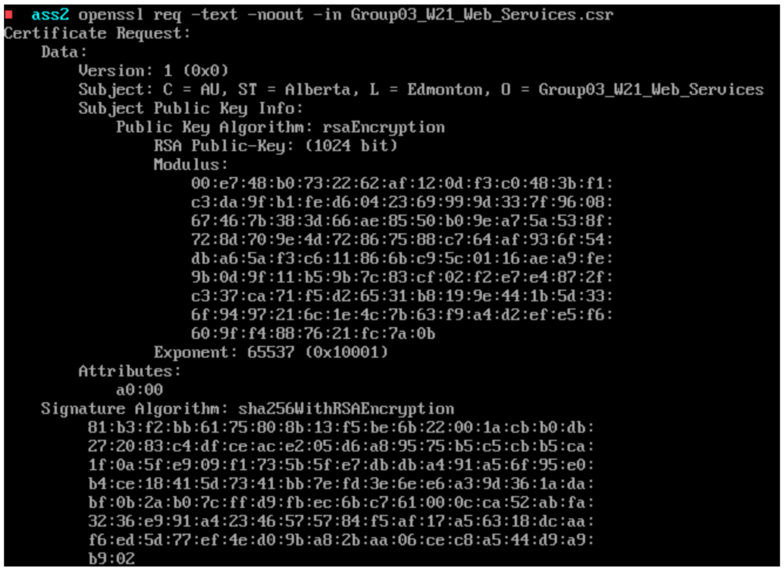
Part 1

1. We had to generate our private key using openssl
    - $ openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:1024 -out CA_private_key.key
    - This is what will be used to sign our root CA certificate (hence the name self signed)
    - This is needed inorder to create our self signed root certificate
2. Encrypt the private key with the password “badboybooboo”
    - $ openssl genpkey -algorithm RSA -out CA_private_key.key -aes-128-cbc -pass pass:badboybooboo
3. Creating self signed(with private key generated) CA certificate
    - $ openssl req -key CA_private_key.key -x509 -new -days 100 -out Group03_W21_CA.crt
    - This will prompt us to fill in some details after this is completed a .crt file will be created which will act as our certificate authority to sign certificates.
    - This .crt is needed inorder to be used to verify other certificates which claims to be signed by us actually are. (we are the trusted CA)
    - To view the created certificate:
        - $ openssl x509 -text -noout -in Group03_W21_CA.crt

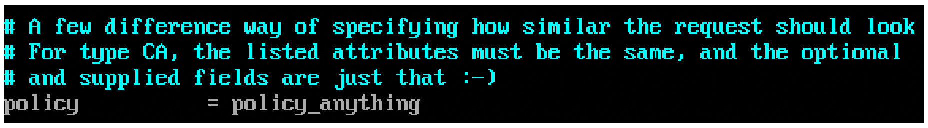
Part 2

- We generate a certificate request
    - $ openssl req -newkey rsa:1024 -keyout CSR_private_key.key -out Group03_W21_Web_Services.csr
    - This will prompt us to enter a password for our new private key that will be generated(“badboybooboo”), then ask us to fill in some details for the certificate request which is the output of this command.
    - To view the created certificate request:
        - $ openssl req -text -noout -in Group03_W21_web_Services.csr
    - The .key file here is needed to create the .csr file
    - The .csr is used “to ask for a signing from a CA”, this will be used to create a .crt file (signed by our CA) which will be used when setting up our https server.

