DESDE LAS PROFUNDIDADES DEL KERNEL: CÓMO CREAR UN ROOTKIT INVISIBLE EN WINDOWS

Descifrando el desarrollo de un Rootkit para Windows 11

[in/vazquez-vazquez-alejandro]

ViCONgal 2025, Galicia

VICON

WHOAMI

- **FRIKI** (Fanático de Revolucionar Internamente Kernels e Inicios del sistema)

- Pastor de ovejas desde los 8 años
- Me gusta el pulpo, de ahí los Rootkits
- Docente en Máster de Análisis de Malware

[in/vazquez-vazquez-alejandro]

TABLE OF CONTENTS

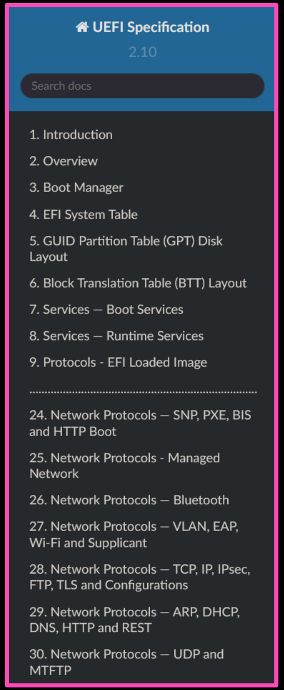
Rootkit Development

/RootedCON

UEFI BOOTKIT DEVELOPMENT

UEFI BOOTKIT DEVELOPMENT

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



UEFI Specification
2.10

Search docs

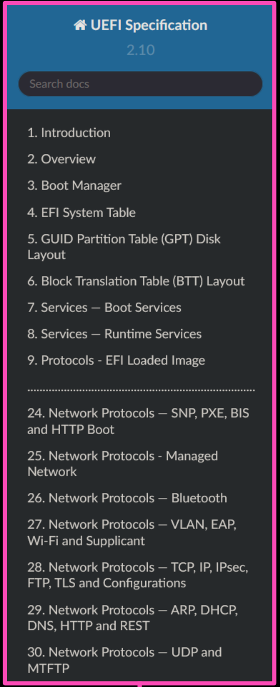
1. Introduction
2. Overview
3. Boot Manager
4. EFI System Table
5. GUID Partition Table (GPT) Disk Layout
6. Block Translation Table (BTT) Layout
7. Services — Boot Services
8. Services — Runtime Services
9. Protocols - EFI Loaded Image

....

24. Network Protocols — SNP, PXE, BIS and HTTP Boot
25. Network Protocols - Managed Network
26. Network Protocols — Bluetooth
27. Network Protocols — VLAN, EAP, Wi-Fi and Supplicant
28. Network Protocols — TCP, IP, IPsec, FTP, TLS and Configurations
29. Network Protocols — ARP, DHCP, DNS, HTTP and REST
30. Network Protocols — UDP and MTFTP

UEFI BOOTKIT DEVELOPMENT

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



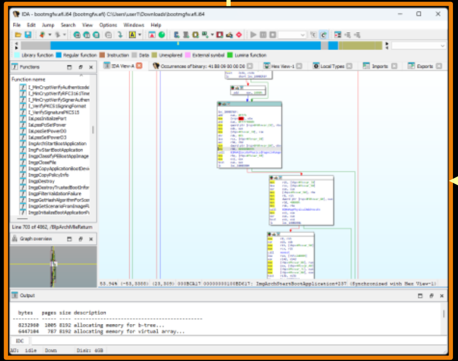
UEFI Specification

2.10

Search docs

1. Introduction
2. Overview
3. Boot Manager
4. EFI System Table
5. GUID Partition Table (GPT) Disk Layout
6. Block Translation Table (BTT) Layout
7. Services — Boot Services
8. Services — Runtime Services
9. Protocols - EFI Loaded Image

24. Network Protocols — SNP, PXE, BIS and HTTP Boot
25. Network Protocols - Managed Network
26. Network Protocols — Bluetooth
27. Network Protocols — VLAN, EAP, Wi-Fi and Supplicant
28. Network Protocols — TCP, IP, IPsec, FTP, TLS and Configurations
29. Network Protocols — ARP, DHCP, DNS, HTTP and REST
30. Network Protocols — UDP and MTFTP



IDA - bootmgfw.efi.i64 (bootmgfw.efi) C:\Users\user1\Downloads\bootmgfw.efi.i64

Function name
- I_MinCryptVerifyAuthenticode
- I_MinCryptVerifyRFC3161Time
- I_MinCryptVerifySignerAuthen
- I_VerifyPKCS1SigningFormat
- I_VerifySignaturePKCS15
- IalpssInitializePort
- IalpssPciSetPower
- IalpssSetPowerD0
- IalpssSetPowerD3
- ImgArchStartBootApplication
- ImgFwStartBootApplication
- ImgpClassifyPEBootAppImage
- ImgpCloseFile
- ImgpCopyApplicationBootDevi
- ImgpCopyPolicyInfo
- ImgpDestroy
- ImgpDestroyTrustedBootInfor
- ImgpFilterValidationFailure
- ImgpGetHashAlgorithmForScen
- ImgpGetScenarioFromImageFi
- ImgpInitializeBootApplicationP

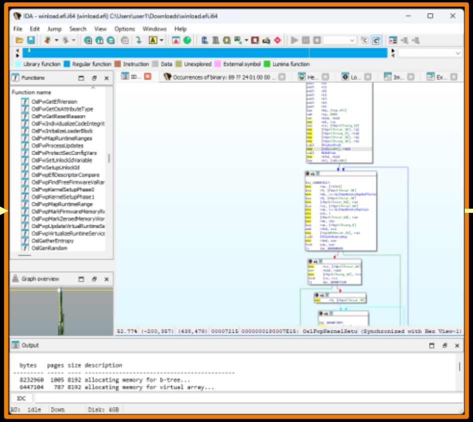
Line 703 of 4862, /BlpArchVileReturn

```
53.94% (-53,3388) (23,309) 000BCA17 0000000010BD617: ImgArchStartBootApplication+237 (Synchronized with Hex View-1)
```

```
 bytes   pages size description
--------- ----- ---- --------------------------------------------
 8232960  1005 8192 allocating memory for b-tree...
 6447104   787 8192 allocating memory for virtual array...
```

```
AU:  idle   Down       Disk: 4GB
```

Windows Boot Manager
\EFI\Microsoft\Boot\
bootmgfw.efi



IDA - winload.efi.i64 (winload.efi) C:\Users\user1\Downloads\winload.efi.i64

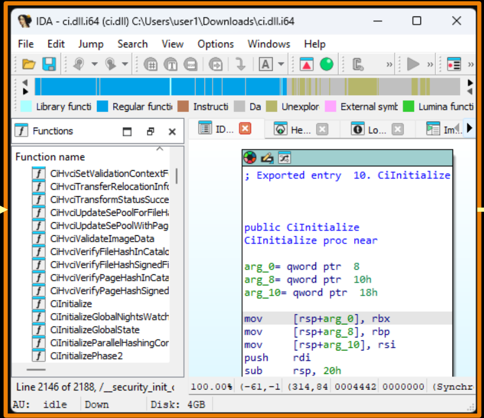
Function name
- OslFwGetEfiVersion
- OslFwGetOsAttributeType
- OslFwGetResetReason
- OslFwIndividualizeCodeIntegrit
- OslFwInitializeLoaderBlock
- OslFwMapRuntimeRanges
- OslFwProcessUpdates
- OslFwProtectSecConfigVars
- OslFwSetUnlockIdVariable
- OslFwSetupUnlockId
- OslFwpEfiDescriptorCompare
- OslFwpFindFreeFirmwareVaRar
- OslFwpKernelSetupPhase0
- OslFwpKernelSetupPhase1
- OslFwpMapRuntimeRange
- OslFwpMarkFirmwareMemoryRu
- OslFwpMarkZeroedMemoryWor
- OslFwpUpdateVirtualRuntimeSe
- OslFwpVirtualizeRuntimeServic
- OslGatherEntropy
- OslGenRandom

```
52.77% (-200,357) (430,470) 00007215 0000000180007E15: OslFwpKernelSetu (Synchronized with Hex View-1)
```

```
 bytes   pages size description
--------- ----- ---- --------------------------------------------
 8232960  1005 8192 allocating memory for b-tree...
 6447104   787 8192 allocating memory for virtual array...
```

```
AU:  idle   Down       Disk: 4GB
```

Windows OS Loader
%SystemRoot%\system32\
winload.efi



IDA - ci.dll.i64 (ci.dll) C:\Users\user1\Downloads\ci.dll.i64

Function name
- CiHvciSetValidationContextF
- CiHvciTransferRelocationInfo
- CiHvciTransformStatusSucce
- CiHvciUpdateSePoolForFileH
- CiHvciUpdateSePoolWithPag
- CiHvciValidateImageData
- CiHvciVerifyFileHashInCatalo
- CiHvciVerifyFileHashSignedFi
- CiHvciVerifyPageHashInCata
- CiHvciVerifyPageHashSigned
- CiInitialize
- CiInitializeGlobalNightsWatch
- CiInitializeGlobalState
- CiInitializeParallelHashingCor
- CiInitializePhase2

