

Lab 5: Securing Apache Web Server

Task : Becoming a CA

Copied the OpenSSL Configuration Template to my directory apachesecure
cp /opt/homebrew/etc/openssl/openssl.cnf ./openssl.cnf

Created Require CA Structure :

```
mkdir certs crl newcerts private  
touch index.txt  
echo 1000 > serial
```

Generated Root CA Certificate (Self Signed):

```
openssl req -new -x509 -days 365 -keyout private/ca.key -out ca.crt -config openssl.cnf
```

Common Name: My Root CA

Set a password

It has created ca.crt and private/ca.key

Task : Generating certificates for example.com and webserverlab.com

I will now act as a CA to issue certificates for both sites

example.com:

Generating private key:

```
openssl genrsa -des3 -out example.key 2048
```

Generate CSR

```
openssl req -new -key example.key -out example.csr -config openssl.cnf
```

Sign CSR with My CA

It will create a signed certificate for the site and a private key

For webserver.com followed the same process

Task Launching HTTPS Test Server :

example.com

Combine Key + Cert for example.com

```
cp example.key example.pem
```

```
cat example.crt >> example.pem
```

Launch OpenSSL Test Server (Checkpoint 1–2)

```
openssl s_server -cert example.pem -www
```

By default, it runs on port 4433.

In browser <https://example.com:4433> this shows all details. It means the HTTP server is ready

Task: Deploying HTTPS into Apache (Checkpoint 3, 4)

Enabling SSL Module

```
sudo /opt/homebrew/bin/httpd -M | grep ssl
```

Restart Apache

```
sudo apachectl restart
```

Configuring HTTPS Virtual Hosts

```
# HTTPS for example.com
<IfModule mod_ssl.c>
<VirtualHost *:443>
    ServerName example.com
    DocumentRoot "/opt/homebrew/var/www/example.com"
    ErrorLog "/opt/homebrew/var/logs/example-ssl-error.log"
    CustomLog "/opt/homebrew/var/logs/example-ssl-access.log" common
```

```
SSLEngine on
```

```
SSLCertificateFile "/Users/shakera/apachesecure/example.crt"
SSLCertificateKeyFile "/Users/shakera/apachesecure/example.key"
</VirtualHost>
</IfModule>
```

```
# HTTPS for webserverlab.com
```

```
<IfModule mod_ssl.c>
<VirtualHost *:443>
    ServerName webserverlab.com
    DocumentRoot "/opt/homebrew/var/www/webserverlab.com"
    ErrorLog "/opt/homebrew/var/logs/webserverlab-ssl-error.log"
    CustomLog "/opt/homebrew/var/logs/webserverlab-ssl-access.log" common
```

```
SSLEngine on
```

```
SSLCertificateFile "/Users/shakera/apachesecure/webserverlab.crt"
SSLCertificateKeyFile "/Users/shakera/apachesecure/webserverlab.key"
</VirtualHost>
</IfModule>
```

Tested in browser (it works, it loads the previously hosted dynamic website using HTML, JS)

Snapshots:



