/apache2/ssl.crt/server-ca.crt
  # Certificate Authority (CA):
  # Set the CA certificate verification path where to find CA
  # certificates for client authentication or alternatively one
  #
  # Write 135 lines
  [Wrote 135 lines]
  ^G Help      ^O Write Out  ^W Where Is   [Wrote 135 lines]
  ^X Exit      ^R Read File  ^\ Replace    ^K Cut        ^T Execute
  ^_          ^_          ^_          ^U Paste      ^J Justify
  ^C Location  M-U Undo
  Go To Line  M-E Redo
  Right Ctrl
```

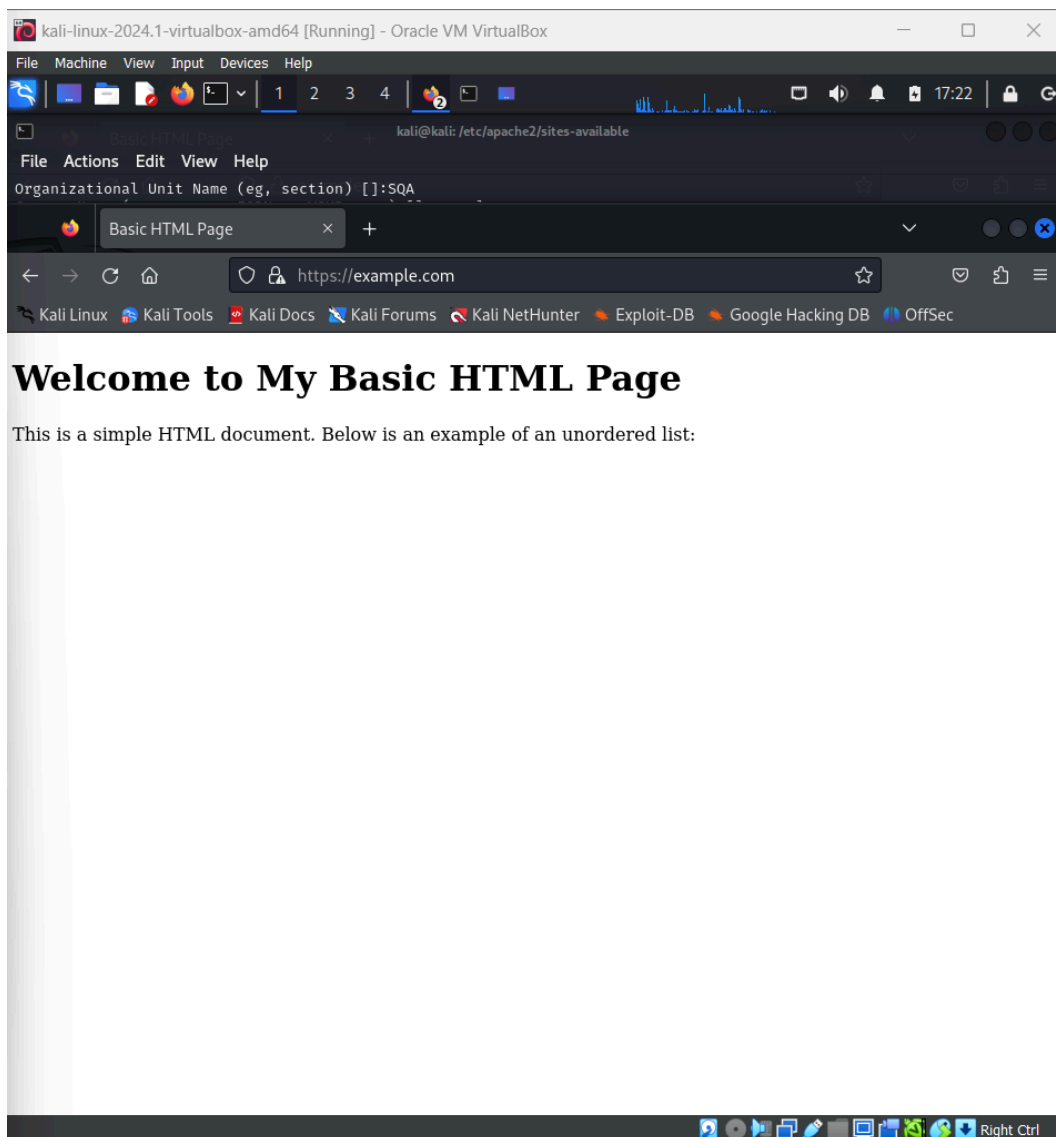

Next, use the following command to test the configuration.

sudo apache2ctl configtest

If a syntax is displayed onto the terminal, it indicates everything is okay.

Next restart the apache server using the restart command shown above.

Now, try to access the <https://example.com>. If everything is properly configured, you should be able to view the webpage in HTTPS.



If your browser is Firefox and it shows a warning, you can fix it by importing the CA certificate as described previously. If you use Chrome and it shows a similar warning, you can also import the CA certificate from the Manage certificate option under the Advanced setting in Chrome.

Checkpoint – 3 (5 marks): Access the <https://example.com> in your browser.

