Automate Extended Validation (EV) code signing

Asked 8 years, 5 months ago Active 4 months ago Viewed 32k times

100

61

We recently purchased a DigiCert EV code signing certificate. We are able to sign .exe files using signtool.exe. However, every time we sign a file, it prompts for the SafeNet eToken password.

How can we automate this process, without user intervention, by storing/caching the password somewhere?

passwords code-signing authenticode code-signing-certificate

Share Follow

edited Aug 24 '15 at 17:33

Josh Kelley
51.7k 19 132 223

asked Jul 29 '13 at 15:21

decasteljau
7,195 7 39 57

The question "How safe are the password prompts of the SafeNet eToken 5110 or similar cryptographic hardware tokens?" is somewhat related, if it ever gets an answer it should be of interest to those assessing whether to automate the password entry. As I'm at it, if someone who currently owns that or a similar token reads this, if you can try to "hack" it and answer that question it would be greatly appreciated :) – gbr May 18 '17 at 17:34

unfortunately the answer that worked for me and that gets the most vote appears at the end of the answers list, so don't lose your time and go directly to Simon Mourier answers stackoverflow.com/a/26126701/27194 – Patrick from NDepend team Oct 3 '19 at 5:49

1 Just a heads up before attempting any of these solutions. Hardware tokens have a "Token Password retries remaining" counter (can be checked in the SafeNet Authentication Client). When experimenting, make sure that it never reaches zero for obvious reasons. Otherwise your will probably be permanently locked out of your hardware token and you will have to order a new one! Learned this the hard way... – Sundae Oct 9 '19 at 11:50

The answer by Simon unfortunately no longer works (see my comment to the answer). And the answer by Austin not only works, but is imo better anyway. – Martin Prikryl Oct 27 '20 at 8:28

1 The method described by Austin Morton works like a charm but it is very important to note that it requires an **up-to-date version** of the `signtool.exe` . With an out-of-date version (mine was from 2016) I got the error ``` Error information: "CryptExportPublicKeyInfoEx failed" (87/0x57) ``` You can get an up-to-date version by installing the Windows SDK. At least at the time of writing this, the version provided with the SDK supports using [method described by Austin Mor – Crown Apr 29 '21 at 14:45

13 Answers

Active | Oldest | Votes

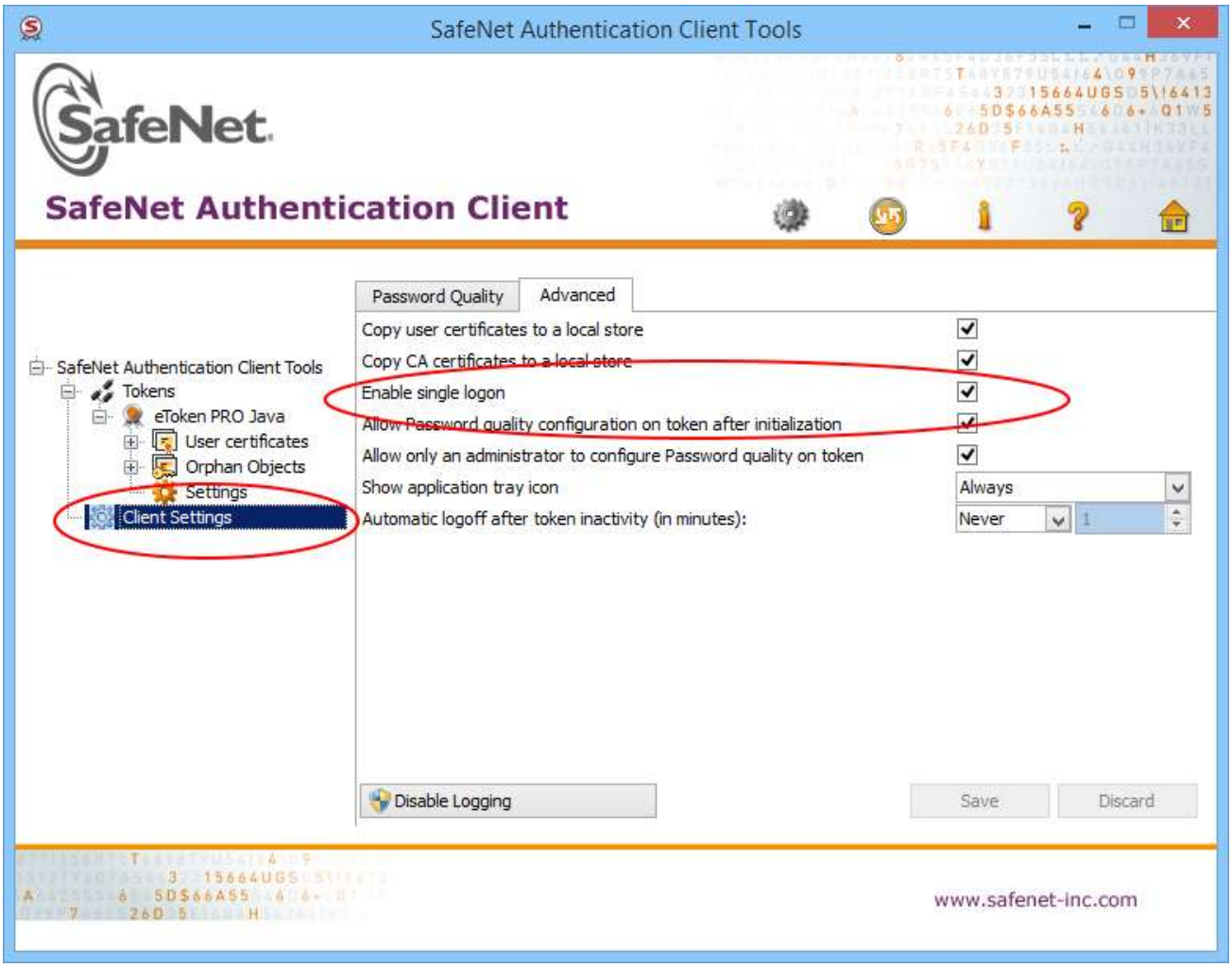
There is no way to bypass the login dialog AFAIK, but what you can do is configure the SafeNet Authentication Client so it only asks it once per login session.

