Project 1d. Create, Sign and Install Web Server Certificate

In this project, we learn how to create and operate Public Key Infrastructure (PKI) with OpenSSL utility on the EAS VI fc28 Linux instance you have cloned. Specifically, we will learn

1. Utilize the script in a Linux system to setup directories to serve as a CA for an organization, learn how to sign certificate request for clients or servers on secure email or secure web access purpose.
2. Learn how to generate server certificate requests as a webmaster, send them to CA for signing and install the signed certificates in Apache web server for secure web access.
3. Learn how to create client certificate requests for secure web access and secure email, send them to CA for signing, combine the returned signed certificate with the private key in password protected .p12 file for importing to browsers or email apps.
4. Learn how to set up an Apache secure directory to be protected with client-server mutual authentication, which requires a user to present the client certificate signed by the specific CA and restricts access to only a user with the specific subject name field of the client certificate that appears in /etc/httpd/conf/httpd.password file.

The assignment will be graded by checking

1. whether the server certificate is properly filled with proper subject field, especially with the proper Common Name.
2. whether the expiration dates are correctly setup.
3. whether the web server is setup properly for presenting server certificate to a browser and establishing a secure web access.
4. whether the CA is imported to the learner's Firefox browser secure store.
5. whether the client .p12 file is properly imported to the learner's Firefox browser secure store.

Assignment Topic:
In this exercise, we will use the Linux instance you already cloned. We use /etc/pki/tls/misc/CA script to create a server certificate request as a web master. Then assume the CA role to sign the server certificate. We install the signed server certificate and related private key in the proper location where the apache web server will read them when it restarts. We will test if the web server is properly presenting the server certificate to the browser. We will also create client certificates to be used for client-server mutual authentication.
Be aware the Linux command option parameter leading dash, is a short dash as the one you type in keyboard. The word processor often substitute - with – (long dash). When you copy and paste commands from this document to the instance for execution, please double check if the option dash is correct, or substitute then with the proper keyboard dash key.

Setup instructions:

Before you begin, remember to login to your instance following the steps in Section 3 of Project 1a, http://ciast.uccs.edu/coursera/pub/project1aV3.pdf. Switch to the /etc/pki/tls directory with "cd /etc/pki/tls".

Due to the removal of CA related scripts on fc28 distribution in favor of 389.ds from mozilla nss, we will get a copy of CA related scripts that work with openssl using the following steps:

```
[root@cs591 tls]# wget http://ciast.uccs.edu/coursera/pub/misc.tbz
--2018-09-27 11:03:59--  http://ciast.uccs.edu/coursera/pub/misc.tbz
Resolving ciast.uccs.edu (ciast.uccs.edu)... 128.198.160.70
Connecting to ciast.uccs.edu (ciast.uccs.edu)|128.198.160.70|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2408 (2.4K)
Saving to: 'misc.tbz'

misc.tbz          100%[===================>]   2.35K  --.-KB/s    in 0s

2018-09-27 11:03:59 (51.7 MB/s) - 'misc.tbz' saved [2408/2408]

[root@cs591 tls]# ls
cert.pem  certs  misc  misc.tbz  openssl.cnf  private
[root@cs591 tls]# tar xjf misc.tbz
[root@cs591 tls]# ls -al misc
total 32
drwxr-xr-x. 2 root root 4096 Mar  7  2018 .
drwxr-xr-x. 5 root root 4096 Sep 27 11:03 ..
-rwxr-xr-x. 1 root root 5178 Nov 13  2017 CA
-rwxr-xr-x. 1 root root  119 Nov 13  2017 c_hash
-rwxr-xr-x. 1 root root  152 Nov 13  2017 c_info
-rwxr-xr-x. 1 root root  112 Nov 13  2017 c_issuer
-rwxr-xr-x. 1 root root  110 Nov 13  2017 c_name
```

Step 1. Setup /etc/pki/tls/openssl.cnf

This configuration file allows all commands using openssl utility to have shared parameters/default values.

Edit openssl.cnf and replace the following line starting at line # 128 with default values reflect your own locale info. Here I show the one I use for UCCS CS Classes.

```
[ req_distinguished_name ]
countryName                     = Country Name (2 letter code)
countryName_default             = US
countryName_min                 = 2
countryName_max                 = 2

stateOrProvinceName             = State or Province Name (full name)
stateOrProvinceName_default     = Colorado

localityName                    = Locality Name (eg, city)
localityName_default            = Colorado Springs

0.organizationName              = Organization Name (eg, company)
0.organizationName_default      = UCCS
```

```
# we can do this but it is not needed normally :-)
#1.organizationName             = Second Organization Name (eg, company)
#1.organizationName_default     = World Wide Web Pty Ltd

organizationalUnitName          = Organizational Unit Name (eg, section)
organizationalUnitName_default  = CS
```

Make sure to remove # in front of Lines 135 and 148.

By setting the same default values, the CSR created by local users will have the uniform locale info and make it easier to recognize the signed certificate and create httpd.password list for further restricting web access.
For Chrome browser, we need to generate the server certificate with new subjectAltName field.
On google support web site https://support.google.com/chrome/a/answer/7391219?hl=en, it indicates

"For Chrome 58 and later, only the subjectAlternativeName extension, not commonName, is used to match the domain name and site certificate. The certificate subject alternative name can be a domain name or IP address."

Therefore we need to add the following lines to your openssl.cnf file after line 204.
Seehttps://www.openssl.org/docs/man1.0.2/apps/x509v3_config.html#Subject-Alternative-Name
or http://wiki.cacert.org/FAQ/subjectAltName

subjectAltName=DNS:*.csnet.uccs.edu

For UCCS CS591 Fall 2018 class, use DNS: *.csnet.uccs.edu as the value for subjectAltName.

Step 2. Setup CA operation.
Run the command "misc/CA -newca". Accept the default locale info set by openssl.cnf. Replace ca.myuccs.net with "ca.<your designated DNS name>" as Common Name. Replace ca@ws2.myuccs.net with your UCCS email address. This is for others to contact you as this CA.
Note that the last command here, we copy the cacert.pem to web server cert directory so that a user can download it to its CA (authorities) list for certificate verification purpose.

```
[root@ip-172-31-14-30 tls]# misc/CA -newca
[root@cs591 tls]# misc/CA -newca
CA certificate filename (or enter to create)

Making CA certificate ...
Generating a 2048 bit RSA private key
.....................................................................................
......+++
.................................................+++
writing new private key to '/etc/pki/CA/private/./cakey.pem'
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) [US]:
State or Province Name (full name) [Colorado]:
Locality Name (eg, city) [Colorado Springs]:
Organization Name (eg, company) [UCCS]:
Organizational Unit Name (eg, section) [CS]:
Common Name (eg, your name or your server's hostname) []:ca.cs591.csnet.uccs.edu
Email Address []:ca@cs591.csnet.uccs.edu

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Using configuration from /etc/pki/tls/openssl.cnf
Enter pass phrase for /etc/pki/CA/private/./cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number:
            b2:da:8e:56:1a:6d:70:6c
        Validity
            Not Before: Sep 27 18:34:42 2018 GMT
            Not After : Sep 26 18:34:42 2021 GMT
        Subject:
            countryName               = US
            stateOrProvinceName       = Colorado
            organizationName          = UCCS
            organizationalUnitName    = CS
            commonName                = ca.cs591.csnet.uccs.edu
            emailAddress              = ca@cs591.csnet.uccs.edu
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                C8:A6:5C:36:27:4C:B5:BF:CB:8A:A1:E2:9F:ED:32:06:70:43:ED:90
            X509v3 Authority Key Identifier:
                keyid:C8:A6:5C:36:27:4C:B5:BF:CB:8A:A1:E2:9F:ED:32:06:70:43:ED:90

            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Subject Alternative Name:
                DNS:cs591.csnet.uccs.edu
            X509v3 Issuer Alternative Name:
                DNS:cs591.csnet.uccs.edu
Certificate is to be certified until Sep 26 18:34:42 2021 GMT (1095 days)

Write out database with 1 new entries
Data Base Updated
[root@cs591 tls]# cp /etc/pki/CA/cacert.pem /var/www/html/cert
cp: overwrite '/var/www/html/cert/cacert.pem'? y $ ls ../CA
cacert.pem  certs  index.txt       index.txt.old  private
careq.pem   crl    index.txt.attr  newcerts       serial
$ sudo mkdir /var/www/html/cert
$ sudo cp ../CA/cacert.pem /var/www/html/cert
```

 # copy ca certificate for others (browser users) to downaload CA cert and trust it).

Step 3. Create Server Certificate Request.

