



Creating Self-Signed Certificates for
Testing SSL and Other Applications
Requiring Security

1 Document History

Date	Version	Change Description	
10/8/09	V1.0	Initial Entry	

2 Table of Contents

1	Document History	2
2	Table of Contents	3
3	Introduction	4
3.1	Problem Solved	4
4	Creating Certificates	4
5	Conclusion	6
6	References	6

3 Introduction

This paper describes a method for creating self-signed certificates for use in testing applications requiring keys and certificates. This paper assumes the use of the OpenSSL product. OpenSSL is not the only product available for creating certificates, but it is the one that was chosen for use in writing this document.

3.1 Problem Solved

When testing applications that implement SSL (secure sockets layer) software engineers sometimes need either client or client and server certificates for testing the efficacy of their application. This paper describes a method for creating certificates that can be used in NET+OS sample applications such as `nahttps_pd`. It should be noted that these certificates are NOT created by a trusted CA (certificate authority) and thus should be used for testing purposes only.

The information in this paper is based on the O'Reilly book entitled Network Security with OpenSSL by Viega, Messier and Chandra.

4 Creating Certificates

The following are the steps required for creating self-signed certificates for use with SSL. For each step you'll be asked questions for data to be placed into the certificates and keys. Be consistent. For the "your name" field for certificates to be used for the secure web server (NET+OS sample application), use the IP address of the device to which you'll be connecting.

For reference throughout this document, the default extension (the one we recommend) to give PEM format files is '.pem'. The default extension (the one we recommend) to give pkc12 files is '.pfx'.

- 1) Install OpenSSL (assuming you have not already installed it on your PC or UNIX box). It is available for MS Windows and various flavors of UNIX. Start at <http://www.openssl.org/>. If you are running windows, the URL that I used for an MS windows-compatible implementation of OpenSSL is at the following URL: <http://www.shininglightpro.com/products/Win32OpenSSL.html>. Additionally keep track of the directory into which OpenSSL is installed. You may need to pull a couple of files from that directory into the directory in which you are creating certificates and keys.
- 2) Create a certificate authority certificate and key. All certificates created will be signed against this.

```
openssl req -x509 -newkey rsa -keyout <ca key file name> -out <ca certificate file name> -outform PEM
```

You will be asked lots of questions.

Please note the <ca key file name> represents the certificate authority key file name. For example you might replace <ca key file name> with mycakeyfile.pem. You might replace <ca certificate file name> with mycacertfile.pem. Feel free to be creative, but including ca and certificate or key might help you remember the purpose of the file.

- 3) Create a certificate request. This can be for client or server. If you are running these on a device, make the “your name” field the ip address of the device (you’ll need one for the client and one for the server). Perform the following steps. It is best to include the word client or server in the name of the file.

```
openssl req -newkey rsa:1024 -keyout <key file name> -keyform PEM -out <certificate file name> -outform PEM
```

Please note that <key file name> represents the key file request and <certificate file name> represents the certificate file request. Also remember that you will be creating both server and client keys and certificates. So you might want to use file names such as myclientkeyreq.pem, myclientcertreq.pem, myserverkeyreq.pem and myservercertreq.pem.

- 4) Create the self-signed certificate from the existing client or server certificate and the ca certificate and key (again one for the client and one for the server). Perform the following steps to create a self signed certificate. Remember you’ll need one for the client and one for the server. It is best when naming them to include the word client and server in the file name.

```
openssl ca -n <the request file created in step 2> -cert <the ca certificate created in step 1> -outdir <where you put all certs and where the new cert will go> -out <the file name of the new certificate>
```

Again, <the request file created in step 2> would be replaced with the name of the certificate request file created in step 2. <the ca certificate created in step 1> would be replaced by the name of the certificate authority certificate file created in step 1. A suitable file name, naming the newly created self signed certificate would replace <the file name of the new certificate>. Again please remember that you’ll want to create certificates for both client and server so chose you output file names wisely.

- 5) The CLIENT certificate must be placed into pkc12 format to be accepted by IE. Perform the following steps to convert the CLIENT certificate into pkc12 format:

```
openssl pkcs12 -export -out <file name for the new pkcs12 format client cert> -in <the existing PEM format client certificate> -name "some name for this cert" -inkey <file name of client key>
```

You'll be asked for a password for the client key. I hope you wrote this down.

You'll be asked for a password to store in the pkcs12 file. You'll be asked this when you try and install this into IE.

Also, <file name for the new pkcs12 format client cert> is the file name of the certificate in pkcs12 format (output). <the existing PEM format client certificate> is the CLIENT certificate created in step 4. Remember the client certificate ONLY needs to be converted to pkcs12 format and ONLY if you are using MS IE. <file name of client key> represents the PEM-format CLIENT key file created in step 3.

5 Conclusion

Conclusion, it takes a couple of steps but if needed, you can successfully create client and server certificates using OpenSSL. Hopefully this document has given you the knowledge and confidence to create your own certificates.

6 References

Viega, John, Messier, Matt and Chandra Pravar. Network Security with OpenSSL. Sebastopol, CA. : O'Reilly Media, Inc., 2002