It works!

```
shakera@Shakera-MacBook-Air-291 ~ % sudo apachectl configtest && sudo apachectl restart
[Syntax OK]
shakera@Shakera-MacBook-Air-291 ~ % curl -I http://example.com
HTTP/1.1 200 OK
Date: Sat, 08 Nov 2025 13:08:31 GMT
Server: Apache/2.4.62 (Unix)
Last-Modified: Sat, 08 Nov 2025 12:54:17 GMT
ETag: "1e-64314cbbf9040"
Accept-Ranges: bytes
Content-Length: 30
Content-Type: text/html
[

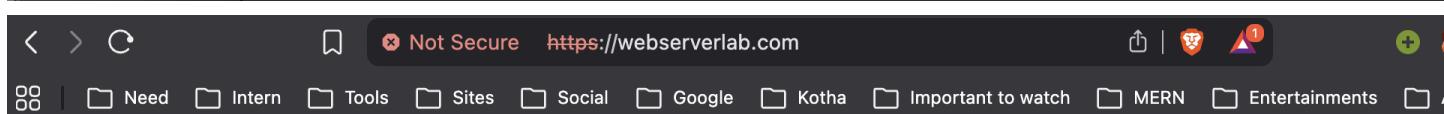
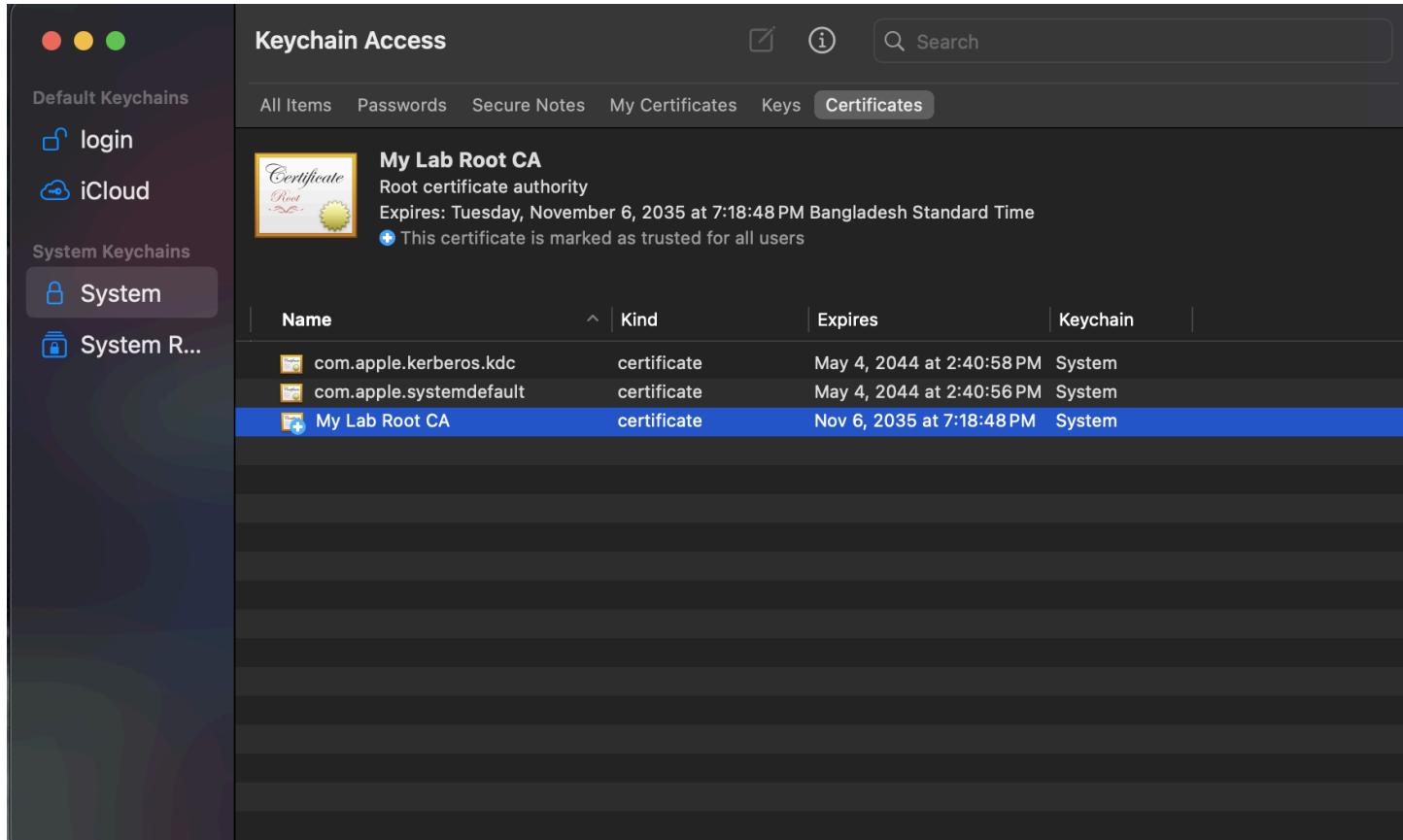
shakera@Shakera-MacBook-Air-291 ~ % curl -I http://webserverlab.com
HTTP/1.1 200 OK
Date: Sat, 08 Nov 2025 13:08:42 GMT
Server: Apache/2.4.62 (Unix)
Last-Modified: Sat, 08 Nov 2025 12:54:27 GMT
ETag: "23-64314cc5826c0"
Accept-Ranges: bytes
Content-Length: 35
Content-Type: text/html

shakera@Shakera-MacBook-Air-291 ~ %
```

```
ssl --zsh -- 78x27

[shakera@Shakera-MacBook-Air-291 ssl % openssl genrsa -aes256 -out example.key ]  
2048  
[Enter PEM pass phrase:  
[Verifying - Enter PEM pass phrase:  
shakera@Shakera-MacBook-Air-291 ssl % openssl req -new -key example.key -out e  
xample.csr -subj "/CN=example.com"  
  
[Enter pass phrase for example.key:  
[shakera@Shakera-MacBook-Air-291 ssl % openssl x509 -req -in example.csr -CA ca  
.crt -CAkey ca.key -CAcreateserial -out example.crt -days 825 -sha256  
Certificate request self-signature ok  
subject=CN=example.com  
[Enter pass phrase for ca.key:  
shakera@Shakera-MacBook-Air-291 ssl % openssl genrsa -aes256 -out webserverlab  