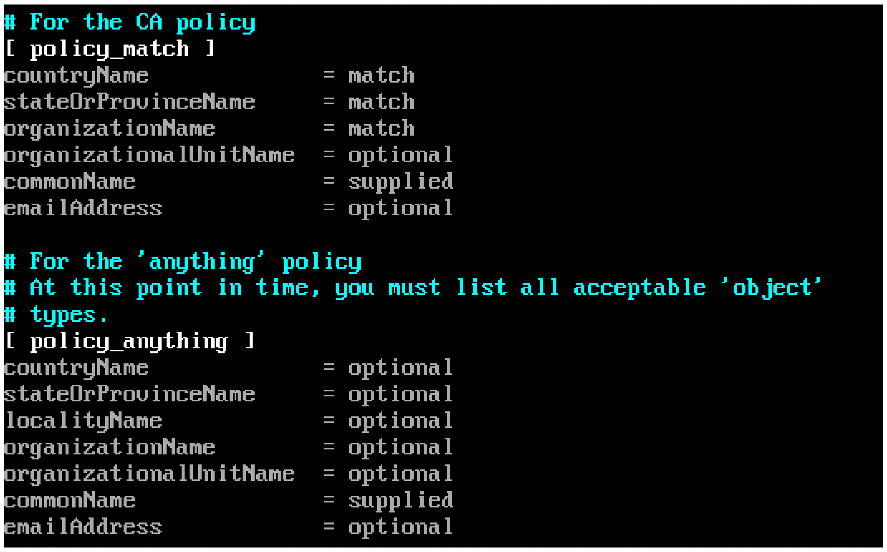
- 

```
■  ass2 openssl req -text -noout -in Group03_W21_Web_Services.csr
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: C = AU, ST = Alberta, L = Edmonton, O = Group03_W21_Web_Services
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (1024 bit)
                Modulus:
                    00:e7:48:b0:73:22:62:af:12:0d:f3:c0:48:3b:f1:
                    c3:da:9f:b1:fe:d6:04:23:69:99:9d:33:7f:96:08:
                    67:46:7b:38:3d:66:ae:85:50:b0:9e:a7:5a:53:8f:
                    72:8d:70:9e:4d:72:86:75:88:c7:64:af:93:6f:54:
                    db:a6:5a:f3:c6:11:86:6b:c9:5c:01:16:ae:a9:fe:
                    9b:0d:9f:11:b5:9b:7c:83:cf:02:f2:e7:e4:87:2f:
                    c3:37:ca:71:f5:d2:65:31:b8:19:9e:44:1b:5d:33:
                    6f:94:97:21:6c:1e:4c:7b:63:f9:a4:d2:ef:e5:f6:
                    60:9f:f4:88:76:21:fc:7a:0b
                Exponent: 65537 (0x10001)
        Attributes:
            a0:00
    Signature Algorithm: sha256WithRSAEncryption
         81:b3:f2:bb:61:75:80:8b:13:f5:be:6b:22:00:1a:cb:b0:db:
         27:20:83:c4:df:ce:ac:e2:05:d6:a8:95:75:b5:c5:cb:b5:ca:
         1f:0a:5f:e9:09:f1:73:5b:5f:e7:db:db:a4:91:a5:6f:95:e0:
         b4:ce:18:41:5d:73:41:bb:7e:fd:3e:6e:e6:a3:9d:36:1a:da:
         bf:0b:2a:b0:7c:ff:d9:fb:ec:6b:c7:61:00:0c:ca:52:ab:fa:
         32:36:e9:91:a4:23:46:57:57:84:f5:af:17:a5:63:18:dc:aa:
         f6:ed:5d:77:ef:4e:d0:9b:a8:2b:aa:06:ce:c8:a5:44:d9:a9:
         b9:02
```

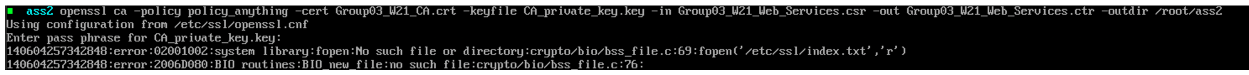
Part 3

1. First we set the policy to “policy_anything”:
    - nvim /etc/ssl/openssl.cnf
    - 

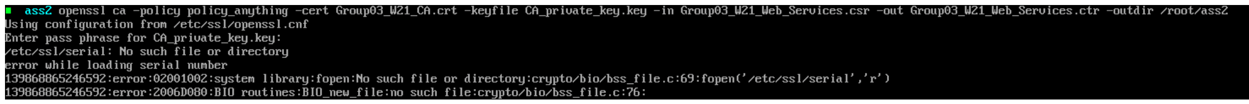
```
# A few difference way of specifying how similar the request should look
# For type CA, the listed attributes must be the same, and the optional
# and supplied fields are just that :-)
policy          = policy_anything
```

