Create an Internal CA

This is the Certificate of Authority. This will be used to validate all of the later certificates you create. You will be putting part of the CA into each and every one of your machines, saying “anything signed by this is valid.”

The premier thing here is security. You will be creating a certificate that will tell everyone connecting “hey, I vouch for this web site, whatever. I trust it, it is really what it says it is.”

This is controlled through your private key, and whatever password you put on it. Anyone who has access to those two things, and can do a little DNS poisoning, can totally own you.

Large companies, like Let's Encrypt, or Thawte, or whatever, are set up so it takes three or four people, together, to get access to the CA.

At Daily Data, the password is available to the owner and the senior technician (who has worked for the company over 20 years). The private key is stored in three secure locations, and the password is written down in our company's safe deposit box.

Hope I've made it clear. Think security, especially if you are something other than an individual or a Mom and Pop (and even then, think security).

Since I work for Daily Data, I'll use that to create the names below. We have a server that has limited access and is generally turned off, and I'll use that machine. It is a Linux machine, so I'll create a directory structure under /opt.

Quick and Dirty

Ok, these instructions are just a guideline. More details follow.

```
# create a random rsa key pair of 2048 bits and ask for encryption
passphrase (min 8 char)
openssl genpkey -algorithm RSA -out dailydataCA.key -des3 -pkeyopt
rsa_keygen_bits:2048
# Create a CA certificate from it. You'll need to answer a bunch of
questions here
# see "create a config file" to keep from having to do that.
openssl req -config openssl.cnf -key dailydataCA.key -new -x509 -days 3650 -
out dailydataCA.crt -extensions CA_default
```

Details

Create the private key

The following command will generate the private key for your CA. I have used \ to make it multi-line

(it is all one command, so make sure no spaces after the \'s)

```
openssl \
   genpkey \
   -algorithm RSA \
   --des3 \
   --out DailyDataCA.key \
   --pkeyopt rsa_keygen_bits:2048
```

When run, it will generate a 2048 bit private key, then ask you for a passphrase (then again to verify). Here is a breakdown of the parameters:

- *genpkey* - openssl has multiple functions. This says you want to generate a private key. Notice there is no dash before the command.
- *-algorithm* - Generate using RSA. You should use this for all private keys unless you know a reason not to.
- *-des3* - use triple-des (des3) to encrypt the key. Will ask for passphrase at end. Minimum of 8 characters, but more is better (like 20 something)
- *-out* - followed by the file name to put the private key in. If not specified, will send output to STDOUT. I use .key as the suffix
- *-pkeyopt* - options specific to the key you are generating. In this case, we are saying the rsa key will be 2048 bits in length

Note: you can use the -pass parameter to accept the password from a file (or stdin). See the options in the *Pass Phrase Options* of *man 1 openssl* on unix machines

Note: you do not have to use a password if you can ensure the key is secure at all times. Simply remove *-des3* from the command.

Note: The old way of generating keys was to use the command

```
openssl genrsa -des3 -out DailyDataCA.key 2048
```

but that has been supercseded by genpkey.

Create the CA Cert

Now that you have a private key, we can use that to create a certificate to be used to sign the certificates. This command assumes you have Create an SSL Configuration File.

The file created (with a .crt suffix) will also be added to each device that needs to access certificates generated. So, for example, workstations where people are accessing internal web sites which will have certificates signed by the CA.

```
openssl \
   req \
   -x509 \
   -new \
   -key DailyDataCA.key \
   -sha256 \
```

```
    -days 3650 \
    -config openssl.cnf \
    -extensions CA_default \
    -out DailyDataCA.crt
```

This will read the key file (.key) and generate a certificate from it. Parameters are:

- *req* - We are doing a certificate request
- *-x509* - we want an X509 certificate created. This is a self-signed certificate (instead of a certificate request). Required for generating a CA
- *-new* - Create a new certificate. This will require you to answer questions to generate a Distinguished Name (DN) if you did not create a config file with that information.
- *-key* - the name of the keyfile created earlier
- *-sha256* - The digest used to create the certificate. This is the default for RSA and only here for documentation
- *-days* - The number of days the certificate is valid. Before this time is up, a new CA must be generated, deployed to all workstations and new certs signed by the new key deployed to all services. Default is 30 days, but we set it to 10 years.
- *-config* - Name of the configuration file to use (if you created one).
- *-extensions* - override the default req_ext and use CA_default instead
- *-out* - name of the output file.

View Cert

You can view the certificate you created using the -text. With this, you can see the issuer (itself, self signed), the Signature Algorithm, the DN (Distinguished Name, the line starting with Subject:) and information about the public key and signature.

```
openssl x509 -in ca.crt -text -noout
```

Install CA on workstations

You are now ready to install the CA's on workstations. **Note**: only install the .crt (the actual certificate file). The key file (.key) is private and should be stored in a secure location.