Make sure you use "*.csnet.uccs.edu" as Common name. The browser will check if this field matches that of the domain name of the url entered by the user. You can use webmaster@<your designated DNS name> for email address. Make sure you observer the "`X509v3 Subject Alternative Name:`" field contains `DNS:*.csnet.uccs.edu`.

```
[ec2-user@ip-172-31-20-120 tls]$ sudo misc/CA -newreq
Generating a 2048 bit RSA private key
.....................................................................................
.............................+++
.......................+++
writing new private key to 'newkey.pem'
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) [US]:
State or Province Name (full name) [Colorado]:
Locality Name (eg, city) [Colorado Springs]:
Organization Name (eg, company) [UCCS]:
Organizational Unit Name (eg, section) [CS]:
Common Name (eg, your name or your server's hostname) []:*.cchow.myuccs.net
Email Address []:webmaster@cchow.myuccs.net

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Request is in newreq.pem, private key is in newkey.pem
[ec2-user@ip-172-31-20-120 tls]$ sudo cp newkey.pem serverKey.pem
[ec2-user@ip-172-31-20-120 tls]$ sudo cp newreq.pem serverReq.pem
 [root@ip-172-31-14-30 tls]$ sudo misc/CA -sign
Using configuration from /etc/pki/tls/openssl.cnf
Enter pass phrase for /etc/pki/CA/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 13418977136347706153 (0xba39c89bd517f329)
        Validity
            Not Before: Sep 28 05:44:40 2017 GMT
            Not After : Sep 28 05:44:40 2018 GMT
        Subject:
```

```
            countryName               = US
            stateOrProvinceName       = Colorado
            localityName              = Colorado Springs
            organizationName          = UCCS
            organizationalUnitName    = CS
            commonName                = *.myuccs.net
            emailAddress              = webmaster@ws2.myuccs.net
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                AD:7B:C4:75:FE:F2:29:BF:6B:62:B9:2F:54:62:24:CF:CD:48:8D:E1
            X509v3 Authority Key Identifier:
                keyid:A3:A8:2F:EF:AB:0F:71:C1:AB:16:42:FA:C4:66:4A:16:49:6A:7D:7D

            X509v3 Subject Alternative Name:
                DNS:*.myuccs.net
            X509v3 Issuer Alternative Name:
                <EMPTY>

Certificate is to be certified until Sep 28 05:44:40 2018 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
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 13418977136347706153 (0xba39c89bd517f329)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=Colorado, O=UCCS, OU=CS,
CN=ca.myuccs.net/emailAddress=ca@ws2.myuccs.net
        Validity
            Not Before: Sep 28 05:44:40 2017 GMT
            Not After : Sep 28 05:44:40 2018 GMT
        Subject: C=US, ST=Colorado, L=Colorado Springs, O=UCCS, OU=CS,
CN=*.myuccs.net/emailAddress=webmaster@ws2.myuccs.net
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:a9:2c:23:10:65:46:e5:b5:e7:f8:75:53:6f:4e:
                    b2:da:10:b4:a0:bf:22:32:c4:b0:d4:46:25:01:cb:
                    06:d0:44:0a:10:fd:5e:62:3f:58:e9:de:2e:bc:fb:
                    70:f1:cd:77:d2:a6:f7:74:97:37:de:e5:51:14:e6:
                    4b:eb:0f:b1:39:0c:d4:30:8e:ec:67:45:e7:a6:62:
                    9d:98:05:af:80:8d:13:db:fa:1b:29:55:a3:7b:80:
                    09:ef:85:92:f3:de:5b:68:0d:20:fe:25:d8:9b:17:
                    ea:ea:04:1a:53:04:9d:6f:a6:82:e7:04:0e:06:17:
                    1b:dc:36:b5:4a:95:85:30:32:66:f6:84:53:2d:d5:
                    06:e0:c4:e1:c9:27:05:5f:56:5f:e0:0c:11:4d:7d:
```

```
                    4b:14:fa:b3:d0:04:9a:9b:b6:5b:38:84:3b:90:ce:
                    79:22:cc:f6:f1:a7:47:08:35:b3:53:e8:de:49:70:
