Building and OpenSSL Certificate Authority - Creating Your Intermediary Certificate

Chase Abbott, 2017-06-11

Creating Your Intermediary Certificate Authority

Previously we created the first part of our OpenSSL CA by building our root certificate. We are now ready to complete our CA chain by creating and signing the intermediary certificate. The intermediary will be responsible for signing client and server certificate requests. It acts as an authoritative proxy for the root certificate hence the name intermediary. The chain of trust will extend from the root certificate to the intermediary certificate down to the certificates you'll deploy within your infrastructure.

Create your directory structure

Create a new subdirectory under `/root/ca` to segregate intermediary files our root configuration .

```
# sudo bash
# mkdir /root/ca/intermediate
```

We're creating the same directory structure previously used under `/root/ca` within `/root/ca/intermediary` . It's your decision if you if you want to do something different. Some of my best friends are flat directory structures and we don't judge personal practices.

Create your intermediary CA database to keep track of signed certificates

```
# cd /root/ca/intermediate
# mkdir certs crl csr private
# touch index.txt
# echo 1000 > serial
```

Create a crlnumber file for the intermediary CA to use

```
# echo 1000 > /root/ca/intermediate/crlnumber
```

Similar to the earlier serial statement, this will create the crlnumber file and start the numerical iteration at 1000. This will be used for future certificate revocation needs.

Create your OpenSSL intermediary config file

Copy the GIST openssl_intermediate.cnf file to `/root/ca/intermediate/openssl_intermediate.cnf` and modify the contents for your own naming conventions. Similar to the `root_ca.cnf` , the `[CA]` is required and will gather it's configuration from the `[CA_default]` section. Changes to the `[int_ca]` include:

```
[ CA_default ]
# Directory and file locations.
dir               = /root/ca/intermediate
private_key       = $dir/private/int.cheese.key.pem
certificate       = $dir/cers/int.cheese.crt.pem
crlnumber         = $dir/crlnumber
crl               = $dir/crl/int.cheese.crl.pem
crl_extensions    = crl_ext
```

```
policy            = policy_loose
```

We have new certificate names for our intermediary use and define `policy_loose` so future certificate requests don't have to match country, state/province, or organization.

Create the Intermediary's Private Key and Certificate Signing Request

Similar to the root certificate, we're following the NSA Suite B requirements and matching the root's elliptical curve, secp384r1. We'll also create the CSR and private key all in one line, making your scripts and life a bit easier.

```
# cd /root/ca
# openssl req -config intermediate/openssl_intermediate.cnf -new -newkey ec:<(openssl ecparam -name secp384r1) -keyo

Generating an EC private key
writing new private key to 'intermediate/private/int.cheese.key.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 [WA]:
Locality Name [Seattle]:
Organization Name [Grilled Cheese Inc.]:
Organizational Unit Name [Grilled Cheese Intermediary CA]:
Common Name []:Grilled Cheese Inc. Intermediary Certificate Authority
Email Address [grilledcheese@yummyinmytummy.us]:
```

Sign the certificate request with the root certificate and use the openssl_intermediate.cnf config file to specify the `[v3_intermediate_ca]` extension instead of the `[v3_ca]` as we did for the root. The `openssl_intermediate.cnf` has a few changes which we need to note.

```
[ v3_intermediate_ca ]
# Extensions for a typical intermediate CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true, pathlen:0
keyUsage = critical, digitalSignature, cRLSign, keyCertSign
crlDistributionPoints = @crl_info
authorityInfoAccess = @ocsp_info

[crl_info]
URI.0 = http://crl.grilledcheese.us/whoremovedmycheese.crl

[ocsp_info]
caIssuers;URI.0 = http://ocsp.grilledcheese.us/cheddarcheeseroot.crt
OCSP;URI.0 = http://ocsp.grilledcheese.us/
```

The Certificate Revocation List (crl) and Online Certificate Status Protocol (OCSP) should be included within the intermediary certificate. This lets systems know where check and see if the intermediary certificate was revoked by the root at any given time. We will cover this in detail later and browsers do not necessarily check the intermediary certificates for revocation, but they absolutely do for the site certificates. We're adding CRL and OCSP to the Intermediary CA for best practices purpose.

