How to setup an enterprise level Certificate Authority using openssl

Introduction

The motivation for writing this paper is that I stumbled upon a situation where I needed a no-cost, signed certificate with a root Certificate Authorities (CA) signing chain in PEM format. This effort is actually pretty simple, but takes some degree of thought in order to put it into effect.

Requirements

You will need at least openssl version 1.0.1 for this task.

I'm writing and performing the creation on a macbook, but linux will also work in the same fashion.

I used brew to install this version, `OpenSSL 1.0.2p, 14 Aug 2018.`

Layout

Create a directory structure in your local home directory

```
$ umask 0022
$ mkdir ~/PKI_20180826_fortress.lan
$ cd ~/PKI_20180826_fortress.lan
```

Create your base directories

```
$ mkdir -p ca/{root-ca,signing-ca}/{db,private}
$ mkdir crl
$ mkdir certs
$ chmod 700 ca/signing-ca/private
$ chmod 700 ca/root-ca/private
```

The ca directory holds the CA resources,
the crl directory holds Certificate Revocation Lists,
and the certs directory holds user and server certificates.

Create your databases

```
cp /dev/null ca/root-ca/db/root-ca.db
cp /dev/null ca/root-ca/db/root-ca.db.attr
cp /dev/null ca/signing-ca/db/signing-ca.db
cp /dev/null ca/signing-ca/db/signing-ca.db.attr
printf "01\n" > ca/root-ca/db/root-ca.crt.srl
printf "01\n" > ca/root-ca/db/root-ca.crl.srl
printf "01\n" > ca/signing-ca/db/signing-ca.crt.srl
printf "01\n" > ca/signing-ca/db/signing-ca.crl.srl
```

The database files must exist before the `openssl ca` command can be used.

With the next four sections starting with Populate… you are creating a heredoc, simply copy and paste from the `cat <<-file_name>EOF` all the way to the end of the section with `EOF` into your terminal. You can paste hundreds of lines of code using this method all at once in order to create documents.

Populate the etc/email.conf file

```
cat <<-etc/email.conf>EOF
# Email certificate request

# This file is used by the openssl req command. Since we cannot know the DN in
# advance the user is prompted for DN information.

[ req ]
default_bits            = 2048                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha1                  # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = yes                   # Prompt for DN
distinguished_name      = email_dn              # DN template
req_extensions          = email_reqext          # Desired extensions

[ email_dn ]
0.domainComponent       = "1. Domain Component          (eg, com)      "
1.domainComponent       = "2. Domain Component          (eg, company)  "
2.domainComponent       = "3. Domain Component          (eg, pki)      "
organizationName        = "4. Organization Name         (eg, company)  "
organizationalUnitName  = "5. Organizational Unit Name (eg, section)  "
commonName              = "6. Common Name               (eg, full name)"
commonName_max          = 64
emailAddress            = "7. Email Address             (eg, name@fqdn)"
emailAddress_max        = 40

[ email_reqext ]
keyUsage                = critical,digitalSignature,keyEncipherment
extendedKeyUsage        = emailProtection,clientAuth
subjectKeyIdentifier    = hash
subjectAltName          = email:move
EOF
```

