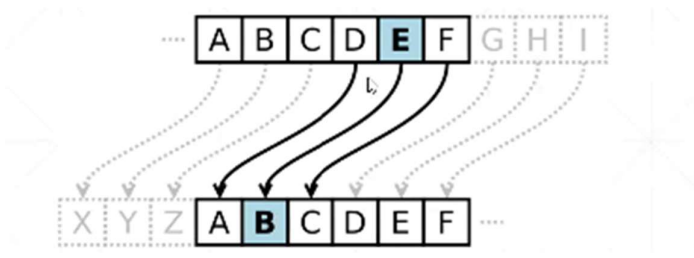


CA

Sertifikat nədir? Bildiyimiz sertifikat üzərində kimin verdiyini, kimə verdiyini, expire müddəti, imza kimi datalar olur. Digital sertifikatlar da daha çox məlumat olur. Orda serial number, CRL, hansı hash alqoritminin istifadə olunduğu kimi məlumatlar da olur. Digital sertifikatlar üzərində Common Name (CN- ən vacibidə budur), Root CA (belə düşünəkki Ingress Academynin bir imzası var), Intermediate Certificate (bu o işə yarıyır ki, hər zaman root sertifikat imzasına ehtiyac olmadan mən Intermediate sertifikatla imzalarımı doğrulayacam. Belə düşünəkki Ruslan müəllim Murad müəllimə imza atma səlahiyyəti verir, biz buna Intermediate imza deyirik. Ruslan müəllimin yeni Root hər zaman gəlib imza atmasına ehtiyac yoxdur. Çünki artıq Murad müəllim var yeni Intermediate). Intermediate imza demək olarki Root imzayla eyni səlahiyyətdədir. Intermediate sertifikat bir neçə dəfə ola bilər. Intermediate sertifikatlarında bir validity müddəti olur. CRL (Revoke list) adlanır. Sertifikatı geri çağırmaq deməkdir. Belə ki sertifikatı revoke etdikdən sonra user sertifikatın revoke olduğunu anlaması üçün CRL də istifadə olunur. Bizim bildiyimiz sertifikatlarla digital sertifikatlar arasındakı məntiq demək olar ki, eynidir.

Sertifikat nə üçündür? IT dünyasında sertifikat bir-birini tanımaq üçün var. Sertifikatın istifadəsi çox böyükdür. Mən sənə güvənirəm demək üçün var. Sertifikat sayəsində danışdığımız serverin düzgün olduğuna əmin oluruq. IP, VPN, SSH, Server, Document, Email, Device, Client sertifikatlar var.

Encrypting/Decrypting nədir? Məsələn salam yazdıqdan, sonra encryption gedir, və bu mənası alır. Buna qarşı dilində nümunə gətirmək olar. Bu alqoritmədən ilk dəfə istifadə edən Yuli Sezar olub.



Bunlar hashing sayılır. Encrypting/Decrypting dinamikdir və açarlardan, keylərdən istifadə olunur. Hash isə statikdir. Biz bir rəqəmini hash etsən hər yerdə eyni nəticə olacaq.

TLS/SSL nədir? Protokoldur. TTL və SSL eyni şeydir, sadəcə artıq SSL dəstəklənir. Həm encryption, həm də authentication edir. Sertifikatda yalnız encryption yox, bizə aydın olmayan çox məsələlər gəlir. TLS handshake zamanı 4 proses gəlir:

- 1) Client və Server qərar verir ki onlar hansı TLS versiyasından istifadə edəcək.
- 2) Cipher Suites
- 3) Key Exchange
- 4) Data Transmission

Sertifikat necə hazırlanır? Hər şeydən öncə bizə Private Key lazımdır. Hansı ki bu bizdə gizli olaraq qalır. Bunu heç kim öyrənməməlidir. Daha sonra CSR qurulur. Burada Public

göndərəcəyimiz məlumat yerləşir. CN(ingressbank.az), OU, ölkə(2 hərf olmalıdır), şəhər və s. Root CA qurarkən CSR-a ehtiyac yoxdur. Çünki Root özü imza atır, başqasından imza almır.