    - This will set the default policy to policy anything(incase the policy was omitted when typing the openssl command)
    - If this was not set to policy anything, then it would have been policy match. This states that the specified fields must match in the CA and certificate request which would have been a problem because our organisation name in both files are different.
    - 

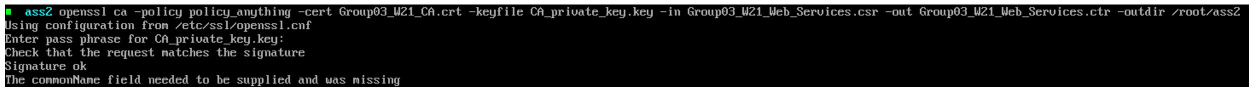
```
# For the CA policy
[ policy_match ]
countryName             = match
stateOrProvinceName     = match
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

# For the 'anything' policy
# At this point in time, you must list all acceptable 'object'
# types.
[ policy_anything ]
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
```

2. Then we sign the certificate request
    - $ openssl ca -policy policy_anything -cert Group03_W21_CA.crt -keyfile CA_private_key.key -in Group03_W21_Web_Services.csr -out Group03_W21_web_Services.crt -outdir /root/ass2
    - When i ran this command i got this error
    - 

```
■ ass2 openssl ca -policy policy_anything -cert Group03_W21_CA.crt -keyfile CA_private_key.key -in Group03_W21_Web_Services.csr -out Group03_W21_Web_Services.ctr -outdir /root/ass2
Using configuration from /etc/ssl/openssl.cnf
Enter pass phrase for CA_private_key.key:
140604257342848:error:02001002:system library:fopen:No such file or directory:crypto/bio/bss_file.c:69:fopen('/etc/ssl/index.txt','r')
140604257342848:error:2006D080:BIO routines:BIO_new_file:no such file:crypto/bio/bss_file.c:76:
```

    - **NOTE: the last file i meant to end with .crt not .ctr this was later corrected**
    - So i had to create an “index.txt” file(it will be initially empty) which is the Database text file file needed in signing this certificate request. This is the database of all signings that have occurred.
        - Went to the required directory: $ cd /etc/ssl
        - $ nvim index.txt
    - When i reran the command i got another error
    - 

```
■ ass2 openssl ca -policy policy_anything -cert Group03_W21_CA.crt -keyfile CA_private_key.key -in Group03_W21_Web_Services.csr -out Group03_W21_Web_Services.ctr -outdir /root/ass2
Using configuration from /etc/ssl/openssl.cnf
Enter pass phrase for CA_private_key.key:
/etc/ssl/serial: No such file or directory
error while loading serial number
139868865246592:error:02001002:system library:fopen:No such file or directory:crypto/bio/bss_file.c:69:fopen('/etc/ssl/serial','r')
139868865246592:error:2006D080:BIO routines:BIO_new_file:no such file:crypto/bio/bss_file.c:76:
```

    - So i had to create a “serial” and added the line “01”, this is a text file needed to indicate the next serial number to use in hex.
    - I then ran into this problem
    - 

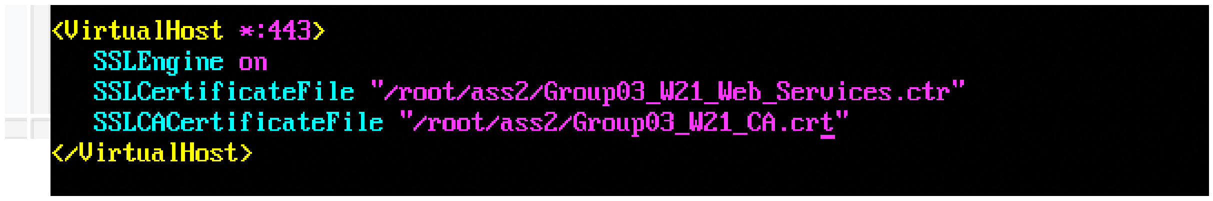
```
■ ass2 openssl ca -policy policy_anything -cert Group03_W21_CA.crt -keyfile CA_private_key.key -in Group03_W21_Web_Services.csr -out Group03_W21_Web_Services.ctr -outdir /root/ass2
Using configuration from /etc/ssl/openssl.cnf
Enter pass phrase for CA_private_key.key:
Check that the request matches the signature
Signature ok
The commonName field needed to be supplied and was missing
```