                    33:56:b6:a7:f7:6c:3b:9c:dc:0f:c5:08:3d:5c:59:
                    e4:5e:f5:af:a1:b0:cc:d4:f6:c5:85:de:15:e4:eb:
                    10:b0:4a:a1:a8:40:80:ee:6b:31:30:c4:f1:c1:39:
                    e8:53:0e:40:65:69:bd:84:e3:ae:96:89:37:52:22:
                    7b:e8:ce:e5:70:f6:66:e9:9b:c4:4b:29:45:a2:59:
                    88:eb
                Exponent: 65537 (0x10001)
            X509v3 extensions:
                X509v3 Basic Constraints:
                    CA:FALSE
                Netscape Comment:
                    OpenSSL Generated Certificate
                X509v3 Subject Key Identifier:
                    AD:7B:C4:75:FE:F2:29:BF:6B:62:B9:2F:54:62:24:CF:CD:48:8D:E1
                X509v3 Authority Key Identifier:
                    keyid:A3:A8:2F:EF:AB:0F:71:C1:AB:16:42:FA:C4:66:4A:16:49:6A:7D:7D

                X509v3 Subject Alternative Name:
                    DNS:*.myuccs.net
                X509v3 Issuer Alternative Name:
                    <EMPTY>

    Signature Algorithm: sha256WithRSAEncryption
         4e:7c:d1:1e:2c:fb:41:7c:5d:fa:0b:6d:72:63:78:86:15:ac:
         41:87:3b:87:9e:74:2a:b5:22:26:b1:a4:93:cd:5e:72:f7:09:
         ad:12:dc:e5:64:3f:45:6a:6b:1c:19:14:3e:fd:05:bf:a5:89:
         81:90:6a:0b:11:8e:bf:7a:1b:01:17:c4:b3:72:38:1b:4d:b6:
         0c:b5:44:88:bb:9e:ff:fe:a2:d3:14:37:d0:cf:5c:e0:37:e0:
         2b:34:b5:62:aa:63:77:62:30:63:9d:51:54:24:95:07:c8:e3:
         b2:84:b3:e2:50:bc:d8:f6:00:79:a5:be:84:47:a9:18:27:92:
         13:53:8d:59:4f:40:13:ee:4d:19:55:c5:19:6e:73:a5:03:3f:
         4d:80:7b:90:88:5d:28:7e:45:7f:60:3c:c6:52:31:a5:d0:4c:
         6f:b2:ff:e1:55:51:92:31:6e:f9:d9:be:f3:a3:41:f2:cd:09:
         77:a6:9b:34:33:8f:0b:c4:f8:8d:91:a8:9c:cc:bd:62:84:c1:
         8b:d3:38:35:cb:ef:9c:ea:58:72:cc:28:67:5a:dc:fe:7b:9e:
         c2:6f:03:95:2b:e5:81:c5:7e:3a:5e:a2:eb:a1:81:e8:23:5d:
         a8:07:66:0a:42:6a:98:b4:72:bb:17:2f:36:79:ba:45:75:7c:
         c7:87:10:49
-----BEGIN CERTIFICATE-----
MIIEMjCCAxqgAwIBAgIJALo5yJvVF/MpMA0GCSqGSIb3DQEBCwUAMHYxCzAJBgNV
BAYTAlVTMREwDwYDVQQIDAhDb2xvcmFkbzENMAsGA1UECgwEVUNDUzELMAkGA1UE
CwwCQ1MxFjAUBgNVBAMMDWNhLm15dWNjcy5uZXQxIDAeBgkqhkiG9w0BCQEWEWNh
QHdzMi5teXVjY3MubmV0MB4XDTE3MDkyODA1NDQ0MFoXDTE4MDkyODA1NDQ0MFow
gZcxCzAJBgNVBAYTAlVTMREwDwYDVQQIDAhDb2xvcmFkbzEZMBcGA1UEBwwQQ29s
b3JhZG8gU3ByaW5nczENMAsGA1UECgwEVUNDUzELMAkGA1UECwwCQ1MxFTATBgNV
BAMMDCoubXl1Y2NzLm5ldDEnMCUGCSqGSIb3DQEJARYYd2VibWFzdGVyQHdzMi5t
eXVjY3MubmV0MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAqSwjEGVG
5bXn+HVTb06y2hC0oL8iMsSw1EYlAcsG0EQKEP1eYj9Y6d4uvPtw8c130qb3dJc3
3uVRFOZL6w+xOQzUMI7sZ0XnpmKdmAWvgI0T2/obKVWje4AJ74WS895baA0g/iXY
mxfq6gQaUwSdb6aC5wQOBhcb3Da1SpWFMDJm9oRTLdUG4MThyScFX1Zf4AwRTX1L
FPqz0ASam7ZbOIQ7kM55Isz28adHCDWzU+jeSXAzVran92w7nNwPxQg9XFnkXvWv
obDM1PbFhd4V5OsQsEqhqECA7msxMMTxwTnoUw5AZWm9hOOulok3UiJ76M7lcPZm
6ZvESylFolmI6wIDAQABo4GgMIGdMAkGA1UdEwQCMAAwLAYJYIZIAYb4QgENBB8W
```

```
HU9wZW5TU0wgR2VuZXJhdGVkIENlcnRpZmljYXRlMB0GA1UdDgQWBBSte8R1/vIp
v2tiuS9UYiTPzUiN4TAfBgNVHSMEGDAWgBSjqC/vqw9xwasWQvrEZkoWSWp9fTAX
BgNVHREEEDAOggwqLm15dWNjcy5uZXQwCQYDVR0SBAIwADANBgkqhkiG9w0BAQsF
AAOCAQEATnzRHiz7QXxd+gttcmN4hhWsQYc7h550KrUiJrGkk81ecvcJrRLc5WQ/
RWprHBkUPv0Fv6WJgZBqCxGOv3obARfEs3I4G022DLVEiLue//6i0xQ30M9c4Dfg
KzS1Yqpjd2IwY51RVCSVB8jjsoSz4lC82PYAeaW+hEepGCeSE1ONWU9AE+5NGVXF
GW5zpQM/TYB7kIhdKH5Ff2A8xlIxpdBMb7L/4VVRkjFu+dm+86NB8s0Jd6abNDOP
C8T4jZGonMy9YoTBi9M4NcvvnOpYcswoZ1rc/nuewm8DlSvlgcV+Ol6i66GB6CNd
qAdmCkJqmLRyuxcvNnm6RXV8x4cQSQ==
-----END CERTIFICATE-----
Signed certificate is in newcert.pem
 [ec2-user@ip-172-31-20-120 tls]$ sudo cp newcert.pem serverCert.pem
[ec2-user@ip-172-31-20-120 tls]$ sudo cp serverCert.pem
/etc/pki/tls/certs/localhost.crt
```

