Kernel-mode code signing - A beginner's guide

Michael Schneider
Offense Department, scip AG
misc@scip.ch
https://www.scip.ch

Marc Ruef (Editor)
Research Department, scip AG
maru@scip.ch
https://www.scip.ch

Abstract: Kernel drivers have to be signed. Requirements have become more stringent with new versions of Windows. Windows 10 requires drivers signed by Microsoft. Drivers with signatures dated prior to July 2015 are still valid.

Keywords: Dashboard, HTTP, Microsoft, Policy, Request, Research, Tool, Trust, USB, Windows

1. Preface

This paper was written in 2019 as part of a research project at scip AG, Switzerland. It was initially published online at *https://www.scip.ch/en/?labs.20190919* and is available in English and German. Providing our clients with innovative research for the information technology of the future is an essential part of our company culture.

2. Introduction

One fascinating aspect of what we do is the fact that, from one day to the next, we have to deal with topics that we don't know much about beforehand and then gather a great deal of knowledge within a short space of time. Usually, we then need to familiarize ourselves with the subject matter so that we can apply what we've just learned straight away.

I recently wanted to run code in kernel mode on Windows and, in so doing, I learned what the prerequisites are and where the pitfalls lurk. And that's what this lab is all about: a beginner's guide to the topic of *code signing* for *Windows kernel mode*.

3. Theory

3.1. An introduction to code signing

The operating system Microsoft Windows makes a distinction between the *user space* and the *system space*, which is also known as the *kernel space*. Normal applications that do not have direct access to internal or sensitive operating system resources are run in the user space. Code executed in the kernel space has more access to and a direct influence on the operating system, either on the stability or security of the system. Consequently, code requirements in the kernel space are higher.

When Windows Vista 64-bit was released, Microsoft required signatures to load code in the kernel space. Windows 8 also required driver packages to be signed too. And with Windows 10, Microsoft required new drivers to be signed by the *Windows Hardware Dev Center* [1].

Signing a driver firstly ensures the software's integrity, since a change to the driver causes the existing signature to become invalid. Secondly, it allows the software's origin to be determined. This is because when you register for the Hardware Dev Center, Microsoft can at least track what account the software comes from. What's more, one of the conditions for taking part is the use of an *Extended Validation Code Signing certificate*, or EV CS certificate.

For an EV CS certificate to be issued, a *certificate authority* (CA) performs a detailed check on the applicant as part of the issuing process. Any company applying for a certificate must provide publicly verifiable information such as a correct address, an official phone number and the people responsible for signing software. These contacts are then verified over the phone. Following successful verification, the *EV Code Signing certificate* is issued in the name of the company and delivered on a USB token.

There is currently discussion going on about the purpose of EV certificates for websites. Security researcher Troy Hunt described them as "outdated" in his blog article entitled *Extended Validation Certificates are (Really, Really) Dead* [2]; he believes that they can be replaced by free solutions. However, there is no alternative to kernel-mode code signing, since Microsoft requires EV CS certificates to be used.

3.2. Windows requirements

Let's take a step back and look at the requirements of the different versions of Windows. Microsoft's *Driver Signing Policy* [3] stipulates that, for Windows 7 64-bit, Windows 8 and Windows 10 up to version 1511, a driver must be signed with SHA1 and the certificate used must come from a CA that is on Microsoft's *Cross-Certificate List* [4]. For Windows 10 versions 1607 to 1709, SHA1 or SHA2 is allowed as the signature algorithm, while only SHA2 is allowed from Windows 10 version 1803 and higher. The signature must come from a Microsoft root authority too. In other words, a new installation of Windows 10 version 1607 will no longer load new kernel drivers that have not been signed by the Hardware Dev Center.

These changes were described in detail in the blog article entitled *Driver Signing changes in Windows 10, version 1607* [5]. In the interests of backward compatibility, Microsoft defined exceptions so that not all drivers have to be re-signed:

- Computers deployed before Windows 10 version 1607 and updated since then still allow the installation of cross-signed drivers
- Computers without *secure boot* still allow the installation of cross-signed drivers
- Drivers with the signature of a certificate issued before 7/29/2015 that contains a supported cross-signed CA in the certificate chain are still allowed

This means that all new drivers for current Windows 10 versions must therefore be signed with an EV CS certificate, then validated by the Windows Hardware Developer Center, then signed by Microsoft.