- 1.Private key
- 2.CSR
3. send CSR to Digicert
- 4
 - 4.1 Public url
 - 4.2 public dns
 - 4.3 mail
- 5.Certificat (public)

Bütün bunlar PKI Infrastructure-da adlanır.

Install openssl

yum install openssl mod_ssl -y

```
root@localhost:~# yum install openssl mod_ssl -y
Last metadata expiration check: 0:00:06 ago on Sun 23 Feb 2025 01:41:05 AM +04.
Package openssl-1:3.2.2-16.el10.x86_64 is already installed.
Dependencies resolved.
=====
Package                        Architecture      Version           Repository        Size
=====
Installing:
mod_ssl                        x86_64            1:2.4.63-1.el10  appstream         111 k
Installing dependencies:
apr                            x86_64            1.7.5-2.el10     appstream         129 k
apr-util                       x86_64            1.6.3-21.el10    appstream         99 k
apr-util-ldap                  x86_64            1.6.3-21.el10    appstream         15 k
httpd-core                     x86_64            2.4.63-1.el10    appstream         1.5 M
httpd-filesystem               noarch            2.4.63-1.el10    appstream         14 k
httpd-tools                    x86_64            2.4.63-1.el10    appstream         83 k
Installing weak dependencies:
apr-util-openssl               x86_64            1.6.3-21.el10    appstream         17 k
Transaction Summary
=====
Install 8 Packages

Total download size: 2.0 M
Installed size: 5.7 M
Downloading Packages:
```

Create required directory

```
mkdir /CA_SERVER
```

```
cd /CA_SERVER
```

```
mkdir conf certs crl newcerts private intermediate csr
```

```
root@localhost:/CA_SERVER# tree
.
├── certs
├── conf
├── crl
├── csr
├── intermediate
├── newcerts
└── private

8 directories, 0 files
```

Edit openssl.cnf

```
cp /etc/pki/tls/openssl.cnf .
```

```
vim openssl.cnf
```

```
dir = /CA_SERVER
```

```
certificate = $dir/certs/ca.cert.pem (Sertifikatların sonluğu pem-dir. Sonluqlar  
key, cert,pfx, pkcs12, pkcs11 ola bilər.)
```

```
private_key = $dir/private/ca.key.pem
```

```
[ CA_default ]
dir                = /CA_SERVER          # Where everything is kept
certs              = $dir/certs          # Where the issued certs are kept
crl_dir            = $dir/crl            # Where the issued crl are kept
database           = $dir/index.txt      # database index file.
#unique_subject    = no                  # Set to 'no' to allow creation of
# several certs with same subject.
new_certs_dir      = $dir/newcerts       # default place for new certs.
certificate        = $dir/certs/ca.cert.pem # The CA certificate
serial             = $dir/serial          # The current serial number
crlnumber           = $dir/crlnumber      # the current crl number
crl                = $dir/crl.pem         # must be commented out to leave a V1 CRL
private_key        = $dir/private/ca.key.pem # The private key
x509_extensions    = usr_cert            # The extensions to add to the cert
```

[v3_ca]

subjectKeyIdentifier = hash

authorityKeyIdentifier = keyid:always,issuer

basicConstraints = critical, CA:true

keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[v3_intermediate_ca]

subjectKeyIdentifier = hash

authorityKeyIdentifier = keyid:always,issuer

basicConstraints = critical, CA:true, pathlen:0

keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[usr_cert]

basicConstraints = CA:FALSE

nsCertType = client, email

nsComment = "OpenSSL Generated Client Certificate"

subjectKeyIdentifier = hash

authorityKeyIdentifier = keyid,issuer

keyUsage = critical, nonRepudiation, digitalSignature,
keyEncipherment

extendedKeyUsage = clientAuth, emailProtection

[server_cert]

basicConstraints = CA:FALSE

nsCertType = server

nsComment = "OpenSSL Generated Server Certificate"

subjectKeyIdentifier = hash

authorityKeyIdentifier = keyid,issuer:always

keyUsage = critical, digitalSignature, keyEncipherment

extendedKeyUsage = serverAuth

```
[ v3_ca ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ v3_intermediate_ca ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true, pathlen:0
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ usr_cert ]
basicConstraints = CA:FALSE
nsCertType = client, email
nsComment = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth, emailProtection

[ server_cert ]
basicConstraints = CA:FALSE
nsCertType = server
nsComment = "OpenSSL Generated Server Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
```

Pathlen:0 bu sertifikatdan sonra neçə intermediate sertifikat gələ biləcəyini göstərir. Yeni intermediate sertifikatdan sonra server və ya user sertifikat gələ bilər. Əgər 1 olarsa, daha sonra 1 dəfə də intermediate gələ biləcəyini göstərir.