Step 3a. Remove encryption protection on server private key!

In the following we remove encryption protection on server private key. Reason?
When a server reboots at night, it will ask for the passphrase that protects the server private key. We donot want to be waked up by paging from the server☺

However, we still need to protect the server private key with root only file access.

```
[ec2-user@ip-172-31-20-120 tls]$ sudo openssl rsa -in serverKey.pem -out
serverUnenc.key
Enter pass phrase for serverKey.pem:
writing RSA key
[ec2-user@ip-172-31-20-120 tls]$ sudo chmod 700 *.key
[ec2-user@ip-172-31-20-120 tls]$ sudo cp serverUnenc.key
/etc/pki/tls/private/localhost.key

# restart web server.
[ec2-user@ip-172-31-20-120 tls]$ sudo systemctl restart httpd.service
```

Step 4. Try access with Firefox browser

Let us use local firefox browser to access the secure web server you just set up
Type in https://cs591.csnet.uccs.edu/ into the url address box. Replace cs591.csnet.uccs.edu with your domain name. You got the following message. Make sure do not use IP address for cs591.csnet.uccs.edu.

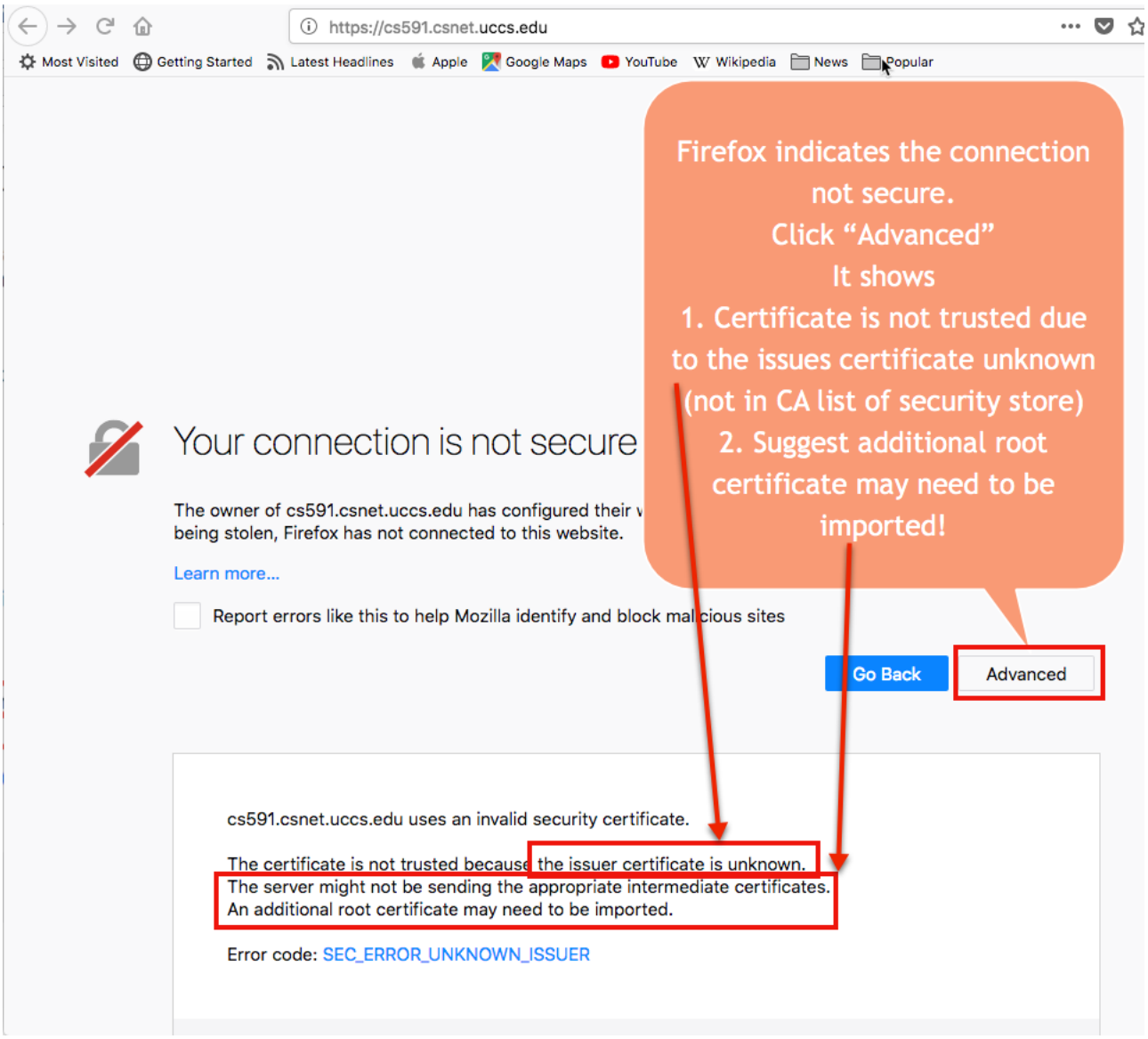


Figure 5.1a. Browser shows the server certificate not truested due to missing issuer certificate.

Click the "Add Exception…" button below the message. Then click "Confirm Security Exception" to accept this web server temporarily,

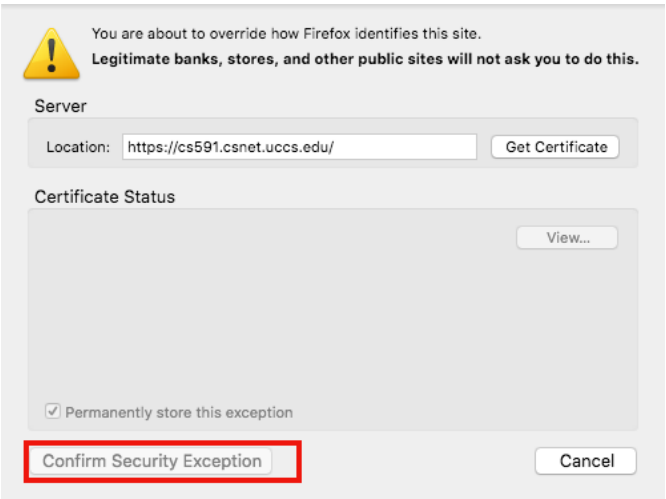


Figure 5.1b. Confirm Security Exception.

Step 5. Fix DNS mapping and issuer certificate unknown problems

Most browser will verify the secure web access using the domain name on the url. Normally a production web site will register the domain name. If you do not want to register domain name and pay for annual fee, for simplicity reason, you can edit the local host file with web server DNS name and to mimic the DNS query without visiting the target DNS name server.

For CS591 Fall 2018 class I have created <login>.csnet.uccs.edu DNS entry for you already. Therefore you can skip the following steps to setup the local DNS lookup.

Step 5a. Setup local DNS local lookup when DNS entry request is not available.

Follow the instruction in the following web site, depending on your machine type, add DNS entry to the host file.

http://wahldev.com/local-dns-settings-map-a-domain-to-a-local-ip-address/

```
On Mac,
% sudo nano /private/etc/host

##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1       localhost
255.255.255.255 broadcasthost
::1             localhost
#ws.cchow.myuccs.net  13.59.2.124  this one not correct, IP first
13.59.2.124 ws.cchow.myuccs.net
```

Add the last line to /private/etc/hosts file. Replace 13.59.2.124 with your own instance IP address. Replace ws.cchow.myuccs.net with ws.<login>.myuccs.net, where <login> is just the username part of your Coursera account or UCCS account.

update your DNS cache from Terminal using the command:

```
% dscacheutil -flushcache
```

 # end of Step 5a

Step 5b. Import CA Certificate to Firefox Security Store.

Let us import our CA certificate to the firefox security store.
First we need to retrieve the cacert.pem CA certificate file.
Now enter http://cs591.csnet.uccs.edu/cert again

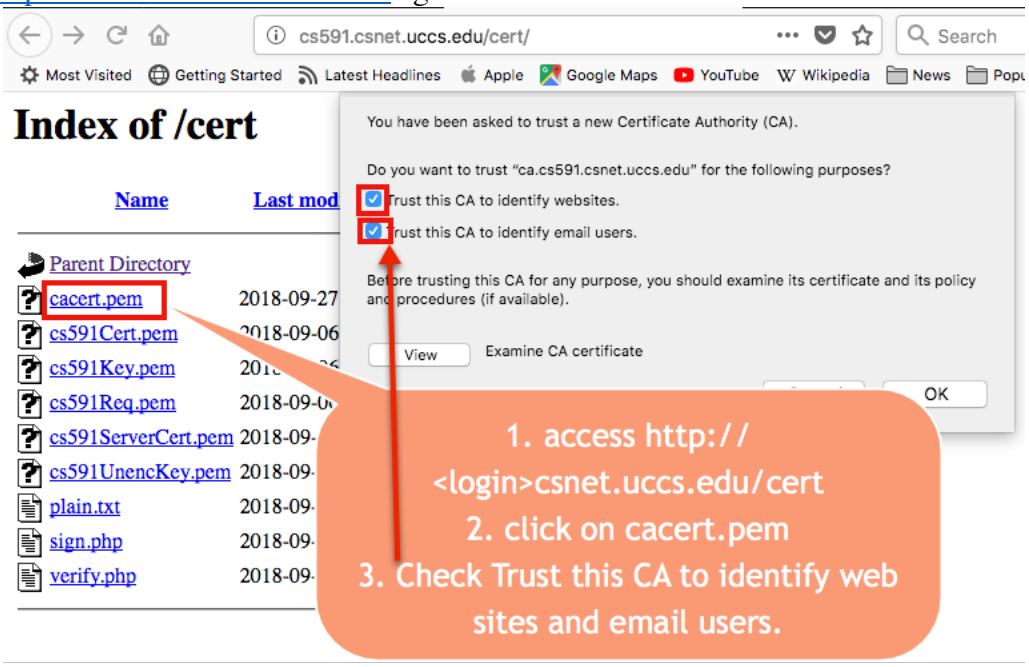


Figure 5.2. retrieve and trust the CA for identifying websites and email users.

Now visit the Trust CA list on Firefox Security Store to see indeed your CA is added.

