BlogtimeException

Creating a rudimentary private certificate authority using OpenSSL

Behrang Saeedzadeh Jan 13th, 2019

Early draft.

Table of Contents

- 1. Introduction
- 2. What we will be building
- 3. Building our CA environment
  - 3.1. Directory structure
  - 3.2. Writing the configuration file for the root CA
  - 3.3. Generating TMNT's root CA certificate and private key
  - 3.4. Checkpoint
  - 3.5. Amending `openssl.root.cnf`
  - 3.6. Writing the configuration for the intermediate CA
  - 3.7. Checkpoint
  - 3.8. Amending our intermediate CA
- 4. Issuing server and client certificates
- 5. Verifying certificate chains
- References
- Appendix A: The `setup-ca.sh` script

1. Introduction

In this article we will use OpenSSL to create a basic certificate authority (CA) for a fictitious company named TMNT Inc and use it to issue two certificates: a wildcard server certificate for `*.tmnt.local` and a client certificate for a user named "Donatello".

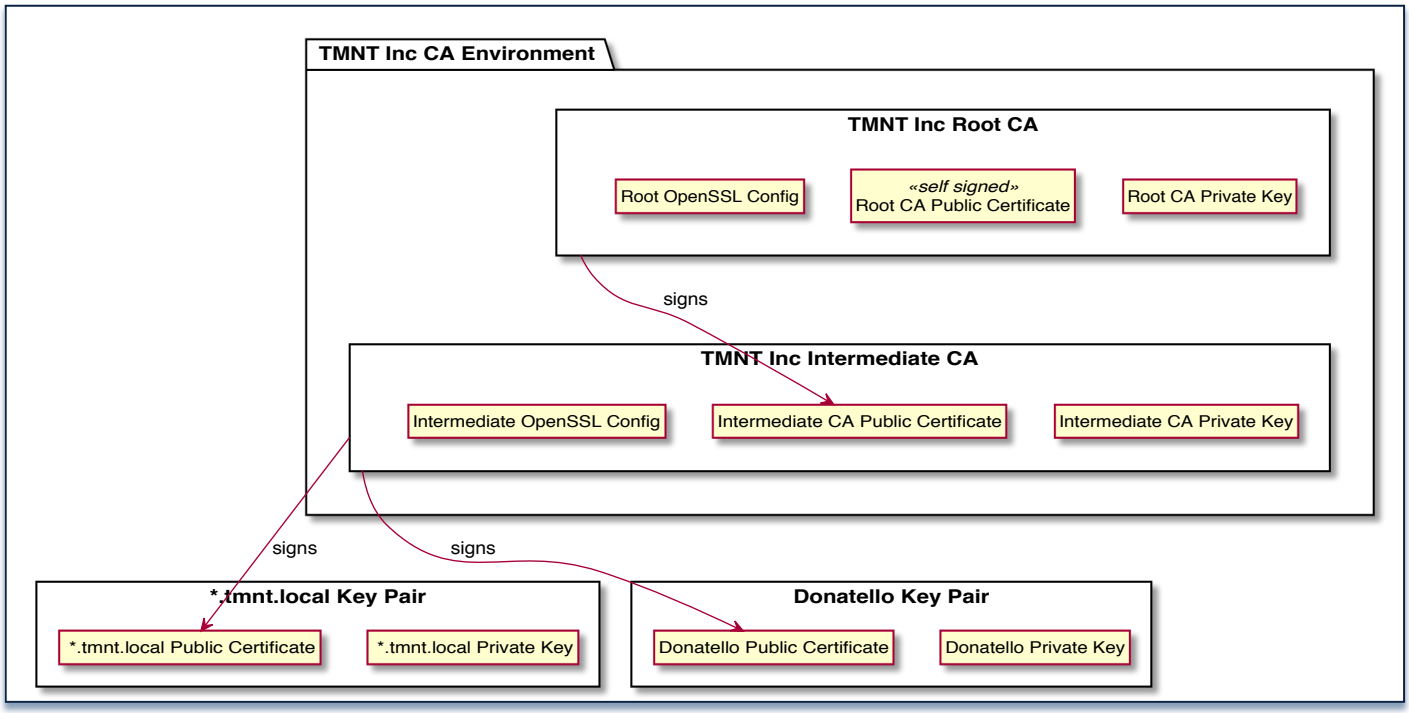
Building a production ready CA is an intricate and major undertaking and out of the scope of this article and our goal here is not to build such a CA; instead we will build a CA that could come in handy for ad-hoc experiments, for issuing key pairs that can be used in integration testing and for mocking APIs that use private CAs—a common setup for internal

services of big organizations such as banks, insurance companies, government agencies, etc.

2. What we will be building

By the end of this article we will build a CA environment depicted in Figure 1:

Figure 1. TMNT Inc CA with two signed certificates



As you can see, TMNT Inc's CA environment is comprised of a root CA and an intermediate CA. The sole purpose of the root CA is to sign the public certificate of the intermediate CA and in our setup client and server certificates must be signed by TMNT Inc's intermediate CA.