User Certificate:

extendedKeyUsage = clientAuth clientləri authentication et. User və server sertifikat arasındakı fərq user sertifikat əlaqə zamanı authentication etmək üçündür, server sertifikat SSL/TLS config etmək üçündür.

Chmod 700 private

touch index.txt

echo 1000 > serial

echo 1000 > crlnumber

yum install tree

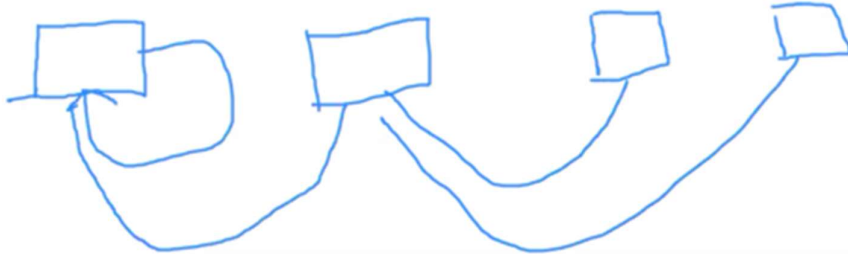
```
root@localhost:/CA_SERVER# tree
.
├── certs
├── conf
├── crl
├── crlnumber
├── csr
├── index.txt
├── intermediate
├── newcerts
├── openssl.cnf
├── private
└── serial

8 directories, 4 files
```

Create Root Private Key

```
openssl genrsa -aes256 -out private/ca.key.pem 4096 (root və intermediate-serfikatdan başqasına parol qoyma)
```

```
chmod 400 private/ca.key.pem
```



Create the root certificate

```
openssl req -config openssl.cnf -key private/ca.key.pem -new -x509 -days 7300 -sha256 -extensions v3_ca -out certs/ca.cert.pem
```

```
root@localhost:/CA_SERVER# openssl req -config openssl.cnf -key private/ca.key.pem -new -x509 -days 7300 -sha256 -extensions v3_ca -out certs/ca.cert.pem
Enter pass phrase for private/ca.key.pem:
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) [XX]:AZ
State or Province Name (full name) []:Baku
Locality Name (eg, city) [Default City]:Baku
Organization Name (eg, company) [Default Company Ltd]:nthad.academy
Organizational Unit Name (eg, section) []:it
Common Name (eg, your name or your server's hostname) []:rootca.nthad.academy
Email Address []:
root@localhost:/CA_SERVER#
```

Verify

```
openssl x509 -noout -text -in certs/ca.cert.pem
```

```
root@localhost:/CA_SERVER# tree
.
├── certs
│   └── ca.cert.pem
├── contr
├── crl
├── crlnumber
├── csr
├── index.txt
├── intermediate
├── newcerts
├── openssl.cnf
├── private
│   └── ca.key.pem
└── serial

8 directories, 6 files
```

Create Intermediate Private key

```
openssl genrsa -aes256 -out private/intermediate.key.pem 4096
```