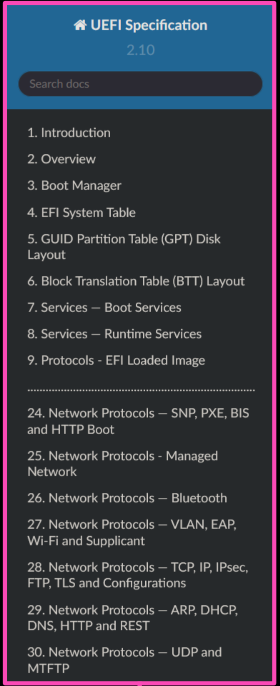
```
; Exported entry  10. CiInitialize



public CiInitialize
CiInitialize proc near

arg_0= qword ptr  8
arg_8= qword ptr  10h
arg_10= qword ptr  18h

mov     [rsp+arg_0], rbx
mov     [rsp+arg_8], rbp
mov     [rsp+arg_10], rsi
push    rdi
sub     rsp, 20h
```

```
Line 2146 of 2188, /__security_init_c   100.00% (-61,-1 (314,84 0004442 0000000 (Synchr
AU:  idle   Down       Disk: 4GB
```

Windows NT OS Kernel
%SystemRoot%\system32\
ntoskrnl.exe

UEFI BOOTKIT DEVELOPMENT

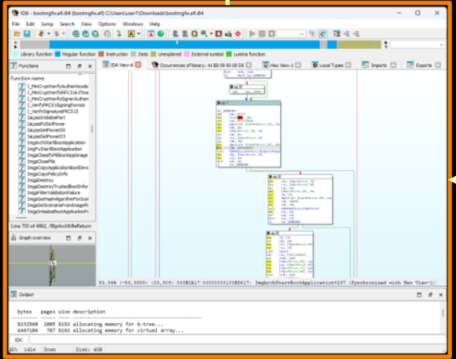
Boot order
Boot0001 = /EFI/Microsoft/boot/bootmgfw.efi
Boot0002 = /EFI/Ubuntu/shimx64.efi
Boot000x = /EFI/Vendor/bootx64.efi



UEFI Specification
2.10
Search docs

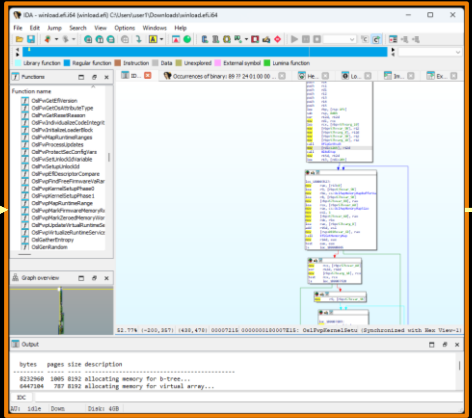
1. Introduction
2. Overview
3. Boot Manager
4. EFI System Table
5. GUID Partition Table (GPT) Disk Layout
6. Block Translation Table (BTT) Layout
7. Services — Boot Services
8. Services — Runtime Services
9. Protocols - EFI Loaded Image

24. Network Protocols — SNP, PXE, BIS and HTTP Boot
25. Network Protocols - Managed Network
26. Network Protocols — Bluetooth
27. Network Protocols — VLAN, EAP, Wi-Fi and Supplicant
28. Network Protocols — TCP, IP, IPsec, FTP, TLS and Configurations
29. Network Protocols — ARP, DHCP, DNS, HTTP and REST
30. Network Protocols — UDP and MTFTP



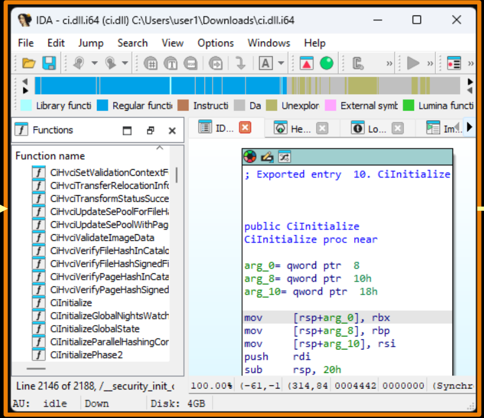
```
bytes   pages size description
8232960  1005 8192 allocating memory for b-tree...
6447104   787 8192 allocating memory for virtual array...
```

Windows Boot Manager
\EFI\Microsoft\Boot\
bootmgfw.efi



```
bytes   pages size description
8232960  1005 8192 allocating memory for b-tree...
6447104   787 8192 allocating memory for virtual array...
```

Windows OS Loader
%SystemRoot%\system32\
winload.efi

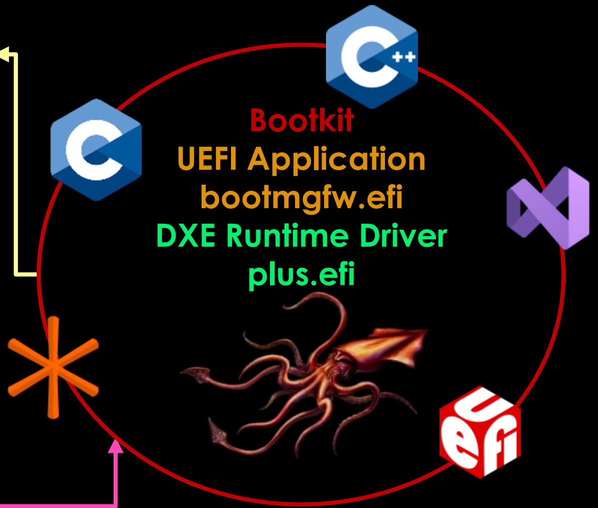


CiHvciSetValidationContextF
CiHvciTransferRelocationInf
CiHvciTransformStatusSucce
CiHvciUpdateSePoolForFileH
CiHvciUpdateSePoolWithPag
CiHvciValidateImageData
CiHvciVerifyFileHashInCatalo
CiHvciVerifyFileHashSignedFi
CiHvciVerifyPageHashInCata
CiHvciVerifyPageHashSigned
CiInitialize
CiInitializeGlobalNightsWatch
CiInitializeGlobalState
CiInitializeParallelHashingCor
CiInitializePhase2

```
; Exported entry  10. CiInitialize


public CiInitialize
CiInitialize proc near

arg_0= qword ptr  8
arg_8= qword ptr  10h
arg_10= qword ptr  18h

mov     [rsp+arg_0], rbx
mov     [rsp+arg_8], rbp
mov     [rsp+arg_10], rsi
push    rdi
sub     rsp, 20h
```

Line 2146 of 2188, /__security_init_c   100.00% (-61,-1 (314,84 0004442 0000000 (Synchr
AU: idle   Down   Disk: 4GB

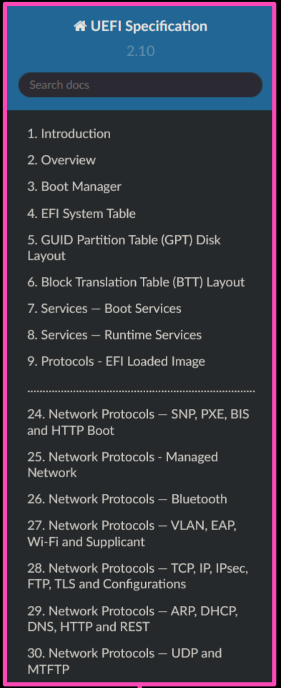
Windows NT OS Kernel
%SystemRoot%\system32\
ntoskrnl.exe



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

UEFI BOOTKIT DEVELOPMENT

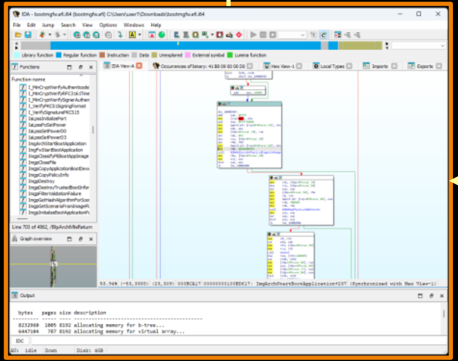
Boot order
Boot0001 = /EFI/Microsoft/boot/bootmgfw.efi
Boot0002 = /EFI/Ubuntu/shimx64.efi
Boot000x = /EFI/Vendor/bootx64.efi



UEFI Specification
2.10
Search docs

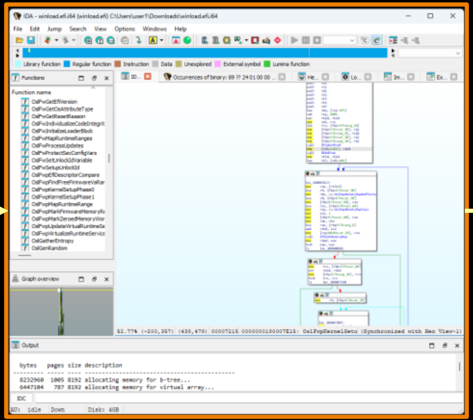
1. Introduction
2. Overview
3. Boot Manager
4. EFI System Table
5. GUID Partition Table (GPT) Disk Layout
6. Block Translation Table (BTT) Layout
7. Services — Boot Services
8. Services — Runtime Services
9. Protocols - EFI Loaded Image