3. Building our CA environment

3.1. Directory structure

First let's build the directory structure for TMNT Inc's CA environment in `/opt/ca/tmnt` by executing the commands in Listing 1:

Listing 1. Creating the CA directory structure

```
$ sudo mkdir -p /opt/ca/tmnt/{certs,newcerts,private}
$ sudo mkdir -p /opt/ca/tmnt/intermediate/{certs,csr,newcerts,private}
$ sudo chown -R $(whoami) /opt/ca/tmnt
```

We also need some auxiliary files to keep track of issued certificates and their serial numbers. Execute the commands in Listing 2 to create these files.

Listing 2. Creating auxiliary files

```
$ cd /opt/ca/tmnt
$ touch index.txt
$ echo "unique_subject = yes" > index.txt.attr
$ echo FFFFFF > serial

$ cd /opt/ca/tmnt/intermediate
$ touch index.txt
$ echo "unique_subject = yes" > index.txt.attr
$ echo FFFFFF > serial
```

These files, in essence, make our CA's database.

3.2. Writing the configuration file for the root CA

Now let's write the first bits of `openssl.root.cnf`, TMNT Inc's root CA OpenSSL configuration file and save it in `/opt/ca/tmnt` and use it to generate the private key and the self signed certificate for TMNT Inc's root CA. Contents of this file is shown in Listing 3:

Listing 3. `/opt/ca/tmnt/openssl.root.cnf`

```
[ req ]
default_bits        = 2048
default_md          = sha256
distinguished_name  = req_distinguished_name  1
prompt              = no
x509_extensions     = v3_ca  2

[ req_distinguished_name ]
commonName          = TMNT Root CA
stateOrProvinceName = Victoria
countryName         = AU
emailAddress        = admin@tmnt.local
organizationName    = TMNT Inc

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

```
keyUsage                = critical, digitalSignature, keyCertSign
```

1. *Default values for the distinguished name of certificates are defined in the section titled `[ req_distinguished_name ]`*
2. *Certificate extension values are defined in the section titled `[ v3_ca ]`.*

3.3. Generating TMNT's root CA certificate and private key

Later we will come back to `/opt/ca/tmnt/openssl.root.cnf` and amend it with more details. But for now it has everything we need in order to create the private key and public certificate for TMNT Inc's root CA. Issue the commands in Listing 4 to create this key pair:

Listing 4. Creating the private key and certificate for our TMNT Inc's root CA

```
$ cd /opt/ca/tmnt
$ openssl req -config openssl.root.cnf \
              -x509 \
              -passout pass:rootpass \
              -days 7300 \
              -newkey rsa \
              -keyout private/root.key.pem \
              -out    certs/root.cert.pem
Generating a 2048 bit RSA private key
..............................+++
....+++
writing new private key to 'private/root.key.pem'
```

Let's inspect the details of the generated certificate using the `openssl x509` command:

Listing 5. Inspecting root CA certificate's details

```
$ cd /opt/ca/tmnt
$ openssl x509 -noout -text \
         -in certs/root.cert.pem \
         -fingerprint -sha256
Certificate:
    Data:
        Version: 3 (0×2)
        Serial Number:
            8e:4b:75:57:05:58:6a:f1
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN = TMNT Root CA, ST = Victoria, C = AU, emailAddress = a
        Validity
```

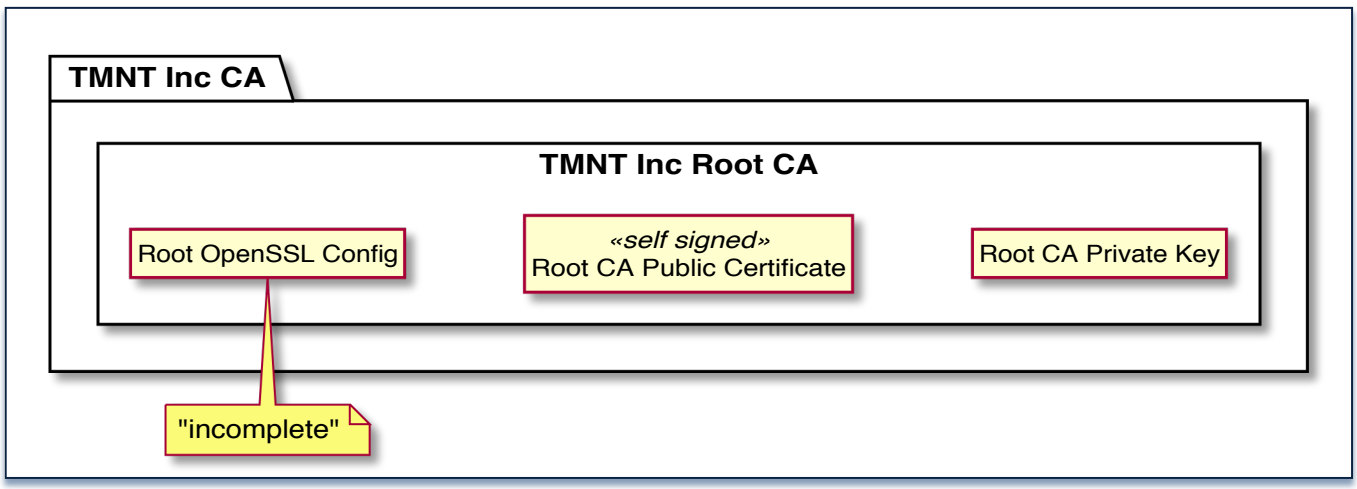
```
        Validity
            Not Before: Jan  2 08:53:53 2019 GMT
            Not After : Dec 28 08:53:53 2038 GMT
        Subject: CN = TMNT Root CA, ST = Victoria, C = AU, emailAddress =
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:ab:25:68:71:7a:86:a9:1b:d9:a8:ec:75:69:88:
                    b9:dd:96:f0:ad:04:fc:9b:4d:f5:ac:09:30:98:7f:
                    7e:ad:b3:aa:66:4d:09:84:2f:b1:16:f2:85:61:3f:
                    ee:be:d8:81:06:7f:74:87:9e:93:62:33:ba:d1:98:
                    75:94:c1:01:3a:3c:37:04:78:af:cc:b7:64:0f:db:
                    88:03:21:26:67:3f:8d:a3:65:ee:e4:69:a0:d6:b3:
                    ab:f7:cb:1f:92:b1:d1:34:7b:08:14:91:19:76:91:
                    f9:5c:25:0c:8e:b0:48:6c:71:5d:53:64:0e:1e:d8:
                    3e:19:36:75:49:04:ab:b5:15:be:43:03:52:16:f6:
                    2b:9c:32:2d:b3:31:a4:62:54:fe:80:25:70:42:5d:
                    0c:46:f4:c8:16:82:b3:64:99:4c:f0:83:25:2c:da:
                    41:4a:a6:b0:7e:b3:5a:35:00:40:34:34:47:12:34:
                    88:3b:c8:d1:48:de:43:f5:ce:d4:32:4e:82:dd:9f:
                    96:e4:b7:4e:d9:57:8f:14:a1:31:d5:ad:9b:50:82:
                    51:c5:e7:da:79:22:57:81:59:79:ac:15:b7:1d:fe:
                    45:48:f7:0d:74:45:8e:be:3b:11:2a:b0:17:e2:ad:
                    bb:7b:18:b8:aa:8c:dc:9b:dc:ac:7c:b1:49:21:1b:
                    cc:db
                Exponent: 65537 (0×10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                4D:92:68:F7:2C:38:CF:AC:23:AE:66:36:6E:7B:E2:FB:13:E7:47:3
            X509v3 Authority Key Identifier:
                keyid:4D:92:68:F7:2C:38:CF:AC:23:AE:66:36:6E:7B:E2:FB:13:E