Populate the etc/root-ca.conf file

```
cat <<-etc/root-ca.conf>EOF
# Root CA

# The [default] section contains global constants that can be referred to from
# the entire configuration file. It may also hold settings pertaining to more
# than one openssl command.

[ default ]
ca                      = root-ca               # CA name
dir                     = .                     # Top dir

# The next part of the configuration file is used by the openssl req command.
# It defines the CA's key pair, its DN, and the desired extensions for the CA
# certificate.

[ req ]
default_bits            = 2048                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha1                  # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = no                    # Don't prompt for DN
distinguished_name      = ca_dn                 # DN section
req_extensions          = ca_reqext             # Desired extensions

[ ca_dn ]
0.domainComponent       = "lan"
1.domainComponent       = "fortress"
organizationName        = "Fortress LAN"
organizationalUnitName  = "Root CA"
commonName              = "Root CA"

[ ca_reqext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true
subjectKeyIdentifier    = hash

# The remainder of the configuration file is used by the openssl ca command.
# The CA section defines the locations of CA assets, as well as the policies
# applying to the CA.

[ ca ]
default_ca              = root_ca               # The default CA section

[ root_ca ]
certificate             = $dir/ca/$ca.crt       # The CA cert
private_key             = $dir/ca/$ca/private/$ca.key # CA private key
new_certs_dir           = $dir/ca/$ca           # Certificate archive
serial                  = $dir/ca/$ca/db/$ca.crt.srl # Serial number file
crlnumber               = $dir/ca/$ca/db/$ca.crl.srl # CRL number file
database                = $dir/ca/$ca/db/$ca.db # Index file
unique_subject          = no                    # Require unique subject
default_days            = 10950                 # How long to certify for (30 years)
default_md              = sha1                  # MD to use
policy                  = match_pol             # Default naming policy
email_in_dn             = no                    # Add email to cert DN
preserve                = no                    # Keep passed DN ordering
name_opt                = ca_default            # Subject DN display options
cert_opt                = ca_default            # Certificate display options
copy_extensions         = none                  # Copy extensions from CSR
x509_extensions         = signing_ca_ext        # Default cert extensions
default_crl_days        = 365                   # How long before next CRL
crl_extensions          = crl_ext               # CRL extensions

# Naming policies control which parts of a DN end up in the certificate and
# under what circumstances certification should be denied.

[ match_pol ]
domainComponent         = match                 # Must match 'fortress.lan'
```

```
organizationName        = match                 # Must match 'Fortress LAN'
organizationalUnitName  = optional              # Included if present
commonName              = supplied              # Must be present

[ any_pol ]
domainComponent         = optional
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = optional
emailAddress            = optional

# Certificate extensions define what types of certificates the CA is able to
# create.

[ root_ca_ext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always

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

# CRL extensions exist solely to point to the CA certificate that has issued
# the CRL.

[ crl_ext ]
authorityKeyIdentifier  = keyid:always
EOF
```

Populate the etc/server.conf file

```
cat <<-etc/server.conf>EOF
# TLS server certificate request

# This file is used by the openssl req command. The subjectAltName cannot be
# prompted for and must be specified in the SAN environment variable.

[ default ]
SAN                     = DNS:*.fortress.lan    # Default value

[ req ]
default_bits            = 2048                  # RSA key size
encrypt_key             = no                    # Protect private key
default_md              = sha1                  # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = yes                   # Prompt for DN
distinguished_name      = server_dn             # DN template
req_extensions          = server_reqext         # Desired extensions

[ server_dn ]
0.domainComponent       = "1. Domain Component         (eg, com)      "
1.domainComponent       = "2. Domain Component         (eg, company)  "
2.domainComponent       = "3. Domain Component         (eg, pki)      "
organizationName        = "4. Organization Name        (eg, company)  "
organizationalUnitName  = "5. Organizational Unit Name (eg, section)  "
commonName              = "6. Common Name              (eg, FQDN)     "
commonName_max          = 64

[ server_reqext ]
keyUsage                = critical,digitalSignature,keyEncipherment
extendedKeyUsage        = serverAuth,clientAuth
subjectKeyIdentifier    = hash
subjectAltName          = $ENV::SAN
EOF
```