Create Request for intermediate certificate

```
openssl req -config openssl.cnf -new -sha256 -key private/intermediate.key.pem -out  
csr/intermediate.csr.pem
```

```
openssl ca -config openssl.cnf -extensions v3_intermediate_ca -days 3650 -notext -md sha256  
-in csr/intermediate.csr.pem -out certs/intermediate.cert.pem
```

```
root@localhost:/CA_SERVER# openssl ca -config openssl.cnf -extensions v3_intermediate_ca -days 3650 -notext -md sha256 -in csr/intermediate.csr.pem -out certs/intermediate.cert.pem
Using configuration from openssl.cnf
Enter pass phrase for /CA_SERVER/private/ca.key.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
  Serial Number: 4096 (0x1000)
  Validity
    Not Before: Feb 22 22:13:51 2025 GMT
    Not After : Feb 20 22:13:51 2035 GMT
  Subject:
    countryName           = AZ
    stateOrProvinceName   = Baku
    organizationName       = nihad.academy
    organizationalUnitName = tt
    commonName             = rootca.nihad.academy
  X509v3 extensions:
    X509v3 Subject Key Identifier:
      0C:70:D0:7E:E1:79:02:40:AA:75:04:55:96:EC:AF:91:83:CC:20:CD
    X509v3 Authority Key Identifier:
      5D:5B:4E:2B:6C:89:AE:8A:AD:75:95:AE:1C:B9:3D:D5:65:E6:49:E4
    X509v3 Basic Constraints: critical
      CA:TRUE, pathlen:0
    X509v3 Key Usage: critical
      Digital Signature, Certificate Sign, CRL Sign
Certificate is to be certified until Feb 20 22:13:51 2035 GMT (3650 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
root@localhost:/CA_SERVER#
```

```
cd intermediate/
mkdir conf certs crl newcerts private intermediate csr
cp ../openssl.cnf .
cp ../private/intermediate.key.pem private/
cp ../certs/intermediate.cert.pem certs/
vim openssl.conf
```

```
dir = /CA_SERVER/intermediate
certificate = $dir/certs/intermediate.cert.pem
private_key = $dir/private/intermediate.key.pem
```

Verify the intermediate certificate

```
openssl x509 -noout -text -in certs/intermediate.cert.pem
```

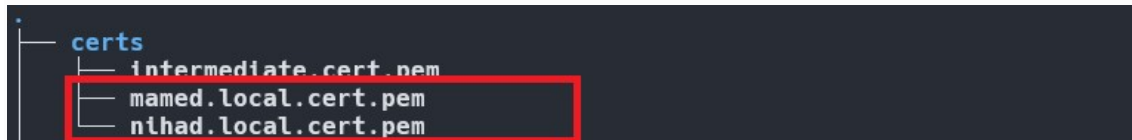
```
cd intermediate/
touch index.txt
echo 2000 > serial
echo 2000 > crlnumber
```

Create clinet certificate

```
openssl genrsa -out private/nihad.local.key.pem 2048
```

```
openssl req -new -key private/nihad.local.key.pem -out csr/nihad.local.key.csr.pem
```

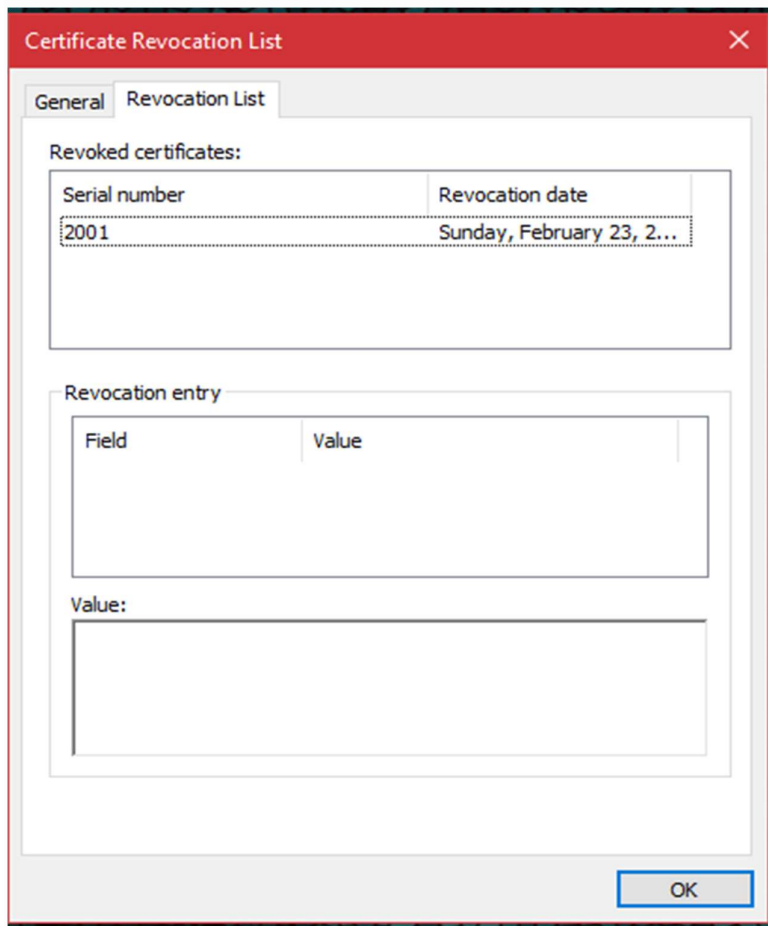
```
openssl ca -config openssl.cnf -extensions usr_cert -notext -md sha256 -in  
csr/nihad.local.key.csr.pem -out certs/nihad.local.cert.pem -batch
```