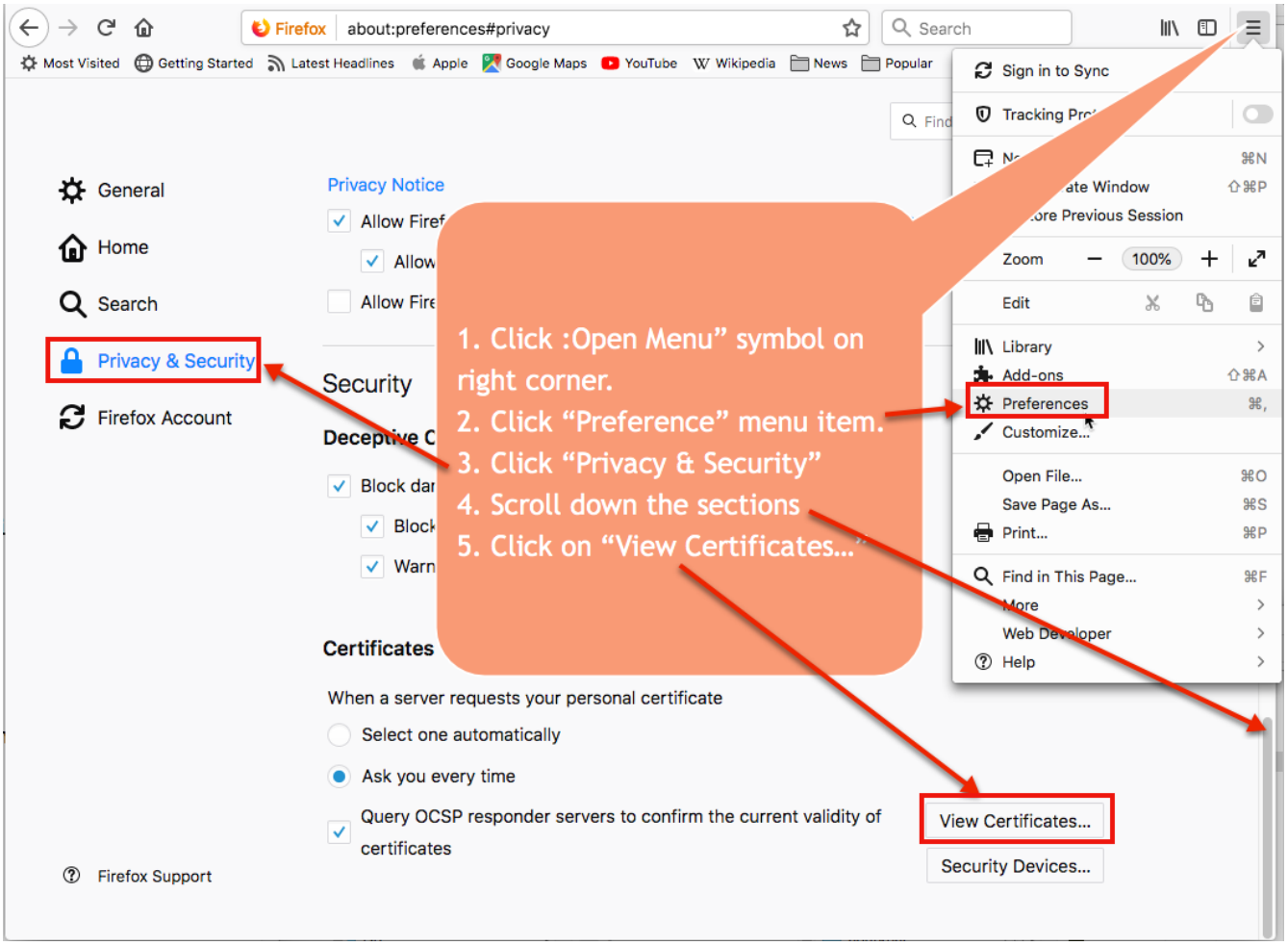


Figure 5.3. Click View Certificates button bring up Certificate Manager.

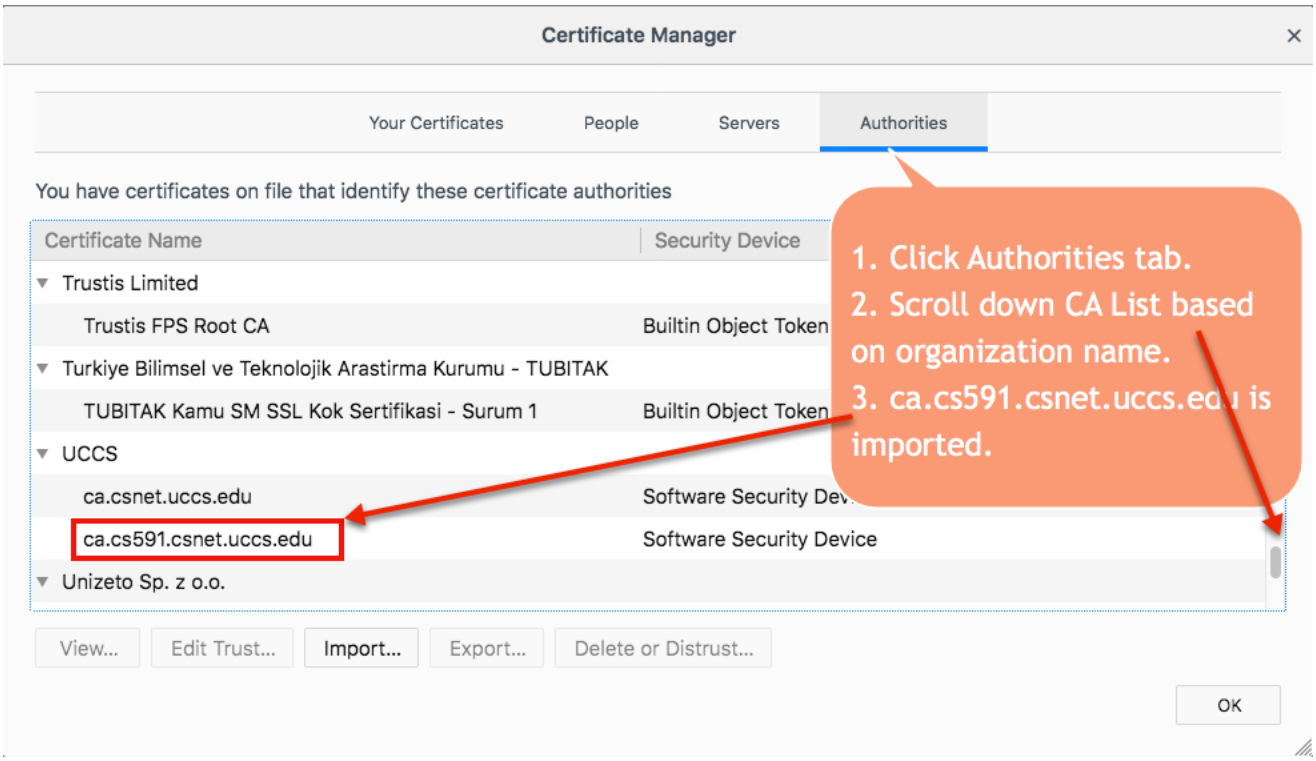


Figure 5.4. Our CA is included in Trust CA list.

Great! Our CA is imported to the local firefox browser's secure store.

Now shift reload https://cs591.csnet.uccs.edu/cert/ again (replace with your own web server name)

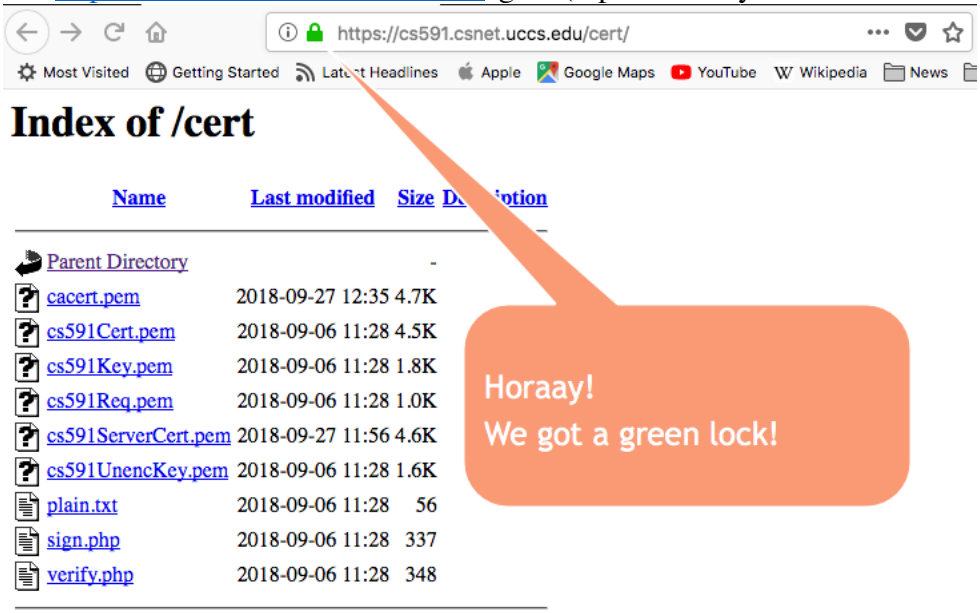


Figure 5.5. Successful Secure Web Page Access with Green Lock Symbol.

