SSN-2nd-Lab-JOEL_OKORE

Task 1 - Firmware Databases

1. Extract the Microsoft certificate that belongs to the key referred to in Step 1 from the
   UEFI firmware, and show its text representation.
   Hint: efitools, openssl x509

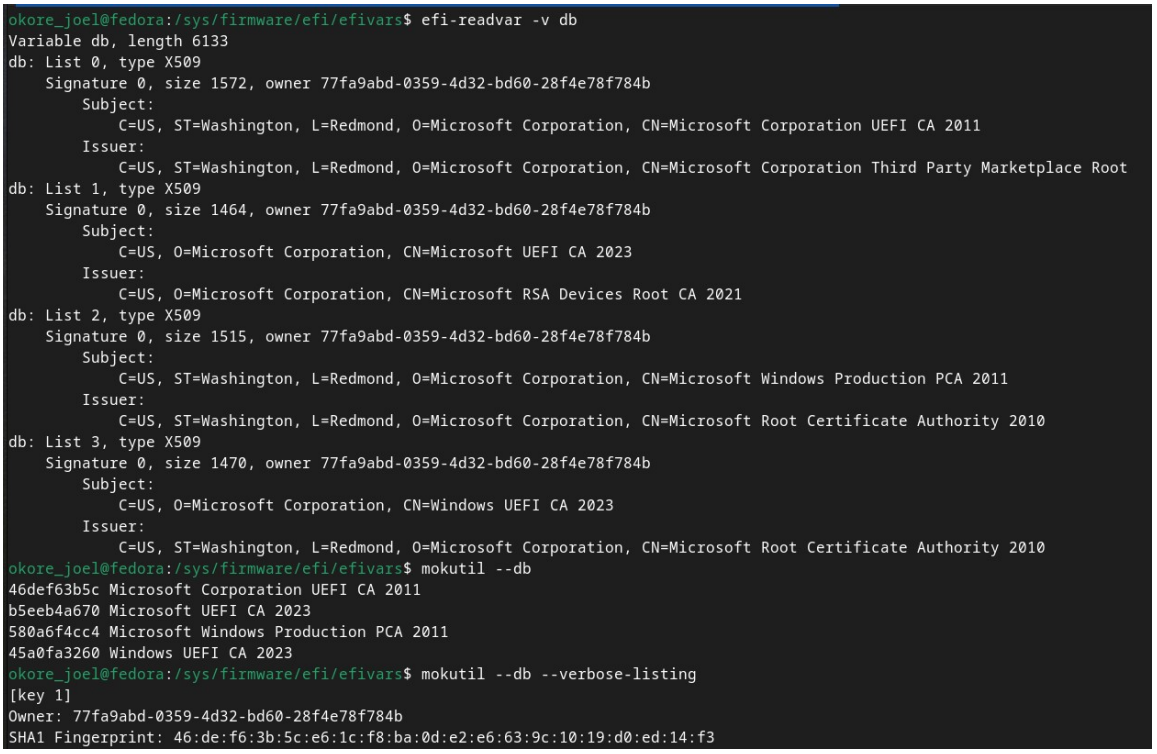
   Having secure boot enabled on my Fedora VM, the following steps were taken in order to obtain certificate within the key used to sign the first stage bootloader (which is the shim) from the signature database. The shim, which acts as an intermediary between the UEFI firmware and the operating system's bootloader is signed by Microsoft UEFI Third-Party Marketplace Root Certificate, which is obtained from Microsoft's Certificate Authority (Microsoft Corporation UEFI CA 2011), all of which are well specified in the certificate below.

   I started by installing every needed utility like 'efitools', 'mokutil' and 'openssl' as shown in pic. 1:

```
okore_joel@fedora:~$ sudo dnf install efitools keyutils mokutil openssl pesign sbsigntools kernel-devel-$(uname -r)

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

For security reasons, the password you type will not be visible.

[sudo] password for okore_joel:
Fedora 40 - x86_64                                                                  2.4 MB/s |  20 MB     00:08
Fedora 40 openh264 (From Cisco) - x86_64                                             88  B/s | 1.4 kB     00:16
Fedora 40 - x86_64 - Updates                                                        2.8 MB/s | 9.8 MB     00:03
Package keyutils-1.6.3-3.fc40.x86_64 is already installed.
Package mokutil-2:0.7.1-1.fc40.x86_64 is already installed.
Dependencies resolved.
================================================================================================================================================
 Package                                 Architecture                Version                                 Repository                     Size
================================================================================================================================================
Installing:
 efitools                                x86_64                      1.9.2-9.fc38                            fedora                        172 k
 kernel-devel                            x86_64                      6.8.5-301.fc40                          fedora                         20 M
 openssl                                 x86_64                      1:3.2.2-3.fc40                          updates                       1.1 M
 pesign                                  x86_64                      116-3.fc40                              fedora                        195 k
 sbsigntools                             x86_64                      0.9.5-6.fc40                            updates                        86 k
Upgrading:
 openssl-libs                            x86_64                      1:3.2.2-3.fc40                          updates                       2.3 M
Installing dependencies:
```

Picture 1 - Installing necessary utilities/dependencies

Making use of the 'efi-readvar' utility which is part of 'efitools', with the '-v' flag to list the content of the signature database as shown in pic. 2:



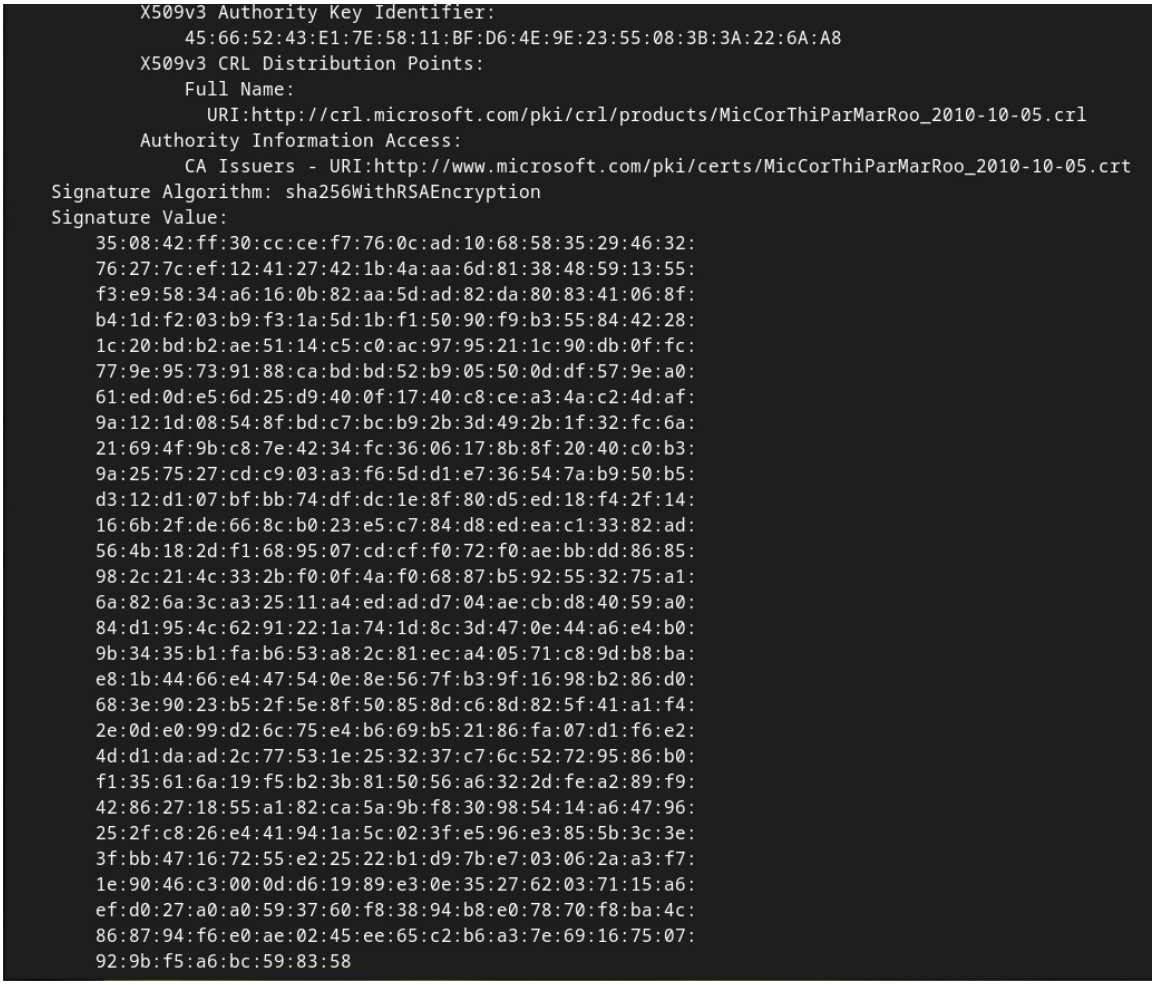
Picture 2 - listing content of DB

To get a more detailed representation of the certificates within the signature database, the ‘mokutil’ utility was used with the ‘—versbose-listing’ option as shown in pic. 3:

```
okore_joel@fedora:/sys/firmware/efi/efivars$ mokutil --db --verbose-listing
[key 1]
Owner: 77fa9abd-0359-4d32-bd60-28f4e78f784b
SHA1 Fingerprint: 46:de:f6:3b:5c:e6:1c:f8:ba:0d:e2:e6:63:9c:10:19:d0:ed:14:f3
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            61:08:d3:c4:00:00:00:00:00:04
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=Washington, L=Redmond, O=Microsoft Corporation, CN=Microsoft Corporation Third Party Marketplace Root
        Validity
            Not Before: Jun 27 21:22:45 2011 GMT
            Not After : Jun 27 21:32:45 2026 GMT
        Subject: C=US, ST=Washington, L=Redmond, O=Microsoft Corporation, CN=Microsoft Corporation UEFI CA 2011
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:a5:08:6c:4c:c7:45:09:6a:4b:0c:a4:c0:87:7f:
                    06:75:0c:43:01:54:64:e0:16:7f:07:ed:92:7d:0b:
                    b2:73:bf:0c:0a:c6:4a:45:61:a0:c5:16:2d:96:d3:
                    f5:2b:a0:fb:4d:49:9b:41:80:90:3c:b9:54:fd:e6:
                    bc:d1:9d:c4:a4:18:8a:7f:41:8a:5c:59:83:68:32:
                    bb:8c:47:c9:ee:71:bc:21:4f:9a:8a:7c:ff:44:3f:
                    8d:8f:32:b2:26:48:ae:75:b5:ee:c9:4c:1e:4a:19:
                    7e:e4:82:9a:1d:78:77:4d:0c:b0:bd:f6:0f:d3:16:
                    d3:bc:fa:2b:a5:51:38:5d:f5:fb:ba:db:78:02:db:
                    ff:ec:0a:1b:96:d5:83:b8:19:13:e9:b6:c0:7b:40:
                    7b:e1:1f:28:27:c9:fa:ef:56:5e:1c:e6:7e:94:7e:
                    c0:f0:44:b2:79:39:e5:da:b2:62:8b:4d:bf:38:70:
                    e2:68:24:14:c9:33:a4:08:37:d5:58:69:5e:d3:7c:
                    ed:c1:04:53:08:e7:4e:b0:2a:87:63:08:61:6f:63:
                    15:59:ea:b2:2b:79:d7:0c:61:67:8a:5b:fd:5e:ad:
                    87:7f:ba:86:67:4f:71:58:12:22:04:22:22:ce:8b:
                    ef:54:71:00:ce:50:35:58:76:95:08:ee:6a:b1:a2:
                    01:d5
                Exponent: 65537 (0x10001)
```

Picture 3 - Verbose listing of signature DB certificates

Below in pic. 4 are some information about the x509 public key certificate definition standard and the signature value of the certificate.



Picture 4 - Certificate signature value

2. Is this certificate the root certificate in the chain of trust? What is the role of the Platform Key (PK) and other variables?

   No, the Microsoft UEFI Third-Party Certificate used to sign the shim is not the root certificate in the UEFI chain of trust. The Platform Key (PK) is usually the root certificate in the UEFI secure boot chain of trust. The Platform Key helps create a relationship between the platform owner and the platform firmware, it is stored in the UEFI platform firmware and is used to validate the Key Exchange Key (KEK), which can in turn be used to make modification to the allow and deny list Databases (DB and DBX respectively).

Task 2 - SHIM

3. Verify that the system indeed boots the shim boot loader in the first stage. What is the full path name of this bootloader?

   To verify that the system indeed boots the shim boot loader in the first stage, I made use of ‘efibootmgr’ utility with the aim of finding a confirmation in form of a reference to ‘shimx64.efi’ as shown in pic. 5 below

```
okore_joel@fedora:/sys/firmware/efi/efivars$ sudo dnf install efibootmgr
Last metadata expiration check: 0:01:55 ago on Thu 05 Sep 2024 02:07:51 PM MSK.
Package efibootmgr-18-6.fc40.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
okore_joel@fedora:/sys/firmware/efi/efivars$ sudo efibootmgr
BootCurrent: 0004
Timeout: 0 seconds
BootOrder: 0004,0000,0001,0002,0003
Boot0000* UiApp FvVol(7cb8bdc9-f8eb-4f34-aaea-3ee4af6516a1)/FvFile(462caa21-7614-4503-836e-8ab6f4662331)
Boot0001* UEFI VBOX CD-ROM VB2-01700376         PciRoot(0x0)/Pci(0x1,0x1)/Ata(1,0,0){auto_created_boot_option}
Boot0002* UEFI VBOX HARDDISK VB431c6dc5-e673c4e6        PciRoot(0x0)/Pci(0xd,0x0)/Sata(0,65535,0){auto_created_boot_option}
Boot0003  EFI Internal Shell    FvVol(7cb8bdc9-f8eb-4f34-aaea-3ee4af6516a1)/FvFile(7c04a583-9e3e-4f1c-ad65-e05268d0b4d1)
Boot0004* Fedora        HD(1,GPT,011ee646-f347-4126-ab31-00dce72d0842,0x800,0x12c000)/\EFI\fedora\shimx64.efi
```

Picture 5 - Shim bootloader verification using ‘efibootmgr’

4. Verify that the shim bootloader is indeed signed with the “Microsoft Corporation UEFI CA” key.

   Upon executing the command below using ‘sbverify’ utility, the result ascertain that the shim bootloader is indeed signed with “Microsoft Corporation UEFI CA” key as shown in pic. 6

```
okore_joel@fedora:/sys/firmware/efi/efivars$ sudo dnf install sbsigntools
Last metadata expiration check: 0:05:26 ago on Thu 05 Sep 2024 02:07:51 PM MSK.
Package sbsigntools-0.9.5-6.fc40.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
okore_joel@fedora:/sys/firmware/efi/efivars$ sudo sbverify --list /boot/efi/EFI/fedora/shimx64.efi
warning: data remaining[823272 vs 949424]: gaps between PE/COFF sections?
signature 1
image signature issuers:
 - /C=US/ST=Washington/L=Redmond/O=Microsoft Corporation/CN=Microsoft Corporation UEFI CA 2011
image signature certificates:
 - subject: /C=US/ST=Washington/L=Redmond/O=Microsoft Corporation/CN=Microsoft Windows UEFI Driver Publisher
   issuer:  /C=US/ST=Washington/L=Redmond/O=Microsoft Corporation/CN=Microsoft Corporation UEFI CA 2011
 - subject: /C=US/ST=Washington/L=Redmond/O=Microsoft Corporation/CN=Microsoft Corporation UEFI CA 2011
   issuer:  /C=US/ST=Washington/L=Redmond/O=Microsoft Corporation/CN=Microsoft Corporation Third Party Marketplace Root
```

Picture 6 - Verifying signature authenticity of shim bootloader

5. What is the exact name of the part of the binary where the actual signature is stored?

   The authenticode signature as per the Windows Authenticode Portable Executable Signature format stipulated based on the Public-Key

Cryptography Standards (PKCS) #7 is stored in the Attribute Certificate Table, which contains several section like ContentInfo, certificates and SignerInfo.

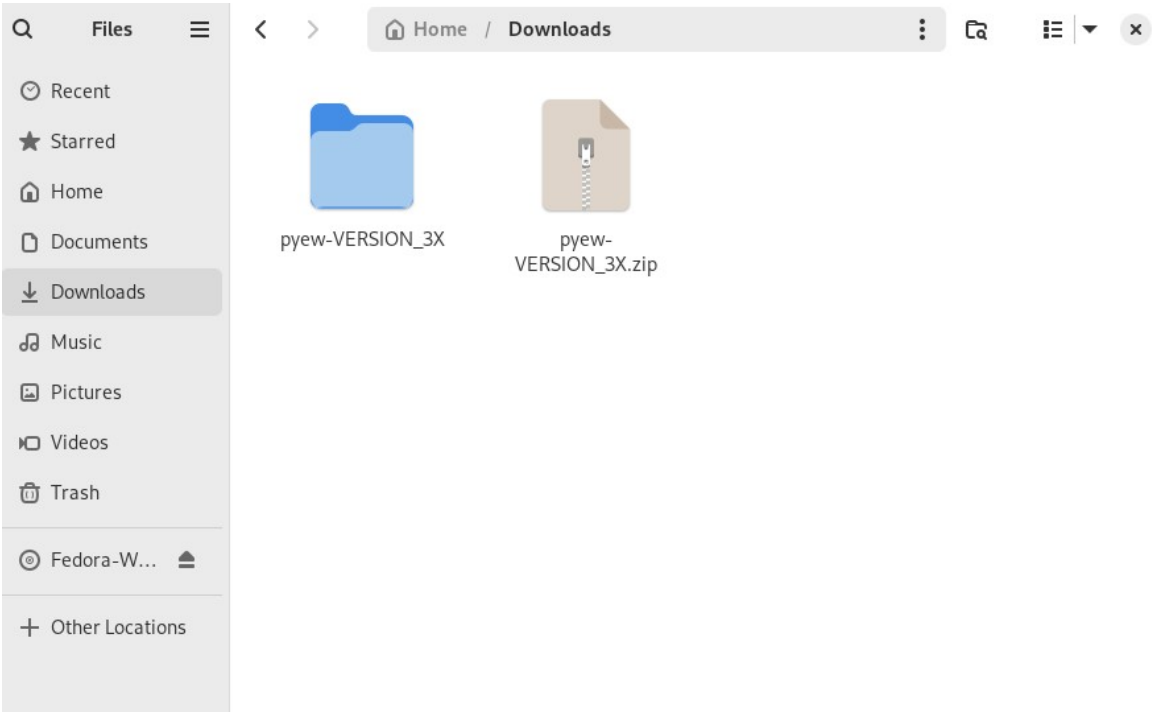
6. In what standard cryptographic format is the signature data stored?