            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign
    Signature Algorithm: sha256WithRSAEncryption
         23:f0:b4:4b:b7:d9:9e:fb:39:32:85:60:29:1f:1b:2b:49:24:
         d9:b6:2d:88:91:5f:dc:21:45:ac:35:eb:27:fb:4e:c2:8a:4c:
         05:26:c9:db:27:27:e6:9d:2b:f1:e9:02:55:0b:3d:cb:63:52:
```

```
        df:7a:a9:3f:16:00:82:40:25:0e:55:a0:cb:af:23:34:f3:fe:
        77:ec:64:80:c9:4e:3e:e2:60:f8:bf:e7:9b:3a:b8:11:e4:a3:
        0b:95:c5:22:2f:9e:64:35:5e:9e:6e:cf:a3:08:c1:6f:7f:c4:
        82:3e:a7:d0:c9:b4:2f:f2:04:ee:25:4c:a2:91:2e:ea:42:e1:
        c3:62:4c:ce:8f:99:c4:f2:02:af:54:3b:22:ec:3a:2f:76:7a:
        52:58:9a:3b:2e:9d:d8:0b:b9:71:be:bb:dd:0c:1e:01:f9:c5:
        1b:08:00:b8:64:69:70:73:dd:00:aa:f2:f2:ad:e5:3f:c3:d0:
        a5:ef:be:2d:af:02:f9:07:86:9c:80:53:9d:da:32:bf:41:2a:
        a5:a4:23:68:65:85:bf:69:64:58:c1:88:f2:60:27:db:6c:13:
        71:67:30:94:bb:69:93:6a:f5:5d:64:c8:68:be:ac:e8:dd:7d:
        5e:ef:63:c6:40:dc:53:7e:6d:68:3d:03:36:10:fa:96:23:3e:
        41:a0:19:a4
SHA256 Fingerprint=1F:F2:90:BA:21:35:5E:3C:85:A6:BB:86:3E:FE:27:6D:1C:81:9
```

3.4. Checkpoint

So far, we have created this subset of TMNT Inc CA's environment:

Figure 2. Checkpoint 1



In the next section, we will amend `openssl.root.cnf` to turn it into a proper—but rudimentary—CA that can be used to sign the public certificate for TMNT Inc's intermediate CA

3.5. Amending `openssl.root.cnf`

Now let's amend `openssl.root.cnf` with the missing `[ ca ]` section. Similar to the `[ req ]` section, the `[ ca ]` section defines default parameter values for the `openssl ca` command—the interface to OpenSSL's minimal CA service. We will also add a section to the config file named `[ v3_intermediate_ca ]` that we will later use whenever we want to sign an intermediate certificate using our root CA. The amendments are shown in Listing 6:

Listing 6. Amending `openssl.root.cnf`

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

[ ca ]
default_ca = ca_tmnt_root

[ca_tmnt_root]
dir                     = /opt/ca/tmnt
database                = $dir/index.txt
new_certs_dir           = $dir/newcerts
serial                  = $dir/serial
private_key             = $dir/private/root.key.pem
certificate             = $dir/certs/root.cert.pem
default_md              = sha256
name_opt                = ca_default
cert_opt                = ca_default
default_days            = 7300
policy                  = ca_tmnt_root_policy

[ca_tmnt_root_policy]
commonName              = supplied
stateOrProvinceName     = match
countryName             = match
emailAddress            = optional
organizationName        = match
organizationalUnitName  = optional
```

