Laboratory_14: CA based on openssl

This laboratory covers CA creation based on openssl and a comparison with Self-Signed Certificate.

Installations

- Follow the lab guidelines in: Lectures_Lab/Secure_Access_Systems_and_Data_Transmission/lab14/ca.md at master · sfl0r3nz05/Lectures_Lab · GitHub
- Read use case: Lectures_Lab/Secure_Access_Systems_and_Data_Transmission/lab14/ca.md at master · sfl0r3nz05/Lectures_Lab · GitHub

Part 1: Self Signed-Certificates

Generate Keys and Certificate Signing Request (CSR)

1. Generate an RSA key for the CA:

   openssl genrsa -out example.org.key 2048

2. Inspect the key

openssl rsa -in example.org.key -noout -text

3. [optional] the rsa public key can be extracted from the private key:

   openssl rsa -in example.org.key -pubout -out example.org.pubkey

4. [optional] public key verification
   openssl rsa -in example.org.pubkey -pubin -noout -text

5. Generate a CSR:

   openssl req -new -key example.org.key -out example.org.csr

```
Country Name (2 letter code) [AU]:PT
State or Province Name (full name) [Some-State]:Lisboa
Locality Name (eg, city) []:Lisboa
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Example Org
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:*.example.org
Email Address []:
```

6. Review the content of the csr file:

   openssl req -in example.org.csr -noout –text

Notice that there are no extensions, to add extensions an additional config file is needed. This makes the process a bit more complicated so when you buy a wildcard certificate you don't usually need to specify the extension SubjectAltName for the naked domain because the issuer will do it for you. This is an example configuration file for a CSR:

```
# The main section is named req because the command we are using is req
# (openssl req ...)
[ req ]
# This specifies the default key size in bits. If not specified then 512 is
# used. It is used if the -new option is used. It can be overridden by using
# the -newkey option.
default_bits = 2048

# This is the default filename to write a private key to. If not specified the
# key is written to standard output. This can be overridden by the -keyout
# option.
default_keyfile = oats.key

# If this is set to no then if a private key is generated it is not encrypted.
# This is equivalent to the -nodes command line option. For compatibility
# encrypt_rsa_key is an equivalent option.
encrypt_key = no

# This option specifies the digest algorithm to use. Possible values include
# md5 sha1 mdc2. If not present then MD5 is used. This option can be overridden
# on the command line.
```

```
default_md = sha1

# if set to the value no this disables prompting of certificate fields and just
# takes values from the config file directly. It also changes the expected
# format of the distinguished_name and attributes sections.
prompt = no

# if set to the value yes then field values to be interpreted as UTF8 strings,
# by default they are interpreted as ASCII. This means that the field values,
# whether prompted from a terminal or obtained from a configuration file, must
# be valid UTF8 strings.
utf8 = yes

# This specifies the section containing the distinguished name fields to
# prompt for when generating a certificate or certificate request.
distinguished_name = my_req_distinguished_name



# this specifies the configuration file section containing a list of extensions
# to add to the certificate request. It can be overridden by the -reqexts
# command line switch. See the x509v3_config(5) manual page for details of the
# extension section format.
req_extensions = my_extensions

[ my_req_distinguished_name ]
C = PT
ST = Lisboa
L = Lisboa
O  = Oats In The Water
CN = *.oats.org

[ my_extensions ]
basicConstraints=CA:FALSE
subjectAltName=@my_subject_alt_names
subjectKeyIdentifier = hash

[ my_subject_alt_names ]
DNS.1 = *.oats.org
DNS.2 = *.oats.net
DNS.3 = *.oats.in
DNS.4 = oats.org
```

```
DNS.5 = oats.net
DNS.6 = oats.in
```