    - After rerunning the initial command, certificate signing was successful

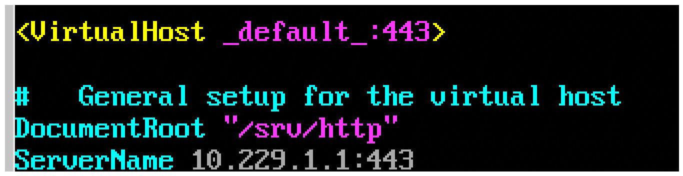
<u>Part 4</u>

1. Edit the file httpd.conf
    - Go to where the file is located: $ cd /etc/httpd/conf
    - $ nvim httpd.conf
    - Uncomment the following lines:
    -

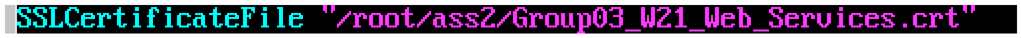
```
LoadModule ssl_module modules/mod_ssl.so
LoadModule socache_shmcb_module modules/mod_socache_shmcb.so
Include conf/extra/httpd-ssl.conf
```

    - Added the following lines
    - 

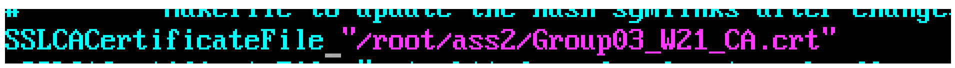
```
<VirtualHost *:443>
   SSLEngine on
   SSLCertificateFile "/root/ass2/Group03_W21_Web_Services.ctr"
   SSLCACertificateFile "/root/ass2/Group03_W21_CA.crt"
</VirtualHost>
```

- **NOTE: the first file i meant to end with .crt not .ctr this was later corrected**
- Edited the httpd-ssl.conf file located in /etc/httpd/conf/extra
- Added the following lines
- 

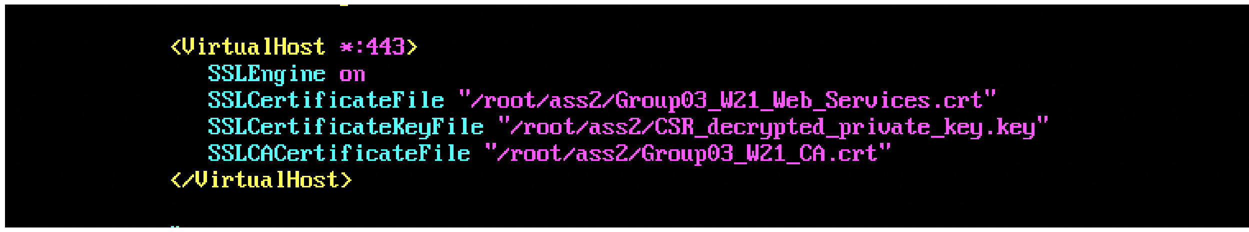
```
<VirtualHost _default_:443>

#   General setup for the virtual host
DocumentRoot "/srv/http"
ServerName 10.229.1.1:443
```



```
SSLCertificateFile "/root/ass2/Group03_W21_Web_Services.crt"
```

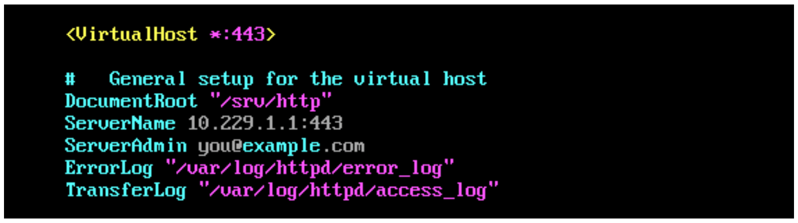


```
SSLCACertificateFile "/root/ass2/Group03_W21_CA.crt"
```

