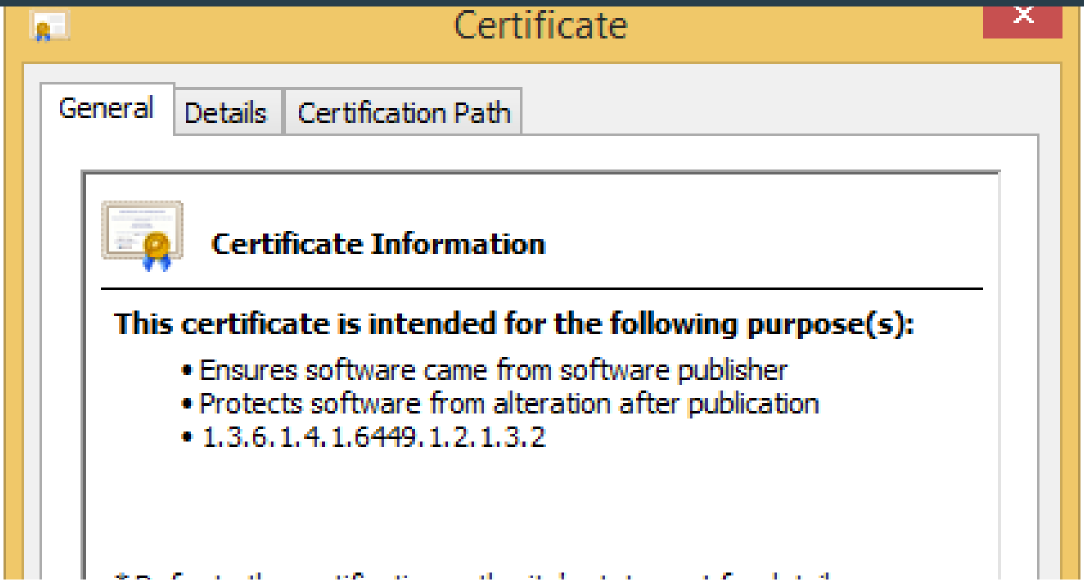

This certificate is intended for the following purpose(s):
- Ensures software came from software publisher
- Protects software from alteration after publication
- 1.3.6.1.4.1.6449.1.2.1.3.2

Code Signing in Visual Studio

Published on March 4, 2015

Jason Brower Follow
CTO | CEO of POS & Payment Processing Software

14 6 0

Strong Name and Trusted Publisher

Note: After teaching myself the exact same process to sign Windows assemblies multiple times in 6 years I decided to put together these notes for myself so when my current code signing certificate expires in two years I remember how to do this. That said, I tried to clean up the notes to share with everyone but please excuse any typos. The process can be more involved than anyone would expect and Google provides only fragments of help.

Trust me that this document will save a lot of your valuable time when trying to sign assemblies using a certificate that you have purchased from a CA such as COMODO. In fact, it can be such a time consuming process if you don't follow this you may give up and decide not to sign your assemblies which is in fact what we often find with software. Signing your assemblies shows the work of a professional.

First and foremost, there are two types of signing that occur with assemblies. Strong Name Signing (sn.exe) and Signing as a Trusted Publisher using signtool.exe. The visual studio tool "sn" is used to strong name sign an assembly (the one which is used when you check the box in Visual Studio to sign the assembly) while “signtool” is used to sign the assembly so that it can be trusted on other computers showing that you the software developer are in fact who you say you are and the software can be trusted knowing that you the publisher can be tracked down. Once you understand those two sign each assembly with a strong name using Visual Studio directly and then run a script after to sign each assembly that will be deployed to a customer's system during production so that it contains a certificate for you, the trusted publisher.

Best and Cheapest Place to Buy a Code Signing Certificate

www.ksoftware.net, a reseller for COMODO is where I always get mine. You can get a two year certificate for $175.00 as of 2015. Note you must get a code signing certificate as other certificate types will not work for code signing. Once you have purchased the certificate, COMODO will handle your order from there including verifying just exactly that you are who you say you are.

Once you have your certificate you will get an email which will provide directions to install the certificate in your computer's trusted store using your browser. You will want to install this on the same computer you requested it from. Once the install has completed in your browser (use Internet Explorer for this tutorial) you will continue with the following process which will transform your certificate into a format that is usable in both sn.exe as well as signtool.exe.

PART1: Create a Fully Chained PFX File for backup Purposes Only

This tutorial uses Internet Explorer version 11.0 running on Windows 8 but is similar in other browsers and versions of Windows. Before running through these steps, follow the directions in the email provided to you by your CA to install the certificate on your computer. You should install it on the same computer it was requested from. Note that at the time this tutorial was written, following the Visual Studio directions provided by COMODO did not result in a successful strong name signing which led me on a wild goose chase resulting in this tutorial.

1. From Internet Explorer select Tools->Internet Options.
2. Select the Content Tab
3. Click the certificates button and scroll through the list until you find the certificate that was just issued to you. If you don't find it then you likely didn't import it per the directions for COMODO. Once you see your certificate, select it and click on export.
4. The certificate wizard will open, click next and select the option to **"Yes, export the private key".**
5. Assure that Personal Information Exchange **PKCS is selected** and **"Include all certificates in the certification path if possible"** option as well as the **"Export all extended properties"** options are checked and click next.

characters as it can cause issues in scripting. Click next.

7. Save this fully chained file in a safe location as “chained.pfx” in case you ever need to reinstall it on your computer after a potential loss of data.