   As mention in the answer to the previous question the signature data in UEFI binaries, stored in the Attribute Certificate Table, is stored in the **PKCS #7** (Public Key Cryptography Standards) format.

   Due to difficulties installing the 'pyew' package through the terminal, I chose to clone it from the Git repository, as demonstrated in pictures 7 and 8.

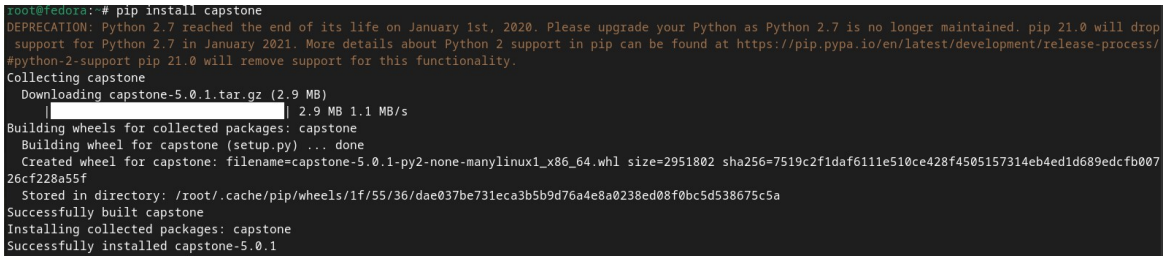
```
okore_joel@fedora:~$ sudo dnf install pyew
[sudo] password for okore_joel:
Last metadata expiration check: 1:57:17 ago on Thu 05 Sep 2024 02:07:51 PM MSK.
No match for argument: pyew
Error: Unable to find a match: pyew
okore_joel@fedora:~$
```

Picture 7 - pyew package installation error



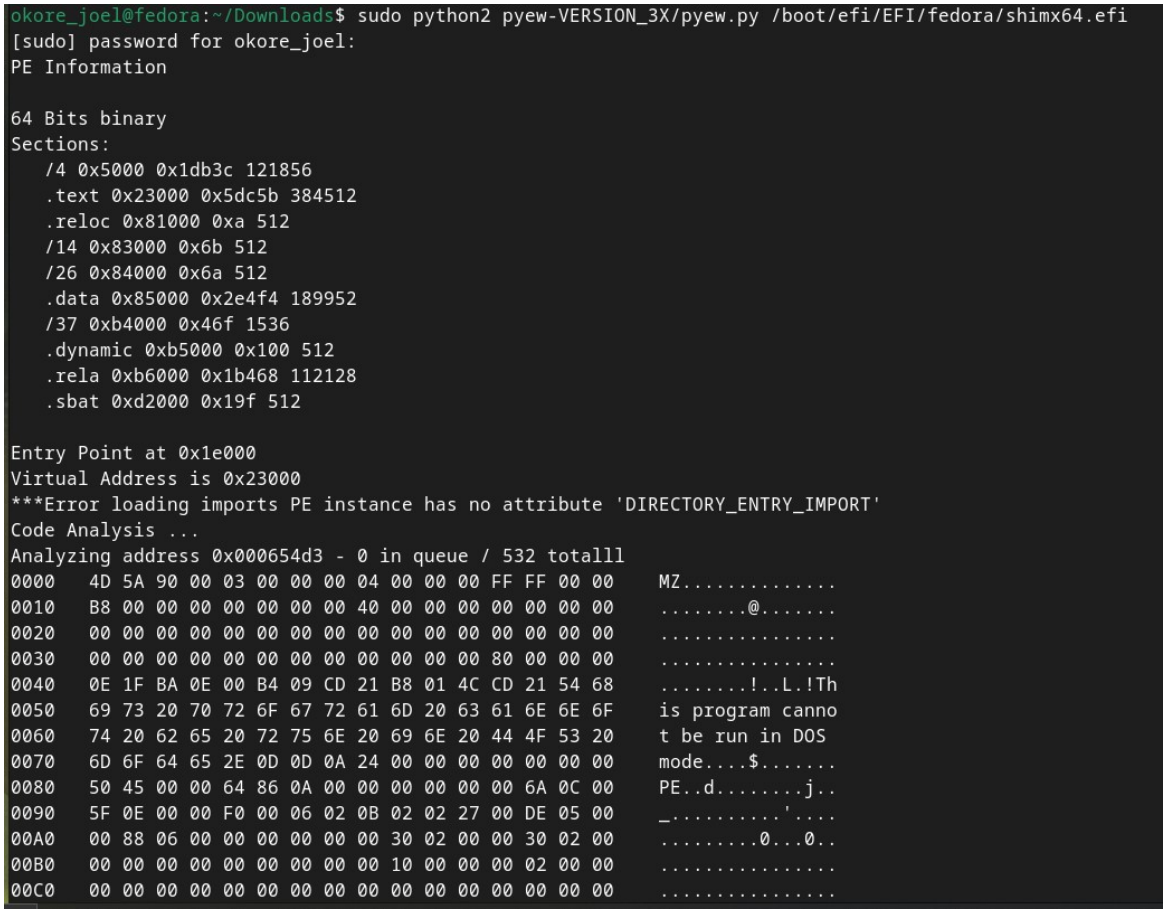
Picture 8 - Clone pyew package from GitHub

Consequently, the Capstone Python library needed to be installed as a dependency for the ‘pyew’ package, as demonstrated in pic. 9.



Picture 9 - Installing Capstone python library

The ‘pyew’ package was then used to aid the extraction of the signature data from the SHIM (‘shimx64.efi’ binary) as demonstrated in pic. 10.



Picture 10 - Running pyew package on SHIM binary

An interactive command line interface was then provided by the pyew package. I obtained the listing of all data entries by executing the command 'pyew.pe.OPTION_HEADER.DATA_DIRECTORY' on pyew command interface (Pic. 11).

```
[0x00000000]> pyew.pe.OPTIONAL_HEADER.DATA_DIRECTORY
[<Structure: [IMAGE_DIRECTORY_ENTRY_EXPORT] 0x108 0x0 VirtualAddress: 0x0 0x10C 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_IMPORT] 0x110 0x0 VirtualAddress: 0x0 0x114 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_RESOURCE] 0x118 0x0 VirtualAddress: 0x0 0x11C 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_EXCEPTION] 0x120 0x0 VirtualAddress: 0x0 0x124 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_SECURITY] 0x128 0x0 VirtualAddress: 0xE56C8 0x12C 0x4 Size: 0x25E8>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_BASERELOC] 0x130 0x0 VirtualAddress: 0x81000 0x134 0x4 Size: 0xA>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_DEBUG] 0x138 0x0 VirtualAddress: 0x0 0x13C 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_COPYRIGHT] 0x140 0x0 VirtualAddress: 0x0 0x144 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_GLOBALPTR] 0x148 0x0 VirtualAddress: 0x0 0x14C 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_TLS] 0x150 0x0 VirtualAddress: 0x0 0x154 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG] 0x158 0x0 VirtualAddress: 0x0 0x15C 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT] 0x160 0x0 VirtualAddress: 0x0 0x164 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_IAT] 0x168 0x0 VirtualAddress: 0x0 0x16C 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT] 0x170 0x0 VirtualAddress: 0x0 0x174 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR] 0x178 0x0 VirtualAddress: 0x0 0x17C 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_RESERVED] 0x180 0x0 VirtualAddress: 0x0 0x184 0x4 Size: 0x0>]
```

Picture 11 - Data entries list from SHIM binary

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).