4. Practice

4.1. Signing a driver

Microsoft offers a comprehensive *Windows Driver Signing Tutorial* [6], which includes instructions for implementing a test signature. However, you will need to boot the operating system in a special mode to deactivate the *driver signature enforcement* option for the session. You can then create your own certificate and use it to sign and load drivers. These steps can be useful for initial attempts and tests. However, self-signed drivers cannot be used on external machines with current Windows versions.

You will need the *Windows Driver Kit* [7] (WDK) to sign drivers. The most important tool in the WDK is the *SignTool* [8]. It is used for signing and potentially verifying drivers. Microsoft advises against using a certificate file (PFX) for signature purposes; instead, it recommends importing the certificate into the operating system's *certificate store* and then performing the signature process. Additionally, an EV CS certificate is delivered on a USB token or smartcard rather than a PFX file.

In its simplest form, the command to sign a driver is:

```
signtool.exe sign /v /n "SubjectName"
DriverFile.sys
```

The parameter `/n` is the certificate's *common name*. But you will need to use a certificate from a cross-signed CA for the driver to also load in kernel mode. You can download the appropriate CA certificate from the *Cross-Certificate List* [9]. This certificate is then integrated using the parameter `/ac`:

```
signtool.exe sign /v /n "SubjectName" /ac
CrossSignedCARoot.cer DriverFile.sys
```

A driver must be signed with SHA2 for Windows 10. The driver should also contain a signature timestamp. You can use the timestamp server of the CA in question:

```
signtool.exe sign /v /n "SubjectName" /ac
CrossSignedCARoot.cer /fd sha256 /td sha256
/tr http://timestamp.example.com/rfc3161
DriverFile.sys
```

Drivers signed in this way can be used on Windows 7, 8 and earlier versions of Windows 10.

4.2. Verifying a signature

The `signtool.exe` application is, in turn, used to verify signatures. During the verification process, a distinction is made between the parameters `/pa` for validating the *Plug and Play driver* (PnP) and `/kp` for the *kernel mode driver*.

```
signtool.exe verify /pa /v DriverFile.sys
```

```
signtool.exe verify /kp /v DriverFile.sys
```

The following errors may occur during verification:

- `The signing certificate is not valid for the requested usage`: An EV CS certificate is required; other certificates are not accepted for *kernel mode*
- `The provided cross-certificate would not be present in the certificate chain`: The certificate downloaded from the Cross-Certificate List does not match the certificate chain; the appropriate root certificate must be selected
- `A certificate chain processed, but terminated in a root certificate which is not trusted by the trust provider`: Message on Windows 10 from version 1607 if the driver was signed using the cross-certificate, but not by the Hardware Dev Center

The certificate chain for a certificate can be checked using `certutil.exe`. Pentesters now have a legitimate reason to run `certutil` on a client:

```
certutil.exe -dump Certificate.cer
```

5. Conclusion

It took a good two weeks, including time waiting for verification of the EV CS certificate application and delivery of the USB token, as well as a few failed attempts with self-signed and normal CS certificates and reading Microsoft documents, until I managed to sign a driver so that Windows would load it in kernel mode. This article should help others to achieve this goal more quickly and easily. Please feel free to send feedback and share your own experiences.

6. External Links

[1] https://docs.microsoft.com/en-us/windows-hardware/drivers/dashboard/get-started-with-the-hardware-dashboard
[2] https://www.troyhunt.com/extended-validation-certificates-are-really-really-dead/
[3] https://docs.microsoft.com/en-us/windows-hardware/drivers/install/kernel-mode-code-signing-policy--windows-vista-and-later-#signing-requirements-by-version
[4] https://docs.microsoft.com/en-us/windows-hardware/drivers/install/cross-certificates-for-kernel-mode-code-signing#cross-certificate-list
[5] https://techcommunity.microsoft.com/t5/Windows-Hardware-Certification/Driver-Signing-changes-in-Windows-10-version-1607/ba-p/364894
[6] https://docs.microsoft.com/en-us/windows-hardware/drivers/install/windows-driver-signing-tutorial

[7] https://docs.microsoft.com/en-us/windows-hardware/drivers/download-the-wdk
[8] https://docs.microsoft.com/en-us/windows-hardware/drivers/devtest/signtool
[9] https://docs.microsoft.com/en-us/windows-hardware/drivers/install/cross-certificates-for-kernel-mode-code-signing#cross-certificate-list