

SSN Lab Assignment: UEFI Secure Boot*

Lucas Hecht

Zhiyang Wang

Feedback deadline:
Monday September 16, 2024 10:00 CET

1 Introduction

UEFI Secure Boot ensures by means of digital signatures that the code that loads the operating system and the operating system itself have not been tampered with. It is used by Microsoft to guarantee safe startup, but also by Ubuntu and Fedora. In this assignment you will learn how Ubuntu implements secure booting of the Linux kernel.

For this assignment you will need access to a Ubuntu machine that was installed with UEFI enabled (your desktop PC, not your server). It is not necessary for UEFI Secure Boot to be enabled on the machine (on your desktop PC it is not), as the same binaries are used by both secure and insecure boot.

Ubuntu provides the following description of its implementation of UEFI Secure Boot at <https://wiki.ubuntu.com/SecurityTeam/SecureBoot>:

“In order to boot on the widest range of systems, Ubuntu uses the following chain of trust:

1. Microsoft signs Canonical’s ‘shim’ 1st stage bootloader with their ‘Microsoft Corporation UEFI CA’. When the system boots and Secure Boot is enabled, firmware verifies that this 1st stage bootloader (from the ‘shim-signed’ package) is signed with a key in DB (in this case ‘Microsoft Corporation UEFI CA’)
2. The second stage bootloader (grub-efi-amd64-signed) is signed with Canonical’s ‘Canonical Ltd. Secure Boot Signing’ key. The shim 1st stage bootloader verifies that the 2nd stage grub2 bootloader is properly signed.
3. The 2nd stage grub2 bootloader boots an Ubuntu kernel (as of 2012/11, if the kernel (linux-signed) is signed with the ‘Canonical Ltd. Secure Boot Signing’ key, then grub2 will boot the kernel which will in turn apply quirks and call ExitBootServices. If the kernel is unsigned, grub2 will call ExitBootServices before booting the unsigned kernel)

*Based on earlier work by J. van der Ham. P. Prjevara. O. Odyurt. V. Breider. R. Vermeulen. Version September 9, 2024.

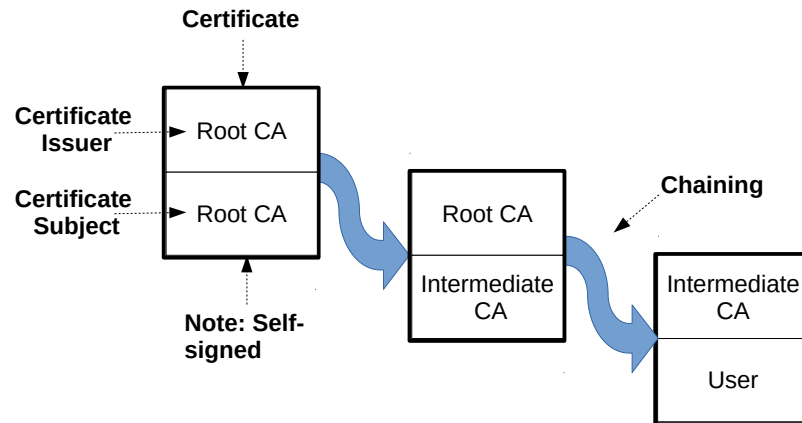


Figure 1: A chain of certificates from the root Certificate Authority to a user's certificate.

4. If signed kernel modules are supported, the signed kernel will verify them during kernel boot."

In this assignment you will verify that Ubuntu actually uses this chain and that the signatures are correct. We will refer to the steps in this description as "Step 1", "Step 2", etc. in the text that follows.

Start by reading the Introduction section of the page on Ubuntu Secure Booting, <https://wiki.ubuntu.com/SecurityTeam/SecureBoot>. If you do not know what a Certification Authority (CA) or certificate chain is (see Figure 1), please read http://en.wikipedia.org/wiki/Certificate_authority.

2 Firmware Databases

Questions

1. Extract the Microsoft certificate that belongs to the key referred to in Step 1 from the UEFI firmware, and show its text representation on your log. *Hint: mokutil, openssl x509*
2. Is this certificate the root certificate in the chain of trust? Is it the ultimate root of trust? What is the role of the Platform Key (PK)?

3 SHIM

Questions

3. Verify that the system indeed boots the 'shim' boot loader in the first stage. What is the full path name of this boot loader?
4. Verify that the 'shim' boot loader is indeed signed with the 'Microsoft Corporation UEFI CA' key. *Hint: sbsigntool, PEM format*

UEFI binaries such as OS/boot loaders, drivers and kernels on Linux are executables in the Windows Authenticode Portable Executable Signature Format. This format is described on https://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599baac1a6c/Authenticode_PE.docx On this page you can download revision 1.0 of the specification of the format.