.key 2048  
[Enter PEM pass phrase:  
[Verifying - Enter PEM pass phrase:  
[shakera@Shakera-MacBook-Air-291 ssl % openssl req -new -key webserverlab.key -]  
out webserverlab.csr -subj "/CN=webserverlab.com"  
[Enter pass phrase for webserverlab.key:  
shakera@Shakera-MacBook-Air-291 ssl % openssl x509 -req -in webserverlab.csr -  
CA ca.crt -CAkey ca.key -CAcreateserial -out webserverlab.crt -days 825 -sha25  
6  
Certificate request self-signature ok  
subject=CN=webserverlab.com  
[Enter pass phrase for ca.key:  
shakera@Shakera-MacBook-Air-291 ssl %  
  
shakera@Shakera-MacBook-Air-291 apachesecure % cp example.key example  
.pem  
shakera@Shakera-MacBook-Air-291 apachesecure % cat example.crt >> exa  
mple.pem  
shakera@Shakera-MacBook-Air-291 apachesecure % openssl s_server -cert  
example.pem -www  
Enter pass phrase for example.pem:  
Enter pass phrase for example.pem:  
Using default temp DH parameters  
ACCEPT  
  
ACCEPT  
40E1DFF001000000:error:0A000416:SSL routines:ssl3_read_bytes:ssl/tls  
alert certificate unknown:ssl/record/rec_layer_s3.c:916:SSL alert num  
ber 46  
40E1DFF001000000:error:0A000416:SSL routines:ssl3_read_bytes:ssl/tls  
alert certificate unknown:ssl/record/rec_layer_s3.c:916:SSL alert num  
ber 46  
40E1DFF001000000:error:0A000416:SSL routines:ssl3_read_bytes:ssl/tls  
alert certificate unknown:ssl/record/rec_layer_s3.c:916:SSL alert num  
ber 46
```