24. Network Protocols — SNP, PXE, BIS and HTTP Boot
25. Network Protocols - Managed Network
26. Network Protocols — Bluetooth
27. Network Protocols — VLAN, EAP, Wi-Fi and Supplicant
28. Network Protocols — TCP, IP, IPsec, FTP, TLS and Configurations
29. Network Protocols — ARP, DHCP, DNS, HTTP and REST
30. Network Protocols — UDP and MTFTP



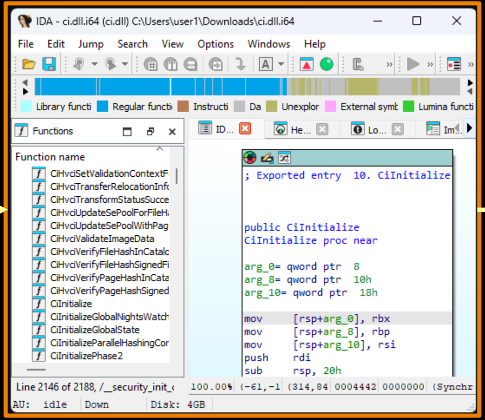
```
IDA - bootmgfw.efi.i64 (bootmgfw.efi) C:\Users\user1\Downloads\bootmgfw.efi.i64

bytes    pages size description
8232960  1005 8192 allocating memory for b-tree...
6447104   787 8192 allocating memory for virtual array...
```

Windows Boot Manager
\EFI\Microsoft\Boot\
bootmgfw.efi



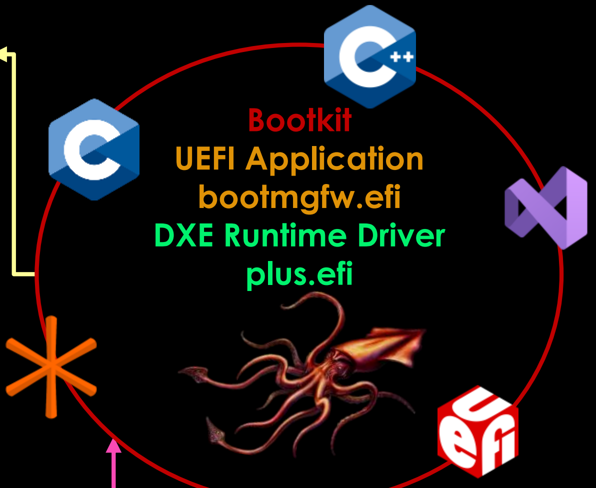
```
IDA - winload.efi.i64 (winload.efi) C:\Users\user1\Downloads\winload.efi.i64

bytes    pages size description
8232960  1005 8192 allocating memory for b-tree...
6447104   787 8192 allocating memory for virtual array...
```

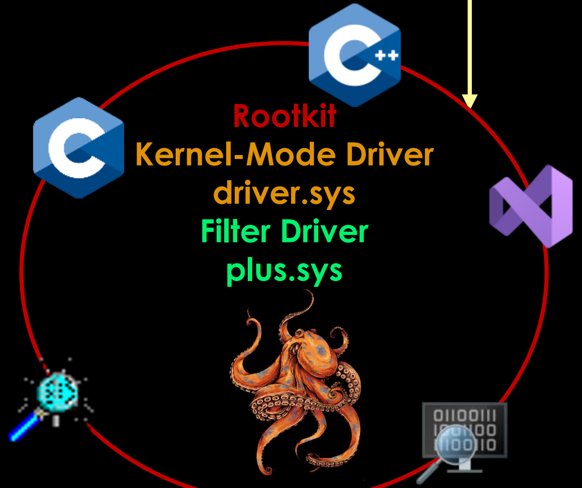
Windows OS Loader
%SystemRoot%\system32\
winload.efi



```
IDA - ci.dll.i64 (ci.dll) C:\Users\user1\Downloads\ci.dll.i64
File  Edit  Jump  Search  View  Options  Windows  Help

Functions
Function name
CiHvciSetValidationContextF
CiHvciTransferRelocationInf
CiHvciTransformStatusSucce
CiHvciUpdateSePoolForFileH
CiHvciUpdateSePoolWithPag
CiHvciValidateImageData
CiHvciVerifyFileHashInCatalo
CiHvciVerifyFileHashSignedFi
CiHvciVerifyPageHashInCata
CiHvciVerifyPageHashSigned
CiInitialize
CiInitializeGlobalNightsWatch
CiInitializeGlobalState
CiInitializeParallelHashingCor
CiInitializePhase2

; Exported entry  10. CiInitialize

public CiInitialize
CiInitialize proc near

arg_0= qword ptr  8
arg_8= qword ptr  10h
arg_10= qword ptr  18h

mov     [rsp+arg_0], rbx
mov     [rsp+arg_8], rbp
mov     [rsp+arg_10], rsi
push    rdi
sub     rsp, 20h

Line 2146 of 2188, /__security_init_c   100.00% (-61,-1 (314,84 0004442 0000000 (Synchr
AU:  idle   Down   Disk: 4GB
```

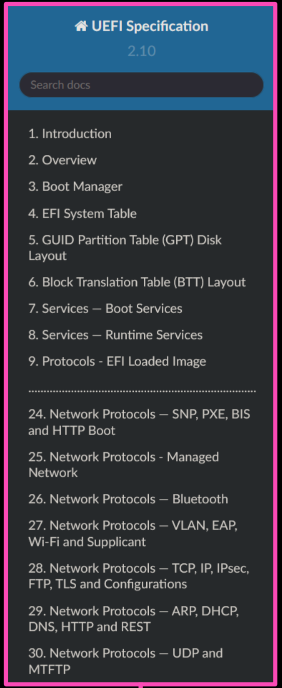
Windows NT OS Kernel
%SystemRoot%\system32\
ntoskrnl.exe





UEFI BOOTKIT DEVELOPMENT

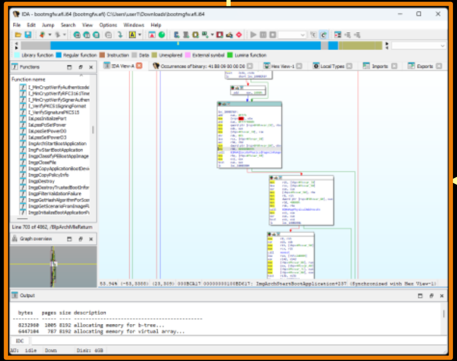
Boot order
Boot0001 = /EFI/Microsoft/boot/bootmgfw.efi
Boot0002 = /EFI/Ubuntu/shimx64.efi
Boot000x = /EFI/Vendor/bootx64.efi



🏠 UEFI Specification
2.10
Search docs

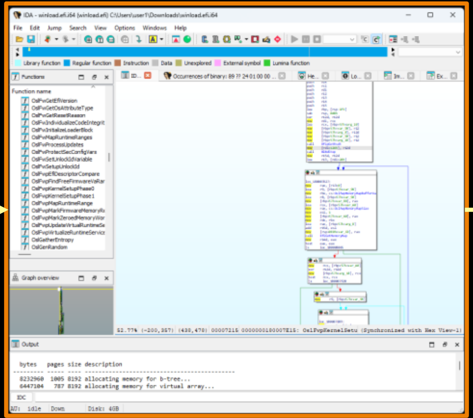
1. Introduction
2. Overview
3. Boot Manager
4. EFI System Table
5. GUID Partition Table (GPT) Disk Layout
6. Block Translation Table (BTT) Layout
7. Services — Boot Services
8. Services — Runtime Services
9. Protocols - EFI Loaded Image

24. Network Protocols — SNP, PXE, BIS and HTTP Boot
25. Network Protocols - Managed Network
26. Network Protocols — Bluetooth
27. Network Protocols — VLAN, EAP, Wi-Fi and Supplicant
28. Network Protocols — TCP, IP, IPsec, FTP, TLS and Configurations
29. Network Protocols — ARP, DHCP, DNS, HTTP and REST
30. Network Protocols — UDP and MTFTP



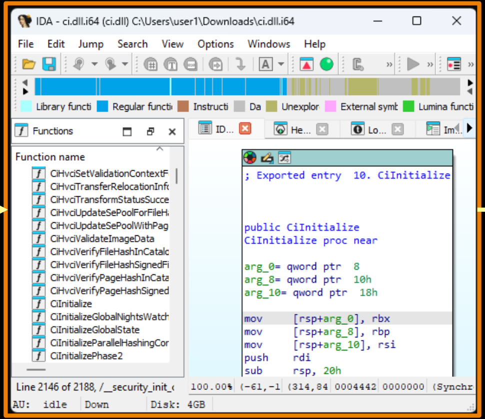
```
bytes    pages size description
8232960  1005  8192 allocating memory for b-tree...
6447104  787   8192 allocating memory for virtual array...
```