3.6. Writing the configuration for the intermediate CA

We need a separate configuration file for TMNT's intermediate CA. The contents of this configuration file, `openssl.intermediate.cnf`, are shown in Listing 7.

Listing 7. `/opt/ca/tmnt/intermediate/openssl.intermediate.cnf`

```
[ req ]
```

```
default_bits            = 2048
default_md              = sha256
distinguished_name      = req_distinguished_name
x509_extensions         = v3_ca

[ req_distinguished_name ]
countryName             = Country Name (2 letter code)
stateOrProvinceName     = State or Province Name
localityName            = Locality Name
organizationName        = Organization Name
organizationalUnitName  = Organizational Unit Name
commonName              = Common Name
emailAddress            = Email Address

stateOrProvinceName_default = Victoria
countryName_default         = AU
emailAddress_default        = admin@tmnt.local
organizationName_default    = TMNT Inc

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

[ ca ]
default_ca = ca_tmnt_intermediate

[ ca_tmnt_intermediate ]
dir                     = /opt/ca/tmnt/intermediate
database                = $dir/index.txt
new_certs_dir           = $dir/newcerts
serial                  = $dir/serial
private_key             = $dir/private/intermediate.key.pem
certificate             = $dir/certs/intermediate.cert.pem
default_md              = sha256
name_opt                = ca_default
cert_opt                = ca_default
default_days            = 7300
policy                  = ca_tmnt_intermediate_policy
```

```
[ ca_tmnt_intermediate_policy ]
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
```

Now let's create an encrypted private key for our intermediate CA and then create a CSR and sign it using our root CA.

Listing 8. Generating the private key for TMNT Inc's intermediate CA

```
$ cd /opt/ca/tmnt/intermediate
$ openssl genrsa \
          -passout pass:interpass \
          -aes256 \
          -out private/intermediate.key.pem 2048
Generating RSA private key, 2048 bit long modulus
.....................................................................
.........+++
e is 65537 (0×010001)
```

Unlike TMNT Inc's root CA certificate that was self-signed, its intermediate CA certificate should be signed by the root CA. Consequently, first we need to create a CSR for our intermediate CA as shown in Listing 9:

Listing 9. Generating the CSR for TMNT's intermediate CA certificate

```
$ cd /opt/ca/tmnt/intermediate
$ openssl req \
          -config openssl.intermediate.cnf \
          -new \
          -days 7300 \
          -sha256 \
          -key private/intermediate.key.pem \
          -passin pass:interpass \
          -subj "/emailAddress=admin@tmnt.local/C=AU/ST=Victoria/O=TMNT In
          -out csr/intermediate.csr.pem
```

Now we can process the CSR using TMNT's root CA and produce the intermediate CA's signed certificate:

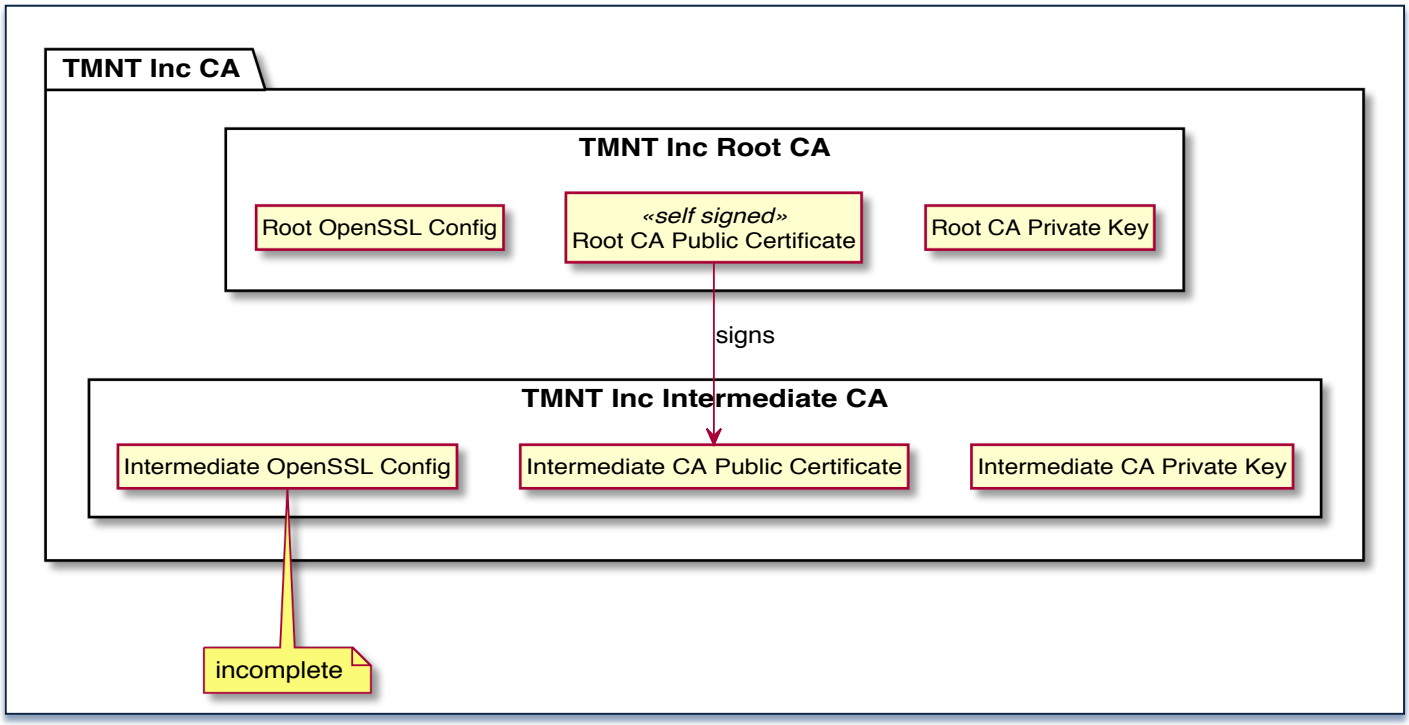
Listing 10. Generating the intermediate CA's signed certificate

```
$ cd /opt/ca/tmnt
$ openssl ca -config openssl.root.cnf \
           -extensions v3_intermediate_ca \
           -notext \
           -passin pass:rootpass \
           -in intermediate/csr/intermediate.csr.pem \
           -out intermediate/certs/intermediate.cert.pem
Using configuration from openssl.root.cnf
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 16777215 (0×ffffff)
        Validity
            Not Before: Jan  3 03:53:27 2019 GMT
            Not After : Dec 29 03:53:27 2038 GMT
        Subject:
            commonName                = TMNT Intermediate CA
            stateOrProvinceName       = Victoria
            countryName               = AU
            emailAddress              = admin@tmnt.local
            organizationName          = TMNT Inc
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                29:5B:BF:1B:C9:C3:31:3A:A7:79:B1:85:4C:CD:8B:41:A6:5C:8D:1
            X509v3 Authority Key Identifier:
                keyid:48:C1:FE:00:4A:B1:3F:6B:4F:69:E9:5E:61:53:8D:EE:72:5

            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign
Certificate is to be certified until Dec 29 03:53:27 2038 GMT (7300 days)
Sign the certificate? [y/n]:y
Data Base Updated
```

3.7. Checkpoint

Our root and intermediate CAs are now almost ready. We now need to amend our intermediate CA's configuration file and add a couple of extra sections to it and it will be good to issue new server and client certificates.

Figure 3. Checkpoint 2



3.8. Amending our intermediate CA

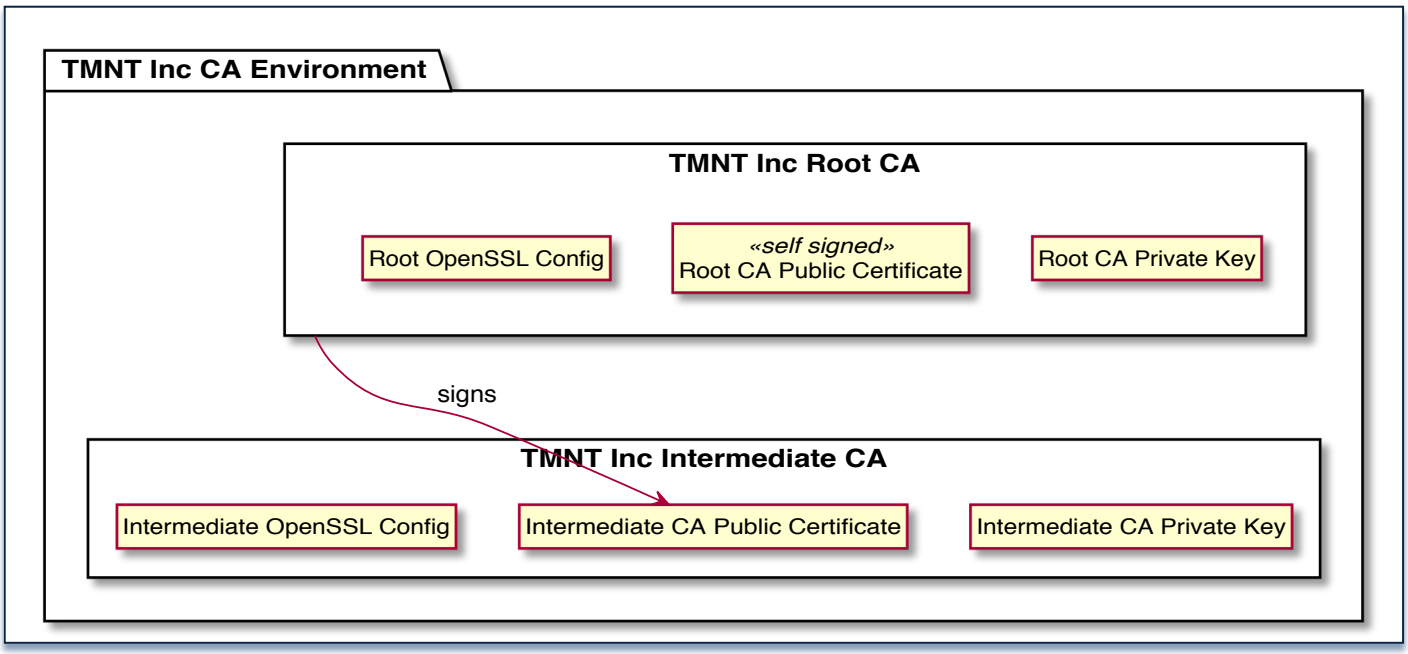
Let's add a `[ client_cert ]` and a `[ server_cert ]` section to `openssl.intermediate.cnf` to have a complete CA that can sign new client and server certificates:

Listing 11. Amending `openssl.intermediate.cnf`

```
[ client_cert ]
basicConstraints       = CA:FALSE
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid,issuer
keyUsage               = critical, nonRepudiation, digitalSignature, keyEn
extendedKeyUsage       = clientAuth

[ server_cert ]
basicConstraints       = CA:FALSE
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage               = critical, digitalSignature, keyEncipherment
extendedKeyUsage       = serverAuth
```

Now our CA environment is ready:

Figure 4. TMNT Inc CA



In the next section we will issue a server certificate for `*.tmnt.local` and a client certificate for "Donatello" using our intermediate CA.

4. Issuing server and client certificates

By executing the commands in Listing 12 we will create a certificate for `*.tmnt.local` that is signed by TMNT Inc's intermediate CA.

Listing 12. Issuing a certificate for `*.tmnt.local`

```
$ cd /opt/ca/tmnt/intermediate

$ cd /opt/ca/tmnt/intermediate
openssl genrsa \
        -passout pass:tmntpass \
        -aes256 \
        -out private/tmnt.local.key.pem 2048
Generating RSA private key, 2048 bit long modulus
.....................................................+++
....+++
e is 65537 (0×010001)

$ cd /opt/ca/tmnt/intermediate
openssl req \
        -config openssl.intermediate.cnf \
        -key private/tmnt.local.key.pem \
        -new \
```

```
        -days 7300 \
        -sha256 \
        -passin pass:tmntpass \
        -subj "/emailAddress=admin@tmnt.local/C=AU/ST=Victoria/O=TMNT Inc/
        -out csr/tmnt.local.csr.pem

$ cd /opt/ca/tmnt/intermediate
openssl ca \
        -config openssl.intermediate.cnf \
        -passin pass:interpass \
        -extensions server_cert \
        -days 7500 \
        -md sha256 \
        -in csr/tmnt.local.csr.pem \
        -out certs/tmnt.local.cert.pem
Using configuration from openssl.intermediate.cnf
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 16777215 (0×ffffff)
        Validity
            Not Before: Jan  3 04:00:12 2019 GMT
            Not After : Jul 17 04:00:12 2039 GMT
        Subject:
            countryName               = AU
            stateOrProvinceName       = Victoria
            organizationName          = TMNT Inc
            commonName                = *.tmnt.local
            emailAddress              = admin@tmnt.local
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Subject Key Identifier:
                E9:76:35:69:13:32:68:53:B1:AE:EF:03:66:E4:05:70:AE:8E:B5:3
            X509v3 Authority Key Identifier:
                keyid:18:85:12:15:94:88:38:5D:46:55:41:3F:F8:F2:91:DA:2C:A
                DirName:/CN=TMNT Root CA/ST=Victoria/C=AU/emailAddress=adm
                serial:FF:FF:FF

            X509v3 Key Usage: critical
```

```
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
Certificate is to be certified until Jul 17 04:00:12 2039 GMT (7500 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
```

Similarly we can generate a client certificate for “Donatello” by executing the commands in Listing 13:

Listing 13. Issuing a certificate for Donatello

```
cd /opt/ca/tmnt/intermediate
openssl genrsa \
          -passout pass:donatellopass \
          -aes256 \
          -out private/donatello.key.pem 2048
Generating RSA private key, 2048 bit long modulus
.....................................................................+++
.........................................+++
e is 65537 (0×010001)

cd /opt/ca/tmnt/intermediate
openssl req \
        -config openssl.intermediate.cnf \
        -key private/donatello.key.pem \
        -new \
        -days 7300 \
        -sha256 \
        -passin pass:donatellopass \
        -subj "/emailAddress=donatello@tmnt.local/C=AU/ST=Victoria/O=TMNT
        -out csr/donatello.csr.pem

cd /opt/ca/tmnt/intermediate
openssl ca \
        -config openssl.intermediate.cnf \
        -passin pass:interpass \
        -extensions client_cert \
```

```
        -days 7500 \
        -md sha256 \
        -in csr/donatello.csr.pem \
        -out certs/donatello.cert.pem
Using configuration from openssl.intermediate.cnf
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 16777216 (0×1000000)
        Validity
            Not Before: Jan  3 04:02:15 2019 GMT
            Not After : Jul 17 04:02:15 2039 GMT
        Subject:
            countryName               = AU
            stateOrProvinceName       = Victoria
            organizationName          = TMNT Inc
            commonName                = Donatello
            emailAddress              = donatello@tmnt.local
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Subject Key Identifier:
                39:D2:C6:57:D7:0B:C4:21:51:A6:2B:D8:5A:79:5A:27:A0:36:CE:A
            X509v3 Authority Key Identifier:
                keyid:18:85:12:15:94:88:38:5D:46:55:41:3F:F8:F2:91:DA:2C:A

            X509v3 Key Usage: critical
                Digital Signature, Non Repudiation, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Client Authentication
Certificate is to be certified until Jul 17 04:02:15 2039 GMT (7500 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
```