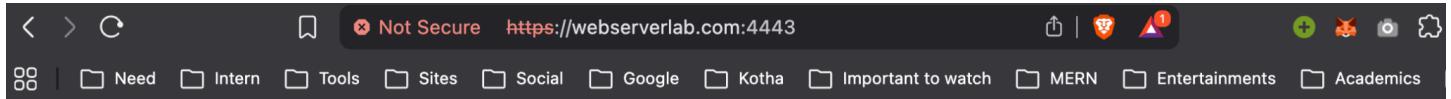
```
[shakera@Shakera-MacBook-Air-291 apachesecure % openssl x509 -req -in example.csr -CA ca.crt -CAkey private/ca.key -CAcreateserial -out example.crt -days 365 -sha256
Certificate request self-signature ok
subject=C=BD, CN=example.com
[Enter pass phrase for private/ca.key:
[shakera@Shakera-MacBook-Air-291 apachesecure %
```



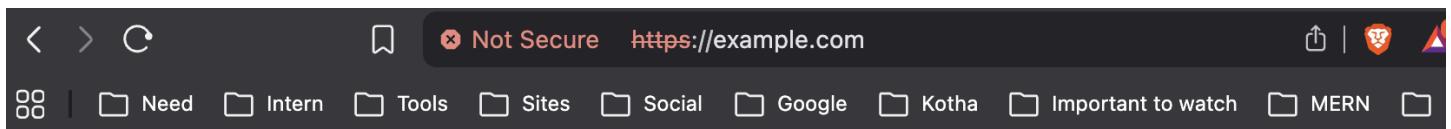
Full Name Generator

...

Show Full Name



```
s_server -cert webserverlab.pem -www -accept 4443
This TLS version forbids renegotiation.
Ciphers supported in s_server binary
TLSv1.3 :TLS_AES_256_GCM_SHA384    TLSv1.3    :TLS_CHACHA20_POLY1305_SHA256
TLSv1.3 :TLS_AES_128_GCM_SHA256    TLSv1.2    :ECDHE-ECDSA-AES256-GCM-SHA384
TLSv1.2 :ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2    :DHE-RSA-AES256-GCM-SHA384
TLSv1.2 :ECDHE-ECDSA-CHACHA20-POLY1305 TLSv1.2    :ECDHE-RSA-CHACHA20-POLY1305
TLSv1.2 :DHE-RSA-CHACHA20-POLY1305 TLSv1.2    :ECDHE-ECDSA-AES128-GCM-SHA256
TLSv1.2 :ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2    :DHE-RSA-AES128-GCM-SHA256
TLSv1.2 :ECDHE-ECDSA-AES256-SHA384 TLSv1.2    :ECDHE-RSA-AES256-SHA384
TLSv1.2 :DHE-RSA-AES256-SHA256    TLSv1.2    :ECDHE-ECDSA-AES128-SHA256
TLSv1.2 :ECDHE-RSA-AES128-SHA256    TLSv1.2    :DHE-RSA-AES128-SHA256
TLSv1.0 :ECDHE-ECDSA-AES256-SHA    TLSv1.0    :ECDHE-RSA-AES256-SHA
SSLv3 :DHE-RSA-AES256-SHA        TLSv1.0    :ECDHE-ECDSA-AES128-SHA
TLSv1.0 :ECDHE-RSA-AES128-SHA        SSLv3     :DHE-RSA-AES128-SHA
TLSv1.2 :RSA-PSK-AES256-GCM-SHA384 TLSv1.2    :DHE-PSK-AES256-GCM-SHA384
TLSv1.2 :RSA-PSK-CHACHA20-POLY1305 TLSv1.2    :DHE-PSK-CHACHA20-POLY1305
TLSv1.2 :ECDHE-PSK-CHACHA20-POLY1305 TLSv1.2    :AES256-GCM-SHA384
TLSv1.2 :PSK-AES256-GCM-SHA384    TLSv1.2    :PSK-CHACHA20-POLY1305