70

I quote the SAC doc (found once installed in `\ProgramFiles\SafeNet\Authentication\SAC\SACHelp.chm`, chapter '`Client Settings`', '`Enabling Client Logon`') here:

> When single logon is enabled, users can access multiple applications with only one request for the Token Password during each computer session. This alleviates the need for the user to log on to each application separately.

To enable this feature which is disabled by default, go to SAC advanced settings, and check the "enable single logon" box:



Restart your computer, and it should now only prompt for the token password once. In our case, we have more than 200 binaries to sign per each build, so this is a total *must*.

Otherwise, here is a small C# console sample code (equivalent to m1st0 one) that allows you to respond automatically to logon dialogs (probably needs to run as admin) (you need to reference from your console project (`UIAutomationClient.dll` and `UIAutomationTypes.dll`):

```
using System;
using System.Windows.Automation;
```

```
namespace AutoSafeNetLogon {
   class Program {
      static void Main(string[] args) {
         SatisfyEverySafeNetTokenPasswordRequest("YOUR_TOKEN_PASSWORD");
      }


      static void SatisfyEverySafeNetTokenPasswordRequest(string password) {
         int count = 0;
         Automation.AddAutomationEventHandler(WindowPattern.WindowOpenedEvent,
AutomationElement.RootElement, TreeScope.Children, (sender, e) =>
         {
            var element = sender as AutomationElement;
            if (element.Current.Name == "Token Logon") {
               WindowPattern pattern =
(WindowPattern)element.GetCurrentPattern(WindowPattern.Pattern);
               pattern.WaitForInputIdle(10000);
               var edit = element.FindFirst(TreeScope.Descendants, new
AndCondition(
                     new PropertyCondition(AutomationElement.ControlTypeProperty,
ControlType.Edit),
                     new PropertyCondition(AutomationElement.NameProperty, "Token
Password:")));

               var ok = element.FindFirst(TreeScope.Descendants, new
AndCondition(
                     new PropertyCondition(AutomationElement.ControlTypeProperty,
ControlType.Button),
                     new PropertyCondition(AutomationElement.NameProperty, "OK")));

               if (edit != null && ok != null) {
                  count++;
                  ValuePattern vp =
(ValuePattern)edit.GetCurrentPattern(ValuePattern.Pattern);
                  vp.SetValue(password);
                  Console.WriteLine("SafeNet window (count: " + count + "
window(s)) detected. Setting password...");

                  InvokePattern ip =
(InvokePattern)ok.GetCurrentPattern(InvokePattern.Pattern);
                  ip.Invoke();
               } else {
                  Console.WriteLine("SafeNet window detected but not with edit
and button...");
               }
            }
         });

         do {
            // press Q to quit...
            ConsoleKeyInfo k = Console.ReadKey(true);
            if (k.Key == ConsoleKey.Q)
               break;
         }
         while (true);
         Automation.RemoveAllEventHandlers();
      }
   }
}
```

Share Follow

edited May 12 '20 at 19:05

Martin Prikryl
160k 47 381 798

answered Sep 30 '14 at 17:30

Simon Mourier
121k 18 233 277

11 This may not be an official answer from DigiCert, but their answer sucks, and this one is awesome! Thanks for the help! – lordjeb Jan 15 '15 at 18:21

2 +1 for a correct answer. It amazes me to see people developing scripts to automate user input and such, defeating the purpose of having a password really, and all they needed to know was where this option was. I doubt this option will ever disappear, as the issuers understand developers can't type in the password every single time a binary is signed. – dyasta Jan 30 '16 at 16:53

4 I can confirm that this works from TeamCity (as long as the TeamCity Windows service has the "Allow service to interact with desktop" box ticked). We also needed to run the password entry process in another thread and to disable the "Interactive Services Detection" service on our build machine. We created a C# wrapper around signtool that performed the signing and handled the password entry as above, all in a self contained app. I can't believe how many hurdles we had to cross to get this working, but for anyone else in the same boat, focus on the C# method described above... – Alan Spark Mar 22 '16 at 14:54

2 FYI...the Symantec EV Certs use SafeNet as well. We had to built a janky solution around this process but after reading your answer and implementing the console app, this has helped out our build process immensely. Thank you. Great solution to a poorly architected code signing process. – Zoltan Oct 24 '16 at 20:01

1 I have been successfully using this useful solution for a while. But now when setting it up on new machine with Windows 10 Pro 2004 with SafeNet client 9.0.34 x64 for Windows 8 and up, it does not work anymore. Where's a new password prompt. It seems to be a Windows built-in one, instead of the custom SafeNet prompt like before. And the password box of the new prompt is not automatable (it's not exposed in `AutomationElement` tree). I do not know if it can be worked around somehow. But visiting this question again and finding the answer by @Austin, I believe it's a better solution anyway. – Martin Prikryl Oct 22 '20 at 9:21