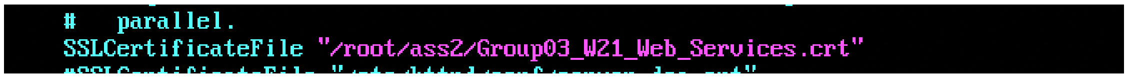
- Before moving on into the next step i ran : $ apachectl configtest
  - This is to test if there any errors in my config files
- When i checked the status of my httpd, it wasn't starting, after going through the log files in /var/log/httpd/error_log. I figured out i had not supplied the ssl certificate private key(also since i encrypted it, i had to supply the httpd server with the decrypted version unless it won't be able to establish the connection because i can’t supply the passphrase at that moment)
  - To decrypt the key i ran $ openssl rsa -in CSR_private_key.key -out CSR_decrypted_private_key.key
  - This will output the decrypted private key of the ssl certificate
- So i corrected the httpf.conf file located in /etc/httpd/conf
- 

```
<VirtualHost *:443>
   SSLEngine on
   SSLCertificateFile "/root/ass2/Group03_W21_Web_Services.crt"
   SSLCertificateKeyFile "/root/ass2/CSR_decrypted_private_key.key"
   SSLCACertificateFile "/root/ass2/Group03_W21_CA.crt"
</VirtualHost>
```

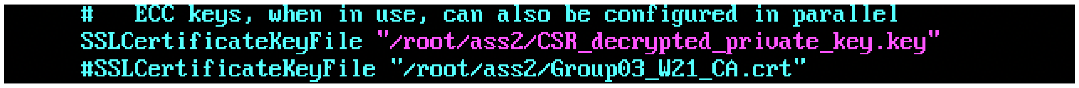
- Also corrected the httpd-ssl.conf file located in /etc/httpd/conf/extra



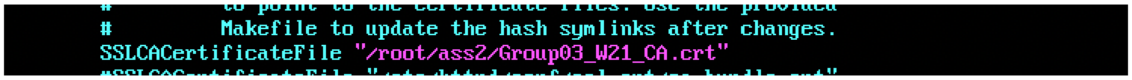
```
<VirtualHost *:443>

#   General setup for the virtual host
DocumentRoot "/srv/http"
ServerName 10.229.1.1:443
ServerAdmin you@example.com
ErrorLog "/var/log/httpd/error_log"
TransferLog "/var/log/httpd/access_log"
```



```
#   parallel.
SSLCertificateFile "/root/ass2/Group03_W21_Web_Services.crt"
```



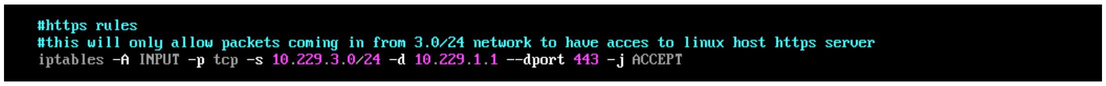
```
 #   ECC keys, when in use, can also be configured in parallel
 SSLCertificateKeyFile "/root/ass2/CSR_decrypted_private_key.key"
 #SSLCertificateKeyFile "/root/ass2/Group03_W21_CA.crt"
```



```
  #         Makefile to update the hash symlinks after changes.
  SSLCACertificateFile "/root/ass2/Group03_W21_CA.crt"
```