   Using the 'dd' Linux command with specific flags and values, I extracted the signature data from the SHIM binary. The image above clearly shows the VirtualAddress of IMAGE_DIRECTORY_ENTRY_SECURITY directory, which holds the signature data, as 0xE56C8 (939720 in decimal). The Size of the directory is 0x25E8 (9704 in decimal). To extract only the signature data, I added 8 bytes to the VirtualAddress of the directory, skipping the certificate structure header. Pic. 12 demonstrates the extraction process.

```
okore_joel@fedora:~$ sudo dd if=/boot/efi/EFI/fedora/shimx64.efi of=signature_data.bin bs=1 skip=939728 count=9704
9696+0 records in
9696+0 records out
9696 bytes (9.7 kB, 9.5 KiB) copied, 0.0360935 s, 269 kB/s
okore_joel@fedora:~$ ls
Desktop  Documents  Downloads  get-pip.py  Music  Pictures  Public  pyew_env  signature_data.bin  Templates  Videos
```

Picture 12 - Signature data extraction process on SHIM binary

8. Show the subject and issuer of all X.509 certificates stored in the signature data. Draw a diagram relating these certificates to the "Microsoft Corporation UEFI CA" certificate.

The subject and issuer of the extracted certificates are shown in pic. 12, 13.

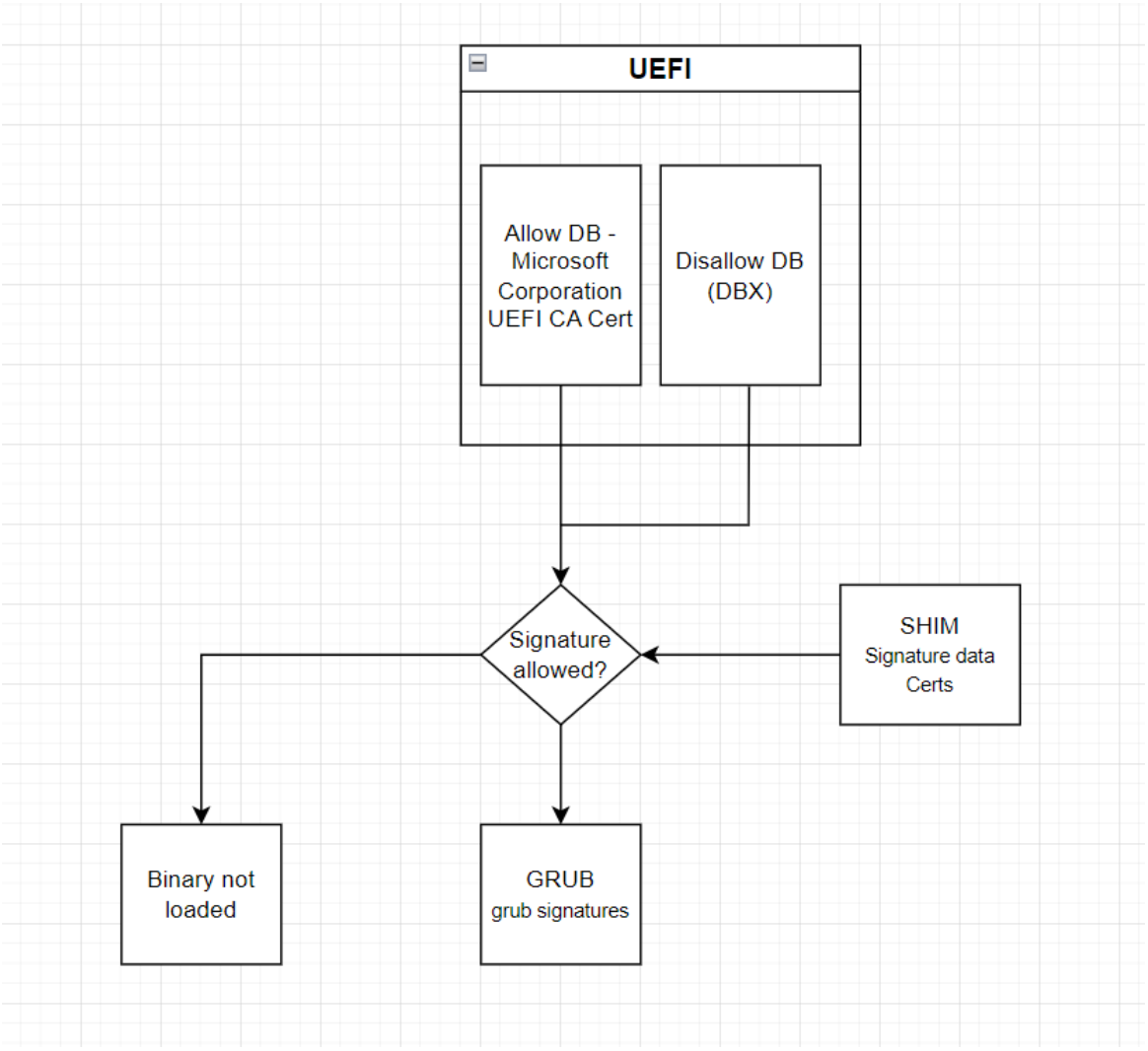
```
okore_joel@fedora:~$ openssl pkcs7 -inform DER -in signature_data.bin -print_certs -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            33:00:00:00:5f:46:39:06:b9:50:a7:39:ed:00:01:00:00:00:5f
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=Washington, L=Redmond, O=Microsoft Corporation, CN=Microsoft Corporation UEFI CA 2011
        Validity
            Not Before: Oct 19 19:53:24 2023 GMT
            Not After : Oct 16 19:53:24 2024 GMT
        Subject: C=US, ST=Washington, L=Redmond, O=Microsoft Corporation, CN=Microsoft Windows UEFI Driver Publisher
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:cd:a4:cc:91:ca:af:ff:27:b6:a9:52:86:3f:67:
                    c4:02:da:16:31:f0:98:c8:e1:e9:9b:07:56:e2:1f:
                    da:01:6c:40:1a:ce:4f:d0:92:fc:cb:14:2a:01:1a:
                    46:ce:bb:53:3b:69:21:ee:a9:fd:5d:b7:e0:3f:01:
                    76:7e:68:c0:b7:4a:af:b4:c4:ba:d9:11:c2:71:99:
                    14:e1:67:87:eb:2f:3a:18:9b:ea:8e:1e:9e:12:aa:
                    bb:78:05:2c:74:6e:55:b9:da:3e:38:5e:b5:6d:48:
                    fc:4f:1d:ec:f9:a4:01:63:2a:a8:c3:07:4c:5c:74:
                    bf:51:a8:da:41:d3:39:d2:bc:a8:8a:26:1e:b6:ef:
                    0f:8d:9e:3c:6d:4a:9c:12:e0:13:52:a7:cc:83:59:
                    f0:8c:5e:29:0d:2e:b6:b6:e6:e3:e3:8a:ba:79:7b:
                    95:10:02:5e:0e:8c:e3:81:a2:19:ec:18:3b:16:12:
                    4a:e7:63:2c:0d:63:3b:d9:0e:7c:ea:16:f8:32:d0:
                    ce:50:18:d3:06:c3:ec:fb:6f:b6:4a:04:df:44:41:
                    18:6e:33:b5:10:7d:94:c2:fe:4f:4f:21:a7:56:50:
                    26:4e:5b:34:4b:31:2c:d4:63:7e:9e:f2:e2:10:f3:
                    e8:54:90:58:9b:d9:22:22:6c:6c:63:f5:d6:8f:98:
                    c4:97
                Exponent: 65537 (0x10001)
```

Picture 13 - Details on extracted UEFI Driver Publisher Certificate

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            61:08:d3:c4:00:00:00:00:00:04
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=Washington, L=Redmond, O=Microsoft Corporation, CN=Microsoft Corporation Third Party Marketplace Root
        Validity
            Not Before: Jun 27 21:22:45 2011 GMT
            Not After : Jun 27 21:32:45 2026 GMT
        Subject: C=US, ST=Washington, L=Redmond, O=Microsoft Corporation, CN=Microsoft Corporation UEFI CA 2011
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:a5:08:6c:4c:c7:45:09:6a:4b:0c:a4:c0:87:7f:
                    06:75:0c:43:01:54:64:e0:16:7f:07:ed:92:7d:0b:
                    b2:73:bf:0c:0a:c6:4a:45:61:a0:c5:16:2d:96:d3:
                    f5:2b:a0:fb:4d:49:9b:41:80:90:3c:b9:54:fd:e6:
                    bc:d1:9d:c4:a4:18:8a:7f:41:8a:5c:59:83:68:32:
                    bb:8c:47:c9:ee:71:bc:21:4f:9a:8a:7c:ff:44:3f:
                    8d:8f:32:b2:26:48:ae:75:b5:ee:c9:4c:1e:4a:19:
                    7e:e4:82:9a:1d:78:77:4d:0c:b0:bd:f6:0f:d3:16:
                    d3:bc:fa:2b:a5:51:38:5d:f5:fb:ba:db:78:02:db:
                    ff:ec:0a:1b:96:d5:83:b8:19:13:e9:b6:c0:7b:40:
                    7b:e1:1f:28:27:c9:fa:ef:56:5e:1c:e6:7e:94:7e:
                    c0:f0:44:b2:79:39:e5:da:b2:62:8b:4d:bf:38:70:
                    e2:68:24:14:c9:33:a4:08:37:d5:58:69:5e:d3:7c:
                    ed:c1:04:53:08:e7:4e:b0:2a:87:63:08:61:6f:63:
                    15:59:ea:b2:2b:79:d7:0c:61:67:8a:5b:fd:5e:ad:
                    87:7f:ba:86:67:4f:71:58:12:22:04:22:22:ce:8b:
                    ef:54:71:00:ce:50:35:58:76:95:08:ee:6a:b1:a2:
                    01:d5
                Exponent: 65537 (0x10001)
```

Picture 14 - Details on extracted UEFI CA 2011 Certificate

Below is a diagram that demonstrated the relationship betwen the extracted signature data certificates from the shim binary to the “Microsoft Corporation UEFI CA” certificate present in the UEFI signature DB (Pic. 15)



Picture 15 - Relationship between extracted certificates and signature DB certificate

TASK 3 - GRUB (BONUS)

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

   To begin this task, I needed to locate the GRUB bootloader binary. I used the 'find' Linux command for this purpose (pic. 16)