65

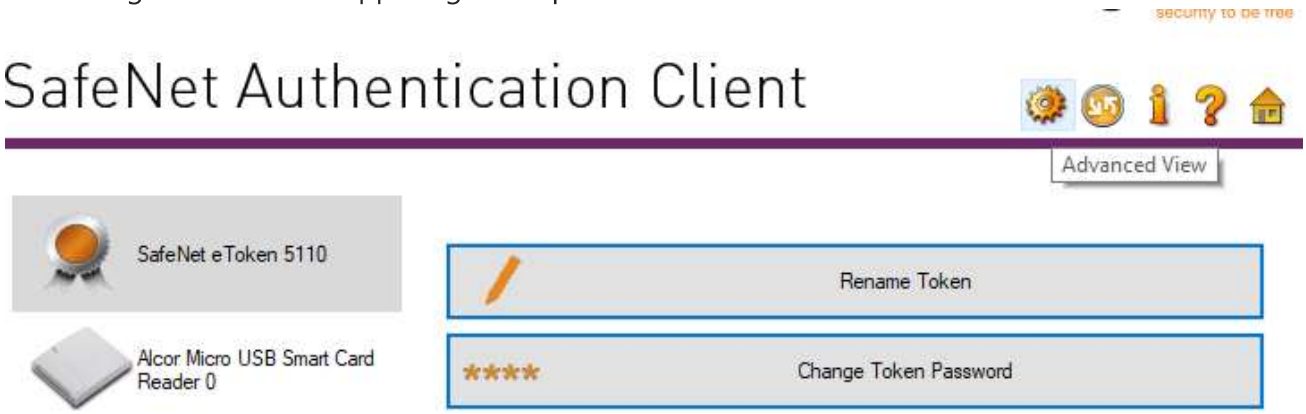
+100

Expanding on answers already in this thread, it is possible to provide the token password using the standard signtool program from microsoft.

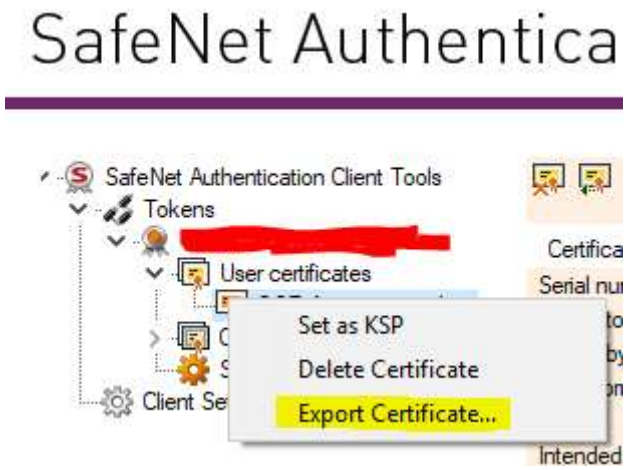
0. Open SafeNet Client in Advanced View

Install paths may vary, but for me the SafeNet client is installed to: `C:\Program Files\SafeNet\Authentication\SAC\x64\SACTools.exe`

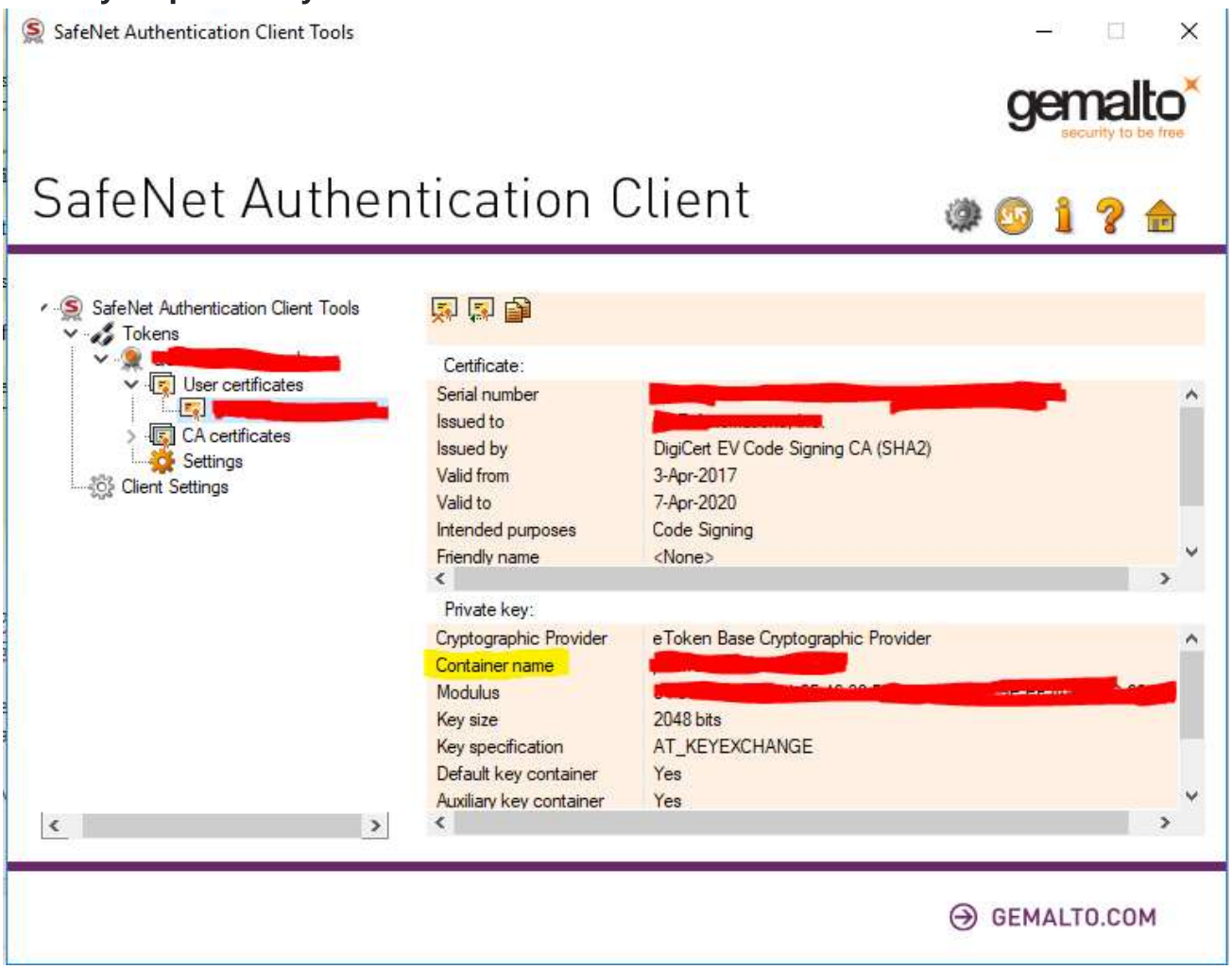
Click the gear icon in the upper right to open "advanced view".