Windows Boot Manager
\EFI\Microsoft\Boot\
bootmgfw.efi



```
bytes    pages size description
8232960  1005  8192 allocating memory for b-tree...
6447104  787   8192 allocating memory for virtual array...
```

Windows OS Loader
%SystemRoot%\system32\
winload.efi

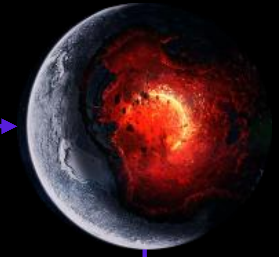

IDA - ci.dll.i64 (ci.dll) C:\Users\user1\Downloads\ci.dll.i64
File Edit Jump Search View Options Windows Help
Library functi
Regular functi
Instructi
Da
Unexplor
External symt
Lumina functi
Functions
Function name

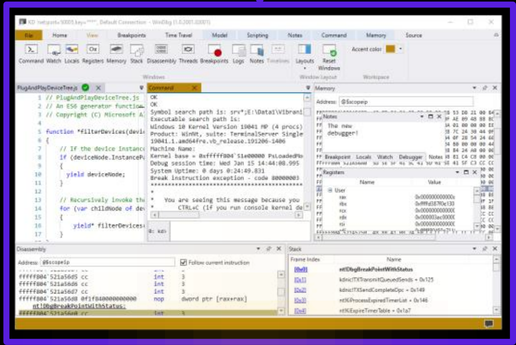
```
CiHvciSetValidationContextF
CiHvciTransferRelocationInf
CiHvciTransformStatusSucce
CiHvciUpdateSePoolForFileH
CiHvciUpdateSePoolWithPag
CiHvciValidateImageData
CiHvciVerifyFileHashInCatalc
CiHvciVerifyFileHashSignedFi
CiHvciVerifyPageHashInCata
CiHvciVerifyPageHashSigned
CiInitialize
CiInitializeGlobalNightsWatch
CiInitializeGlobalState
CiInitializeParallelHashingCor
CiInitializePhase2
```

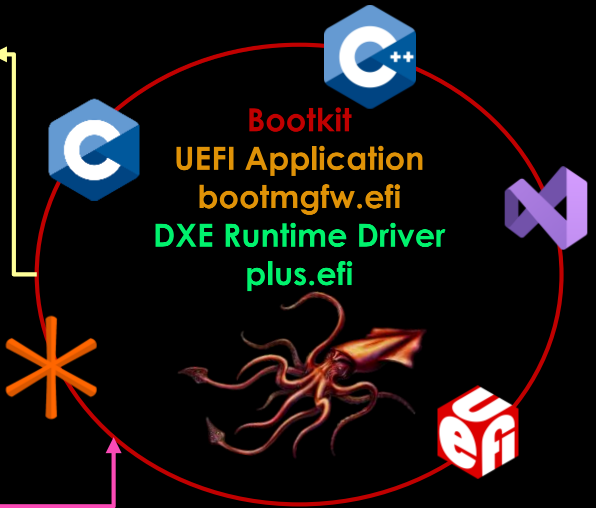
```
; Exported entry  10. CiInitialize

public CiInitialize
CiInitialize proc near

arg_0= qword ptr  8
arg_8= qword ptr  10h
arg_10= qword ptr  18h

mov     [rsp+arg_0], rbx
mov     [rsp+arg_8], rbp
mov     [rsp+arg_10], rsi
push    rdi
sub     rsp, 20h
```

```
Line 2146 of 2188, /__security_init_c  100.00% (-61,-1 (314,84 0004442 0000000 (Synchr
AU:  idle    Down      Disk: 4GB
```

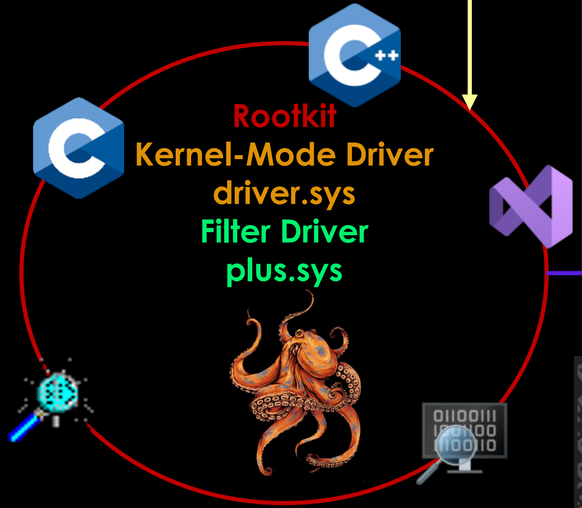
Windows NT OS Kernel
%SystemRoot%\system32\
ntoskrnl.exe







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

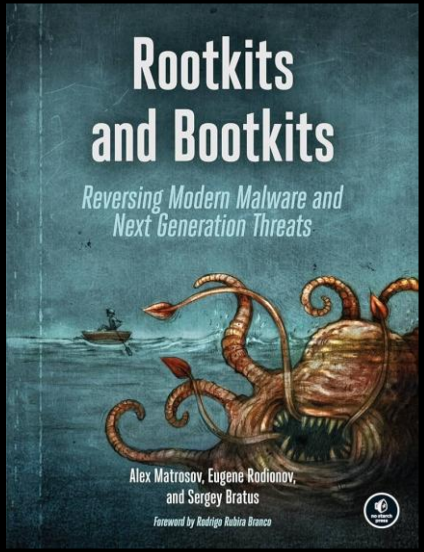


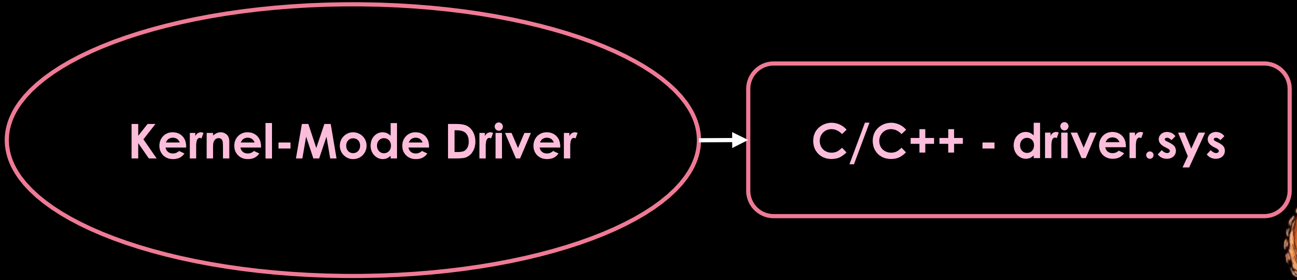
Rootkit
Kernel-Mode Driver
driver.sys
Filter Driver
plus.sys

ROOTKIT

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


Rootkits and Bootkits
Reversing Modern Malware and Next Generation Threats
Alex Matrosov, Eugene Rodionov, and Sergey Bratus
Foreword by Rodrigo Rubira Branco



SECURITY MECHANISMS

[*Anti-Rootkit Installation*]

- Driver Signature Enforcement (DSE)

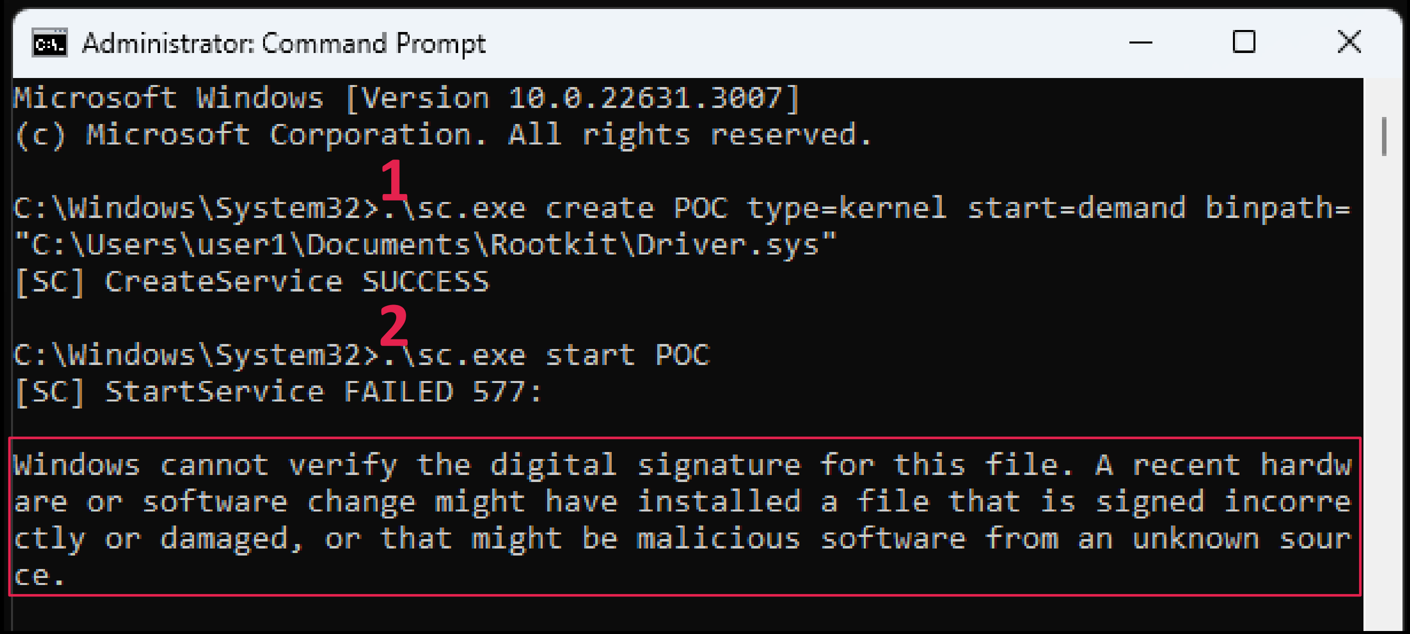
  Windows won't run drivers not certified by Microsoft