```
okore_joel@fedora:/$ sudo find / -iname grubx64.efi -type f
[sudo] password for okore_joel:
/boot/efi/EFI/fedora/grubx64.efi
find: ‘/run/user/1000/gvfs’: Permission denied
find: ‘/run/user/1000/doc’: Permission denied
/run/media/okore_joel/Fedora-WS-Live-40-1-14/EFI/BOOT/grubx64.efi
```

Picture 16 - using ‘find’ command to locate GRUB binary

I then analyzed the binary using the 'pyew' package, performing an extraction process identical to the one used for the SHIM binary as demonstrated in pic. 17.

```
okore_joel@fedora:/$ sudo python2 /home/okore_joel/Downloads/pyew-VERSION_3X/./pyew.py /boot/efi/EFI/fedora/grubx64.efi
[sudo] password for okore_joel:
PE Information

64 Bits binary
Sections:
   .text 0x1000 0x19000 102400
   .data 0x1a000 0x12000 73728
   mods 0x2c000 0x399000 3772416
   .sbat 0x3c5000 0x1000 4096
   .reloc 0x3c6000 0x2000 8192

Entry Point at 0x1000
Virtual Address is 0x1000
***Error loading imports PE instance has no attribute 'DIRECTORY_ENTRY_IMPORT'
Code Analysis ...
Analyzing address 0x00006dfe - 17 in queue / 246 totall
0000   4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00    MZ..............
0010   B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00    ........@.......
0020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0030   00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00    ................
0040   0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68    ........!..L.!Th
0050   69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F    is program canno
0060   74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20    t be run in DOS
0070   6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00    mode....$.......
0080   50 45 00 00 64 86 05 00 00 8E A4 54 00 00 00 00    PE..d......T....
0090   00 00 00 00 F0 00 0E 02 0B 02 00 00 00 90 01 00    ................
00A0   00 C0 3A 00 00 00 00 00 00 10 00 00 00 10 00 00    ..:.............
00B0   00 00 00 00 00 00 00 00 00 10 00 00 00 10 00 00    ................
00C0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00D0   00 80 3C 00 00 10 00 00 00 00 00 00 0A 00 00 01    ..<.............
00E0   00 00 01 00 00 00 00 00 00 00 01 00 00 00 00 00    ................
00F0   00 00 01 00 00 00 00 00 00 00 01 00 00 00 00 00    ................
0100   00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00    ................
0110   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
```

Picture 17 - Running pyew package on GRUB binary

I listed the data entries by executing the command 'pyew.pe.OPTION_HEADER.DATA_DIRECTORY' in the pyew command interface (Pic. 18)

```
[0x00000000]> pyew.pe.OPTIONAL_HEADER.DATA_DIRECTORY
[<Structure: [IMAGE_DIRECTORY_ENTRY_EXPORT] 0x108 0x0 VirtualAddress: 0x0 0x10C 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_IMPORT] 0x110 0x0 VirtualAddress: 0x0 0x114 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_RESOURCE] 0x118 0x0 VirtualAddress: 0x0 0x11C 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_EXCEPTION] 0x120 0x0 VirtualAddress: 0x0 0x124 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_SECURITY] 0x128 0x0 VirtualAddress: 0x3C8000 0x12C 0x4 Size: 0xD40>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_BASERELOC] 0x130 0x0 VirtualAddress: 0x3C6000 0x134 0x4 Size: 0x2000>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_DEBUG] 0x138 0x0 VirtualAddress: 0x0 0x13C 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_COPYRIGHT] 0x140 0x0 VirtualAddress: 0x0 0x144 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_GLOBALPTR] 0x148 0x0 VirtualAddress: 0x0 0x14C 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_TLS] 0x150 0x0 VirtualAddress: 0x0 0x154 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG] 0x158 0x0 VirtualAddress: 0x0 0x15C 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT] 0x160 0x0 VirtualAddress: 0x0 0x164 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_IAT] 0x168 0x0 VirtualAddress: 0x0 0x16C 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT] 0x170 0x0 VirtualAddress: 0x0 0x174 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR] 0x178 0x0 VirtualAddress: 0x0 0x17C 0x4 Size: 0x0>,
 <Structure: [IMAGE_DIRECTORY_ENTRY_RESERVED] 0x180 0x0 VirtualAddress: 0x0 0x184 0x4 Size: 0x0>]
[0x00000000]> exit
```

Picture 18 - Data entries list from GRUB binary

I extracted the signature data from the GRUB binary using an 8-byte offset from the VirtualAddress of 'IMAGE_DIRECTORY_ENTRY_SECURITY', mirroring the process used for the SHIM binary (Pic. 19).

```
okore_joel@fedora:/$ sudo dd if=/boot/efi/EFI/fedora/grubx64.efi of=signature_data_grub.bin bs=1 skip=3964936 count=3392
3384+0 records in
3384+0 records out
3384 bytes (3.4 kB, 3.3 KiB) copied, 0.0063255 s, 535 kB/s
```

Picture 19 - Signature data extraction process on GRUB binary

The details of the extracted signature data from the GRUB binary are shown below (Pic. 20)

```
okore_joel@fedora:/$ openssl pkcs7 -inform DER -in signature_data_grub.bin -print_certs -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 2574710357 (0x9976f655)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=Fedora Secure Boot CA
        Validity
            Not Before: Dec  7 16:04:24 2012 GMT
            Not After : Dec  5 16:04:24 2022 GMT
        Subject: CN=Fedora Secure Boot Signer
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:c2:6e:3c:b1:7f:56:ab:09:56:5f:b2:32:98:ac:
                    33:8b:d3:9a:99:96:03:63:ee:19:d6:42:d9:3b:52:
                    8b:eb:33:99:53:c2:50:77:bf:88:2f:57:6b:e6:03:
                    81:e4:33:44:d8:95:3c:36:2e:04:99:dd:1b:ce:74:
                    4a:53:de:2b:0a:a0:63:0f:56:6b:1e:ed:5f:53:a2:
                    d3:0d:ed:e3:fd:0e:d2:b6:39:bb:e2:8b:45:4e:6e:
                    ce:c9:ac:29:1a:2f:08:d4:cf:12:a3:87:9b:fb:46:
                    ef:64:b7:49:18:2f:ae:7a:58:25:7c:ff:4f:a2:15:
                    fe:c7:96:03:04:6e:ec:9e:0e:aa:4b:0e:e4:24:d6:
                    c0:f7:ce:06:3a:b9:86:82:13:87:0b:96:4d:3d:e9:
                    21:5a:c8:53:e4:8e:69:88:be:ff:a7:0c:39:ff:b4:
                    e0:a3:c2:2d:f2:83:02:e9:64:9e:94:5f:76:a2:42:
                    69:45:d7:39:a8:59:51:88:d8:2b:bc:3a:5e:72:46:
                    5e:72:a7:67:ab:e2:d5:61:bd:46:1a:80:3f:c6:8b:
                    3b:92:6f:e4:c3:4b:75:b4:35:33:1e:84:33:1f:07:
                    13:01:97:8a:b3:62:68:78:de:59:0a:55:7f:1f:7e:
                    90:6e:4a:9b:fc:56:56:48:58:ad:80:46:a0:fc:22:
                    60:95
```

Picture 20 - Signature data extraction details (Issuer, Subject)

10. Why is storing the certificate X in the shim binary secure?

    Because UEFI Secure Boot establishes a chain of trust, storing the Canonical Ltd certificate in the shim binary is secure. The trusted root certificate (Microsoft Corporation UEFI CA) is used by the UEFI firmware to validate the shim binary, which is signed by Canonical Ltd. This ensures that during boot, only verified and reliable shims are loaded. Then, using its own certificate, the shim itself authenticates further bootloaders (such as GRUB). This helps to preserve security via a chain of trust that begins with the UEFI firmware and reaches the various parts of the operating system.

11. What do you think is the subject CommonName (CN) of this X certificate?

    The subject CommonName (CN) of the above certificate, as shown in pic. 20, is **Fedora Secure Boot Signer**

12. Obtain the X certificate used by the shim to verify the GRUB binary.

    The content of the certificate extracted from the data is shown below (Pic. 21)