1. Export your public certificate to a file from the SafeNet Client



2. Find your private key container name



3. Find your reader name

4. Format it all together

The eToken CSP has hidden (or at least not widely advertised) functionality to parse the token password out of the container name.

The format is one of the following

```
[]=name
[reader]=name
[{{password}}]=name
[reader{{password}}]=name
```

Where:

- `reader` is the "Reader name" from the SafeNet Client UI
- `password` is your token password
- `name` is the "Container name" from the SafeNet Client UI

Presumably you must specify the reader name if you have more than one reader connected - as I only have one reader I cannot confirm this.

5. Pass the information to signtool

- `/f certfile.cer`
- `/csp "eToken Base Cryptographic Provider"`
- `/k "<value from step 4>"`
- any other signtool flags you require

Example signtool command as follows

```
signtool sign /f mycert.cer /csp "eToken Base Cryptographic Provider" /k "
[{{TokenPasswordHere}}]=KeyContainerNameHere" myfile.exe
```

Some Images taken from this answer: https://stackoverflow.com/a/47894907/5420193

Share Follow

edited Oct 4 '19 at 6:31

answered Jan 30 '19 at 11:38

Austin Morton
783 6 6

2 Works great, sadly i found this after two days digging and implementing own "signtool" :D thx – Lukáš Koten Mar 13 '19 at 10:10

3 Warning: This solution may lock you out of your hardware token if you enter the wrong password, even just twice! Somehow the password retries remaining counter decreased from 15 to 3 after I executed the command just once with an invalid password. The "etokensign.exe" solution seems to work ok though, the remaining password counter decreased as it should from 15 to 14 after one invalid password attempt. – Sundae Oct 9 '19 at 11:37

1 Wonderful, this worked perfectly for me with the SafeNet USB dongle that came to me from Sectigo. – JacobJ Dec 9 '19 at 23:50

4 As far as I know, this syntax is not publicly documented anywhere. I found this functionality by reverse engineering the driver binary in IDA Pro. – Austin Morton Mar 11 '20 at 17:04

3 This is better than the other answers that fake password inputs to GUI prompts. Among other, this works even in non-interactive Windows sessions. And the other answers seem not to work with the latest tools. – Martin Prikryl Oct 22 '20 at 10:31

21

Expanding on this answer, this can be automated using CryptAcquireContext and CryptSetProvParam to enter the token PIN programmatically and CryptUIWizDigitalSign to perform the signing programmatically. I created a console app (code below) that takes as input the certificate file (exported by right clicking the certificate in SafeNet Authentication Client and selecting "Export..."), the private key container name (found in SafeNet Authentication Client), the token PIN, timestamp URL, and the path of the file to sign. This console app worked when called by the TeamCity build agent where the USB token was connected.

Example Usage:

```
etokensign.exe c:\CodeSigning.cert CONTAINER PIN http://timestamp.digicert.com
C:\program.exe
```