SECURITY MECHANISMS

[Anti-Rootkit Installation]

- Driver Signature Enforcement (DSE)

  Windows won't run drivers not certified by Microsoft



Administrator: Command Prompt

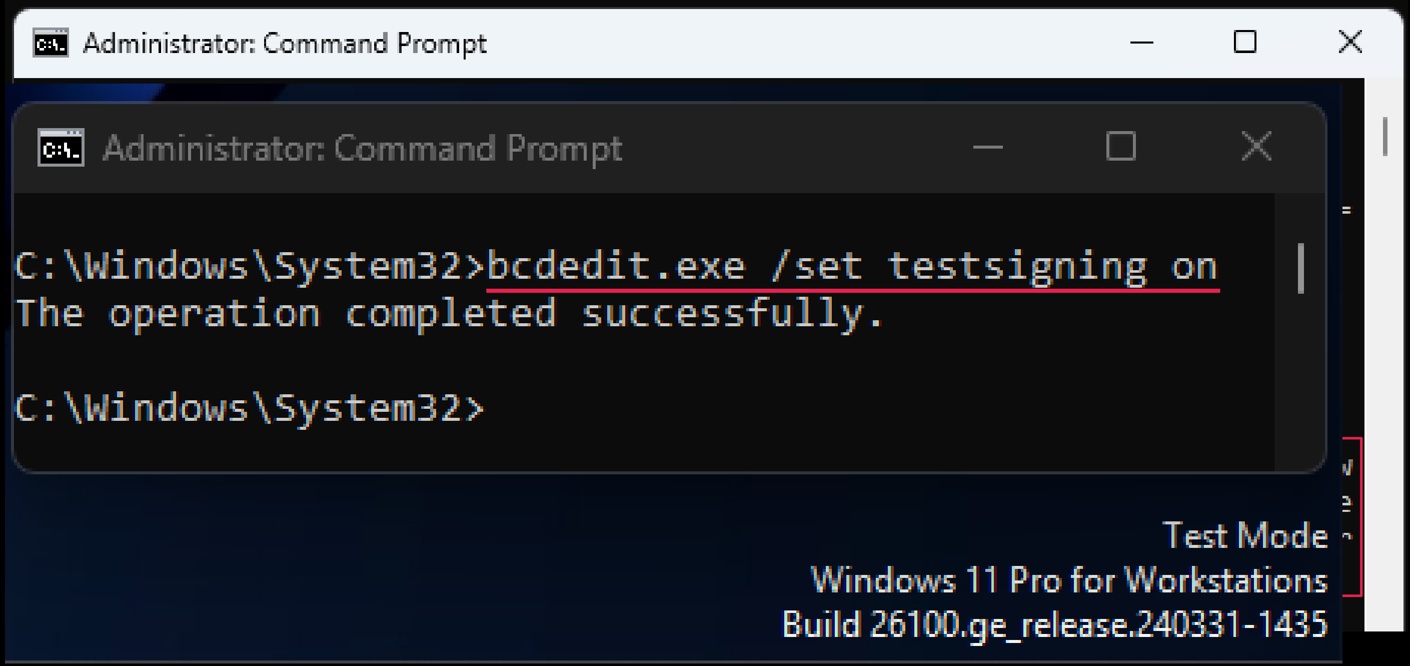
```
Microsoft Windows [Version 10.0.22631.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>.\sc.exe create POC type=kernel start=demand binpath=
"C:\Users\user1\Documents\Rootkit\Driver.sys"
[SC] CreateService SUCCESS

C:\Windows\System32>.\sc.exe start POC
[SC] StartService FAILED 577:

Windows cannot verify the digital signature for this file. A recent hardw
are or software change might have installed a file that is signed incorre
ctly or damaged, or that might be malicious software from an unknown sour
ce.
```

SECURITY MECHANISMS

[Anti-Rootkit Installation]

- Driver Signature Enforcement (DSE)

  Windows won't run drivers not certified by Microsoft

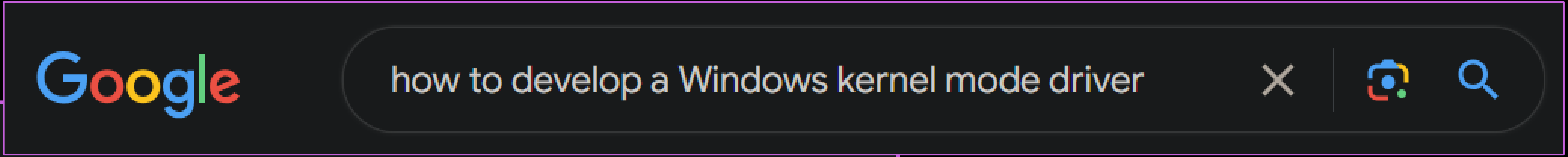


Administrator: Command Prompt

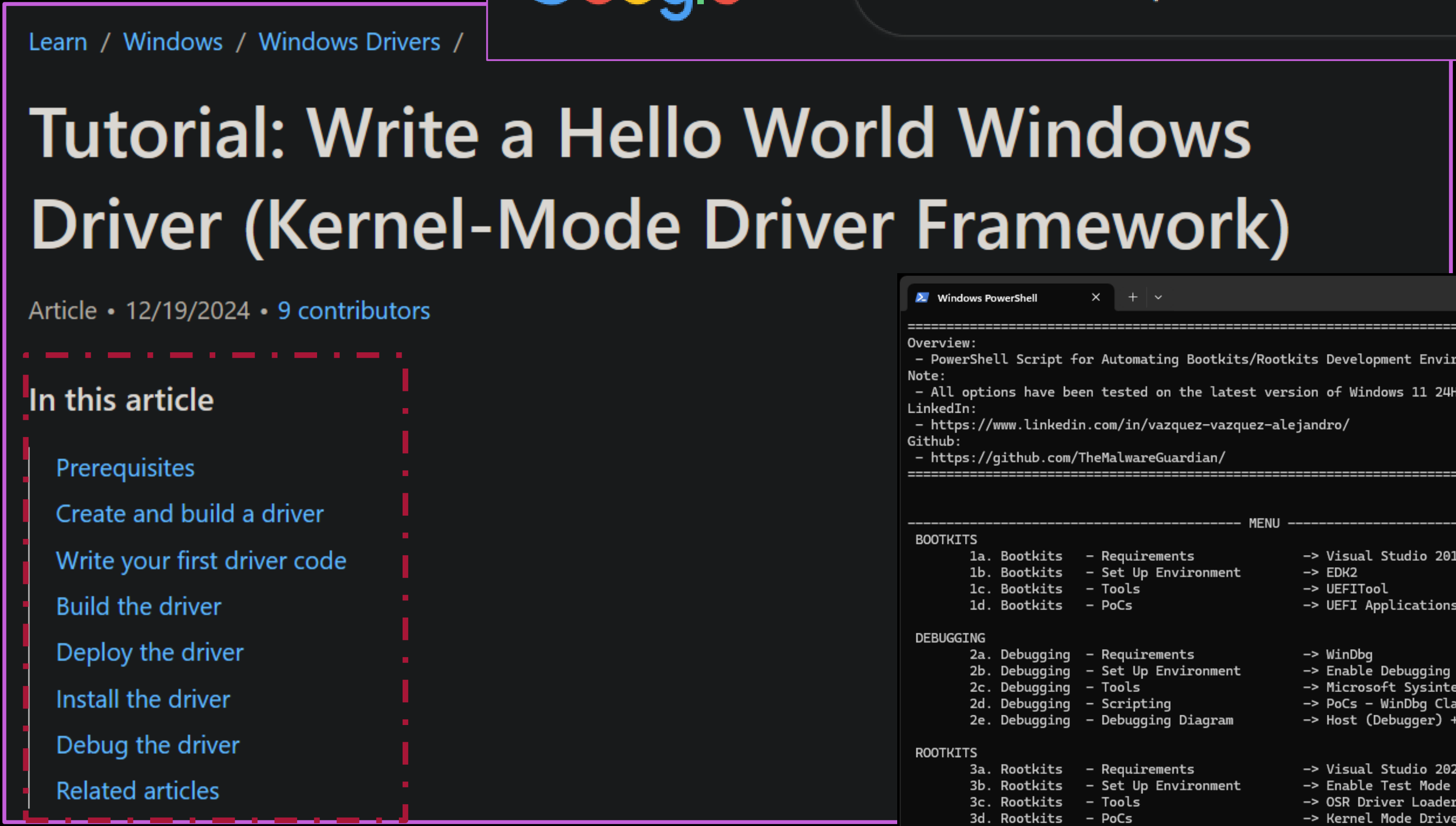
```
C:\Windows\System32>bcdedit.exe /set testsigning on
The operation completed successfully.

C:\Windows\System32>
```