```
-----BEGIN CERTIFICATE-----
MIIDbDCCAlSgAwIBAgIFAJl29lUwDQYJKoZIhvcNAQELBQAwIDEeMBwGA1UEAxMV
RmVkb3JhIFNlY3VyZSBCb290IENBMB4XDTEyMTIwNzE2MDQyNFoXDTIyMTIwNTE2
MDQyNFowJDEiMCAGA1UEAxMZRmVkb3JhIFNlY3VyZSBCb290IFNpZ25lcjCCASIw
DQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMJuPLF/VqsJVl+yMpisM4vTmpmW
A2PuGdZC2TtSi+szmVPCUHe/iC9Xa+YDgeQzRNiVPDYuBJndG850SlPeKwqgYw9W
ax7tX1Oi0w3t4/0O0rY5u+KLRU5uzsmsKRovCNTPEqOHm/tG72S3SRgvrnpYJXz/
T6IV/seWAwRu7J4OqksO5CTWwPfOBjq5hoIThwuWTT3pIVrIU+SOaYi+/6cMOf+0
4KPCLfKDAulknpRfdqJCaUXXOahZUYjYK7w6XnJGXnKnZ6vi1WG9RhqAP8aLO5Jv
5MNLdbQ1Mx6EMx8HEwGXirNiaHjeWQpVfx9+kG5Km/xWVkhYrYBGoPwiYJUCAwEA
AaOBqDCBpTBOBggrBgEFBQcBAQRCMEAwPgYIKwYBBQUHMAKGMmh0dHBzOi8vZmVk
b3JhcHJvamVjdC5vcmcvd2lraS9GZWF0dXJlcy9TZWN1cmVCb290MB8GA1UdIwQY
MBaAFP3jJZnC1h2xv1gHM117IOTNljtCMBMGA1UdJQQMMAoGCCsGAQUFBwMDMB0G
A1UdDgQWBBQPpy6PPwV+OWgKeHWLOsL5qIWJZzANBgkqhkiG9w0BAQsFAAOCAQEA
l9D5DhLlxOholAMf1jr5NcGTcMl5+vwOzTvR/tB76Xk5iFcQDQ9RU+uqrvRzeH9m
bm5uG1WDu9QHwdsWKmr9vUxEDKe1m3FLSP6UhA0wI0YXADGAOYFM2wwfLLy7NgKt
R4vZa4c0CcQGz1B5RXulh33eeUUh5aCclJg8+RfS7v4ElsO/jsfnkeWAhB8I83bp
KFOU7Z0xsEd2zWRb7nJnbHtESUdi5oB9WCok6tUpMTGTyIk7qhfQ3V8lKSIU7jx5
YC7jr+ZOeTvEb4tcA9cp277KsQ5bYDQ4nERUE+NMFvggG4DsuWFM24JWgYu5kQ34
nik84fAujJIEQsPfl4c6zQ==
-----END CERTIFICATE-----
```

Picture 21 - Certificate from GRUB binary

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

    To verify that the Canonical Ltd certificate was indeed used to sign the GRUB binary, I first needed to copy the content of the certificate—extracted from the GRUB binary's signature data—into a Privacy-Enhanced Mail (.PEM) key file as shown in pic. 22.

```
okore_joel@fedora:/$ cd home/okore_joel/
okore_joel@fedora:~$ gedit extracted_cert.pem
okore_joel@fedora:~$ cat extracted_cert.pem
-----BEGIN CERTIFICATE-----
MIIDbDCCAlSgAwIBAgIFAJl29lUwDQYJKoZIhvcNAQELBQAwIDEeMBwGA1UEAxMV
RmVkb3JhIFNlY3VyZSBCb290IENBMB4XDTEyMTIwNzE2MDQyNFoXDTIyMTIwNTE2
MDQyNFowJDEiMCAGA1UEAxMZRmVkb3JhIFNlY3VyZSBCb290IFNpZ25lcjCCASIw
DQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMJuPLF/VqsJVl+yMpisM4vTmpmW
A2PuGdZC2TtSi+szmVPCUHe/iC9Xa+YDgeQzRNiVPDYuBJndG850SlPeKwqgYw9W
ax7tX1Oi0w3t4/0O0rY5u+KLRU5uzsmsKRovCNTPEqOHm/tG72S3SRgvrnpYJXz/
T6IV/seWAwRu7J4OqksO5CTWwPfOBjq5hoIThwuWTT3pIVrIU+SOaYi+/6cMOf+0
4KPCLfKDAulknpRfdqJCaUXXOahZUYjYK7w6XnJGXnKnZ6vi1WG9RhqAP8aLO5Jv
5MNLdbQ1Mx6EMx8HEwGXirNiaHjeWQpVfx9+kG5Km/xWVkhYrYBGoPwiYJUCAwEA
AaOBqDCBpTBOBggrBgEFBQcBAQRCMEAwPgYIKwYBBQUHMAKGMmh0dHBzOi8vZmVk
b3JhcHJvamVjdC5vcmcvd2lraS9GZWF0dXJlcy9TZWN1cmVCb290MB8GA1UdIwQY
MBaAFP3jJZnC1h2xv1gHM117IOTNljtCMBMGA1UdJQQMMAoGCCsGAQUFBwMDMB0G
A1UdDgQWBBQPpy6PPwV+OWgKeHWLOsL5qIWJZzANBgkqhkiG9w0BAQsFAAOCAQEA
l9D5DhLlxOholAMf1jr5NcGTcMl5+vwOzTvR/tB76Xk5iFcQDQ9RU+uqrvRzeH9m
bm5uG1WDu9QHwdsWKmr9vUxEDKe1m3FLSP6UhA0wI0YXADGAOYFM2wwfLLy7NgKt
R4vZa4c0CcQGz1B5RXulh33eeUUh5aCclJg8+RfS7v4ElsO/jsfnkeWAhB8I83bp
KFOU7Z0xsEd2zWRb7nJnbHtESUdi5oB9WCok6tUpMTGTyIk7qhfQ3V8lKSIU7jx5
YC7jr+ZOeTvEb4tcA9cp277KsQ5bYDQ4nERUE+NMFvggG4DsuWFM24JWgYu5kQ34
nik84fAujJIEQsPfl4c6zQ==
-----END CERTIFICATE-----
```

Picture 22 - Copying Canonical Ltd Certificate to key file

I then made use of the Linux utility 'sbverify' to verify that indeed, the extracted certificate was used to sign the GRUB binary as shown in pic. 23.

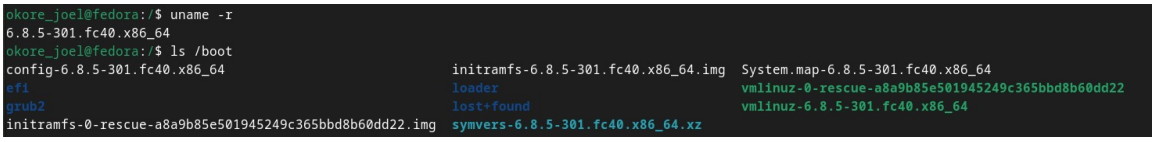
```
okore_joel@fedora:~$ sudo sbverify --cert extracted_cert.pem /boot/efi/EFI/fedora/grubx64.efi
Signature verification OK
okore_joel@fedora:~$
```

Picture 23 - Signature Verified Successfully

TASK 4 - THE KERNEL (OPTIONAL)

14. Verify the kernel you booted against the X certificate

To verify that the kernel I booted is signed using the trusted X certificate, I needed to identify the current kernel image which is usually located in the '/boot' directory and contains as part of its name the kernel version gotten from executing the command 'uname -r' on the terminal (Pic. 24)



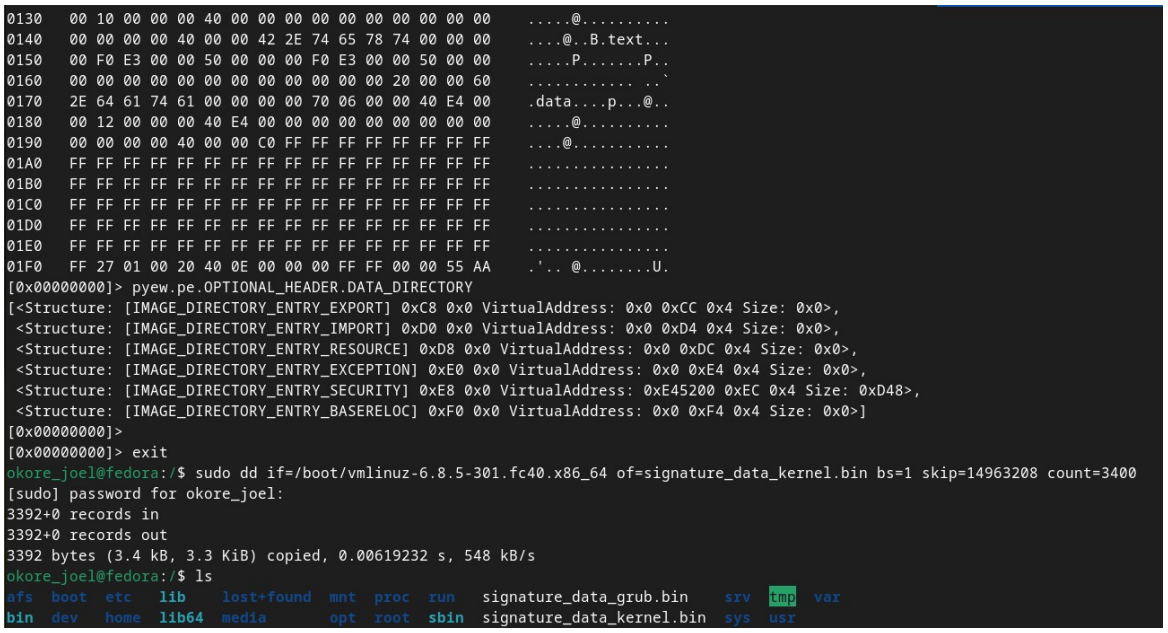
Picture 24 - Locating currently booted kernel image

I then went on to analyze the kernel binary using 'pyew' package as shown in pic. 25.

Picture 25 - Analyzing kernel binary using 'pyew' package

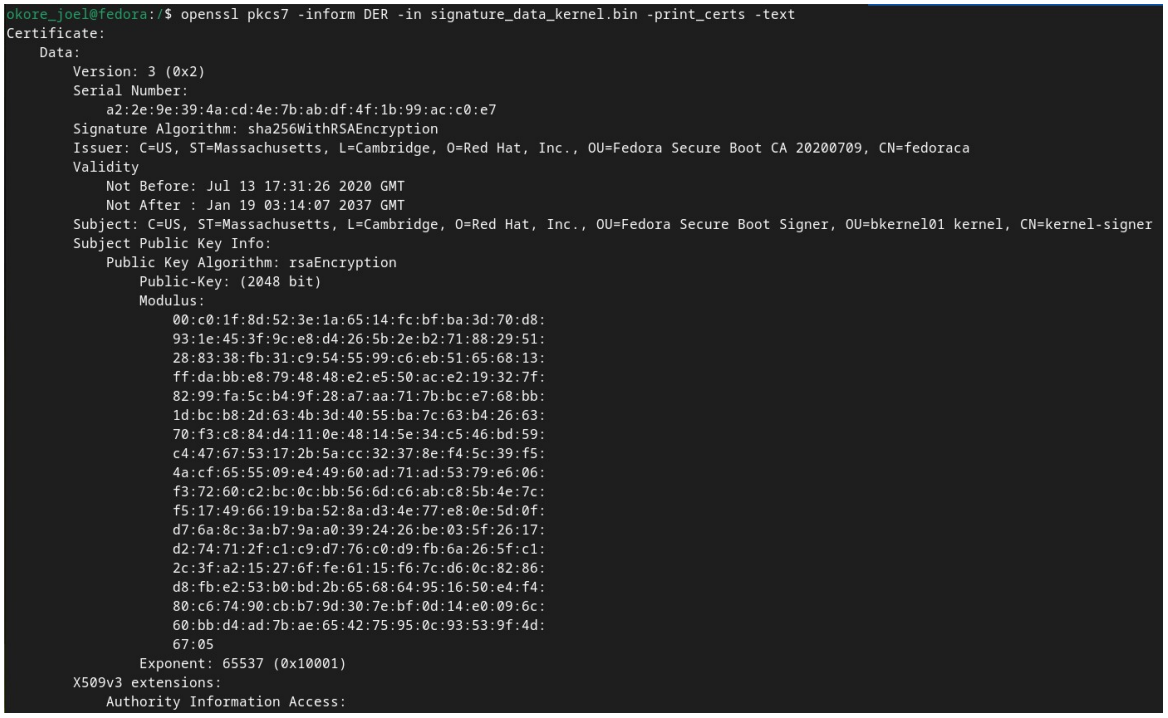
I extracted the signature data into a binary file name "signature_data_kernel.bin", making sure to offset the VirtualAddress of the

'IMAGE_DIRECTORY_ENTRY_SECURITY' directory, mirroring the process used for the two previous binaries, setting the block size flag (bs) to 1 and count flag to the size of the section that needs to be extracted (Pic. 26).



Picture 26 - Signature data extraction process on Kernel binary

I leveraged the ‘openssl’ Linux utility to display the extracted data in text format (Pic. 27)



Picture 27 - Using openssl utility to display extracted data (binary format) in text format

Here is the certificate section of the signature data, with subject CommonName ‘kernel-signer’, that was extracted from the kernel binary (Pic. 28)