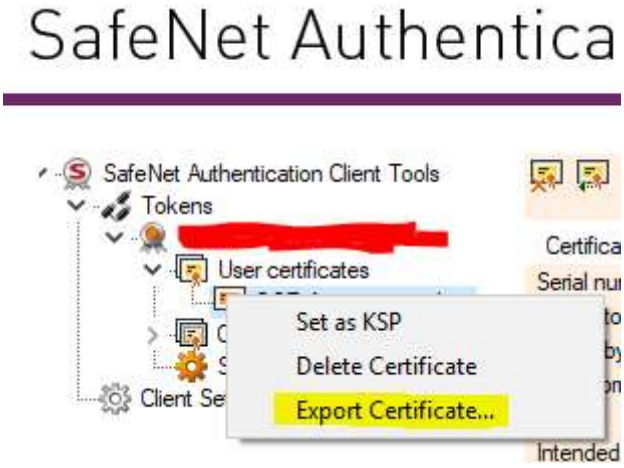
Code:

```
#include <windows.h>
#include <cryptuiapi.h>
#include <iostream>
#include <string>

const std::wstring ETOKEN_BASE_CRYPT_PROV_NAME = L"eToken Base Cryptographic
Provider";

std::string utf16_to_utf8(const std::wstring& str)
{
    if (str.empty())
    {
        return "";
    }

    auto utf8len = ::WideCharToMultiByte(CP_UTF8, 0, str.data(), str.size(),
NULL, 0, NULL, NULL);
    if (utf8len == 0)
    {
        return "";
    }

    std::string utf8Str;
    utf8Str.resize(utf8len);
    ::WideCharToMultiByte(CP_UTF8, 0, str.data(), str.size(), &utf8Str[0],
utf8Str.size(), NULL, NULL);

    return utf8Str;
}

struct CryptProvHandle
{
    HCRYPTPROV Handle = NULL;
    CryptProvHandle(HCRYPTPROV handle = NULL) : Handle(handle) {}
    ~CryptProvHandle() { if (Handle) ::CryptReleaseContext(Handle, 0); }
};

HCRYPTPROV token_logon(const std::wstring& containerName, const std::string&
tokenPin)
{
    CryptProvHandle cryptProv;
    if (!::CryptAcquireContext(&cryptProv.Handle, containerName.c_str(),
ETOKEN_BASE_CRYPT_PROV_NAME.c_str(), PROV_RSA_FULL, CRYPT_SILENT))
    {
        std::wcerr << L"CryptAcquireContext failed, error " << std::hex <<
std::showbase << ::GetLastError() << L"\n";
        return NULL;
    }

    if (!::CryptSetProvParam(cryptProv.Handle, PP_SIGNATURE_PIN,
reinterpret_cast<const BYTE*>(tokenPin.c_str()), 0))
    {
        std::wcerr << L"CryptSetProvParam failed, error " << std::hex <<
```

```
std::showbase << ::GetLastError() << L"\n";
        return NULL;
    }

    auto result = cryptProv.Handle;
    cryptProv.Handle = NULL;
    return result;
}

int wmain(int argc, wchar_t** argv)
{
    if (argc < 6)
    {
        std::wcerr << L"usage: etokensign.exe <certificate file path> <private
key container name> <token PIN> <timestamp URL> <path to file to sign>\n";
        return 1;
    }

    const std::wstring certFile = argv[1];
    const std::wstring containerName = argv[2];
    const std::wstring tokenPin = argv[3];
    const std::wstring timestampUrl = argv[4];
    const std::wstring fileToSign = argv[5];

    CryptProvHandle cryptProv = token_logon(containerName,
utf16_to_utf8(tokenPin));
    if (!cryptProv.Handle)
    {
        return 1;
    }

    CRYPTUI_WIZ_DIGITAL_SIGN_EXTENDED_INFO extInfo = {};
    extInfo.dwSize = sizeof(extInfo);
    extInfo.pszHashAlg = szOID_NIST_sha256; // Use SHA256 instead of default SHA1

    CRYPT_KEY_PROV_INFO keyProvInfo = {};
    keyProvInfo.pwszContainerName = const_cast<wchar_t*>(containerName.c_str());
    keyProvInfo.pwszProvName = const_cast<wchar_t*>
(ETOKEN_BASE_CRYPT_PROV_NAME.c_str());
    keyProvInfo.dwProvType = PROV_RSA_FULL;

    CRYPTUI_WIZ_DIGITAL_SIGN_CERT_PVK_INFO pvkInfo = {};
    pvkInfo.dwSize = sizeof(pvkInfo);
    pvkInfo.pwszSigningCertFileName = const_cast<wchar_t*>(certFile.c_str());
    pvkInfo.dwPvkChoice = CRYPTUI_WIZ_DIGITAL_SIGN_PVK_PROV;
    pvkInfo.pPvkProvInfo = &keyProvInfo;

    CRYPTUI_WIZ_DIGITAL_SIGN_INFO signInfo = {};
    signInfo.dwSize = sizeof(signInfo);
    signInfo.dwSubjectChoice = CRYPTUI_WIZ_DIGITAL_SIGN_SUBJECT_FILE;
    signInfo.pwszFileName = fileToSign.c_str();
    signInfo.dwSigningCertChoice = CRYPTUI_WIZ_DIGITAL_SIGN_PVK;
    signInfo.pSigningCertPvkInfo = &pvkInfo;
    signInfo.pwszTimestampURL = timestampUrl.c_str();
    signInfo.pSignExtInfo = &extInfo;

    if (!::CryptUIWizDigitalSign(CRYPTUI_WIZ_NO_UI, NULL, NULL, &signInfo, NULL))
    {
        std::wcerr << L"CryptUIWizDigitalSign failed, error " << std::hex <<
std::showbase << ::GetLastError() << L"\n";
        return 1;
    }
```

```
    std::wcout << L"Successfully signed " << fileToSign << L"\n";
    return 0;
}
```

Exporting the Certificate to a File:



Private Key Container Name:

Share Follow

edited Feb 15 '18 at 10:07
IInspectable

answered Dec 19 '17 at 20:44
draketb

 **39.2k** 8 75 159 **414** 4 4

2 This one should be the accepted answer, it works like a charm! – shawn Mar 27 '18 at 4:29

2 This is perfect. In particular after I realized that I only need to sign some dummy file with this tool. If the "single logon" is enabled (SafeNet driver), all subsequent steps work with the standard signtool. This is very useful for signing Office Addins (VSTO), which use a different tool and it also meant that only minimal changes were required for my build script / process. – Stefan Egli Oct 25 '18 at 5:54

This answer is a good addition to the one provided by avzhatkin. At this point the code is close to replacing signtools.exe. This program would need to support cross signing. Luckily there is another SO post with now to perform cross signing. – sdc Dec 6 '18 at 14:46

This answer ended up helping me the most. I missed an external reference when building in VS2017, but when adding some pragma comments as suggested here I managed to get Bamboo (CI/CD by Atlassian) to sign. – HTBR Mar 7 '19 at 14:16

Link to "Cryptui.lib" to build successfully. For Visual C++ it is sufficient to insert `#pragma comment(lib, "Cryptui.lib")` after the `#include` s. – zett42 Feb 19 '21 at 17:19

12

I'm made beta tool which will help to automate build process.