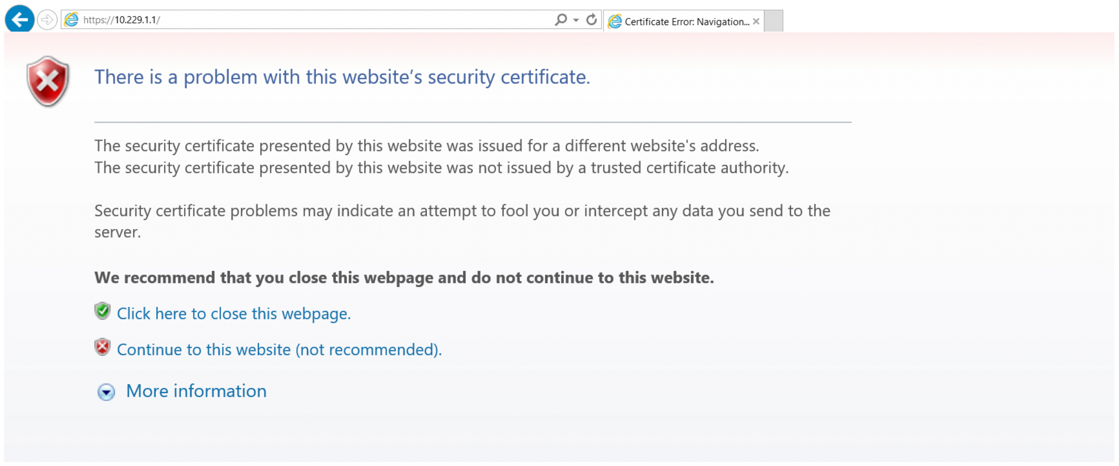
Part 5

1. First of all since we are using a firewall we had to enable hosts to have access to https server we just configured. We added the following rule to our iptables.



```
#https rules
#this will only allow packets coming in from 3.0/24 network to have acces to linux host https server
iptables -A INPUT -p tcp -s 10.229.3.0/24 -d 10.229.1.1 --dport 443 -j ACCEPT
```

2. When we tried to connect to the linux https server on the windows initially



https://10.229.1.1/

Certificate Error: Navigation...

There is a problem with this website's security certificate.

The security certificate presented by this website was issued for a different website's address.
The security certificate presented by this website was not issued by a trusted certificate authority.

Security certificate problems may indicate an attempt to fool you or intercept any data you send to the server.

We recommend that you close this webpage and do not continue to this website.

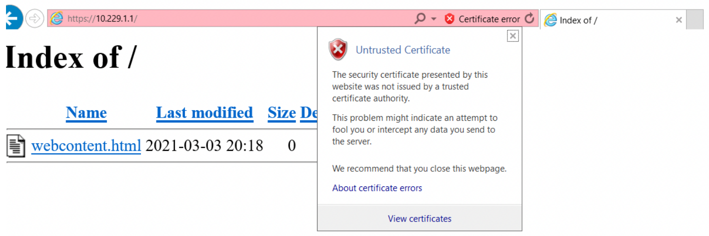
Click here to close this webpage.

Continue to this website (not recommended).

More information

3. How we installed our root CA so internet explorer trusts it
   - First of all i sent my root CA certificate from my linux to my mac then to my windows using shared folders i had set up earlier
   - Try connect to the webserver with internet explorer, and when it shows that warning message, click continue to this website and log in normally.

- Click the certificate error



https://10.229.1.1/ Certificate error Index of /

Index of /

Name Last modified Size De

webcontent.html 2021-03-03 20:18 0

Untrusted Certificate

The security certificate presented by this website was not issued by a trusted certificate authority.

This problem might indicate an attempt to fool you or intercept any data you send to the server.

