T: 0191 222 5000
E: enquiries@netskills.ac.uk
www.netskills.ac.uk

Using OpenSSL

Tips for CSC3504

Creating Certificates and Keys

OpenSSL is a freely available, open-source toolkit for all things encryption & SSL. It provides simple command-line tools to enable you to create, read and manage keys, certificates and other associated files. It is available on most Linux distributions and can also be downloaded for Windows.

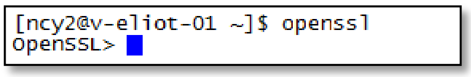
Once the client is running (i.e. you are at the `OpenSSL>` command prompt all versions behave in exactly the same way.

Running OpenSSL

Login to your VM and from within a suitable directory run the command:

```
openssl
```

Your command prompt should change to show you are now using the OpenSSL client:



```
[ncy2@v-eliot-01 ~]$ openssl
OpenSSL>
```

Figure 1. OpenSSL prompt

Generate a CSR

There are many OpenSSL command line incantations that will generate you a CSR (Certificate Signing Request). One pretty reliable one is below:

```
req -out CSRFILENAME -pubkey -new -keyout KEYFILENAME
```

This will create CSR and private key files in the current directory.

It will require you to provide a password to protect your private key, before prompting you for the subject information for the certificate request.

Another password?

This password is just for access to your private key.

In production, this should be a fairly robust string, but for your purposes (and because you're going to be using it a lot in a short space of time)…

…choose something simple that you can remember!

Use the CSR to create a certificate… and sign it with your private key

Again, there are several ways to do this, but you can simply do:

```
req -x509 -days 365 -key KEYFILENAME -in CSRFILENAME
-out CERTIFICATENAME
```

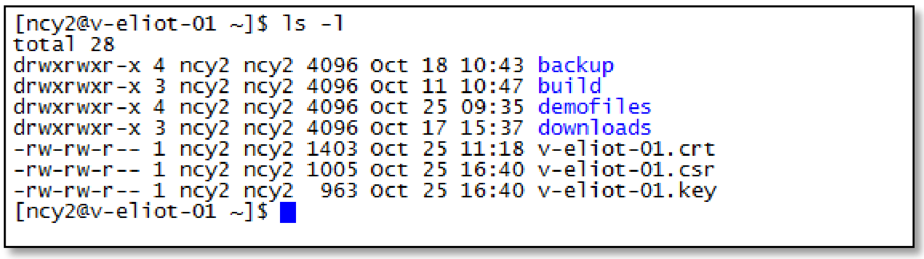
You will be prompted for the password you set earlier to use your private key.

Quit OpenSSL

To quit OpenSSL, type…… **`quit`** ☺

Check out the new files

Get a listing of your directory to see the new files you can now use in your server configuration.



```
[ncy2@v-eliot-01 ~]$ ls -l
total 28
drwxrwxr-x 4 ncy2 ncy2 4096 Oct 18 10:43 backup
drwxrwxr-x 3 ncy2 ncy2 4096 Oct 11 10:47 build
drwxrwxr-x 4 ncy2 ncy2 4096 Oct 25 09:35 demofiles
drwxrwxr-x 3 ncy2 ncy2 4096 Oct 17 15:37 downloads
-rw-rw-r-- 1 ncy2 ncy2 1403 Oct 25 11:18 v-eliot-01.crt
-rw-rw-r-- 1 ncy2 ncy2 1005 Oct 25 16:40 v-eliot-01.csr
-rw-rw-r-- 1 ncy2 ncy2  963 Oct 25 16:40 v-eliot-01.key
[ncy2@v-eliot-01 ~]$
```

Figure 2. Directory listing showing new key, certificate and signing request files

More OpenSSL commands

If you're interested... the following *should*:

Decode and read a certificate:

```
x509 -text -in vm-eliot-01.crt
```

Decode and read a CSR

```
rsa -text -in vm-eliot-01.csr
```

Decode and read a key

```
asn1parse -in vm-eliot-01.key
```

Learn more

There are several ways you can get OpenSSL to do your bidding.

For example, it is perfectly possible to create a certificate key pair in one step and you will certainly find online guides that show you how to do so , thus skipping the CSR -> signing step.

You'll find lots more examples at sites like:

http://www.madboa.com/geek/openssl

Do have look at the main OpenSSL site too – but notice that the documentation here isn't particularly easy to follow!

http://www.openssl.org

Configuring Apache

You should capable of figuring out how to do this but to help you…

Quick recipe

1. Check for (and if necessary compile in) `mod_ssl`
2. Copy your new certificate and key to `conf` directory
3. Set up server configuration in appropriate file(s)
4. Stop/start server
5. Test access at: `https://vm-eliot-NN.ncl.ac.uk`

Private key passphrase?

This is the password you set when you made your private key and Apache is now going to ask you for it ***every*** time you restart it.

If this becomes too much for you, look up how you can remove the password using OpenSSL (you may also find how to create a key that doesn't have one in the first place!).

Whilst this is not considered best practice – a compromised private key can be used if there is no password protecting it – this is a common technique used by many web managers.

The security for the key is provided instead by its location on the server and the permission set at the operating system level. The thinking is, that if someone has compromised your system to the level at which they can get hold of the private key, you have much bigger problems ☺