5. Read the first 9 pages of the specification (up to "Authenticode-Specific Structures"). Focus on the structure of the binaries. What is the name of the part of the binary where the actual signing info is stored?
6. In what standard cryptographic format is the signature data stored?

To extract the signature data from the binary, one needs to determine the exact location and size of this data in the binary. This information is stored in the Data Directories section of the Optional Header of the executable, as shown in Figure 1 of the specification. To retrieve location and size you can use the pyew software, as follows:

0. Clone pyew from <https://github.com/joxeankoret/pyew>. Then install the Ubuntu package python2, and python-pip. Finally, use pip2 capstone to install the capstone decoding library for Python 2.
 - i. Start python2 pyew.py from the command line with the full name of the 'shim' binary as argument.
 - ii. Wait for the pyew prompt to appear (it will do an analysis on the binary which you can interrupt with Ctrl-C if it takes too long)
 - iii. Type `pyew.pe.OPTIONAL_HEADER.DATA_DIRECTORY` to get a listing of all data directory entries.
 - iv. Locate the one called `IMAGE_DIRECTORY_ENTRY_SECURITY`. The location and size are given as the "VirtualAddress:" and "Size:" fields.

Hint: if pyew gives problems to install because of python2, you can also use readpe from the pev package.

We will now study this signature data:

7. Extract the signature data from the 'shim' binary using dd. Add 8 bytes to the location as given in the data directory to skip over the Microsoft WIN_CERTIFICATE structure header (see page 14 of the specification if you are interested). Show the command you used.
8. Show the subject and issuer of any X.509 certificates stored in the signature data. Draw a diagram relating these certificates to the 'Microsoft Corporation UEFI CA' certificate. *Hint: openssl, strongswan-starter*

Now we know that the system indeed boots the 'shim', and that this OS loader is indeed signed by Microsoft and where this signature is stored.

4 GRUB

In Step 2 of the Ubuntu description it says that "the second stage bootloader (grub-efi-amd64-signed) is signed with Canonical's 'Canonical Ltd. Secure Boot Signing' key." This boot loader binary is called 'grubx64.efi' and is located in the same directory as the 'shim' binary.

9. BONUS: Using your new knowledge about Authenticode binaries, extract the signing certificates from the GRUB boot loader, and show the subject and issuer.

If you do this, it will reveal the following certificates:

```
subject=/C=GB/ST=Isle of Man/O=Canonical Ltd./OU=Secure Boot/CN=Canonical
Ltd. Secure Boot Signing
issuer=/C=GB/ST=Isle of Man/L=Douglas/O=Canonical Ltd./CN=Canonical Ltd.
Master Certificate Authority
```

You now know that the GRUB binary is indeed signed, but how does 'shim' verify this signature? To do so 'shim' needs a certificate that it trusts and which must not be modifiable by an external party without detection. The solution chosen by Ubuntu is to store a certificate in the 'shim' binary. We will call this certificate X.

10. Why is storing the certificate X in the 'shim' binary secure?
11. What do you think is the subject CommonName (CN) of this X certificate?
12. Obtain the X certificate used by 'shim' to verify the GRUB binary. There are two ways to obtain it: from the source code or from the binary directly. Hints for the latter case:
 - The certificate is in X.509 DER/ASN.1 format (see `openssl asn1parse`).
 - DER/ASN.1 leaves the CommonName readable.
 - The certificate is 1080 bytes long.

Show the X certificate on your log in text format.

13. Verify that this X certificate's corresponding private key was indeed used to sign the GRUB binary.

5 The Kernel (Bonus)

GRUB allows the user to select from a list of kernels. According to Step 3, GRUB will check if the chosen kernel is signed. If so, GRUB leaves the system in UEFI mode before loading and running the kernel, so that it still has access to the UEFI services.

As with 'shim', GRUB needs a trusted certificate against which it can verify the signed kernel. This is the same certificate as 'shim' uses.

14. Verify the kernel you booted against the X certificate.
15. Where does GRUB get its trusted certificate from? Hint: It is not stored in the binary, and it is not stored on the file system.

We have now verified Steps 1–3 of the Ubuntu chain of trust. Verifying Step 4 is left as a bonus. (Note that the next question is mandatory!)

16. Draw a diagram that shows the chain of trust from the UEFI PK key to the signed kernel. Show all certificates, binaries and signing relations involved.