How to configure HTTPS certificates with alternative names

VIAVI Solutions

Description:

How to configure HTTPS with alternative names to be used with nginx

Scenario:

How to configure HTTPS with alternative names in order use encrypted traffic via port 443

Solution/Steps Taken:

This procedure will assume we have two servers: onmsi-server1 and onmsi-server2 whose fully qualified domain names like onmsi-server1.example.com and onmsi-server2.example.com

1. **Generate a Private key, a CSR and a certificate**

Open a shell as root on onmsi-server1:
$ su

Nginx directories

Create the folders and ensure privacy of your private key.

$ cd /etc/nginx/
$ mkdir -p ssl/keep
$ mkdir -p ssl/private/
$ chmod go-rwx ssl/private/
$ cd ssl

Generate a Private Key
Generate a 2048-bit private key for the server

$ *openssl genrsa -out /etc/nginx/ssl/private/private.key 2048*

- **NOTE**

The key should be an ASCII text:

$ file /etc/nginx/ssl/private/private.key
/etc/nginx/ssl/private/private.key: ASCII text

2. **Create an OpenSSL configuration file**

Create an OpenSSL configuration file in /etc/nginx/ssl/keep/openssl.conf and make sure the content looks like this (edit to suit your needs):

[req]
prompt = no
req_extensions = req_ext
distinguished_name = dn
[dn]
CN = onmsi-server1
emailAddress = ssl@example.com
O = the_Organization
OU = the_Organization_Unit
L = the_Location
ST = the_State
C = the_Country
[req_ext]
subjectAltName = @alternate_names
[alternate_names]
DNS.1 = onmsi-server1.example.com
DNS.2 = onmsi-server2.example.com
IP.1 = 127.0.0.1
IP.2 = 192.168.56.3

With C=Country, S=State, L=Location, O=Organization, OU=Organization Unit, CN=Common Name (as in the server's name).

NOTE that in this example, the certificate will cover two host names and 2 IP addresses. In the general case, only one DNS name will be needed.

Generate a Certificate Signing Request (CSR)
Launch the following command to createba myrequestform.csr in the keep folder:

$ openssl req -new -key private/private.key -out ./keep/myrequestform.csr -config ./keep/openssl.conf

Check the CSR content:
$ openssl req -text -noout -verify -in keep/myrequestform.csr

Use the following commands to be sure the csr has been created correctly after the private key

$ openssl pkey -in private/private.key -pubout -outform pem | sha256sum
b5ad4531436c19fd0f4ed245ec087979c808ef44aa01bd7fd84540da42abdfb0 -

$ openssl req -in keep/myrequestform.csr -pubkey -noout -outform pem | sha256sum
b5ad4531436c19fd0f4ed245ec087979c808ef44aa01bd7fd84540da42abdfb0 -

Note: the above output is just an example but the two must match each other

3. **Ask for a certificate**

Dump the CSR
$ cat keep/myrequestform.csr

Send the CSR to whoever is responsible to generate the certificate (can be a certificate vendor orthe IT department of your company).
You should then receive:

a certnew.cer (certificate)
a certnew.p7b (chain towards your CA root)

4. **Create the full certificate**

- In case you received a .p7b file and the certificate in extended form like

Intermediate Certificate
-----BEGIN CERTIFICATE-----
MIIFITCCAwmgAwIBAgIQVn4yLvJvtu27gzxYdW2cvTANBgkqhkiG9w0BAQsFADBQ
.
.
1y+4waVSA2//q+N9YzBPoNDhdL9z
-----END CERTIFICATE-----

End Entity Certificate
-----BEGIN CERTIFICATE-----
MIIE7jCCA9agAwIBAgIQBQI/72FWDr7xCtoDTJUCIDANBgkqhkiG9w0BAQsFADBZ
.
.
2A2gPXW+Gbsu1j5PR5GeXc6a
-----END CERTIFICATE-----

You must create manually certnew.cer using the above certificates making sure that:

- You use only the parts within “-----BEGIN CERTIFICATE-----” and “-----END CERTIFICATE-----”, including those lines

- Start from the “End Entity Certificate” part and proceed with the “Intermediate Certificate”

To create a file do the following

$ vi /etc/nginx/ssl/keep/certnew.cer

copy and paste the certificates as explained above.

The full file should like (in this example)

-----BEGIN CERTIFICATE-----
MIIE7jCCA9agAwIBAgIQBQI/72FWDr7xCtoDTJUCIDANBgkqhkiG9w0BAQsFADBZ
.
.
2A2gPXW+Gbsu1j5PR5GeXc6a
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIFITCCAwmgAwIBAgIQVn4yLvJvtu27gzxYdW2cvTANBgkqhkiG9w0BAQsFADBQ
.
.
1y+4waVSA2//q+N9YzBPoNDhdL9z
-----END CERTIFICATE-----

Copy certnew.p7b to the keep directory.

Extract the chain to a suitable format with

$ openssl pkcs7 -outform PEM -in ./keep/certnew.p7b -print_certs > /tmp/server-chain.cer

then, since NGINX is using the certificate concatenated with the chain

$ cat ./keep/certnew.cer /tmp/server-chain.cer > server.fullchain.crt

- In case you received a .p7b file and the certificate in form like

-----BEGIN CERTIFICATE-----
MIIE7jCCA9agAwIBAgIQBQI/72FWDr7xCtoDTJUCIDANBgkqhkiG9w0BAQsFADBZ
.
.
1y+4waVSA2//q+N9YzBPoNDhdL9z
-----END CERTIFICATE-----

It is enough to create directly the certificate with

$ openssl pkcs7 -outform PEM -in ./keep/certnew.p7b -printcerts > server.fullchain.crt

- In case you received a single .crt or .pem file that could be already usable.

To make sure:

Check the certificate is a PEM on:

$ file /etc/nginx/ssl/server.fullchain.crt
/etc/nginx/ssl/server.fullchain.crt: PEM certificate

Check the outputs below are a match:

$ openssl pkey -in /path/to/private.key -pubout -outform pem | sha256sum

$ openssl x509 -in /path/to/certificate.crt -pubkey -noout -outform pem | sha256sum

If so, it simply has to be renamed and put in /etc/nginx/ssl/server.fullchain.crt

5. **Test the certificate.**

Test the .crt file with

$ openssl x509 -in server.fullchain.crt -pubkey -noout -outform pem | sha256sum
b5ad4531436c19fd0f4ed245ec087979c808ef44aa01bd7fd84540da42abdfb0 -

that should give a matching output to similar command above.

6. **Test the files and folders**

At this point your directory structure should be like this:

```
/etc/nginx/ssl
 ├── keep/
 │     ├── certnew.cer
 │     ├── certnew.p7b
 |     ├── myrequestform.csr
 |     ├──  openssl.conf
  ├──private/              (==>  drwx------  root  root)
 │     └── private.key
 └── server.fullchain.crt
```

In case of a second server like this example, move private.key and server.fullchain.crt onto the second server in the same folders (to be created as above).