Creating SSL (Self signed Certificate)

This document describes how to sign your own SSL certificate requests using the OpenSSL toolkit and use these self-signed certificates to allow HTTPS connections to Microsoft's IIS 5 web server (as supplied with Windows 2000).
If you know what a self-signed certificate is and understand the concept of a certificate authority, great. If not, this should still work but you'll have no idea what you've acheived when it does :)
Command transcripts are shown in monospaced type, with the bits you type shown in bold. *Bits in italics are comments to explain what's going on and what you should be doing.*

Disclaimer

I'm by no means a security expert, and I'm not an OpenSSL guru. If you find these notes helpful, great - if you don't, there's plenty of more detailed resources out there which **will** answer your questions if you take the time to read them properly. Contributions and testimonials are welcome; questions will be read and possibly answered but I'm making no guarantees, and please don't rely on this information for anything important. I don't know whether it's the most secure or most effective way of doing this, but it works and that's good enough for me. If it's not good enough for you, don't use it :)
These instructions were tested using OpenSSL 0.9.6g (v1.0 Final) on Windows 2000 Server running Service Pack 3.

Ingredients

- Windows 2000 running Internet Information Services (IIS)
- The OpenSSL tools for Windows from Shining Light Productions. This is a Windows port of the popular OpenSSL toolkit.

Walkthrough

Install and configure the OpenSSL toolkit

1. Get OpenSSL from the address above, and run the installer, accepting the defaults. These instructions assume OpenSSL is installed in **C:\OpenSSL**.
2. Add **C:\OpenSSL\bin** to your system path (Control Panel, System, Advanced, Environment Variables, System Variables) - this isn't strictly necessary but it makes things a lot easier.
3. Create a working directory - here, we'll use **c:\ssl** as our working folder.
4. Download this copy of openssl.conf to your working folder. (**Note:** I have no idea what most of the options in this file mean. I just hacked it around until it worked...)
5. Set up the directory structure and files required by OpenSSL:

6. **C:\ssl>md keys**

   C:\ssl>md requests

   C:\ssl>md certs

7. Create the file **database.txt** - an empty (zero-byte) text file. This can be done using the 'touch' command if you have it (it's a Unix tool not available on Windows by default, but you might have one lying around), or by creating an empty file manually:
8. **c:\ssl>copy con database.txt**
   ^Z

   C:\ssl>

9. MS-DOS veterans will recognise this particular invocation. We're copying from CON (the console) to a file called database.txt, and that's a Control-Z end-of-file character on the first line. This should produce a zero-byte file called c:\ssl\database.txt
10. Create the serial number file serial.txt. This is a plain ASCII file containing the string "01" on the first line, followed by a newline. Again, we can use a little bit of ancient DOS magic:
11. **C:\ssl>copy con serial.txt**
    01
    ^Z

    C:\ssl>

12. to achieve the desired effect. (That's keystrokes zero, one, return, control-Z, return)

Set up a Certificate Authority (CA)

1. First, we create a 1024-bit private key to use when creating our CA.:
2. **C:\ssl>openssl genrsa -des3 -out keys/ca.key 1024**
   Loading 'screen' into random state - done
   warning, not much extra random data, consider using the -rand option
   Generating RSA private key, 1024 bit long modulus
   .....++++++
   .....++++++
   e is 65537 (0x10001)
   Enter PEM pass phrase: *- choose a memorable pass phrase to use for this key*
   Verifying password - Enter PEM pass phrase: *- type your pass phrase again for verification*

3. The pass phrase will be requested whenever you use this certificate for anything, so make sure you remember it. This will create a file called **c:\ssl\keys\ca.key,**containing our certificate authority private key.
4. Next, we create a master certificate based on this key, to use when signing other certificates:
5. **C:\ssl>openssl req -config openssl.conf -new -x509 -days 1001 -key keys/ca.key -out certs/ca.cer**
   Using configuration from openssl.conf
   Enter PEM pass phrase: *- type your passphrase here.*
   You are about to be asked to enter information that will be incorporated into your certificate request.
   What you are about to enter is what is called a Distinguished Name or a DN.
   There are quite a 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 blank.