7. Use this new config file:

   openssl req -new -out oats.csr -config oats.conf

8. Because we did not specify a key, OpenSSL uses the information on our configuration (default_bits and default_keyfile) to create one. Now we can see that there is a Request Extensions section with our coveted Subject Alternative Name field. Verify the csr:

   openssl req -in oats.csr -noout -text

9. CA Key and self-signed Certificate

openssl genrsa -out ca.key 2048

10. Generate a self-signed certificate for the CA:

openssl req -new -x509 -key ca.key -out ca.crt

```
Country Name (2 letter code) [AU]:PT
State or Province Name (full name) [Some-State]:Lisboa
Locality Name (eg, city) []:Lisboa
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Sz CA
Organizational Unit Name (eg, section) []:SZ CA
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:An optional company name []:
```

11. Signing:

openssl x509 -req -in example.org.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out example.org.crt

12. Each issued certificate must contain a unique serial number assigned by the CA. It must be unique for each certificate given by a given CA. OpenSSL keeps the used serial numbers on a file, by default it has the same name as the CA certificate file with the extension replace by srl. So a file named ca.srl is created:

cat ca.srl

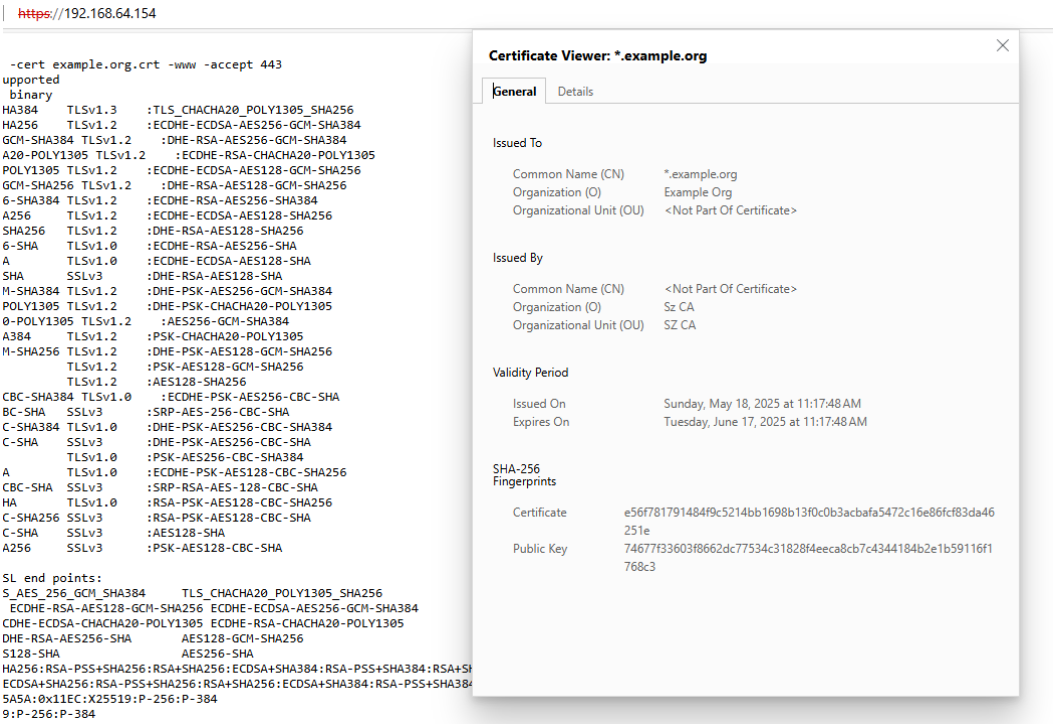
13. This command produces the file example.org.crt which we can examine. Notice the serial number, in hex is exactly the contents of the created ca.srl file.

openssl x509 -in example.org.crt -noout –text

14. Mount a basic web server:

    sudo openssl s_server -key example.org.key -cert example.org.crt -www -accept 443

