TLS CA and server key HOWTO

TLS CA and server key HOWTO

Justin Davies

v1.0, December 2002

Secure Socket Layer is a communications encryption system. It is commonly used to encrypt a http link for secure on−line transactions, but can be used for any communications protocol, such as e−mail, telnet, FTP etc. SSL uses certificates to encrypt and verify a connection session.

Next Previous Contents Next Previous Contents

---

1. Introduction

SSL certificates follow the PPK model (Public/Private key). To establish a connection, a public and private certificate is used to verify and encrypt a session. To verify these certificates, a Certificate Authority (CA) is used to sign them. A CA is a known and trusted third party that signs certificates and allows the hosts participating in the communications to be confident that they are both authorised by a separate entity (the CA).

There are a few CAs on the Internet, the most popular being Verisign (www.verisign.com). The get a CA key, you must apply to a CA and in most cases you must also pay the CA for their service. You are able to become your own CA, which means you can sign newly created certificates yourself. This is acceptable if an organisation is offering services to employees as both parties trust the CA (the organisation). For public SSL sites it is advisable that you apply for a CA certificate from a known authority so that a client can present the server certificate as authentic to the end user. In the following pages, we will create self signed certificates based on a self created CA.

Next Previous Contents Next Previous Contents

---

2. SSL on Linux

The most popular open source SSL implementation is OpenSSL. It is in the n (networking) series of the SuSE distribution, this will be different for other distributions as the *package series* is centric to SuSE.

Note: OpenSSL is only available in the European SuSE distributions. This is due to American import restrictions on munitions.

Next Previous Contents Next Previous Contents

3. Setting up a Certificate Authority

Once you have installed the OpenSSL package, change directory to /usr/ssl/misc (again, may be different based on your distribution).The OpenSSL distribution provides a perl script that greatly simplifies the creation of a CA, certificate requests and certificate signing. In the misc directory execute the following:

```
root@zen:/usr/ssl/misc > ./CA.pl −newca
CA certname (or enter to create)
Making CA certificate ...
Using configuration from /usr/ssl/openssl.cnf
Generating a 1024 bit RSA private key...++++++ ....++++++
writing new private key to ./DemoCA/private/cakey.pem
Enter PEM pass phrase:
Verifying password −
Enter PEM pass phrase:

−−−−−

You are about to be asked to enter information that will be incorporated into your certi
request.
What you are about to enter is what is called a Distinguished Name or a DN. There are qu
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 blan
−−−−−
Country Name (2 letter code) [UK]:UK
State or Province Name (full name) [Some−State]:Herts
Locality Name (eg, city) []:London
Organization Name (eg, company) [Internet Widgits Pty Ltd]:SuSE Linux UK Ltd
Organizational Unit Name (eg, section) []:SUSEUK
Common Name (eg, YOUR name) []:SuSE Linux UK
Certificate Authority Email Address []:justin@suse.co.uk
```

This creates the CA certificate and a private key. It is very important to use a good, solid pass phrase for the certificate as anyone who has access to the certificate can fake an authentic certificate from your CA. The default location of the CA files is in the ./DemoCA directory. To change this you will need to edit the CA.pl script to make the CA in a different directory. We will go with the default here as it helps keep things nice and simple.

Next Previous Contents Next Previous Contents

4. Creating a server key

Once we have created the CA, we need to create a certificate for the a server or a client (the way to make these is exactly the same).

We need to create a certificate request. This creates a certificate that requests to be signed (this is not an automatic process, and is handled by the sign process later on).

In the misc directory execute:

```
root@zen:/usr/ssl/misc > ./CA.pl −newreq
Using configuration from /usr/ssl/openssl.cnf
Generating a 1024 bit RSA private key...........................++++++ ....++++++
writing new private key to newreq.pem
−−−−−
You are about to be asked to enter information that will be incorporated into your certi
request.
What you are about to enter is what is called a Distinguished Name or a DN. There are qu
few fields but you can leave some blankFor some fields there will be a default value, If
the field will be left blank.
−−−−−

Country Name (2 letter code) [UK]:UK
State or Province Name (full name) [Some−State]:Herts
Locality Name (eg, city) []:London
Organization Name (eg, company) [Internet Widgits Pty Ltd]:SusE Linux UK Ltd
Organizational Unit Name (eg, section) []:SUSEUK−SERVER
Common Name (eg, YOUR name) []:mail.suse.co.uk
Email Address []:postmaster@suse.co.uk
Please enter the following extra attributesto be sent with your certificate request
A challenge password []:
An optional company name []:
Request (and private key) is in newreq.pem
```

Notice the value used for the Common Name. It is the FQDN of the machine this certificate will be used on. If this is used as a server machine, the client will lookup the host name given by the certificate to see if it is indeed connecting to the machine the certificate was made for. This is not applicable to a certificate for a client as it is unlikely the hostname of the machine can be resolved.

It is advisable to still use the host name of the client if it has one to uniquely identify the certificate in transactions. If you are using the certificate in an automated client/server model, it is not going to be possible to setup a pass phrase for the certificates as the connection cannot be initiated until the pass phrase has been entered. The certificate is always encrypted using a private key that is stored in the certificate request constructed via the *CA.pl −newreq* command. You will need to specify the new request (**newreq.pem**) as the private key to use in all client transactions regarding this certificate. It is advisable to rename this file to something meaningful like the host name of the machine it is being used on (in my case I called it *zen.suse.co.uk.key*). You can also concatenate the certificate and key together into one manageable file. To do this, just issue the following command:

```
cat newcert.pem newreq.pem > zen.suse.co.uk.pem
```

You now have one file with the private key and the certificate in it.You can edit the new file and take out the data between the **BEGIN CERTIFICATE REQUEST** and **END CERTIFICATE REQUEST** inclusive to tidy up the file. All certificates must be signed by your CA to provide the authentication of the certificates you use in your system.

4.1 Caveats

One thing that is useful to know is that OpenSSL is sometimes emphatic about how you name a certificate. It does this to use a hash to lookup a certificate in a directory. The hash is generated by issuing the c_rehash /path/to/certificates command. This generates something like:

```
root@zen:~ > c_rehash /usr/ssl/misc/
Doing /usr/ssl/misc/
newcert.pem => 66be5b2a.0
```

This creates a symbolic link to the certificate with the link being the 8 byte hash of the certificate. This is what the OpenSSL library looks for when it loads the certificate.

Next Previous Contents Next Previous Contents

---

5. Links

- OpenSSL Home
- SuSE Home
- Home of the Postfix/TLS HOWTOS

Next Previous Contents