Populate the etc/signing-ca.conf file

```
cat <<-etc/signing-ca.conf>EOF
# Signing CA

# The [default] section contains global constants that can be referred to from
# the entire configuration file. It may also hold settings pertaining to more
# than one openssl command.

[ default ]
ca                      = signing-ca            # CA name
dir                     = .                     # Top dir

# The next part of the configuration file is used by the openssl req command.
# It defines the CA's key pair, its DN, and the desired extensions for the CA
# certificate.

[ req ]
default_bits            = 2048                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha1                  # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = no                    # Don't prompt for DN
distinguished_name      = ca_dn                 # DN section
req_extensions          = ca_reqext             # Desired extensions

[ ca_dn ]
0.domainComponent       = "lan"
1.domainComponent       = "fortress"
organizationName        = "Fortress LAN"
organizationalUnitName  = "Signing CA"
commonName              = "Signing CA"

[ ca_reqext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true,pathlen:0
subjectKeyIdentifier    = hash

# The remainder of the configuration file is used by the openssl ca command.
# The CA section defines the locations of CA assets, as well as the policies
# applying to the CA.

[ ca ]
default_ca              = signing_ca            # The default CA section

[ signing_ca ]
certificate             = $dir/ca/$ca.crt       # The CA cert
private_key             = $dir/ca/$ca/private/$ca.key # CA private key
new_certs_dir           = $dir/ca/$ca           # Certificate archive
serial                  = $dir/ca/$ca/db/$ca.crt.srl # Serial number file
crlnumber               = $dir/ca/$ca/db/$ca.crl.srl # CRL number file
database                = $dir/ca/$ca/db/$ca.db # Index file
unique_subject          = no                    # Require unique subject
default_days            = 1825                  # How long to certify for
default_md              = sha1                  # MD to use
policy                  = match_pol             # Default naming policy
email_in_dn             = no                    # Add email to cert DN
preserve                = no                    # Keep passed DN ordering
name_opt                = ca_default            # Subject DN display options
cert_opt                = ca_default            # Certificate display options
copy_extensions         = copy                  # Copy extensions from CSR
x509_extensions         = email_ext             # Default cert extensions
default_crl_days        = 7                     # How long before next CRL
crl_extensions          = crl_ext               # CRL extensions

# Naming policies control which parts of a DN end up in the certificate and
# under what circumstances certification should be denied.

[ match_pol ]
domainComponent         = match                 # Must match 'fortress.lan'
```

```
organizationName        = match                 # Must match 'Fortress LAN'
organizationalUnitName  = optional              # Included if present
commonName              = supplied              # Must be present

[ any_pol ]
domainComponent         = optional
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = optional
emailAddress            = optional

# Certificate extensions define what types of certificates the CA is able to
# create.

[ email_ext ]
keyUsage                = critical,digitalSignature,keyEncipherment
basicConstraints        = CA:false
extendedKeyUsage        = emailProtection,clientAuth
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always

[ server_ext ]
keyUsage                = critical,digitalSignature,keyEncipherment
basicConstraints        = CA:false
extendedKeyUsage        = serverAuth,clientAuth
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always

# CRL extensions exist solely to point to the CA certificate that has issued
# the CRL.

[ crl_ext ]
authorityKeyIdentifier  = keyid:always
EOF
```

Generate the initial CA request (CSR)

```
openssl req -new \
  -config etc/root-ca.conf \
  -out ca/root-ca.csr \
  -keyout ca/root-ca/private/root-ca.key
```

With the `openssl req -new` command we create a private key and a certificate signing request (CSR) for the root CA. You will be asked for a passphrase to protect the private key.
The `openssl req` command takes its configuration from the [req] section of the configuration file.

Sign the CA Certificate with the root-CA

```
openssl ca -selfsign \
  -config etc/root-ca.conf \
  -in ca/root-ca.csr \
  -out ca/root-ca.crt \
  -extensions root_ca_ext
```

With the `openssl ca` command we issue a root CA certificate based on the CSR. The root certificate is self-signed and serves as the starting point for all trust relationships in the PKI. The `openssl ca` command takes its configuration from the [ca] section of the configuration file.

Create the Signing CA request (CSR)

```
openssl req -new \
  -config etc/signing-ca.conf \
  -out ca/signing-ca.csr \
  -keyout ca/signing-ca/private/signing-ca.key
```

Geneate the Signing CA certificate

```
openssl ca \
  -config etc/root-ca.conf \
  -in ca/signing-ca.csr \
  -out ca/signing-ca.crt \
  -extensions signing_ca_ext
```

With the `openssl ca` command we issue a certificate based on the CSR. The command takes its configuration from the [ca] section of the configuration file. Note that it is the root CA that issues the signing CA certificate! Note also that we attach a different set of extensions.

