Show

Generating a PKCS#12 Private Key and Public Certificate

This article discusses how to generate a PKCS#12 private key and public certificate file that is suitable for use with HTTPS, FTPS, and the administrative port for Secure FTP Server-FIPS. (To convert an incompatible PKCS#12 format file, refer to Converting an Incompatible PKCS#12 Format File to a Compatible PKCS#12.)

General Information

- When operating in a FIPS-approved mode, PKI key/certificates must be between 1024- bits and 4096-bits, inclusive.
- The supported cipher combinations allowed for SSL negotiation are limited to:
  - SSLv3/TLSv1 - RSA Key Exchange, RSA Authentication, 256 bit AES encryption, and SHA1 HMAC
  - SSLv3/TLSv1 - RSA Key Exchange, RSA Authentication, 168 bit 3DES encryption, and SHA1 HMAC
  - SSLv3/TLSv1 - RSA Key Exchange, RSA Authentication, 128 bit AES encryption, and SHA1 HMAC

  Each of the above combinations uses RSA key exchange; therefore, RSA based key/certificates must be used.
- In FIPS Mode, the PKCS#12 format must use compatible encryption and hashing algorithms when encrypting the file. The necessary strong encryption will use 3DES and SHA1 encryption.

Procedure

These instructions assume you have downloaded and installed the Windows binary distribution of OpenSSL. Refer to Using OpenSSL for the general instructions

1. Generate an RSA private key:

   ```
   >C:\Openssl\bin\openssl.exe genrsa -out <Key Filename> <Key Size>
   ```

   Where:

   - `<Key Filename>` is the desired filename for the private key file
   - `<Key Size>` is the desired key length of either 1024, 2048, or 4096

   For example, type:

   ```
   >C:\Openssl\bin\openssl.exe genrsa -out my_key.key 2048
   ```

2. Generate a Certificate Signing Request:

   In version 0.9.8g:

   ```
   >C:\Openssl\bin\openssl.exe req -new -key <Key Filename> -out <Request
   Filename> -config C:\Openssl\bin\openssl.cnf
   ```

   -OR-

   In version 0.9.8h and later:

   ```
   >C:\Openssl\bin\openssl.exe req -new -key <Key Filename> -out <Request
   Filename> -config C:\Openssl\bin\openssl.cfg
   ```

   Where:

   - `<Key Filename>` is the input filename of the previously generated private key
   - `<Request Filename>` is the output filename of the certificate signing request

   For example, type:

   ```
   >C:\Openssl\bin\openssl.exe req -new -key my_key.key -out my_request.csr -
   config C:\Openssl\bin\openssl.cnf
   ```

3. Follow the on-screen prompts for the required certificate request information.
4. Generate a self-signed public certificate based on the request

   ```
   >C:\Openssl\bin\openssl.exe x509 -req -days 3650 -in <Request Filename> -
   signkey <Key Filename> -out <Certificate Filename>
   ```

Where:

- <Request Filename> is the input filename of the certificate signing request
- <Key Filename> is the input filename of the previously generated private key
- <Certificate Filename> is the output filename of the public certificate

For example, type:

```
>C:\Openssl\bin\openssl.exe x509 -req -days 3650 -in my_request.csr -signkey
my_key.key -out my_cert.crt
```

5. Generate a PKCS#12 file:

```
>C:\Openssl\bin\openssl.exe pkcs12 -keypbe PBE-SHA1-3DES -certpbe PBE-
SHA1-3DES -export -in <Public Certificate Filename> -inkey <Private Key
Filename> -out <PKCS#12 Filename> -name "<Display Name>"
```

Where:

- <Public Certificate Filename> is the input filename of the public certificate, in PEM format
- <Private Key Filename> is the input filename of the private key
- <PKCS#12 Filename> is the output filename of the pkcs#12 format file
- <Display Name> is the desired name that will sometimes be displayed in user interfaces.

For example, type:

```
>C:\Openssl\bin\openssl.exe pkcs12 -keypbe PBE-SHA1-3DES -certpbe PBE-
SHA1-3DES -export -in my_cert.crt -inkey my_key.key -out my_pkcs12.pfx -name
"my-name"
```

6. (Optional) Delete unneeded files. At this point, you only need the PKCS#12 format file, so you can delete the certificate signing request (.csr) file, the private key (.key) file, and the public certificate (.crt) file.

The resulting PKCS#12 format file may now be used within Secure FTP Server - FIPS.