Create the intermediate certificate

Sign the `csr/int.cheese.cs`r with the root's certificate. We are going to drop down to `/root/ca` so the creation of the intermediary certificate is stored within the root's `index.txt` and we'll also use the root's OpenSSL Config file `openssl_root.cnf`.

```
# openssl ca -config openssl_root.cnf -extensions v3_intermediate_ca -days 3600 -md sha384 -in intermediate/csr/int.chee

Using configuration from openssl_root.cnf
Enter pass phrase for /root/ca/private/ca.cheese.key.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4097 (0x1001)
        Validity
            Not Before: Aug 24 21:51:07 2017 GMT
            Not After : Jul  3 21:51:07 2027 GMT
        Subject:
            countryName               = US
            stateOrProvinceName       = WA
            organizationName          = Grilled Cheese Inc.
            organizationalUnitName    = Grilled Cheese Intermediary CA
            commonName                = Grilled Cheese Inc. Intermediary Certificate Authority
            emailAddress              = grilledcheese@yummyinmytummy.us
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                7E:2D:A5:D0:9B:70:B9:E3:D2:F7:C0:0A:CF:70:9A:8B:80:38:B1:CD
            X509v3 Authority Key Identifier:
                keyid:27:C8:F7:34:2F:30:81:97:DE:2E:FC:DD:E2:1D:FD:B6:8F:5A:AF:BB

            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
            X509v3 CRL Distribution Points:

                Full Name:
                  URI:http://crl.grilledcheese.us/whomovedmycheese.crl

            Authority Information Access:
                CA Issuers - URI:http://ocsp.grilledcheese.us/cheddarcheeseroot.crt
                OCSP - URI:http://ocsp.grilledcheese.us/

Certificate is to be certified until Jul  3 21:51:07 2027 GMT (3600 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
```

Validate the Certificate Contents with OpenSSL.

```
# openssl x509 -noout -text -in intermediate/certs/int.cheese.crt.pem

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4097 (0x1001)
    Signature Algorithm: ecdsa-with-SHA384
        Issuer: C = US, ST = WA, L = Seattle, O = Grilled Cheese Inc., OU = Grilled Cheese Root CA, C
        Validity
            Not Before: Aug 24 21:51:07 2017 GMT
            Not After : Jul  3 21:51:07 2027 GMT
        Subject: C = US, ST = WA, O = Grilled Cheese Inc., OU = Grilled Cheese Intermediary CA, CN =
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (384 bit)
                pub:
                    04:9b:14:9a:55:6d:db:15:7f:d7:8b:fd:37:4d:ba:
                    e8:50:8e:88:32:99:27:4e:20:36:25:8b:7b:ac:bb:
                    2f:d6:61:c1:5a:c8:e6:4c:98:20:3f:cf:86:3c:bf:
                    f4:f3:b0:1c:1c:0b:cc:7f:e4:4b:13:59:58:a1:53:
                    87:cb:4c:17:66:04:21:01:6a:44:5f:22:31:7d:3d:
                    fe:a2:e7:73:c8:77:7c:1a:f9:9c:4a:9d:e7:77:6a:
                    c7:9e:3e:f0:4a:b0:37
                ASN1 OID: secp384r1
                NIST CURVE: P-384
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                7E:2D:A5:D0:9B:70:B9:E3:D2:F7:C0:0A:CF:70:9A:8B:80:38:B1:CD
            X509v3 Authority Key Identifier:
                keyid:27:C8:F7:34:2F:30:81:97:DE:2E:FC:DD:E2:1D:FD:B6:8F:5A:AF:BB

            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
            X509v3 CRL Distribution Points:

                Full Name:
                  URI:http://crl.grilledcheese.us/whomovedmycheese.crl

            Authority Information Access:
                CA Issuers - URI:http://ocsp.grilledcheese.us/cheddarcheeseroot.crt
                OCSP - URI:http://ocsp.grilledcheese.us/

    Signature Algorithm: ecdsa-with-SHA384
         30:65:02:30:74:07:ba:fe:4b:71:78:d8:d2:7f:84:c0:50:b4:
         b6:df:6c:f6:57:f5:d9:2c:4b:e1:d4:d8:1d:78:fd:7e:bf:0a:
         81:86:bb:40:c5:9b:97:6f:83:04:5f:d3:85:36:6c:d6:02:31:
         00:d3:08:78:1c:da:6d:ef:1d:bb:27:df:0b:76:eb:ab:84:b2:
         91:04:25:1a:85:5b:d5:c3:cd:66:e4:9e:14:b2:c0:ed:9c:59:
         b7:18:c3:26:eb:df:78:13:68:47:66:b5:43
```