It's Client-Server windows application. You can start server on computer where EV token inserted. Enter password for token on server-side application startup. After this you can sign files remotely. Client side application fully replaces signtool.exe so you can use existing build scripts.

Source code located here: https://github.com/SirAlex/RemoteSignTool

Edit: We successfully used this tool for code signing last half-year 24x7 on our Build server. All works fine.

Share Follow

edited Jan 26 '17 at 12:20

answered Feb 12 '16 at 17:59

Aleksey Kharlanov

411 3 8

1 How secure is this approach? Doesn't it mean that anyone who can connect to your signing server with HTTP, can sign any binary they want with your EV certificate? – Gene Pavlovsky May 21 '19 at 9:51

7

signtool.exe sign /fd sha256 /f "signing.cer" /csp "eToken Base Cryptographic Provider" /kc "[{{token password here}}]=Container name here" "ConsoleApp1.exe"

Use Microsoft Windows SDK 10 for signtool

Share Follow

answered Jan 29 '20 at 4:40

Lakmal

599 1 6 14

1 Superb! This one-liner certainly puts all the other answers to shame (although at first I was puzzled how to figure out my "Container name", until I found the instructions in draketb's answer above). – Dan Z Jun 5 '20 at 9:55

6

Actually on Windows you can specify the token password fully programmatically. This can be done by creating a context (CryptAcquireContext) with flag CRYPT_SILENT using token name in form "\\.\AKS ifdh 0" or token container name, which is some guid visible in cerificate properties in the Authentication Client application. You then need to use CryptSetProvParam with parameter PP_SIGNATURE_PIN to specify your token password. After that the process can use certificates on that token to sign files.

Note: once you create the context is seems to just work for current process entirely, no need to pass it to other Crypto API functions or anything. But feel free to comment if you find a situation when some more efforts will be required.

Edit: added code sample

```
HCRYPTPROV OpenToken(const std::wstring& TokenName, const std::string& TokenPin)
{
    const wchar_t DefProviderName[] = L"eToken Base Cryptographic Provider";

    HCRYPTPROV hProv = NULL;
    // Token naming can be found in "eToken Software Developer's Guide"
    // Basically you can either use "\\.\AKS ifdh 0" form
    // Or use token's default container name, which looks like "ab-c0473610-8e6f-4a6a-
ae2c-af944d09e01c"
    if(!CryptAcquireContextW(&hProv, TokenName.c_str(), DefProviderName, PROV_RSA_FULL,
CRYPT_SILENT))
    {
        DWORD Error = GetLastError();
        //TracePrint("CryptAcquireContext for token %ws failed, error 0x%08X\n",
TokenName.c_str(), Error);
        return NULL;
    }
    if(!CryptSetProvParam(hProv, PP_SIGNATURE_PIN, (BYTE*)TokenPin.c_str(), 0))
    {
        DWORD Error = GetLastError();
        //TracePrint("Token %ws unlock failed, error 0x%08X\n", TokenName.c_str(),
Error);
        CryptReleaseContext(hProv, 0);
        return NULL;
    }
    else
    {
        //TracePrint("Unlocked token %ws\n", TokenName.c_str());
        return hProv;
    }
}
```

Share Follow

edited Sep 24 '17 at 16:06

answered Sep 4 '17 at 8:20

avzhatkin

101 1 4

1 Interesting. Seems promising, you should IMHO elaborate on that (enhance explanation, provide code, etc.) – Simon Mourier Sep 12 '17 at 14:47

Please post a full example. This sounds really useful – dten Sep 13 '17 at 9:59

thanks for the extra details. is this the guide you mention ? read.pudn.com/downloads128/ebook/549477/eToken_SDK_3_50[1].pdf – dten Oct 1 '17 at 12:29

I believe it is not the exact version I had, but it does seem to contain similar information about creating context and providing the PIN, though for different use scenario. – avzhatkin Oct 3 '17 at 12:04

I guess you call this function OpenToken(L"\\\\.\\AKS ifdh 0",<token password>)... well it worked for me! – Michael Haephrati Aug 21 '18 at 18:32

5

Install https://chocolatey.org/docs/installation (Can be done using one command from administrative command prompt)

(Restart command prompt)

Supress choco's constant prompting for each installation:

```
choco feature enable -n=allowGlobalConfirmation
```

Install python, using command:

```
choco install python
```

(Restart command prompt) Install additional python module:

```
pip install pypiwin32
```

Save following text into `disableAutoprompt.py` :

```
import pywintypes
import win32con
import win32gui
import time



DIALOG_CAPTION = 'Token Logon'
DIALOG_CLASS = '#32770'
PASSWORD_EDIT_ID = 0x3ea
TOKEN_PASSWORD_FILE = 'password.txt'
SLEEP_TIME = 10
```

```
def get_token_password():
    password = getattr(get_token_password, '_password', None)
    if password is None:
        with open(TOKEN_PASSWORD_FILE, 'r') as f:
            password = get_token_password._password = f.read()

    return password

def enumHandler(hwnd, lParam):
    if win32gui.IsWindowVisible(hwnd):
        if win32gui.GetWindowText(hwnd) == DIALOG_CAPTION and
win32gui.GetClassName(hwnd) == DIALOG_CLASS:
            print('Token logon dialog has been detected, trying to enter password...')
            try:
                ed_hwnd = win32gui.GetDlgItem(hwnd, PASSWORD_EDIT_ID)
                win32gui.SendMessage(ed_hwnd, win32con.WM_SETTEXT, None,
get_token_password())
                win32gui.PostMessage(ed_hwnd, win32con.WM_KEYDOWN, win32con.VK_RETURN,
0)
                print('Success.')
            except Exception as e:
                print('Fail: {}'.format(str(e)))
                return False
```

Save your password into passwd.txt, and after that run

```
python disableAutoprompt.py
```

From `SafeNet Authentication Client` - configuration > `Client Settings` > `Advanced` > `Enable Single Log On` option can be enabled to minimize amount of password prompts, but it does not fully disable them (Tested on version 10.4.26.0)

C# application (E.g. https://github.com/ganl/safenetpass) does not work with lock screen on, but does work with this python script.