Test Mode
Windows 11 Pro for Workstations
Build 26100.ge_release.240331-1435

DEVELOPMENT ENVIRONMENT



how to develop a Windows kernel mode driver



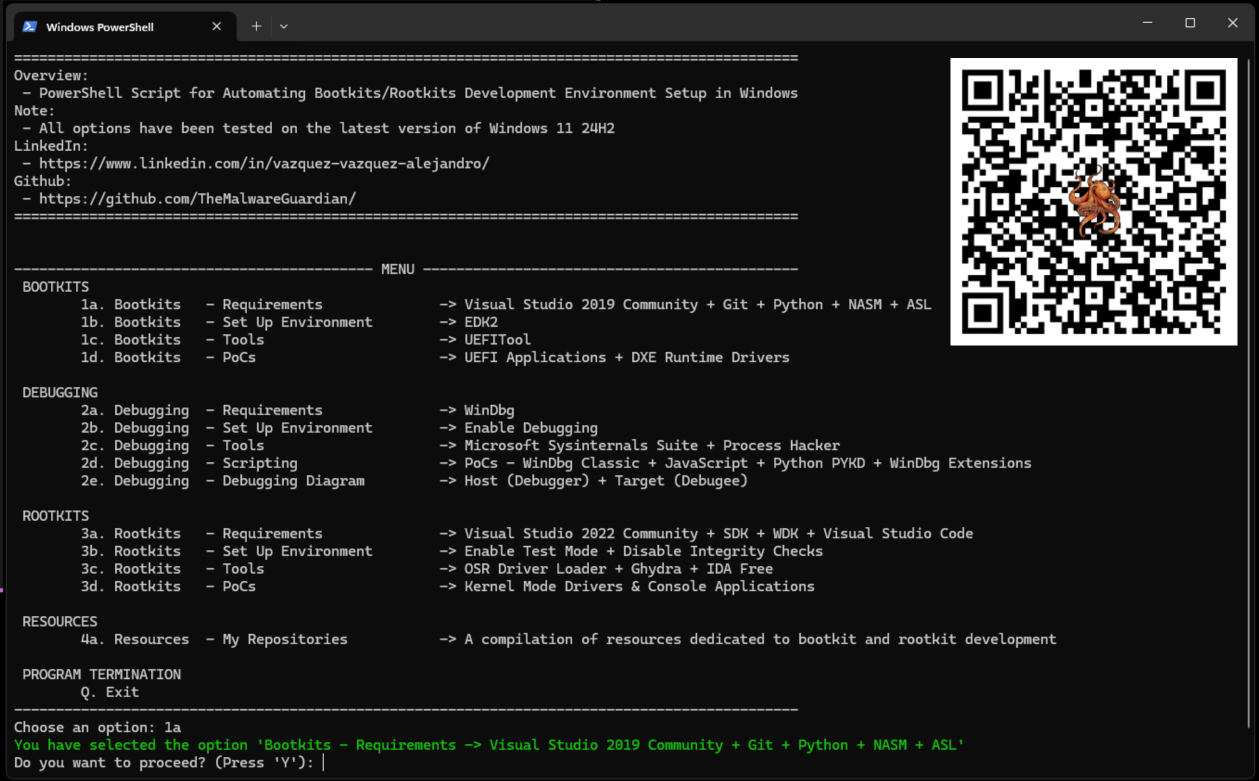
Learn / Windows / Windows Drivers /

Tutorial: Write a Hello World Windows Driver (Kernel-Mode Driver Framework)

Article • 12/19/2024 • 9 contributors

In this article

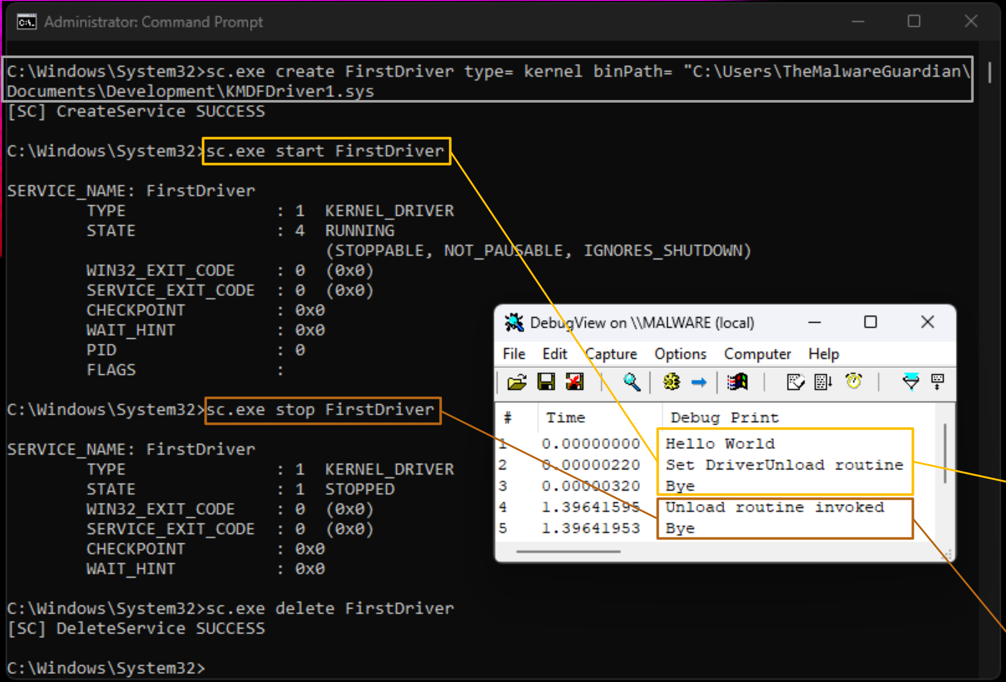
- Prerequisites
- Create and build a driver
- Write your first driver code
- Build the driver
- Deploy the driver
- Install the driver
- Debug the driver
- Related articles



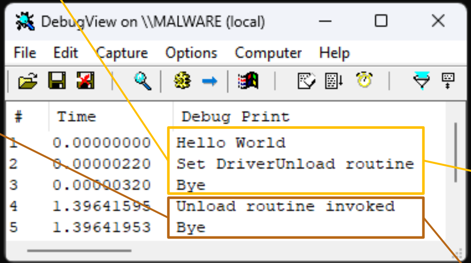
Windows PowerShell

```
==============================================================================================
Overview:
 - PowerShell Script for Automating Bootkits/Rootkits Development Environment Setup in Windows
Note:
 - All options have been tested on the latest version of Windows 11 24H2
LinkedIn:
 - https://www.linkedin.com/in/vazquez-vazquez-alejandro/
Github:
 - https://github.com/TheMalwareGuardian/
==============================================================================================


------------------------------------------- MENU ---------------------------------------------
 BOOTKITS
        1a. Bootkits   - Requirements           -> Visual Studio 2019 Community + Git + Python + NASM + ASL
        1b. Bootkits   - Set Up Environment     -> EDK2
        1c. Bootkits   - Tools                  -> UEFITool
        1d. Bootkits   - PoCs                   -> UEFI Applications + DXE Runtime Drivers

 DEBUGGING
        2a. Debugging  - Requirements           -> WinDbg
        2b. Debugging  - Set Up Environment     -> Enable Debugging
        2c. Debugging  - Tools                  -> Microsoft Sysinternals Suite + Process Hacker
        2d. Debugging  - Scripting              -> PoCs - WinDbg Classic + JavaScript + Python PYKD + WinDbg Extensions
        2e. Debugging  - Debugging Diagram      -> Host (Debugger) + Target (Debugee)

 ROOTKITS
        3a. Rootkits   - Requirements           -> Visual Studio 2022 Community + SDK + WDK + Visual Studio Code
        3b. Rootkits   - Set Up Environment     -> Enable Test Mode + Disable Integrity Checks
        3c. Rootkits   - Tools                  -> OSR Driver Loader + Ghydra + IDA Free
        3d. Rootkits   - PoCs                   -> Kernel Mode Drivers & Console Applications

 RESOURCES
        4a. Resources  - My Repositories        -> A compilation of resources dedicated to bootkit and rootkit development

 PROGRAM TERMINATION
        Q. Exit
----------------------------------------------------------------------------------------------
Choose an option: 1a
You have selected the option 'Bootkits - Requirements -> Visual Studio 2019 Community + Git + Python + NASM + ASL'
Do you want to proceed? (Press 'Y'):
```