We have now built everything depicted Figure 1.

5. Verifying certificate chains

In order to put our CA environment and certificates issued by our CA to use, we have to distribute our public CA to all the departments and teams within our organization.

On the other hand, we should configure our servers to send their terminal certificate plus the intermediate certificate that has signed them to clients that want to make a secure connection.

This will allow clients to validate the chain of trust. In Listing 14, we verify that our terminal certificates are indeed signed by the intermediate CA, and the intermediate CA's public certificate is also signed by the root CA:

Listing 14. Verifying chain of trust

```
$ openssl verify \
        -CAfile /opt/ca/tmnt/certs/root.cert.pem \
        -untrusted /opt/ca/tmnt/intermediate/certs/intermediate.cert.pem \
        /opt/ca/tmnt/intermediate/certs/tmnt.local.cert.pem
/opt/ca/tmnt/intermediate/certs/tmnt.local.cert.pem: OK

$ openssl verify \
        -CAfile /opt/ca/tmnt/certs/root.cert.pem \
        -untrusted /opt/ca/tmnt/intermediate/certs/intermediate.cert.pem \
        /opt/ca/tmnt/intermediate/certs/donatello.cert.pem
/opt/ca/tmnt/intermediate/certs/donatello.cert.pem: OK
```

References

- [1] John Viega, Matt Messier & Pravir Chandra. Network Security with OpenSSL: Cryptography for Secure Communications. O'Reilly Media. 2009.
- [2] Jamie Nguyen. OpenSSL Certificate Authority.

Appendix A: The `setup-ca.sh` script

Listing 15. `setup-ca.sh`

```
#!/bin/bash

read -p "Step 0 - Press enter to delete /opt/ca/tmnt"
rm -fr /opt/ca/tmnt
```

```
read -p "Step 1 - Press enter to make the /opt/ca/tmnt directory tree"
sudo mkdir -p /opt/ca/tmnt/{certs,newcerts,private}
sudo mkdir -p /opt/ca/tmnt/intermediate/{certs,csr,newcerts,private}
sudo chown -R $(whoami) /opt/ca/tmnt

tree /opt/ca/tmnt

read -p "Step 2 - Press enter to prepare auxiliary files"
cd /opt/ca/tmnt
touch index.txt
echo "unique_subject = yes" > index.txt.attr
echo FFFFFF > serial

cd /opt/ca/tmnt/intermediate
touch index.txt
echo "unique_subject = yes" > index.txt.attr
echo FFFFFF > serial

tree /opt/ca/tmnt

read -p "Step 3 - Press enter to prepare /opt/ca/tmnt/openssl.root.cnf"
cat << ROOT_CONF > /opt/ca/tmnt/openssl.root.cnf
[ req ]
default_bits        = 2048
default_md          = sha256
distinguished_name  = req_distinguished_name
prompt              = no
x509_extensions     = v3_ca

[ req_distinguished_name ]
commonName          = TMNT Root CA
stateOrProvinceName = Victoria
countryName         = AU
emailAddress        = admin@tmnt.local
organizationName    = TMNT Inc

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

```
keyUsage               = critical, digitalSignature, keyCertSign

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

[ ca ]
default_ca = ca_tmnt_root

[ca_tmnt_root]
dir                     = /opt/ca/tmnt
database                = \$dir/index.txt
new_certs_dir           = \$dir/newcerts
serial                  = \$dir/serial
private_key             = \$dir/private/root.key.pem
certificate             = \$dir/certs/root.cert.pem
default_md              = sha256
name_opt                = ca_default
cert_opt                = ca_default
default_days            = 7300
policy                  = ca_tmnt_root_policy

[ca_tmnt_root_policy]
commonName              = supplied
stateOrProvinceName     = match
countryName             = match
emailAddress            = optional
organizationName        = match
organizationalUnitName  = optional
ROOT_CONF

cat /opt/ca/tmnt/openssl.root.cnf

read -p "Step 4 - Press enter to generate the root key pair"
cd /opt/ca/tmnt
openssl req -config openssl.root.cnf \
            -x509 \
            -passout pass:rootpass \
```

```
            -days 7300 \
            -newkey rsa \
            -keyout private/root.key.pem \
            -out    certs/root.cert.pem

echo "Inspecting root.cert.pem"

cd /opt/ca/tmnt
openssl x509 -noout -text \
       -in certs/root.cert.pem \
       -fingerprint -sha256

read -p "Step 5 - Press enter to prepare /opt/ca/tmnt/intermediate/openssl
cat << INTERMEDIATE_CONF > /opt/ca/tmnt/intermediate/openssl.intermediate.
[ req ]
default_bits            = 2048
default_md              = sha256
distinguished_name      = req_distinguished_name
x509_extensions         = v3_ca