Hooray! we got a green lock symbol. The browser is happy with the webserver certificate! Sometimes the alert symbol stayed and you need to choose to remove the security exception explicitly before green symbol will appear!

Step 6. Mutual Authentication

Step 6a. Setup Apache with certain directory for certificate based mutual authentication

Here we show how to set up Apache to protect the secure directory by requiring users to present their client certificates.

- Add the following directives to /etc/httpd/conf/httpd.conf after line 381

```
SSLCACertificateFile /etc/pki/CA/cacert.pem
SSLCACertificatePath /etc/pki/CA
SSLVerifyClient none
<Directory /var/www/html/secure>
SSLVerifyClient require
SSLVerifyDepth 5

SSLOptions +FakeBasicAuth
SSLRequireSSL
AuthName "UCCS CS Department"
AuthType Basic
AuthUserFile /etc/httpd/conf/httpd.passwd
require valid-user
</Directory>
```

Note that this set of restriction should set up on the main httpd.conf to block plain http access to your secure directory. If you include it in /etc/httpd/conf.d/ssl.conf, you may allow hackers to access your secure directory through plain http! This shows the configuration process is tightly related to the security of a system.

- Add the following line to the /etc/httpd/conf/httpd.passwd

  ```
  /C=US/ST=Colorado/L=Colorado
  Springs/O=UCCS/OU=CS/CN=cchow/emailAddress=cchow@uccs.edu:xxj31ZMTZzkVA
  ```

  Note that the above should be all in one line. There should be a single space character separated Colorado and Springs.

  You should replace cchow with your own CN and cchow@uccs.edu with your own email address.
  The encrypted password xxj31ZMTZzkVA should not be changed. All entries in httpd.passwd should this exact the same fake password. No space after A.
  If you did not see httpd.passwd, create one
- The above client certificate verification using FakeBasicAuth option is implemented in line1836 of ssl_hook_UserCheck() of ssl_engine_kernel.c.
  http://cs.uccs.edu/~cs591/httpd-2.0.54/modules/ssl/ssl_engine_kernel.c
- Note that string "xxj31ZMTZzkVA" is used as the password in the user file and it is just the crypted variant of the word "password".

- Make sure to include all subject field of the client certificate. The example in SSL-how-to does not include /ST, /O, and /emailAddress fields.
- Create a secure directory under /var/www/html. With an index.html contains " <h1> This is a secure area only accessible via client certificate and subject field match the httpd.passwd list.</h1>
- Now let us restart httpd

  $ sudo service httpd restart
- Make sure /var/www/html/secure does not contain .htaccess file with apache directory that interfere with the client certificate based mutual authentication we set up here.

- To verify the normal user or hacker cannot access the secure directory, let us type in https://cs591.csnet.uccs.edu/secure
  - We see

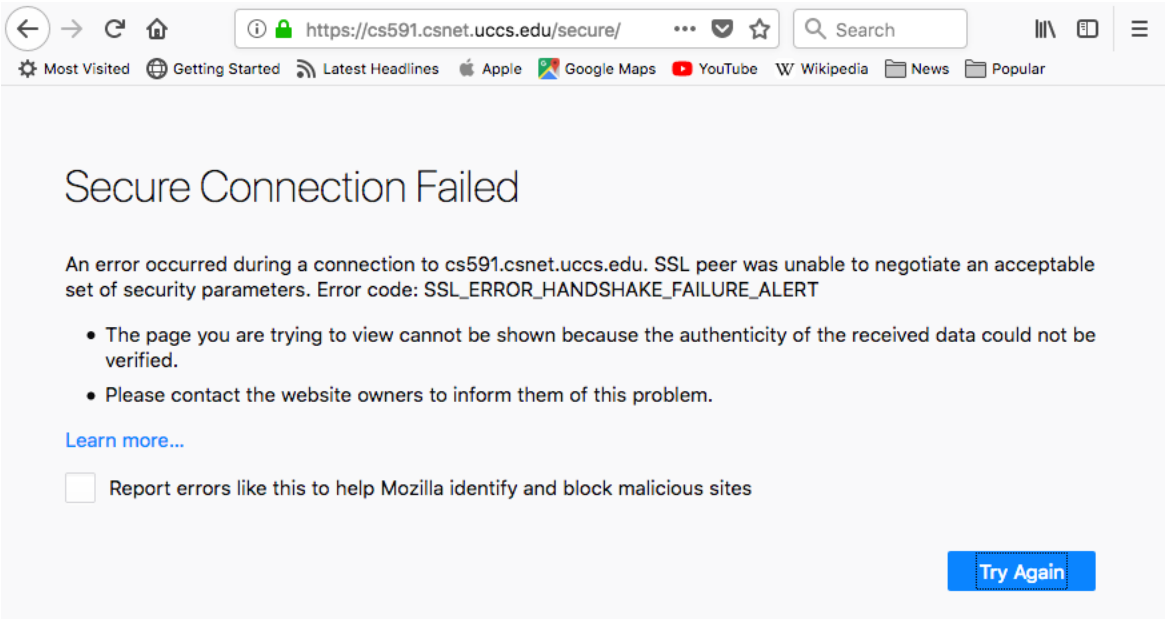


Figure 5.6. Access to secure web site failed.

- The apache error log for secure access shows the following

```
[root@cs591 secure]# tail -f /var/log/httpd/ssl_error_log
…
[Fri Sep 28 11:02:15.190131 2018] [autoindex:error] [pid 24197:tid 140204987877120] [client 128.198.169.164:50828] AH01276: Cannot serve directory /var/www/html/: No matching DirectoryIndex (index.html,index.php) found, and server-generated directory index forbidden by Options directive
[Fri Sep 28 11:29:42.548268 2018] [ssl:error] [pid 24483:tid 139968513533696] [client 128.198.169.164:57733] AH02261: Re-negotiation handshake failed
[Fri Sep 28 11:29:42.548386 2018] [ssl:error] [pid 24483:tid 139968513533696] SSL Library Error: error:1417C0C7:SSL routines:tls_process_client_certificate:peer did not return a certificate -- No CAs known to server for verification?
```

- We need to create and import the client certificate to the browser.

Step 6b. Set up Client browser to use the new client certificate.