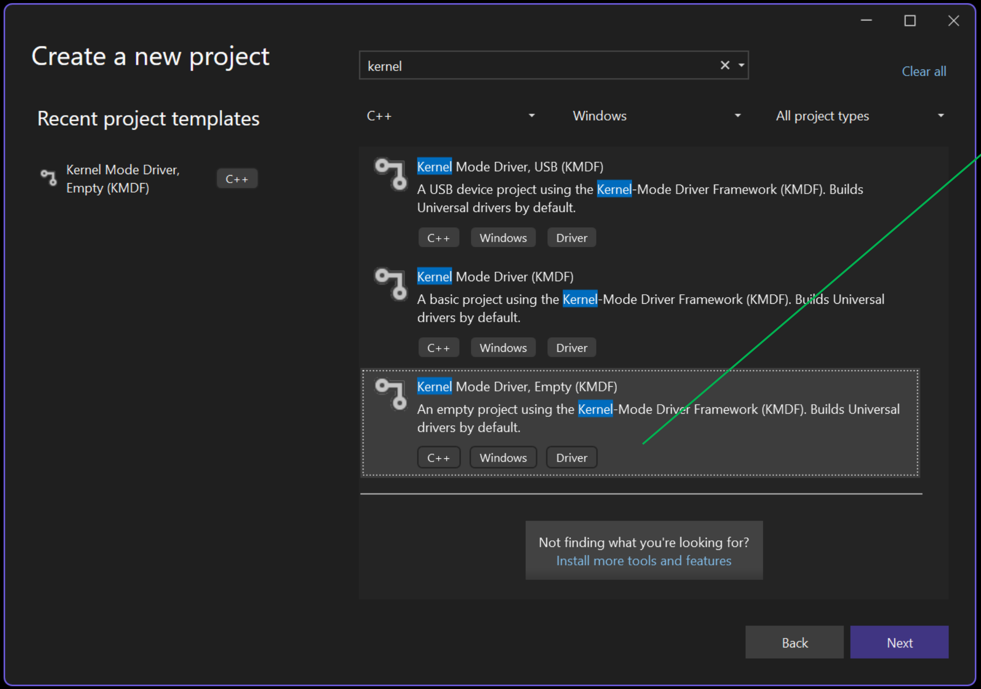
KERNEL MODE DRIVER



```
C:\Windows\System32>sc.exe create FirstDriver type= kernel binPath= "C:\Users\TheMalwareGuardian\
Documents\Development\KMDFDriver1.sys
[SC] CreateService SUCCESS

C:\Windows\System32>sc.exe start FirstDriver

SERVICE_NAME: FirstDriver
        TYPE               : 1  KERNEL_DRIVER
        STATE              : 4  RUNNING
                                (STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
        PID                : 0
        FLAGS              :

C:\Windows\System32>sc.exe stop FirstDriver

SERVICE_NAME: FirstDriver
        TYPE               : 1  KERNEL_DRIVER
        STATE              : 1  STOPPED
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0

C:\Windows\System32>sc.exe delete FirstDriver
[SC] DeleteService SUCCESS

C:\Windows\System32>
```



```
#   Time          Debug Print
1   0.00000000    Hello World
2   0.00000220    Set DriverUnload routine
3   0.00000320    Bye
4   1.39641595    Unload routine invoked
5   1.39641953    Bye
```



Kernel Mode Driver, Empty (KMDF) C++

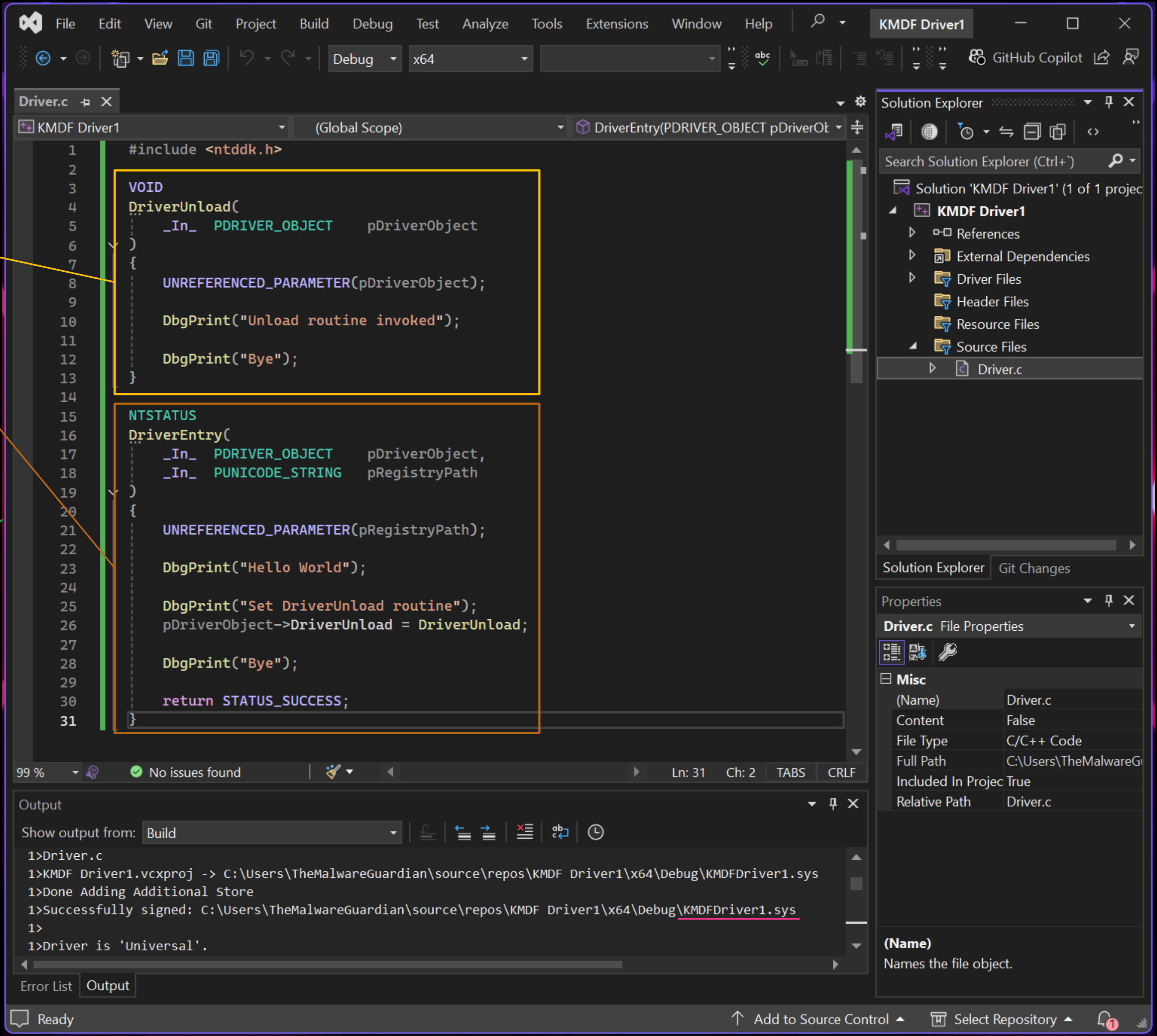
kernel

Kernel Mode Driver, USB (KMDF)
A USB device project using the Kernel-Mode Driver Framework (KMDF). Builds Universal drivers by default.
C++ Windows Driver

Kernel Mode Driver (KMDF)
A basic project using the Kernel-Mode Driver Framework (KMDF). Builds Universal drivers by default.
C++ Windows Driver

Kernel Mode Driver, Empty (KMDF)
An empty project using the Kernel-Mode Driver Framework (KMDF). Builds Universal drivers by default.
C++ Windows Driver

Not finding what you're looking for?
Install more tools and features



```c
#include <ntddk.h>

VOID
DriverUnload(
    _In_  PDRIVER_OBJECT    pDriverObject
)
{
    UNREFERENCED_PARAMETER(pDriverObject);

    DbgPrint("Unload routine invoked");

    DbgPrint("Bye");
}

NTSTATUS
DriverEntry(
    _In_  PDRIVER_OBJECT    pDriverObject,
    _In_  PUNICODE_STRING   pRegistryPath
)
{
    UNREFERENCED_PARAMETER(pRegistryPath);

    DbgPrint("Hello World");

    DbgPrint("Set DriverUnload routine");
    pDriverObject->DriverUnload = DriverUnload;

    DbgPrint("Bye");

    return STATUS_SUCCESS;
}
```

Misc	
(Name)	Driver.c
Content	False
File Type	C/C++ Code
Full Path	C:\Users\TheMalwareG
Included In Projec	True
Relative Path	Driver.c

(Name)
Names the file object.

```
1>Driver.c
1>KMDF Driver1.vcxproj -> C:\Users\TheMalwareGuardian\source\repos\KMDF Driver1\x64\Debug\KMDFDriver1.sys
1>Done Adding Additional Store
1>Successfully signed: C:\Users\TheMalwareGuardian\source\repos\KMDF Driver1\x64\Debug\KMDFDriver1.sys
1>
1>Driver is 'Universal'.
```

OUR MALICIOUS DRIVER

Rootkit Development

1. User Mode - Kernel Mode Communication ntddk.h	Toolkit Communication
2. Direct Kernel Object Modification ntddk.h	Hide Processes DKOM
3. Keyboard and Mouse Filter ntddk.h	Keylogger Keyboard Filter
4. Windows Filtering Platform fwpmk.h, fwpsk.h, fwpmu.h	Network Control WFP
5. Windows Kernel Sockets wsk.h	Network Requests WSK
6. File System Minifilter Driver fltKernel.h	Hide Folders Minifilter

DEMO
LOADIND...