Now that you have saved the file for backup, place that file in a safe place as we do not need it for the remaining portion of this tutorial.

PART2: Transform your certificate into a format that is usable by both sn.exe and signtool.exe

To void potential confusion, please note that we do not use the file from PART1 in any of these steps. Move it aside as the chaining information in the backup file will cause obscure errors in the sn.exe strong name signing process.

It is time to create a file that you can use with the strong name tool (sn.exe) as well as the sign tool (signtool.exe). *Note: that I have found that the Fully Chained PFX file indeed works with signtool.exe however it does not work with the strong name signing tool sn.exe. For that reason, I use the final file from this tutorial to both create a strong name and sign the assembly.*

1. Repeat the process in **PART1** used to create the fully chained PFX file **BUT DO NOT INCLUDE** "all certificates in path" **AND DO NOT INCLUDE** "extended properties”. Yes, **include** the private key. For the sake of sanity, use the same password you used to create the fully chained file in PART1 and remember no special characters. Save this file as "unchained.pfx".

2. Now you must use openssl.exe to create a PFX file in a format that Visual Studio and sn.exe does not choke on. You can download openssl for windows via Cygwin at http://www.cygwin.com/ . Once you have installed Cygwin using the default features please open a command prompt in Cygwin with the shortcut that was provided during install. From the Cygwin command prompt navigate to the location of your unchained.pfx key you just created. Again, please note that we will be using the unchained.pfx file and **not** the chained.pfx file.If you’re not familiar with Cygwin it lets you run commands in a shell in a Unix / Linux like environment. As a tip, your "C" drive in Cygwin is at the following virtual path /cygdrive/c/.

3. After you have navigated to the location of your unchained.pfx file using the Cygwin command prompt you are going to run the following two commands. In both cases and for the sake of sanity enter the same password you have used throughout this entire process. You are going to be prompted for the password a total of 6 times. Make sure you get it right. Note the dollar sign below represents the command prompt.

```
$ openssl.exe pkcs12 -in unchained.pfx -out keyopenssl.key
```

```
$ openssl.exe pkcs12 -export -out strongname.pfx -keysig -in keyopenssl.key
```

After you have run the two commands above in the Cygwin prompt you FINALLY have a usable PFX file that will work with both strong name signing sn.exe as well as trusted publisher signing with signtool.exe

Using the new Certificate to Strong Name Sign the Assemblies

Navigate to the properties of any assembly project in Visual Studio and click on the signing tab. Click "sign the assembly" and select the last file you created using openssl called the **strongname.pfx** file. When prompted enter that same password again. Now you understand why I said choose the same one for sanity? Once you build your solution using Visual Studio you could potentially encounter the following errors.

- Any assembly that you sign with a strong name which includes other assemblies will require the other assemblies also to be signed with a strong name. That stated, you could potentially see an error telling you that the other assemblies are not signed. It can be a tedious process to assure all assemblies are strong name signed in a large project but you should do it.
- There may be a chance that you get an error when compiling with visual studio that refers to containers starting with "VS_KEY_XYZ" where XYZ is an alpha numeric string. If that error occurs then using the Visual Studio command prompt run the following command. Note I said Visual Studio command prompt and not Cygwin.

```
sn -i unchained.pfx VS_KEY_XYZ
```

Note that the sn.exe tool can be found in the Windows Kits and that you should include the full path to the unchained file in the command above as I have not tested in any other way. Also, replace the XYZ suffix with the alpha numeric string reported in the Visual Studio error if you get it.

Using the new Certificate to Sign your Assembly as a Trusted Publisher

The following dos command should be run in a dos window and not Cygwin. Note that in the example below anything with a “%” symbol prefixed and suffixed to it is a dos variable. Simply replace that with the actual values for your setup. Also, **xys.dll** is the name of the assembly you want to sign. On my example system the signtool.exe can be found at "C:\Program Files (x86)\Windows Kits\8.0\bin\x86\signtool". In this step, if you used any special characters in your password when creating the **strongname.pfx** you used. That is outside the scope of this tutorial.

```
“C:\Program Files (x86)\Windows Kits\8.0\bin\x86\signtool.exe” sign /f
%STRONGNAME_KEY_PATH%\strongname.pfx /p
%STRONGNAME_PASSWORD% %PATH_TO_ASSEMBLY%\xyz.dll
```

Report this

Jason Brower
CTO | CEO of POS & Payment Processing Software
4 articles

Follow

6 comments

Newest

Leave your thoughts here…

Neil Larson ··· 2mo
Contract Programmer at Outerwall

Thanks for getting me unstuck on signing with a Comodo purchased cert in VS 2015. Works great now.

Like Reply

Robert Zeff ··· 7mo
CEO at Nikola, Inc.

Thank you Jason, I have enough trouble writing code, dealing with certificates is a huge time sink. You have saved me a lot of time. BTW, rather that installing Cygwin, under Windows 10 pre-release, I ran a Bash shell and apt-get install openssl and executed your commands. Removed the .exe. Worked like a charm. Thanks again, Robert Zeff

Like Reply

There are 4 other comments. Show more.

Don't miss more articles by Jason Brower

Plunging from Engineer to Entrepreneur

Jason Brower on LinkedIn

Vicariously through a Cat in a Shark Tank

Jason Brower on LinkedIn

An Engineers POV: Tablet vs. Traditional POS Terminals

Jason Brower on LinkedIn

Looking for more of the latest headlines on LinkedIn?

Discover more stories