Share Follow

edited Feb 13 '20 at 12:08
TarmoPikaro
4,000 1 34 48

answered Apr 18 '17 at 10:37
A.N.
285 2 10

This script is awesome, and I was able to easily adapt it to my needs working with a Yubikey dongle. However, Windows 10 breaks it. Windows 10 changes to XAML and so the win32gui.xxxx() functions won't work. /SIGH. Thanks Microsoft. This is why we can't have nice things. – John Rocha Feb 21 '20 at 17:57

4

I used AutoHotKey to automate the password entry using the script following. We have been trying to make a Web based front end for our developers to send the binaries to the Windows box with this script running so that it can be signed and returned.

```
Loop
{
  Sleep 2000
```

```
  if (WinExist("Token Logon"))
  {
    WinActivate ; use the window found above
    SendInput [your_password]
    SendInput {Enter}
  }
  if (WinExist("DigiCert Certificate Utility for Windows©"))
  {
    WinActivate ; use the window found above
    SendInput [your_password]
    SendInput {Enter}
  }
}
```

I must note that what I shared is not completely not secure, but we also hit this issue requiring either purchasing signing keys for each developer or assigning a job of a signing manager that would approve the signature of released software. I believe those are the better, secure processes--that once things pass quality assurance and are approved for release, they can be officially signed. However, smaller company needs may dictate that this be done in some other automated way.

I originally used osslsigncode on Linux (before EV certificates) to automate signing of Windows executables (since we had a Linux server doing a lot of work for developer ease and collaboration). I have contacted the developer of osslsigncode to see if he can make use of the DigiCert SafeNet tokens to help automate it in a different way since I can see them on Linux. His reply provided hope but I am unsure of any progress and I could not dedicate more time to help

Share Follow

answered Jun 7 '14 at 19:48

m1st0
119 2 10

See the other answer. There is an option to unlock only once per session, which suffices for most users. – dyasta Jan 30 '16 at 16:54

2

Got an answer from Digicert:

> Unfortunately, part of the security with the EV Code Signing Certificate is that you must enter the password everytime. There is not a way to automate it.

Share Follow

answered Jul 29 '13 at 15:35

decasteljau
7,195 7 39 57

We got the same response, although they are looking into a solution they have no timeframe for when one might be available. They are aware of this SO post though so hopefully they will realise how much of an issue it is. – Alan Spark Mar 22 '16 at 14:58

We found a way around it: github.com/mareklinka/SafeNetTokenSigner/issues/8 – gunslingor Oct 15 '19 at 14:28

2

I my case Digicert issue an Standard (OV) certificate for the CI, for free if you already have an EV certificate.

I know this is not the solution but if you cant put the token in the server (a cloud server) this is the way to go.

Share Follow

answered Sep 8 '15 at 19:47

Ricardo Polo Jaramillo
11.6k 13 53 79

2

I am using a globalsign certificate and they nicely said the same thing.

> It is not possible to script the signature with a standard EV code signing, they are promoting the use of an HSM platform

...which is far beyond my budget. Contrary what they said, I succeeded to make it work :

```
"C:\Program Files (x86)\Microsoft SDKs\ClickOnce\SignTool\signtool.exe" sign /fd sha256
/f "MyCertificate.cer" /csp "eToken Base Cryptographic Provider" /kc "
[{{TokenPassword}}]=ContainerTame" "FileToSign"
```

=> this command returns the following error :

```
Error information: "CryptExportPublicKeyInfoEx failed" (87/0x57)
```

I dont really understand this issue. BUT if you run another time the following command it works

```
"C:\Program Files (x86)\Microsoft SDKs\ClickOnce\SignTool\SignTool.exe" sign /tr
http://timestamp.globalsign.com/scripts/timestamp.dll "MyFileToSign"
Done Adding Additional Store
Successfully signed: MyFileToSign
```

This works within teamcity build, and no need for an active account log in in teamcity build agent.

Edit : this solution does not work anymore for me, globalsign changed the timestamp url to http://rfc3161timestamp.globalsign.com/advanced. Since that i cant sign with TokenPassword/ContainerName anymore. The only solution I found is to unable single logon and make sure the server does not logoff (i run a video on the server so my account is not loggued off

automatically). this is a quick and dirty solution but the only one I found. Thank you globalsign for your poor support.

Share Follow

edited Mar 18 '21 at 7:16

answered Dec 15 '20 at 14:47

SebastienAuroux
21 3

1

The way I do it is :

1. **Open the token**

   PCCERT_CONTEXT cert = OpenToken(SAFENET_TOKEN, EV_PASS);

2. **Sign the file** using the token, root / cross certificates when required and the EV certificate loaded into memory.