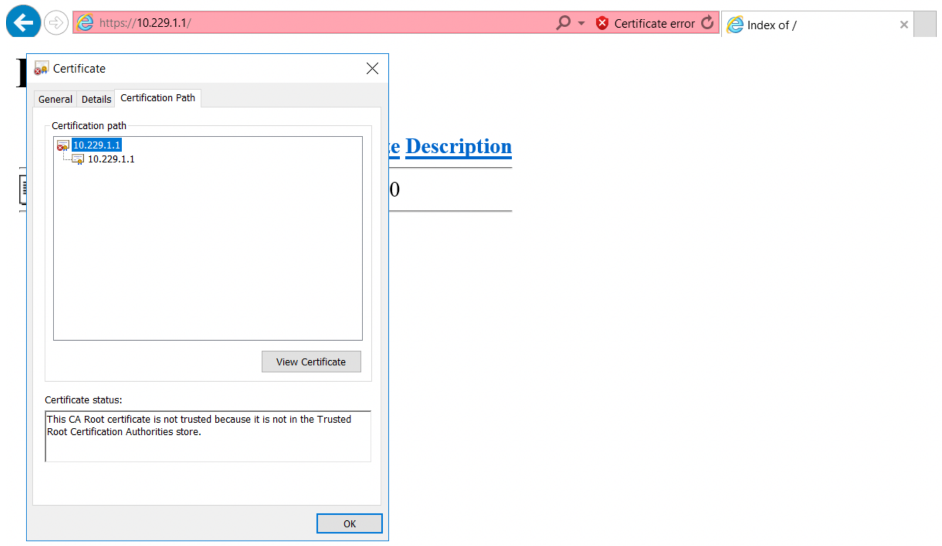
We recommend that you close this webpage.

About certificate errors

View certificates

  -

- Then click view certificates, click certificate path then view certificate



https://10.229.1.1/ Certificate error Index of /

Certificate

General Details Certification Path

Certification path

10.229.1.1
10.229.1.1

View Certificate

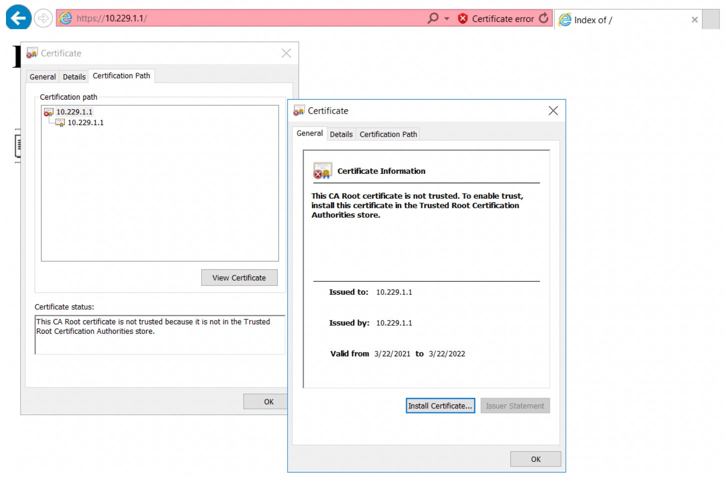
Certificate status:

This CA Root certificate is not trusted because it is not in the Trusted Root Certification Authorities store.