Create CRL

```
openssl ca -config openssl.cnf -gencrl -out crl/intermediate.crl.pem
```

```
openssl ca -config openssl.cnf -revoke certs/nihad.local.cert.pem
```



Sign server and certificates

```
openssl genrsa -aes256 out private/ser.nihad.local.key.pem 2048
```

```
openssl req -config openssl.cnf -key private/ser.nihad.local.key.pem -new -sha256 -out  
csr/ser.nihad.local.csr.pem
```

```
openssl ca -config openssl.cnf -extensions server_cert -days 375 -notext -md sha256 -in  
csr/ser.nihad.local.csr.pem -out certs/ser.nihad.local.cert.pem
```

Create the certificate chain file

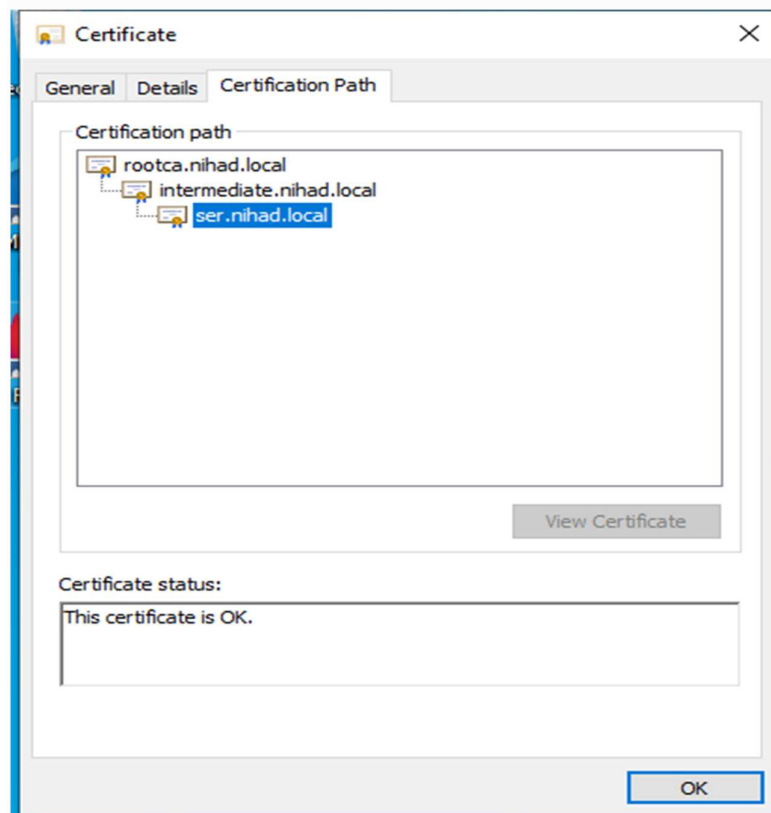
When an application (eg, a web browser) tries to verify a certificate signed by the intermediate CA, it must also verify the intermediate certificate against the root certificate. To complete the chain of trust, create a CA certificate chain to present to the application. To create the CA certificate chain, concatenate the intermediate and root certificates together. We will use this file later to verify certificates signed by the intermediate CA.

```
cd /CA_SERVER
```

```
cat /CA_SERVER/certs/ca.cert.pem >> ca-chain.pem
```

```
cat /CA_SERVER/intermediate/certs/intermediate.cert.pem >> ca-chain.pem
```

```
openssl verify -CAfile ca-chain.pem intermediate/certs/ser.nihad.local.cert.pem
```



| serve_nihad.local | intermediate.nihad.local | rootca.nihad.local |
|---|--------------------------|--------------------|
| Subject Name Country AZ State/Province Baku Organization nihad.local Organizational Unit it Common Name serve_nihad.local | | |
| Issuer Name Country AZ State/Province Baku Organization nihad.local Organizational Unit it Common Name intermediate.nihad.local | | |