[ req_distinguished_name ]
countryName             = Country Name (2 letter code)
stateOrProvinceName     = State or Province Name
localityName            = Locality Name
organizationName        = Organization Name
organizationalUnitName  = Organizational Unit Name
commonName              = Common Name
emailAddress            = Email Address

stateOrProvinceName_default = Victoria
countryName_default         = AU
emailAddress_default        = admin@tmnt.local
organizationName_default    = TMNT Inc

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

[ ca ]
```

```
[ ca ]
default_ca = ca_tmnt_intermediate

[ ca_tmnt_intermediate ]
dir                     = /opt/ca/tmnt/intermediate
database                = \$dir/index.txt
new_certs_dir           = \$dir/newcerts
serial                  = \$dir/serial
private_key             = \$dir/private/intermediate.key.pem
certificate             = \$dir/certs/intermediate.cert.pem
default_md              = sha256
name_opt                = ca_default
cert_opt                = ca_default
default_days            = 7300
policy                  = ca_tmnt_intermediate_policy

[ ca_tmnt_intermediate_policy ]
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

[ client_cert ]
basicConstraints       = CA:FALSE
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid,issuer
keyUsage               = critical, nonRepudiation, digitalSignature, keyEn
extendedKeyUsage       = clientAuth

[ server_cert ]
basicConstraints       = CA:FALSE
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage               = critical, digitalSignature, keyEncipherment
extendedKeyUsage       = serverAuth
INTERMEDIATE_CONF

cat /opt/ca/tmnt/intermediate/openssl.intermediate.cnf
```

```
read -p "Step 6 - Press enter to generate the intermediate private key"
cd /opt/ca/tmnt/intermediate
openssl genrsa \
        -aes256 \
        -passout pass:interpass \
        -out private/intermediate.key.pem 2048

read -p "Step 7 - Press enter to generate the CSR for the intermediate CA'
cd /opt/ca/tmnt/intermediate
openssl req \
        -config openssl.intermediate.cnf \
        -new \
        -days 7300 \
        -sha256 \
        -key private/intermediate.key.pem \
        -passin pass:interpass \
        -subj "/emailAddress=admin@tmnt.local/C=AU/ST=Victoria/O=TMNT Inc/
        -out csr/intermediate.csr.pem

read -p "Step 8 - Press enter to sign the the intermediate CA's certificat
cd /opt/ca/tmnt
openssl ca -config openssl.root.cnf \
           -extensions v3_intermediate_ca \
           -notext \
           -passin pass:rootpass \
           -in intermediate/csr/intermediate.csr.pem \
           -out intermediate/certs/intermediate.cert.pem

read -p "Step 9 - Press enter to generate the private key for *.tmnt.local
cd /opt/ca/tmnt/intermediate
openssl genrsa \
        -passout pass:tmntpass \
        -aes256 \
        -out private/tmnt.local.key.pem 2048

read -p "Step 10 - Press enter to generate the CSR for *.tmnt.local"
cd /opt/ca/tmnt/intermediate
openssl req \
        -config openssl.intermediate.cnf \
```

```
        -key private/tmnt.local.key.pem \
        -new \
        -days 7300 \
        -sha256 \
        -passin pass:tmntpass \
        -subj "/emailAddress=admin@tmnt.local/C=AU/ST=Victoria/O=TMNT Inc/
        -out csr/tmnt.local.csr.pem

read -p "Step 11 - Press enter to sign the certificate for *.tmnt.local"
cd /opt/ca/tmnt/intermediate
openssl ca \
        -config openssl.intermediate.cnf \
        -passin pass:interpass \
        -extensions server_cert \
        -days 7500 \
        -md sha256 \
        -in csr/tmnt.local.csr.pem \
        -out certs/tmnt.local.cert.pem

read -p "Step 12 - Press enter to generate the private key for Donatello"
cd /opt/ca/tmnt/intermediate
openssl genrsa \
          -passout pass:donatellopass \
          -aes256 \
          -out private/donatello.key.pem 2048

read -p "Step 13 - Press enter to generate the CSR for Donatello"
cd /opt/ca/tmnt/intermediate
openssl req \
        -config openssl.intermediate.cnf \
        -key private/donatello.key.pem \
        -new \
        -days 7300 \
        -sha256 \
        -passin pass:donatellopass \
        -subj "/emailAddress=donatello@tmnt.local/C=AU/ST=Victoria/O=TMNT
        -out csr/donatello.csr.pem

read -p "Step 14 - Press enter to sign the certificate for Donatello"
cd /opt/ca/tmnt/intermediate
openssl ca \
```

```
openssl ca \
        -config openssl.intermediate.cnf \
        -passin pass:interpass \
        -extensions client_cert \
        -days 7500 \
        -md sha256 \
        -in csr/donatello.csr.pem \
        -out certs/donatello.cert.pem
```


“Begin at once to live, and count each separate day as a separate life.” — Seneca