   HRESULT hr = SignAppxPackage(cert, FILETOSIGN);

Using SignerSignEx2():

The file is signed using SignerSignEx2() which needs to be loaded into memory using LoadLibrary() and GetProcAddress():

```
// Type definition for invoking SignerSignEx2 via GetProcAddress
typedef HRESULT(WINAPI *SignerSignEx2Function)(
    DWORD,
    PSIGNER_SUBJECT_INFO,
    PSIGNER_CERT,
    PSIGNER_SIGNATURE_INFO,
    PSIGNER_PROVIDER_INFO,
    DWORD,
    PCSTR,
    PCWSTR,
    PCRYPT_ATTRIBUTES,
    PVOID,
    PSIGNER_CONTEXT *,
    PVOID,
    PVOID);

// Load the SignerSignEx2 function from MSSign32.dll
HMODULE msSignModule = LoadLibraryEx(
    L"MSSign32.dll",
    NULL,
    LOAD_LIBRARY_SEARCH_SYSTEM32);

if (msSignModule)
{
    SignerSignEx2Function SignerSignEx2 = reinterpret_cast<SignerSignEx2Function>(
        GetProcAddress(msSignModule, "SignerSignEx2"));
    if (SignerSignEx2)
    {
        hr = SignerSignEx2(
            signerParams.dwFlags,
            signerParams.pSubjectInfo,
            signerParams.pSigningCert,
```

```
            signerParams.pSignatureInfo,
            signerParams.pProviderInfo,
            signerParams.dwTimestampFlags,
            signerParams.pszAlgorithmOid,
            signerParams.pwszTimestampURL,
```

Time Stamping

Further, you must Time Stamp your signed file and do that using a Time Stamping authority to which you connect.

That is done by securely checking a Time Stamping server via URL for the current date and time. Each Signing authority have their own Time Stamping server. Time Stamping is an extra step in the Code Signing process, but when it comes to EV Code Signing it is a requirement which adds an additional layer of security to the signed PE. For that reason, add to your code a check whether the user is connected to the Internet.

```
DWORD dwReturnedFlag;
if (InternetGetConnectedState(&dwReturnedFlag,0) == NULL) // use
https://docs.microsoft.com/en-us/windows/desktop/api/netlistmgr/nf-netlistmgr-
inetworklistmanager-getconnectivity
{
    wprintf(L"Certificate can't be dated with no Internet connection\n");
    return 1;
}
```

Loading a certificate from a file

```
std::tuple<DWORD, DWORD, std::string> GetCertificateFromFile
(const wchar_t*                          FileName
    , std::shared_ptr<const CERT_CONTEXT>*   ResultCert)
{
    std::vector<unsigned char> vecAsn1CertBuffer;
    auto tuple_result = ReadFileToVector(FileName, &vecAsn1CertBuffer);

    if (std::get<0>(tuple_result) != 0)
    {
        return tuple_result;
    }

    return GetCertificateFromMemory(vecAsn1CertBuffer, ResultCert);
}
```

Loading a certificate into memory

```
std::tuple<DWORD, DWORD, std::string> GetCertificateFromMemory
(const std::vector<unsigned char>&      CertData
    , std::shared_ptr<const CERT_CONTEXT>*   ResultCert)
{
    const CERT_CONTEXT* crtResultCert = ::CertCreateCertificateContext
    (X509_ASN_ENCODING | PKCS_7_ASN_ENCODING
        , &CertData[0]
        , static_cast<DWORD>(CertData.size()));
```

```
    if (crtResultCert == NULL)
    {
        return std::make_tuple(E_FAIL
            , ::GetLastError()
            , "CertCreateCertificateContext");
    }

    *ResultCert = std::shared_ptr<const CERT_CONTEXT>(crtResultCert
        , ::CertFreeCertificateContext);
    return std::make_tuple(0, 0, "");
}
```

After the certificate has been loaded following accessing the hardware token we load it:

```
std::vector<unsigned char> dataCertEV(signingCertContext->pbCertEncoded,
        signingCertContext->pbCertEncoded + signingCertContext->cbCertEncoded);
```

Finally, the signing is done in the following function:

```
HRESULT SignAppxPackage(
    _In_ PCCERT_CONTEXT signingCertContext,
    _In_ LPCWSTR packageFilePath)
{
    HRESULT hr = S_OK;
    if (PathFileExists(CertAuthority_ROOT))
    {
        wprintf(L"Cross Certificate '%s' was found\n", CertAuthority_ROOT);
    }
    else
    {
        wprintf(L"Error: Cross Certificate '%s' was not found\n", CertAuthority_ROOT);
        return 3;
    }
    DWORD dwReturnedFlag;
    if (InternetGetConnectedState(&dwReturnedFlag,0) == NULL)
    {
        wprintf(L"Certificate can't be dated with no Internet connection\n");
        return 1;
    }
    if (PathFileExists(CertAuthority_RSA))
    {
        wprintf(L"Cross Certificate '%s' was found\n", CertAuthority_RSA);
    }
    else
    {
        wprintf(L"Error: Cross Certificate '%s' was not found\n", CertAuthority_RSA);
        return 2;
    }
    if (PathFileExists(CROSSCERTPATH))
    {
        wprintf(L"Microsoft Cross Certificate '%s' was found\n", CROSSCERTPATH);

    }
    else
    {
        wprintf(L"Error: Microsoft Cross Certificate '%s' was not found\n",
CROSSCERTPATH);
```

See this article I wrote.

Share Follow

answered Dec 15 '18 at 19:42

Michael Haephrati
3,185 1 29 52

-1

Our eSigner cloud signing solution for EV Code Signing certs at SSL.com eliminates the need for hardware tokens and makes automated signing (ie CI/CD) using EV Code Signing certificates much easier.

See more here https://www.ssl.com/how-to/automate-esigner-ev-code-signing/

The demo signing service can also be found at https://try.esigner.com

Share Follow

answered Sep 7 '21 at 2:15

Leo Grove
184 1 4