TLSv1.2 :RSA-PSK-AES128-GCM-SHA256 TLSv1.2    :DHE-PSK-AES128-GCM-SHA256
TLSv1.2 :AES128-GCM-SHA256    TLSv1.2    :PSK-AES128-GCM-SHA256
TLSv1.2 :AES256-SHA256    TLSv1.2    :AES128-SHA256
TLSv1.0 :ECDHE-PSK-AES256-CBC-SHA384 TLSv1.0    :ECDHE-PSK-AES256-CBC-SHA
SSLv3 :SRP-RSA-AES-256-CBC-SHA    SSLv3     :SRP-AES-256-CBC-SHA
TLSv1.0 :RSA-PSK-AES256-CBC-SHA384 TLSv1.0    :DHE-PSK-AES256-CBC-SHA384
SSLv3 :RSA-PSK-AES256-CBC-SHA    SSLv3     :DHE-PSK-AES256-CBC-SHA
SSLv3 :AES256-SHA        TLSv1.0    :PSK-AES256-CBC-SHA384
SSLv3 :PSK-AES256-CBC-SHA    TLSv1.0    :ECDHE-PSK-AES128-CBC-SHA256
TLSv1.0 :ECDHE-PSK-AES128-CBC-SHA    SSLv3     :SRP-RSA-AES-128-CBC-SHA
SSLv3 :SRP-AES-128-CBC-SHA    TLSv1.0    :RSA-PSK-AES128-CBC-SHA256
TLSv1.0 :DHE-PSK-AES128-CBC-SHA256 SSLv3     :RSA-PSK-AES128-CBC-SHA
SSLv3 :DHE-PSK-AES128-CBC-SHA    SSLv3     :AES128-SHA
TLSv1.0 :PSK-AES128-CBC-SHA256    SSLv3     :PSK-AES128-CBC-SHA
_____
Ciphers common between both SSL end points:
TLS_AES_128_GCM_SHA256    TLS_AES_256_GCM_SHA384    TLS_CHACHA20_POLY1305_SHA256
ECDHE-ECDSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-GCM-SHA256 ECDHE-ECDSA-AES256-GCM-SHA384
ECDHE-RSA-AES256-GCM-SHA384 ECDHE-ECDSA-CHACHA20-POLY1305 ECDHE-RSA-CHACHA20-POLY1305
ECDHE-RSA-AES128-SHA        ECDHE-RSA-AES256-SHA    AES128-GCM-SHA256
AES256-GCM-SHA384        AES128-SHA        AES256-SHA
Signature Algorithms: ECDSA+SHA256:RSA+PSS+SHA256:RSA+SHA256:ECDSA+SHA384:RSA+PSS+SHA384:RSA+SHA384:RSA+PSS+SHA512:RSA+SHA512
Shared Signature Algorithms: ECDSA+SHA256:RSA+PSS+SHA256:RSA+SHA256:ECDSA+SHA384:RSA+PSS+SHA384:RSA+SHA384:RSA+PSS+SHA512:RSA+SHA512
Supported groups: :X25519MLKEM768:x25519:secp256r1:secp384r1
Shared groups: X25519MLKEM768:x25519:secp256r1:secp384r1
```