OK

  -

- Then click View Certificate



Certificate

Certificate Information

This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

Issued to: 10.229.1.1

Issued by: 10.229.1.1

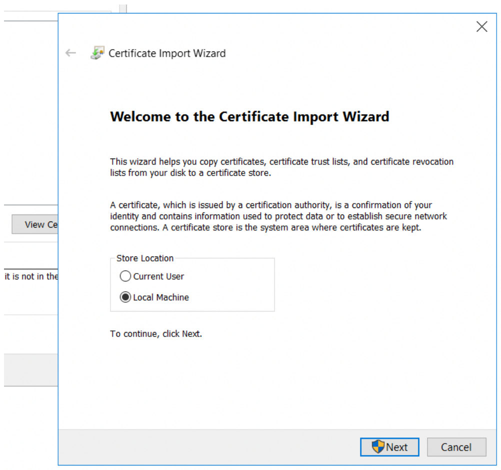
Valid from 3/22/2021 to 3/22/2022

Install Certificate... Issuer Statement

OK

  -

- Then click install certificate



Certificate Import Wizard

Welcome to the Certificate Import Wizard

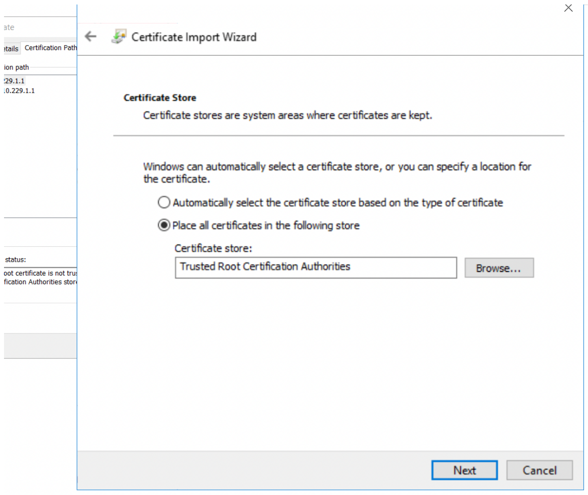
This wizard helps you copy certificates, certificate trust lists, and certificate revocation lists from your disk to a certificate store.

A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

Store Location
- ○ Current User
- ● Local Machine

To continue, click Next.

- Click local machine and next. Then click place all certificates, browse, then select Trusted Root certificate Authorities



Certificate Import Wizard

Certificate Store

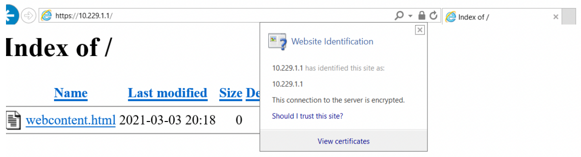
Certificate stores are system areas where certificates are kept.

Windows can automatically select a certificate store, or you can specify a location for the certificate.

- ○ Automatically select the certificate store based on the type of certificate
- ● Place all certificates in the following store

Certificate store: Trusted Root Certification Authorities

- Click next, then select where you stored the certificate
- Then click install/finish
- After this process restart your internet explorer and try to connect to the server. A warning should not pop up and you can verify it is secured by clicking on the lock sign in the address bar



https://10.229.1.1/

Index of /

Name	Last modified	Size	De
webcontent.html	2021-03-03 20:18	0	

Website Identification

10.229.1.1 has identified this site as:

10.229.1.1

This connection to the server is encrypted.

Part 7

- First we go into the directory of the .jar file(which is basically a bundled applet)
    - C:\Program Files\Java\jdk1.8.0\bin\
    - We need to be in this directory because it will contain all the files needed and tools to carry out the other steps below.
- First we need to generate an RSA keypair using keytool
    - $ keytool -genkey -keyalg rsa -alias MyCert
    - This will create and RSA key pair and a certificate, this will prompt us to enter some certificate details(and a password for keystore)
- Then we generate a certificate signing request using keytool
    - $ keytool -certreq -alias MyCert
    - This will generate the certificate signing request which will be signed by me (my CA)
- Then we import our CA into java keystore, this is needed so we can use it to sign the certificate request.
    - $ keytool -import -alias MyCert -file Group_W21_CA.crt
- Then we can simply use jarsigner to sign our certificate
    - $ jarsigner C:\TestApplet.jar MyCert
    - This will prompt us to enter our keystore password
    - Then our .jar file will be successfully signed by my CA which i already linked with MyCert previously.
    - We can verify or .jar file
        - $ jarsigner -verify -verbose -certs

<u>Differences in signing .jar files and web server certificates</u>

- One of the main differences are the tools used.
- The commands used for signing .jar files are simpler.
- The process for signing the .jar files is shorter
    - One line does everything
    - But to sign the web server certificates we have to create some new files, and change some configuration files

<u>References</u>

During the completion of this assignment we used the following sites as guides to complete various parts of the assignment:

- https://wiki.archlinux.org/index.php/OpenSSL#Generate_an_RSA_private_key
- https://linux.die.net/man/1/req
- https://linux.die.net/man/1/genpkey
- https://www.openssl.org/docs/man1.0.2/man1/openssl-ca.html
- https://stackoverflow.com/questions/21297139/how-do-you-sign-a-certificate-signing-request-with-your-certification-authority
- https://www.ssl.com/how-to/manually-generate-a-certificate-signing-request-csr-using-openssl/
- https://docs.oracle.com/javase/7/docs/technotes/guides/jweb/security/rsa_signing.html