Step 6b.1. First create a <login>Client .p12

```
$ mkdir client      # note that here we assume you are in your ec2-user homedirectory
$ cd /etc/pki/tls
```

- Use misc/CA -newreq and misc/CA -sign to create clientCert.pem and clientKey.pem. Similar to the creation of server certificate. Use your login name as Common Name. For CS591 class, your UCCS email address as emailAddress. For Coursera classes, use your Coursera login.

```
$ sudo misc/CA –newreq
$ sudo cp newreq.pem ~/client/clientReq.pem
$ sudo cp newkey.pem ~/client/clientKey.pem
```

- Normally we will submit the clientReq.pem to a CA that signs secure email certificate. Some like commodore will sign it for free. Here we submit to ourselves as a CA for signing. We leave the newreq.pem in /etc/pki/tls which read in by misc/CA -sign as CSR.

```
$ sudo misc/CA -sign
$ sudo mv newcert.pem ~/client/clientCert.pem
```

- Combine client private key and signed certificate as .p12 file using the following command. Since the netscape or firefox only accepts pkcs12 format. We use pkcs12 utility command of openssl to export the client certificate with the format.

```
openssl pkcs12 -export -in clientCert.pem -inkey clientKey.pem -out <yourLogin>Client.p12

$ cd ~/client
$ openssl pkcs12 -export -in clientCert.pem -inkey clientKey.pem -out chowClient.p12
Enter pass phrase for clientKey.pem: XXXXXXX
Enter Export Password: YYYYYYYY
Verifying - Enter Export Password: YYYYYYYY
```

- You will be asked to enter the pass phrase and export password. Export password will be asked when we installed on client browser.
- Note that here we are exporting both the private key and the client certificate, all in the same file protected by the export password. Make sure you carry/transmit them in a secure manner.

Step6b.2. Create cs591Client.p12 file for testing the secure web access.

Repeat the above process and create a client certificate with cs591 as CN and cs591@uccs.edu as email address field. It will allow us to check whether server reject the access when presented with an client certificate not in the httpd.passwd list.

Step6b.3. Import client.p12 files to your browser.

- Download password protected <login>Client.p12 and cs591Client.p12files to the local machine or the machine you used for mutual authentication for installation.
- Here are the steps for importing your client certificate to your browser.
    - On firebox browser, select Open Menu | Preference or just hit command-, on MAC. On Windows, select Open Menu | Options.

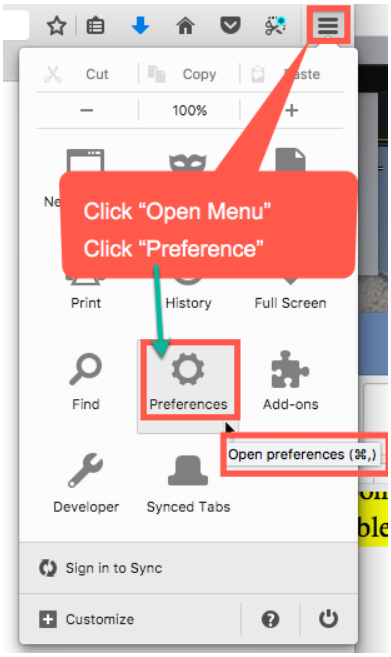


Figure 6a . On Mac, select Preferences

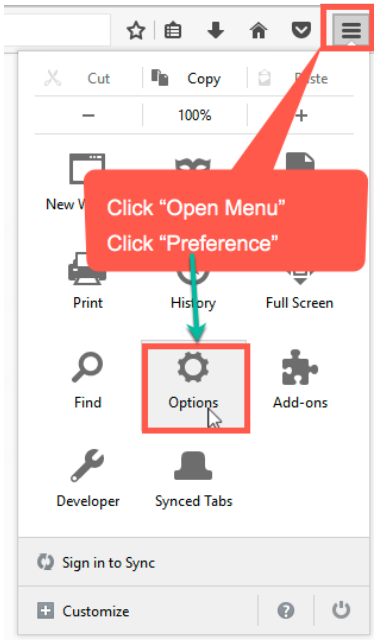


Figure 6b. On Windows, select Options.

- Bring up Preference dialog window. Select Advanced settings, select certificates tab, make sure “Ask you every time” is set, then click on View Certificate.

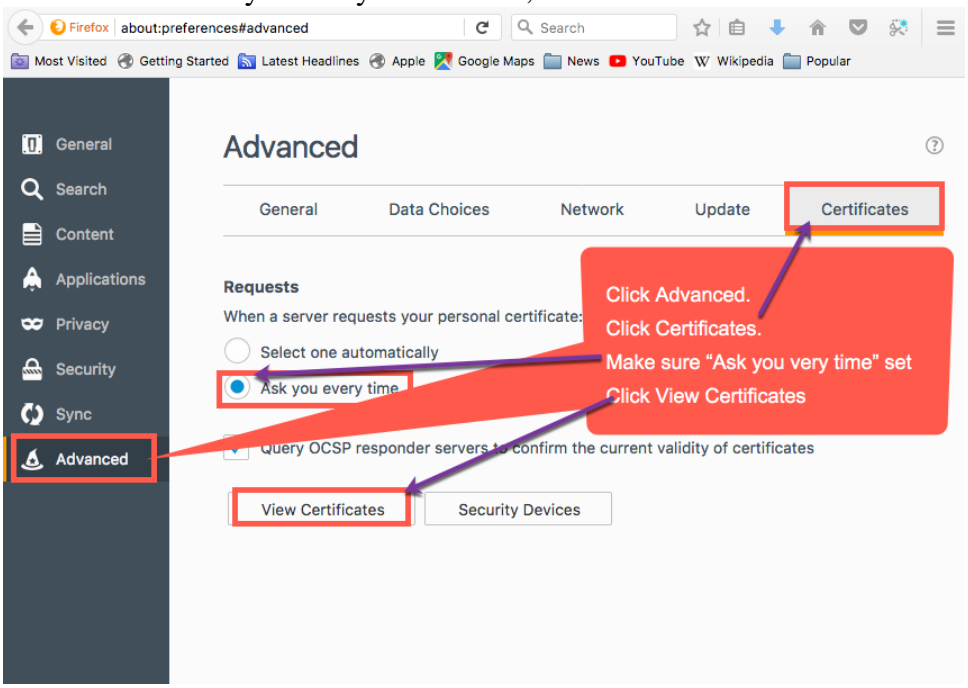


Figure 6c. Bring up Certificate Manager.

- Click Your Certificates tab (since we are importing personal client certificates. Click Import button.

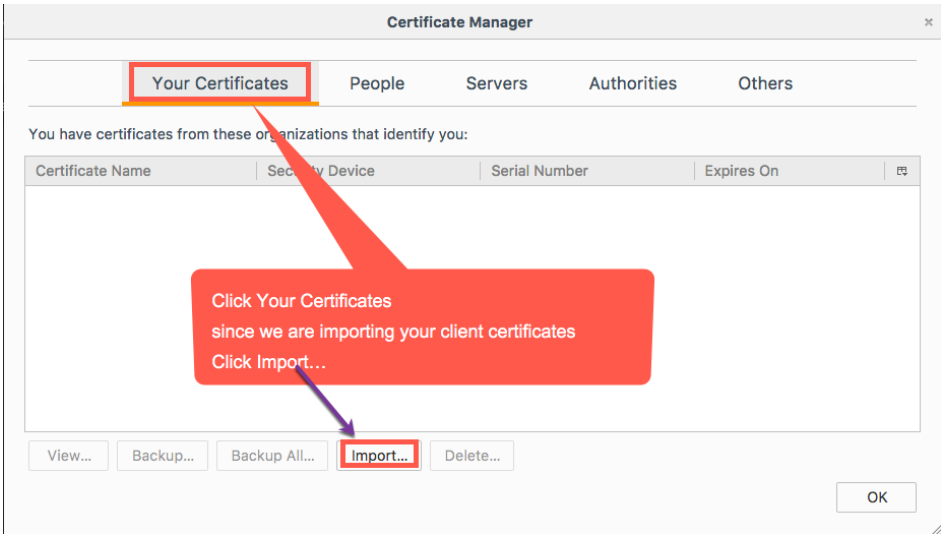
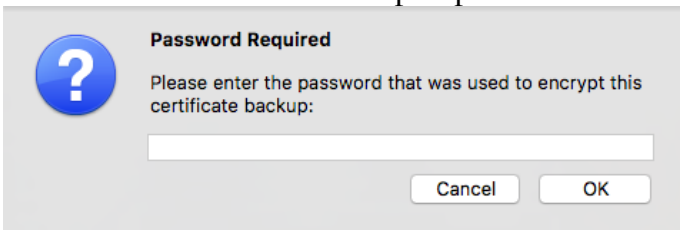


Figure 6d. Choose Your Certificates category to import your client certificates.