Deploy the certificate

You can now either deploy your new certificate to a server, or distribute the certificate to a client. When deploying to a server application (eg, Apache), you need to make the following files available:

Bundle contain

1. Certificate Chain file
2. Certificate Private key
3. Certificate

Create PFX

```
openssl pkcs12 -export -out certs/nihad.full.pfx -inkey
intermediate/private/ser.nihad.local.key.pem -in intermediate/certs/ser.nihad.local.cert.pem
-certfile ca-chain.pem -passout pass:salam.12
```

```
root@localhost:/CA_SERVER/certs# ls -l
total 16
-rw-r--r--. 1 root root 2065 Feb 23 12:31 ca.cert.pem
-rw-r--r--. 1 root root 2033 Feb 23 12:34 intermediate.cert.pem
-rw-----. 1 root root 6195 Feb 23 16:50 nihad.full.pfx
```

Edit server.xml

<Connector

```
port="8443" minProcessors="5" maxProcessors="75"  
enableLookups="true" disableUploadTimeout="true"  
acceptCount="100" maxThreads="200"  
scheme="https" secure="true" SSLEnabled="true"  
keystoreFile="/Tomcat/Multi-Instances/nihad.full.pfx"  
keystorePass="salam.12"  
keystoreType="PKCS12" />
```

```
<Connector port="8081" protocol="HTTP/1.1"  
connectionTimeout="20000"  
redirectPort="8443"  
maxParameterCount="1000"  
/>  
<Connector  
port="8443" minProcessors="5" maxProcessors="75"  
enableLookups="true" disableUploadTimeout="true"  
acceptCount="100" maxThreads="200"  
scheme="https" secure="true" SSLEnabled="true"  
keystoreFile="/Tomcat/Multi-Instances/nihad.full.pfx"  
keystorePass="salam.12"  
keystoreType="PKCS12" />
```

Edit web.xml

```
<security-constraint>

  <web-resource-collection>

    <web-resource-name>Protected Context</web-resource-name>

    <url-pattern>/*</url-pattern>

  </web-resource-collection>

  <user-data-constraint>

    <transport-guarantee>CONFIDENTIAL</transport-guarantee>

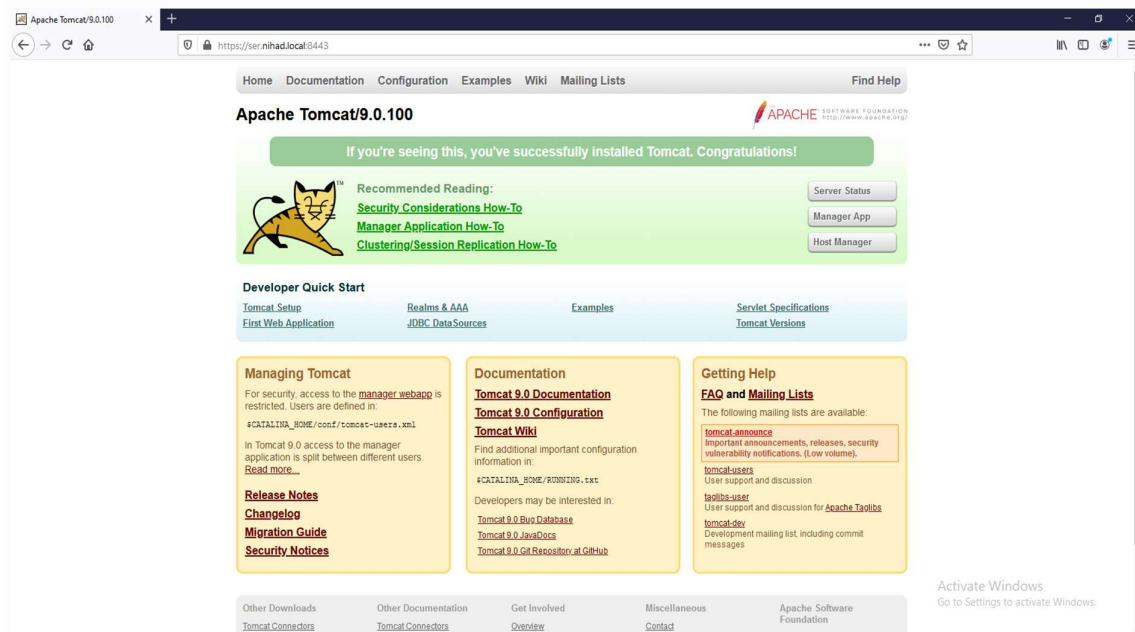
  </user-data-constraint>

</security-constraint>
```