Creating Server Certificates

Create a Server request (CSR)

```
SAN=DNS:www.fortess.lan \
  openssl req -new \
    -config etc/server.conf \
    -out certs/fortress.lan.csr \
    -keyout certs/fortress.lan.key
```

Next we create the private key and CSR for a TLS-server certificate using another request configuration file. When prompted enter these DN components: `DC=lan, DC=fortress, O=Fortress LAN,`

CN=www.fortress.lan. Note that the subjectAltName must be specified and passed in as an environment variable. Also of note is that server keys typically have no passphrase.

Sign the TLS Server certificate

```
openssl ca \
  -config etc/signing-ca.conf \
  -in certs/fortress.lan.csr \
  -out certs/fortress.lan.crt \
  -extensions server_ext
```

How to Revoke a certificate

```
openssl ca \
  -config etc/signing-ca.conf \
  -revoke ca/signing-ca/01.pem \
  -crl_reason superseded
```

Certain events, like a certificates replacement or loss of private key, requires that a certificate to be revoked before its scheduled expiration date. The `openssl ca -revoke` command marks a certificate as revoked in the CA database. It will from then on be included in CRLs issued by the CA. The above command revokes the certificate with serial number `01` (hex). You have to know the serial number of the certificate that you wish to revoke. This assumes that you plan to public the CRL.

How to Generate a CRL

```
openssl ca -gencrl \
  -config etc/signing-ca.conf \
  -out crl/signing-ca.crl
```

The `openssl ca -gencrl` command creates a certificate revocation list (CRL).
The CRL contains all revoked, not-yet-expired certificates from the CA database.
A new CRL must be issued at regular intervals.

How to create a root certificate chain PEM bundle

```
$ cat ca/signing-ca.crt ca/root-ca.crt | \
  sed -n '/-----BEGIN CERTIFICATE-----/,/-----END CERTIFICATE-----/p' > ca/signing-ca-chain.pem
```

Backups

It should go without saying that once this structure is created and used, you will want to backup your laptop to at least 2 separate external hard drives to increase the ability of recovery when your local hard drive fails. Also, keep these drives in a safe and secure environment, e.g. locked office drawer for one and the other in a safe. You get the general idea. You don't need anyone stealing your backup to use to generate new certs for malicious purposes. If you plan on making a lot of certs, I would use the following crontab to automate backing up the cert repo into a tarball locally. You will still need to back up weekly or more frequently to an external hard drive.

Crontab Example (this is all one line)

```
15 09 * * 2 /usr/bin/tar cf - ~/PKI_20180826_fortress.lan | /usr/bin/gzip -4c >
/opt/backups/PKI_$(date +\%Y-\%m-\%d-\%Hh\%Mm\%Ss_\%A).tar.gz
```

Conclusion

By following this guide, the user has created a folder structure to create and hold new certificates on your laptop/workstation. For deployment of these certs, you will most likely want the .crt file as well as the .key file. The .crt file is your public key, and the .key file is your private key that you will not want to share or reveal to anyone. Finally add the signing-ca-chain.pem to the bundle, which can be dependent on the application using them. In example, for Elasticsearch version 6.4, you need all three certs in order to secure management and transport channels. The last nugget of wisdom here, is the magic of Wildcard certificates. You can go through and set a wildcard in the Common Name, e.g. CN=*.fortress.lan. What happens is, when you go to use that on any web server or application server in your domain '.fortress.lan', it will use the wildcard. Think about this for a second, with three files, you can secure all of your servers in one domain. This means less work for you if you choose to automate the process, i.e. using puppet/ansible/chef to deploy your encryption certificates to all of your application and web servers.