   Country Name (2 letter code) []:GB
   State or Province Name (full name) []:Hampshire
   Locality Name (eg, city) []:Southampton
   Organization Name (eg, company) []:dylanbeattie.net
   Organizational Unit Name (eg, section) []:
   Common Name (eg, your websites domain name) []:ssl.dylanbeattie.net
   Email Address []:ssl@dylanbeattie.net

   C:\ssl>

6. This will create our CA certificate and store it as c:\ssl\certs\ca.cer
7. (optional) Finally, we export our CA certificate in PKCS12 format - this will allow Windows users to import the PKCS12 certificate into their Trusted Root Store, so they don't get warning messages every time they use one of our certificates. **From the OpenSSL FAQ:**
8. 12. How do I install a CA certificate into a browser?
9. The usual way is to send the DER encoded certificate to the browser as MIME type application/x-x509-ca-cert, for example by clicking on an appropriate link. On MSIE certain extensions such as .der or .cacert may also work, or you can import the certificate using the certificate import wizard.
10. You can convert a certificate to DER form using the command:
11. **openssl x509 -in ca.pem -outform DER -out ca.der**
12. Occasionally someone* suggests using a command such as:
13. **openssl pkcs12 -export -out cacert.p12 -in cacert.pem -inkey cakey.pem**
14. **DO NOT DO THIS!** This command will give away your CAs private key and reduces its security to zero: allowing anyone to forge certificates in whatever name they choose.
15. * Guilty as charged - sorry! This guide originally recommended the insecure method warned about above. Thanks to Baahl for pointing out the error and Marco Fagiolini for the correct method.

Create an IIS Certificate Request

This is described in detail elsewhere on the web - see Microsoft Knowledge Base Article Q228821. You should end up with a file called certreq.txt.

Sign the Certificate Request

1. Copy the **certreq.txt** file into c:\ssl\requests
2. Sign the request
3. **C:\ssl>openssl ca -policy policy_anything -config openssl.conf -cert certs/ca.cer -in requests/certreq.txt -keyfile keys/ca.key -days 360 -out certs/iis.cer**
   Using configuration from openssl.conf
   Loading 'screen' into random state - done
   Enter PEM pass phrase:
   Check that the request matches the signature
   Signature ok
   The Subjects Distinguished Name is as follows
   commonName :PRINTABLE:'myCommonName'
   organizationalUnitName:PRINTABLE:'myOrganisationalUnit'
   organizationName :PRINTABLE:'myOrganisation'
   localityName :PRINTABLE:'myLocality'
   stateOrProvinceName :PRINTABLE:'myProvince'
   countryName :PRINTABLE:'GB'
   Certificate is to be certified until Feb 2 01:13:14 2004 GMT (360 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

   C:\ssl>

4. Let's just take a look at those command-line options in a bit more detail:
   -policy policy_anything - specifies that we're using the 'policy_anything' policy from our **openssl.conf** file. This is a relaxed policy in which the name, country, etc. in the certificate don't need to match those used by the certification authority. Use **-policy policy_match** for a more restrictive CA.
   -config openssl.conf - specifies we're reading our configuration from **openssl.conf** in the current directory.
   -cert certs/ca.cer - specifies we're using our CA master certificate to sign the request.
   -in requests/certreq.txt - the certificate request we're signing.
   -keyfile keys/ca.key - the private key for our CA master certificate, which proves we're allowed to use it.

-days 360 - the time until the certficate will expire
-out certs/iis.cer - the file in which to place our newly-signed certificate

5. Convert the signed certificate into x509 format for use with IIS:
6. **C:\ssl>openssl x509 -in certs/iis.cer -out certs/iisx509.cer**

   C:\ssl>

7. This will leave the new certificate in **c:\ssl\certs\iisx509.cer** - signed, sealed and ready to install

Install the new certificate under IIS

Again, this is described elsewhere on the web - remember that the **iisx509.cer** file is our **certificate response file**, and the instructions in Knowledge Base article 228836should make everything clear.