```
        f5:2c:8e:dc:e8:ac:4e:7a:f2:b8:14:7b:f1:df:df:95:a8:c0:
        0e:71:6d:85:22:aa:95:b1:57:f7:bf:7c:7b:60:9a:79:0e:4a:
        a9:7e:ed:4c
-----BEGIN CERTIFICATE-----
MIIEeTCCA2GgAwIBAgIRAKIunjlKzU57q99PG5mswOcwDQYJKoZIhvcNAQELBQAw
gY0xCzAJBgNVBAYTAlVTMRYwFAYDVQQIEw1NYXNzYWNodXNldHRzMRIwEAYDVQQH
EwlDYW1icmlkZ2UxFjAUBgNVBAoTDVJlZCBIYXQsIEluYy4xJzAlBgNVBAsTHkZl
ZG9yYSBTZWN1cmUgQm9vdCBDQSAyMDIwMDcwOTERMA8GA1UEAxMIZmVkb3JhY2Ew
HhcNMjAwNzEzMTczMTI2WhcNMzcwMTE5MDMxNDA3WjCBqDELMAkGA1UEBhMCVVMx
FjAUBgNVBAgTDU1hc3NhY2h1c2V0dHMxEjAQBgNVBAcTCUNhbWJyaWRnZTEWMBQG
A1UEChMNUmVkIEhhdCwgSW5jLjEiMCAGA1UECxMZRmVkb3JhIFNlY3VyZSBCb290
IFNpZ25lcjEZMBcGA1UECxMQYmtlcm5lbDAxIGtlcm5lbDEWMBQGA1UEAxMNa2Vy
bmVsLXNpZ25lcjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMAfjVI+
GmUU/L+6PXDYkx5FP5zo1CZbLrJxiClRKIM4+zHJVFWZxutRZWgT/9q76HlISOLl
UKziGTJ/gpn6XLSfKKeqcXu852i7Hby4LWNLPUBVunxjtCZjcPPIhNQRDkgUXjTF
Rr1ZxEdnUxcrWswyN470XDn1Ss9lVQnkSWCtca1TeeYG83JgwrwMu1ZtxqvIW058
9RdJZhm6UorTTnfoDl0P12qMOreaoDkkJr4DXyYX0nRxL8HJ13bA2ftqJl/BLD+i
FSdv/mEV9nzWDIKG2PviU7C9K2VoZJUWUOT0gMZ0kMu3nTB+vw0U4AlsYLvUrXuu
ZUJ1lQyTU59NZwUCAwEAAaOBtjCBszBOBggrBgEFBQcBAQRCMEAwPgYIKwYBBQUH
MAKGMmh0dHBzOi8vZmVkb3JhcHJvamVjdC5vcmcvd2lraS9GZWF0dXJlcy9TZWN1
cmVCb290MB8GA1UdIwQYMBaAFLKAx65riE4PTSoNhyTCXq9sZcMmMBMGA1UdJQQM
MAoGCCsGAQUFBwMDMAwGA1UdEwEB/wQCMAAwHQYDVR0OBBYEFBvxkUE2zboKJXIZ
HMsHOF5AiN+6MA0GCSqGSIb3DQEBCwUAA4IBAQB/LqUgDAbULzsTQ1EsHl2PbGJF
0K19hCI1S22d+t6wh0JfQV0PWwq7Zw4m96Oq0dXAdpbZBBib7HCUdU2L+W81gRpL
mZqWk+k+Cn0YY31npVDRNxZM22ZNDxQBTlcY3L1jwpjRSrMYJmhdTG7NSYaE6Jhp
DXBvmd+l5eCrbjuSynca6vkweHcYJMkCVTy04ytBfVFJ8SzverWdJRVFGpnXlWOW
O+dVCB5e2zq+AZa8oBmuqXNOmWViL7W8xPGjqjZF+KiqCQnCg1jc+WXAQtuuTuhp
210NAWn1LI7c6KxOevK4FHvx39+VqMAOcW2FIqqVsVf3v3x7YJp5Dkqpfu1M
-----END CERTIFICATE-----
```

Picture 28 - Certificate section of signature

I then copied the certificate section to a PEM file for verification as shown in picture below (Pic. 29)