Simple Calculator

Add

```
s_server -cert example.pem -www
This TLS version forbids renegotiation.
Ciphers supported in s_server binary
TLSv1.3 :TLS_AES_256_GCM_SHA384      TLSv1.3     :TLS_CHACHA20_POLY1305_SHA256
TLSv1.3 :TLS_AES_128_GCM_SHA256      TLSv1.2     :ECDHE-ECDSA-AES256-GCM-SHA384
TLSv1.2 :ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2     :DHE-RSA-AES256-GCM-SHA384
TLSv1.2 :ECDHE-ECDSA-CHACHA20-POLY1305 TLSv1.2     :ECDHE-RSA-CHACHA20-POLY1305
TLSv1.2 :DHE-RSA-CHACHA20-POLY1305 TLSv1.2     :ECDHE-ECDSA-AES128-GCM-SHA256
TLSv1.2 :ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2     :DHE-RSA-AES128-GCM-SHA256
TLSv1.2 :ECDHE-ECDSA-AES256-SHA384 TLSv1.2     :ECDHE-RSA-AES256-SHA384
TLSv1.2 :DHE-RSA-AES256-SHA256       TLSv1.2     :ECDHE-ECDSA-AES128-SHA256
TLSv1.2 :ECDHE-RSA-AES128-SHA256   TLSv1.2     :DHE-RSA-AES128-SHA256
TLSv1.0 :ECDHE-ECDSA-AES256-SHA    TLSv1.0     :ECDHE-RSA-AES256-SHA
SSLv3   :DHE-RSA-AES256-SHA        TLSv1.0     :ECDHE-ECDSA-AES128-SHA
TLSv1.0 :ECDHE-RSA-AES128-SHA    SSLv3      :DHE-RSA-AES128-SHA
TLSv1.2 :RSA-PSK-AES256-GCM-SHA384 TLSv1.2     :DHE-PSK-AES256-GCM-SHA384
TLSv1.2 :RSA-PSK-CHACHA20-POLY1305 TLSv1.2     :DHE-PSK-CHACHA20-POLY1305
TLSv1.2 :ECDHE-PSK-CHACHA20-POLY1305 TLSv1.2     :AES256-GCM-SHA384
TLSv1.2 :PSK-AES256-GCM-SHA384    TLSv1.2     :PSK-CHACHA20-POLY1305
TLSv1.2 :RSA-PSK-AES128-GCM-SHA256 TLSv1.2     :DHE-PSK-AES128-GCM-SHA256
TLSv1.2 :AES128-GCM-SHA256       TLSv1.2     :PSK-AES128-GCM-SHA256
TLSv1.2 :AES256-SHA256          TLSv1.2     :AES128-SHA256
TLSv1.0 :ECDHE-PSK-AES256-CBC-SHA384 TLSv1.0     :ECDHE-PSK-AES256-CBC-SHA
SSLv3   :SRP-RSA-AES-256-CBC-SHA   SSLv3      :SRP-AES-256-CBC-SHA
TLSv1.0 :RSA-PSK-AES256-CBC-SHA384 TLSv1.0     :DHE-PSK-AES256-CBC-SHA384
SSLv3   :RSA-PSK-AES256-CBC-SHA   SSLv3      :DHE-PSK-AES256-CBC-SHA
SSLv3   :AES256-SHA             TLSv1.0     :PSK-AES256-CBC-SHA384
SSLv3   :PSK-AES256-CBC-SHA     TLSv1.0     :ECDHE-PSK-AES128-CBC-SHA256
TLSv1.0 :ECDHE-PSK-AES128-CBC-SHA TLSv3      :SRP-RSA-AES-128-CBC-SHA
SSLv3   :SRP-AES-128-CBC-SHA     TLSv1.0     :RSA-PSK-AES128-CBC-SHA256
TLSv1.0 :DHE-PSK-AES128-CBC-SHA256 SSLv3      :RSA-PSK-AES128-CBC-SHA
SSLv3   :DHE-PSK-AES128-CBC-SHA   SSLv3      :AES128-SHA
TLSv1.0 :PSK-AES128-CBC-SHA256   SSLv3      :PSK-AES128-CBC-SHA
---
Ciphers common between both SSL end points:
TLS_AES_128_GCM_SHA256      TLS_AES_256_GCM_SHA384      TLS_CHACHA20_POLY1305_SHA256
ECDHE-ECDSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-GCM-SHA256 ECDHE-ECDSA-AES256-GCM-SHA384
ECDHE-RSA-AES256-GCM-SHA384 ECDHE-ECDSA-CHACHA20-POLY1305 ECDHE-RSA-CHACHA20-POLY1305
ECDHE-RSA-AES128-SHA        ECDHE-RSA-AES256-SHA        AES128-GCM-SHA256
AES256-GCM-SHA384          AES128-SHA          AES256-SHA
Signature Algorithms: ECDSA+SHA256:RSA-PSS+SHA256:RSA+SHA256:ECDSA+SHA384:RSA-PSS+SHA384:RSA+SHA384:RSA-PS
Shared Signature Algorithms: ECDSA+SHA256:RSA-PSS+SHA256:RSA+SHA256:ECDSA+SHA384:RSA-PSS+SHA384:RSA+SHA384
Supported groups: :X25519MLKEM768:x25519:secp256r1:secp384r1
Shared groups: X25519MLKEM768:x25519:secp256r1:secp384r1
```