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Protected Context</web-resource-name>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>

</web-app>
```

Add “192.168.1.11 ser.nihad.local” to C:\Windows\System32\Drivers\etc\hosts file



Standart Method

```
yum groupinstall "Development Tools"
```

```
cd /Tomcat/Multi-Instances/instances1/bin
```

```
wget https://archive.apache.org/dist/apr/apr-1.7.1.tar.gz
```

```
tar -xvf apr-1.7.1.tar.gz - cd apr-1.7.1-rc2
```

```
./configure -- make - make install
```

```
cd ..
```

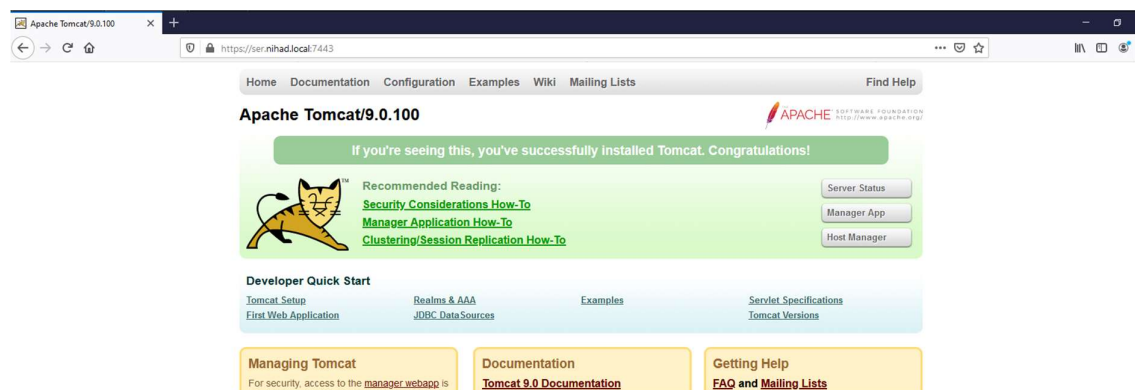
```
tar -xvf tomcat-native.tar.gz
```

```
cd tomcat-native-1.3.1-src
```

```
./configure -with-java-home=/usr/lib/jvm/java-21-openjdk - make
```

```
echo "export CATALINA_OPTS=-Djava.library.path=/usr/local/apr/lib" >> ~/.bashrc
```

```
<Connector port="7443" protocol="org.apache.coyote.http11.Http11AprProtocol"
  maxThreads="150" SSLEnabled="true"
  maxParameterCount="1000"
  >
  <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
  <SSLHostConfig>
    <Certificate certificateKeyFile="/CA_SERVER/intermediate/private/ser.nihad.local.key.nopass.pem"
      certificateFile="/CA_SERVER/intermediate/certs/ser.nihad.local.nopass.cert.pem"
      certificateChainFile="/CA_SERVER/ca-chain.pem"
      type="RSA" />
  </SSLHostConfig>
</Connector>
```



Nginx

```
yum install nginx -y
```

```
vim /etc/nginx/nginx.conf
```

```
server {
    listen      80;
    listen      [::]:80;
    server_name ;
    return 301 https://$host$request_url;
    root        /usr/share/nginx/html;