```
okore_joel@fedora:/$ cd home/okore_joel/
okore_joel@fedora:~$ gedit extracted_cert_kernel.pem
okore_joel@fedora:~$ cat extracted_cert_kernel.pem
-----BEGIN CERTIFICATE-----
MIIEeTCCA2GgAwIBAgIRAKIunjlKzU57q99PG5mswOcwDQYJKoZIhvcNAQELBQAw
gY0xCzAJBgNVBAYTAlVTMRYwFAYDVQQIEw1NYXNzYWNodXNldHRzMRIwEAYDVQQH
EwlDYW1icmlkZ2UxFjAUBgNVBAoTDVJlZCBIYXQsIEluYy4xJzAlBgNVBAsTHkZl
ZG9yYSBTZWN1cmUgQm9vdCBDQSAyMDIwMDcwOTERMA8GA1UEAxMIZmVkb3JhY2Ew
HhcNMjAwNzEzMTczMTI2WhcNMzcwMTE5MDMxNDA3WjCBqDELMAkGA1UEBhMCVVMx
FjAUBgNVBAgTDU1hc3NhY2h1c2V0dHMxEjAQBgNVBAcTCUNhbWJyaWRnZTEWMBQG
A1UEChMNUmVkIEhhdCwgSW5jLjEiMCAGA1UECxMZRmVkb3JhIFNlY3VyZSBCb290
IFNpZ25lcjEZMBcGA1UECxMQYmtlcm5lbDAxIGtlcm5lbDEWMBQGA1UEAxMNa2Vy
bmVsLXNpZ25lcjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMAfjVI+
GmUU/L+6PXDYkx5FP5zo1CZbLrJxiClRKIM4+zHJVFWZxutRZWgT/9q76HlISOLl
UKziGTJ/gpn6XLSfKKeqcXu852i7Hby4LWNLPUBVunxjtCZjcPPIhNQRDkgUXjTF
Rr1ZxEdnUxcrWswyN470XDn1Ss9lVQnkSWCtca1TeeYG83JgwrwMu1ZtxqvIW058
9RdJZhm6UorTTnfoDl0P12qMOreaoDkkJr4DXyYX0nRxL8HJ13bA2ftqJl/BLD+i
FSdv/mEV9nzWDIKG2PviU7C9K2VoZJUWUOT0gMZ0kMu3nTB+vw0U4AlsYLvUrXuu
ZUJ1lQyTU59NZwUCAwEAAaOBtjCBszBOBggrBgEFBQcBAQRCMEAwPgYIKwYBBQUH
MAKGMmh0dHBzOi8vZmVkb3JhcHJvamVjdC5vcmcvd2lraS9GZWF0dXJlcy9TZWN1
cmVCb290MB8GA1UdIwQYMBaAFLKAx65riE4PTSoNhyTCXq9sZcMmMBMGA1UdJQQM
MAoGCCsGAQUFBwMDMAwGA1UdEwEB/wQCMAAwHQYDVR0OBBYEFBvxkUE2zboKJXIZ
HMsHOF5AiN+6MA0GCSqGSIb3DQEBCwUAA4IBAQB/LqUgDAbULzsTQ1EsHl2PbGJF
0K19hCI1S22d+t6wh0JfQV0PWwq7Zw4m96Oq0dXAdpbZBBib7HCUdU2L+W81gRpL
mZqWk+k+Cn0YY31npVDRNxZM22ZNDxQBTlcY3L1jwpjRSrMYJmhdTG7NSYaE6Jhp
DXBvmd+l5eCrbjuSynca6vkweHcYJMkCVTy04ytBfVFJ8SzverWdJRVFGpnXlWOW
O+dVCB5e2zq+AZa8oBmuqXNOmWViL7W8xPGjqjZF+KiqCQnCg1jc+WXAQtuuTuhp
210NAWn1LI7c6KxOevK4FHvx39+VqMAOcW2FIqqVsVf3v3x7YJp5Dkqpfu1M
-----END CERTIFICATE-----
```

Picture 29 - Certificate copied to PEM file for kernel verification

Finally, the kernel was verified to be indeed booted from the Kernel-Signer signature certificate as shown in pic. 30.

```
okore_joel@fedora:~$ sudo sbverify --cert extracted_cert_kernel.pem /boot/vmlinuz-6.8.5-301.fc40.x86_64
Signature verification OK
okore_joel@fedora:~$
```

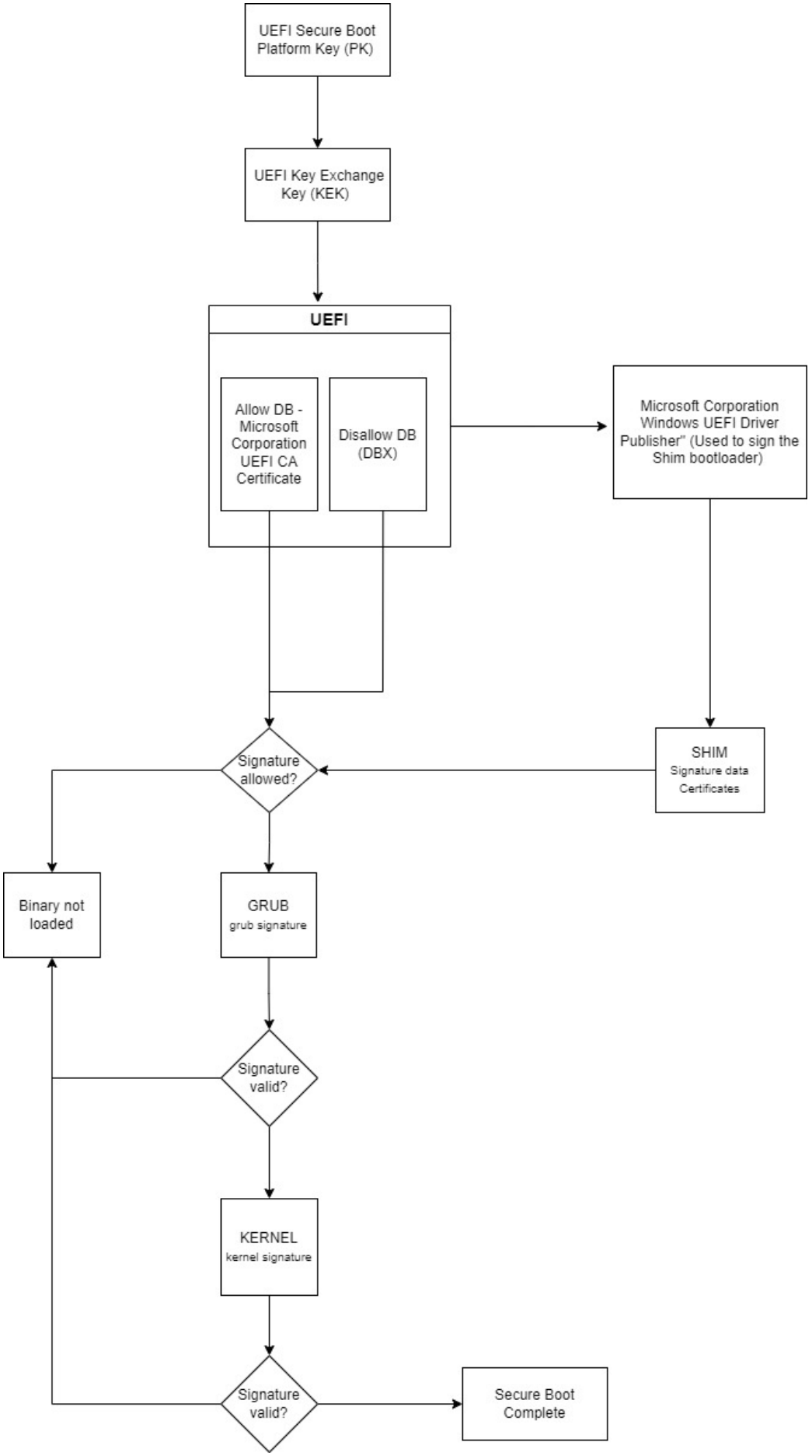
Picture 30 - Kernel Verified Successfully

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.

    The trusted certificate is not kept by GRUB in the file system or its binary. UEFI Secure Boot is where GRUB instead obtains its trusted certificates. GRUB verifies signatures using the certificates in the UEFI Secure Boot database (UEFI's DB variable). Instead of being stored on a disk or in GRUB, these certificates are loaded into the UEFI firmware.

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.

    The UEFI chain of trust is illustrated in the diagram below (Pic. 31)


UEFI Secure Boot Platform Key (PK)
UEFI Key Exchange Key (KEK)
UEFI
Allow DB - Microsoft Corporation UEFI CA Certificate
Disallow DB (DBX)
Microsoft Corporation Windows UEFI Driver Publisher" (Used to sign the Shim bootloader)
Signature allowed?
SHIM
Signature data
Certificates
Binary not loaded
GRUB
grub signature
Signature valid?
KERNEL
kernel signature
Signature valid?
Secure Boot Complete

Picture 30 - UEFI Chain of Trust