15. Review invalid certificate in the browser:



16. Extended validation is missing, that is an extension we did not include in the certificate that usually requires the CA to verify the legal identification of the subject. Just to check it, we can ask the browser to export the certificate into a file we can query with openssl: Comparte with GitHbu certificate:

openssl x509 -in github.com.crt -noout –text

17. Create openssl ca configuring ca.conf:
    Lectures_Lab/Secure_Access_Systems_and_Data_Transmission/lab14/ca.md at master · sfl0r3nz05/Lectures_Lab · GitHub.

```
we use 'ca' as the default section because we're usign the ca command
we use 'ca' as the default section because we're usign the ca command
[ ca ] default_ca = my_ca
[ my_ca ]
a text file containing the next serial number to use in hex. Mandatory.
This file must be present and contain a valid serial number.
serial = ./serial
the text database file to use. Mandatory. This file must be present though
initially it will be empty.
database = ./index.txt
specifies the directory where new certificates will be placed. Mandatory.
new_certs_dir = ./newcerts
the file containing the CA certificate. Mandatory
certificate = ./ca.crt
the file contaning the CA private key. Mandatory
private_key = ./ca.key
the message digest algorithm. Remember to not use MD5
default_md = sha1
for how many days will the signed certificate be valid
default_days = 365
a section with a set of variables corresponding to DN fields
policy = my_policy
[ my_policy ]
```

```
if the value is "match" then the field value must match the same field in the
CA certificate. If the value is "supplied" then it must be present.
Optional means it may be present. Any fields not mentioned are silently
deleted.
countryName = match stateOrProvinceName = supplied organizationName = supplied
commonName = supplied organizationalUnitName = optional commonName = supplied
[ ca ] default_ca = my_ca
[ my_ca ]
a text file containing the next serial number to use in hex. Mandatory.
This file must be present and contain a valid serial number.
serial = ./serial
the text database file to use. Mandatory. This file must be present though
initially it will be empty.
database = ./index.txt
specifies the directory where new certificates will be placed. Mandatory.
new_certs_dir = ./newcerts
the file containing the CA certificate. Mandatory
certificate = ./ca.crt
the file contaning the CA private key. Mandatory
private_key = ./ca.key
the message digest algorithm. Remember to not use MD5
default_md = sha1
for how many days will the signed certificate be valid
default_days = 365
a section with a set of variables corresponding to DN fields
policy = my_policy
[ my_policy ]
if the value is "match" then the field value must match the same field in the
```

```
CA certificate. If the value is "supplied" then it must be present.

Optional means it may be present. Any fields not mentioned are silently

deleted.

countryName = match stateOrProvinceName = supplied organizationName = supplied
commonName = supplied organizationalUnitName = optional commonName = supplied
```

18. We need to setup some structure first. The configuration file expects a newcerts directory, and the index.txt and serial files:

    $ mkdir newcerts
    $ touch index.txt
    $ echo '01' > serial

19. We can sign the certificate:

    ```
    openssl ca -config ca.cnf -out example.org.crt -infiles example.org.csr
    ```

20. If we wish to add extensions, or even to keep the extensions sent in a CSR (openssl will remove them when signing), then we need to also include that configuration. This is an extra configuration file `oats.extensions.cnf`:

    basicConstraints=CA:FALSE subjectAltName=@my_subject_alt_names
    subjectKeyIdentifier = hash
    [my_subject_alt_names] DNS.1 = *.oats.org DNS.2 = *.oats.net DNS.3 = *.oats.in
    DNS.4 = oats.org DNS.5 = oats.net DNS.6 = oats.in

21. And now:

    ```
    openssl ca -config ca.cnf -out oats.crt -extfile oats.extensions.cnf -in
    oats.csr
    ```

```
Using configuration from ca.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'PT'
stateOrProvinceName   :PRINTABLE:'Lisboa'
localityName          :PRINTABLE:'Lisboa'
organizationName      :PRINTABLE:'Oats In The Water'
commonName            :T61STRING:'*.oats.org'
Certificate is to be certified until Mar 21 01:43:11 2015 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
```

22. We have a certificate that includes the SubjectAltNames we wanted:

```
openssl x509 -in oats.crt -noout –text
```

23. We can verify the certificate is correct:

```
openssl verify -CAfile ca.crt oats.crt
```