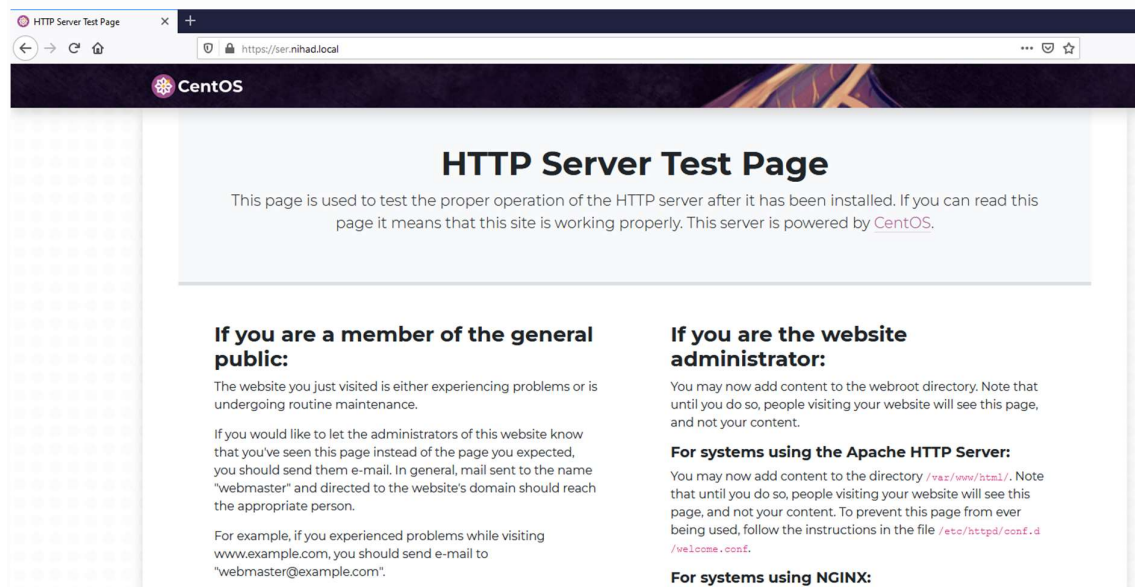
    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;
}

# Settings for a TLS enabled server.

server {
    listen      443 ssl;
    listen      [::]:443 ssl;
    http2       on;
    server_name ser.nihad.local www.ser.nihad.local;
    root        /usr/share/nginx/html;

    #
    ssl_certificate "/CA_SERVER/intermediate/certs/ser.nihad.local.nopass.cert.pem";
    ssl_certificate_key "/CA_SERVER/intermediate/private/ser.nihad.local.key.nopass.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout 10m;
    ssl_ciphers PROFILE=SYSTEM;
    ssl_prefer_server_ciphers on;
}
```

Add “192.168.1.11 ser.nihad.local” to C:\Windows\System32\Drivers\etc\hosts file



HTTP Server Test Page

This page is used to test the proper operation of the HTTP server after it has been installed. If you can read this page it means that this site is working properly. This server is powered by [CentOS](#).

If you are a member of the general public:

The website you just visited is either experiencing problems or is undergoing routine maintenance.

If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

For example, if you experienced problems while visiting [www.example.com](#), you should send e-mail to "webmaster@example.com".

If you are the website administrator:

You may now add content to the webroot directory. Note that until you do so, people visiting your website will see this page, and not your content.

For systems using the Apache HTTP Server:

You may now add content to the directory `/var/www/html/`. Note that until you do so, people visiting your website will see this page, and not your content. To prevent this page from ever being used, follow the instructions in the file `/etc/httpd/conf.d/welcome.conf`.

For systems using NGINX:

Apache httpd SSL config

```
Listen 443
```

```
<VirtualHost *:443>
```

```
    SSLEngine On
```

```
    SSLCertificateFile /etc/httpd/certs/servera.cert.pem
```

```
    SSLCertificateChainFile /etc/httpd/certs/ca-chain.cert.pem
```

```
    SSLCertificateKeyFile /etc/httpd/certs/servera.key.pem
```

```
    DocumentRoot /var/www/html
```

```
    ServerName servera.ingress.com
```

```
</VirtualHost>
```

Linux maşın üçün RootCA –ni sistemə tanıtdırmaq

```
yum install ca-certificates
```

```
cp ingressrootCA.cert.pem /etc/pki/ca-trust/source/anchors/
```

```
update-ca-trust
```