Similar to the root, we can note the usage and algorithms but we have the addition of:

```
* X509v3 CRL Distribution Points:
    Full Name:
      URI:http://crl.grilledcheese.us/whomovedmycheese.crl
*Authority Information Access:
    CA Issuers - URI:http://ocsp.grilledcheese.us/cheddarcheeseroot.crt
    OCSP - URI:http://ocsp.grilledcheese.us/
```

Create the certificate chain

The root certificate and intermediary certificate must be available to the requesting client/server in order to validate the chain of trust. To complete the trust validation, a certificate chain must be available to the client application. A certificate chain usually takes the form of separate certificates installed into Root and Intermediary containers (as the case for Windows), or bundled together either in a .pfx cert and cert chain bundle or a PEM formatted text file. Concatenate the root and intermediate certificates together to create a PEM certificate chain text file.

```
# cd /root/ca
# $cat intermediate/certs/int.cheese.crt.pem certs/ca.cheese.crt.pem > intermediate/certs/chain.cheese.crt.pem
```

The file should look similar to this with two separate BEGIN and END statements for each certificate (example condensed for space):

```
# cat intermediate/certs/chain.cheese.crt.pem
-----BEGIN CERTIFICATE-----
MIID/TCCA4OgAwIBAgICEAEwCgYIKoZIzj0EAwMwgdQxCzAJBgNVBAYTAlVTMQsw
CQYDVQQIDAJXQTEQMA4GA1UEBwwHU2VhdHRsZTEcMBoGA1UECgwTR3JpbGxlZCBD
......
hkjOPQQDAwNoADBlAjB0B7r+S3F42NJ/hMBQtLbfbPZX9dksS+HU2B14/X6/CoGG
u0DFm5dvgwRf04U2bNYCMQDTCHgc2m3vHbsn3wt266uEspEEJRqFW9XDzWbknhSy
wO2cWbcYwybr33gTaEdmtUM=
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIDQTCCAsegAwIBAgIJAP+99S/FDT0CMAoGCCqGSM49BAMDMIHUMQswCQYDVQQG
EwJVUzELMAkGA1UECAwCV0ExEDAOBgNVBAcMB1NlYXR0bGUxHDAaBgNVBAoME0dy
......
CgYIKoZIzj0EAwMDaAAwZQIwd6H54qs6WkvOjWouMD8Bz4523fYfA9mzXKE9bTYE
+wH3MycDhd4kVhfJGuQ7NcSoAjEAzQ5s4NUm0/uIVvpnn+m+tI+UHCy3dBnO7BXS
/kiTCl//67LTrlpoh9zJLFSNBGh/
-----END CERTIFICATE-----
```

Note: In the real world hosting application should never have the entire chain available as it defeats a core principle of PKI. It's recommended in test labs to distribute the root certificate to all testing client applications and systems and include only the intermediary along with the server certificate. This way the client can establish the trust between the intermediary and root certificates. Next we'll move on to creating our CLR endpoint list and OCSP certificate.

Our intermediary certificate is now created and signed and we are ready to move on. To complete the CA our next article we will create our certificate revocation list (CRL) endpoint and online certificate status protocol (OCSP) certificate allowing us to revoke certificates. Lab environments rarely need revocation functionality but modern clients check for CLR and OCSP URIs so it's nessisary to have the configruation defined at minimum. Let's proceed.

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com