You will be ask to enter the export password that use to protect your client.p12 file.



Then you will see the client certificate imported.

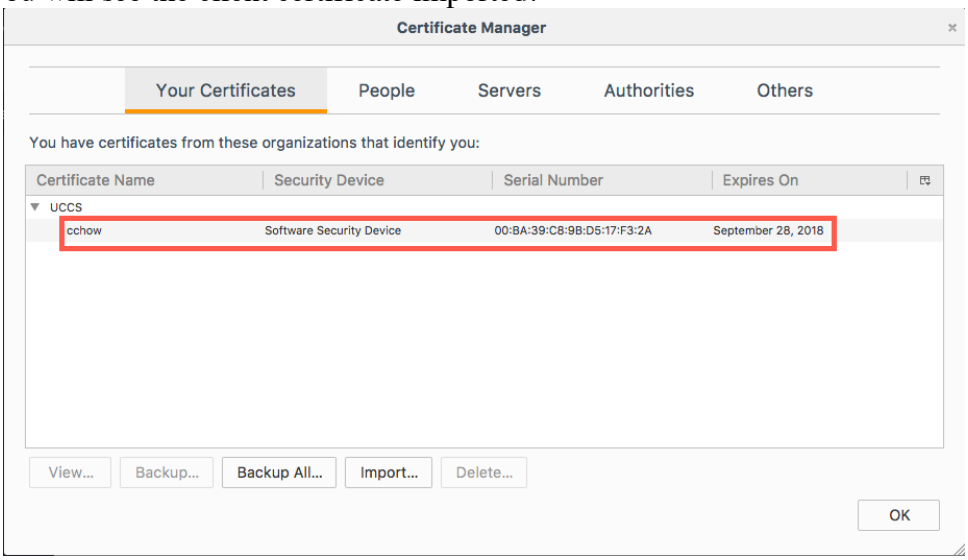


Figure 6e. Certificate Manager displays the imported client cdertificate.

- Click import again and chose cs591Client.p12 file in your download directory. Repeat the above process to import cs591 client certificate.

Step 6c. Verify Client Certificate Access.

- Type https://<yourInstanceDNSname>/secure/
- You should see a dialog box popup

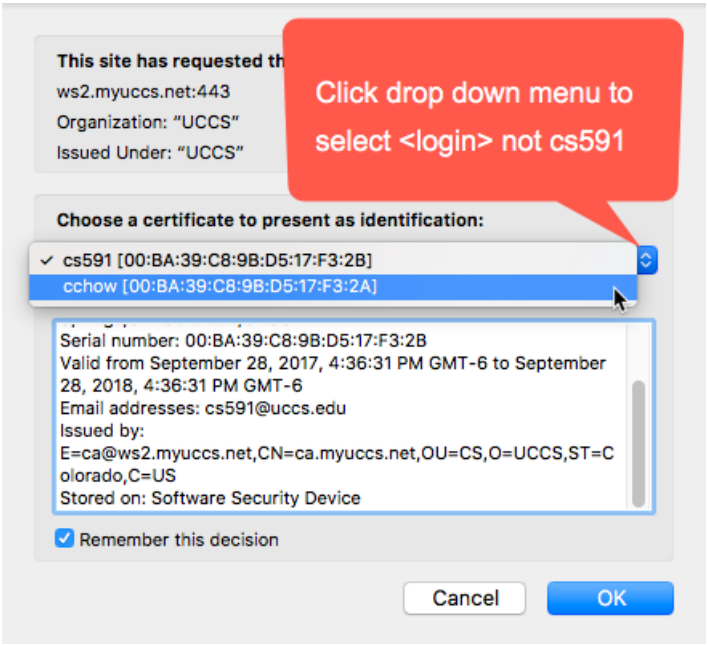


Figure 6f. Select right client certificate to access

- After select <login> client certificate, you should see

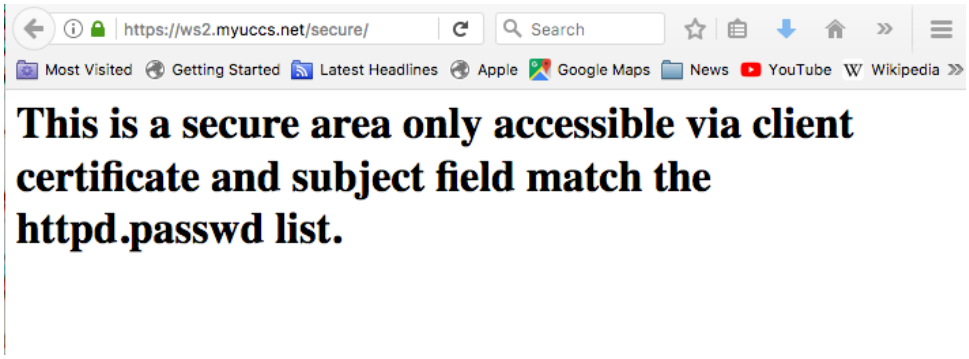


Figure 6g. Successful access of secure directory content.

- A successful mutual client-server authentication!
- Let us now test the access with cs591 client certificate.
- Since your browser will remember the last client certificate you used, you need to quit your browser to start testing cs591 client certificate access. Then
- Type https://<yourInstanceDNSname>/secure/
- You will be prompted for selecting client certificates. Choose cs591 this time.
- You will get the following popup window to ask for username and password. At this point, you are refused to access the secure directory by the web server. No matter what credential you enter, you cannot access secure directory.

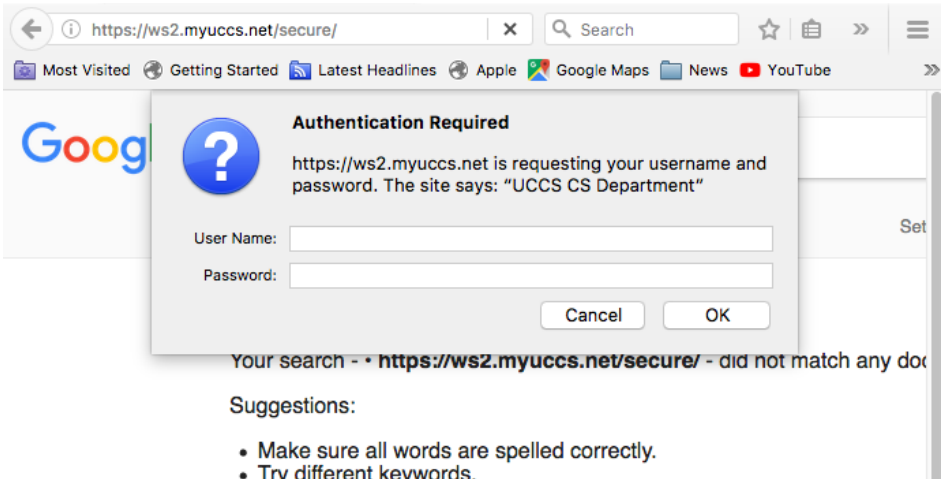


Figure 6h. Unsuccessful access using other client certificate.

- This completes the verification that the client certificate even though signed by the same CA will not be able to access the secure directory, unless their subject field name is included in the httpd.password file. Another layer of protection!