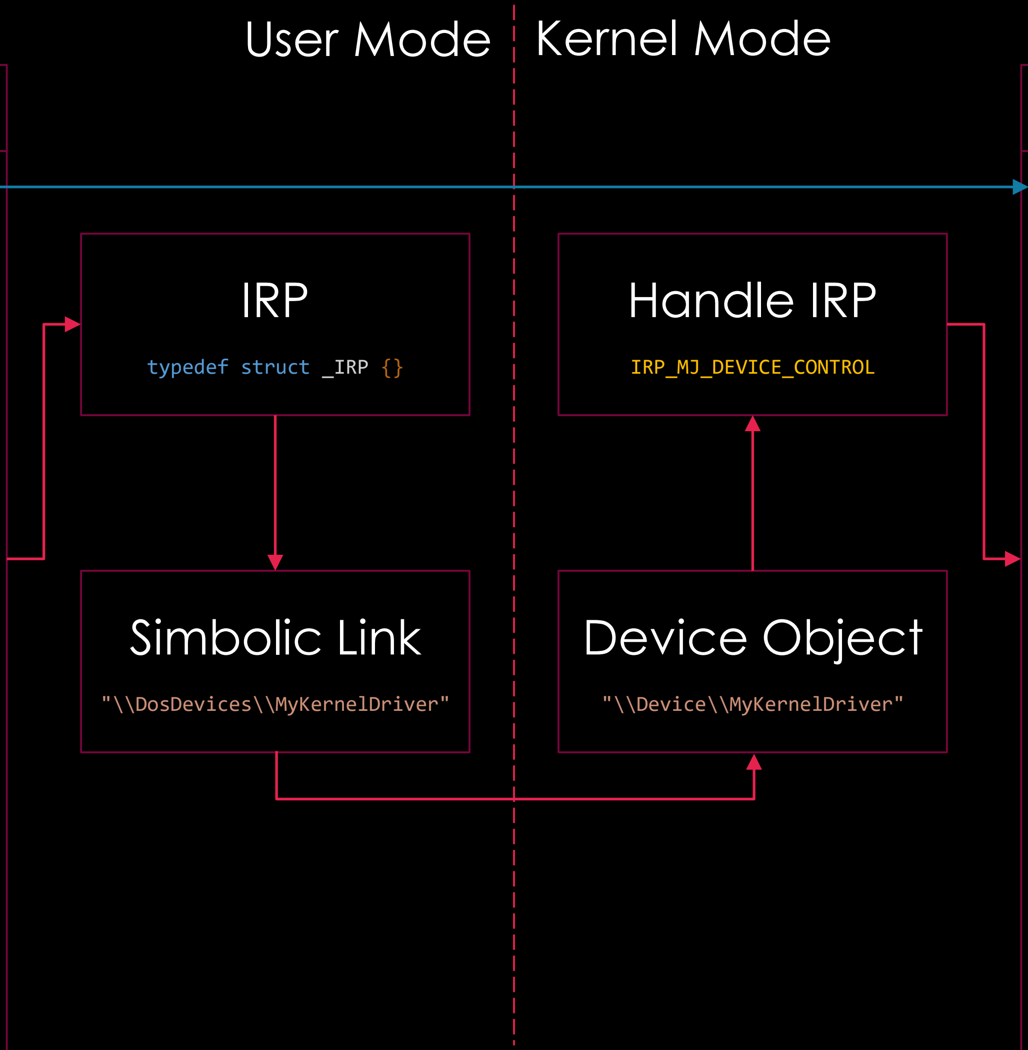
I

COMMUNICATION

“The bridge between user mode and kernel mode:
IOCTL requests initiate communication, while IRPs manage data exchange and driver actions.”

- ✓ Via Input/Output Control Codes and Input/Output Request Packets
- ✗ Via Filter Communication Ports
- ✗ Via Network Requests
- ✗ Via Shared Memory
- ✗ Via Registry Keys
- ✗ Via Files
- ✗ Via …

COMMUNICATION



console_application.exe

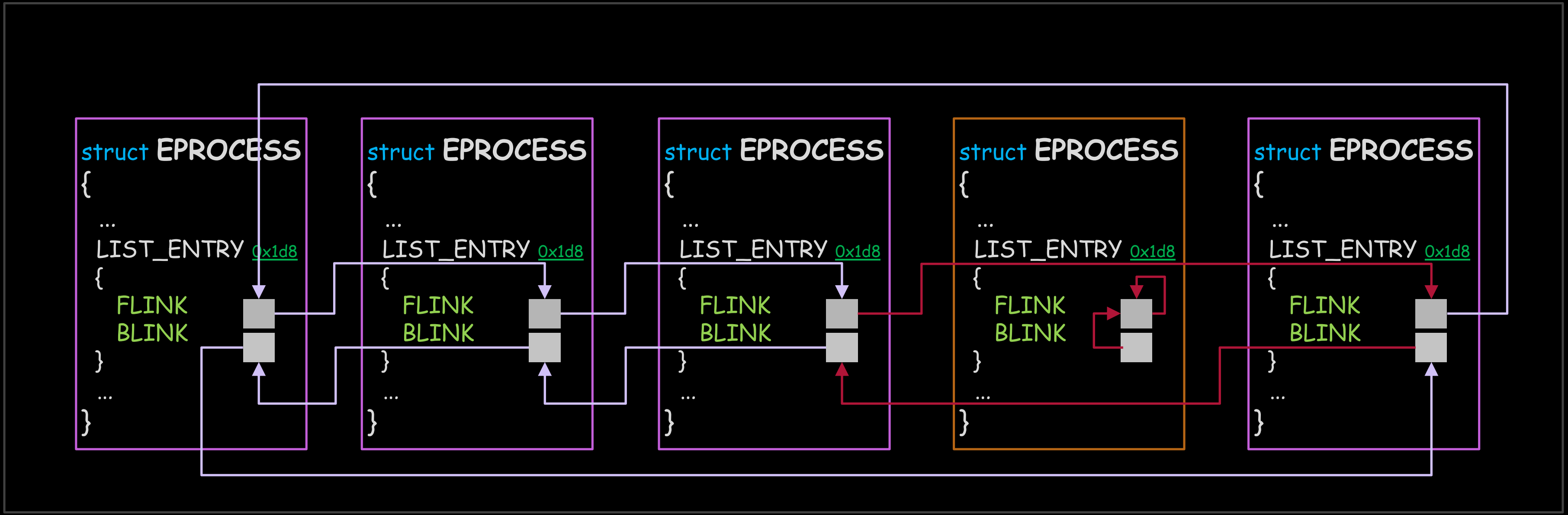
```c
// ----------------------
#define IOCTL_COMM_0 CTL_CODE(FILE_DEVICE_UNKNOWN, 0x800, METHOD_BUFFERED, FILE_ANY_ACCESS)


// ----------------------
hDevice = CreateFile(L"\\\\.\\MyKernelDriver", GENERIC_READ | GENERIC_WRITE, ...


// ----------------------
BOOL success = DeviceIoControl(hDevice,
                        1 IOCTL_COMM_0,
                        2 inBuffer, sizeof(inBuffer),
                        3 outBuffer, sizeof(outBuffer),
                         &bytesReturned, NULL);
```

kernel_mode_driver.sys

```c
// ----------------------
#define IOCTL_COMM_0 CTL_CODE(FILE_DEVICE_UNKNOWN, 0x800, METHOD_BUFFERED, FILE_ANY_ACCESS)

// ----------------------
status = IoCreateDevice(...);
status = IoCreateSymbolicLink(...);

// ----------------------
pDriverObject->MajorFunction[...] = ...;
pDriverObject->MajorFunction[IRP_MJ_DEVICE_CONTROL] = DriverHandleIOCTLs;

// ----------------------
NTSTATUS
DriverHandleIOCTLs(
    _In_  PDEVICE_OBJECT  pDeviceObject,
    _In_  PIRP            pIrp
)
{
    PIO_STACK_LOCATION stack = IoGetCurrentIrpStackLocation(pIrp);

    ULONG controlCode = stack->Parameters.DeviceIoControl.IoControlCode;

    switch (controlCode)
    {

        case IOCTL_COMM_0:
            ...
            break;

        case IOCTL_COMM_1:
            ...
            break;

    pIrp->IoStatus.Status = STATUS_SUCCESS;

    IoCompleteRequest(pIrp, IO_NO_INCREMENT);

    return STATUS_SUCCESS;
}
```

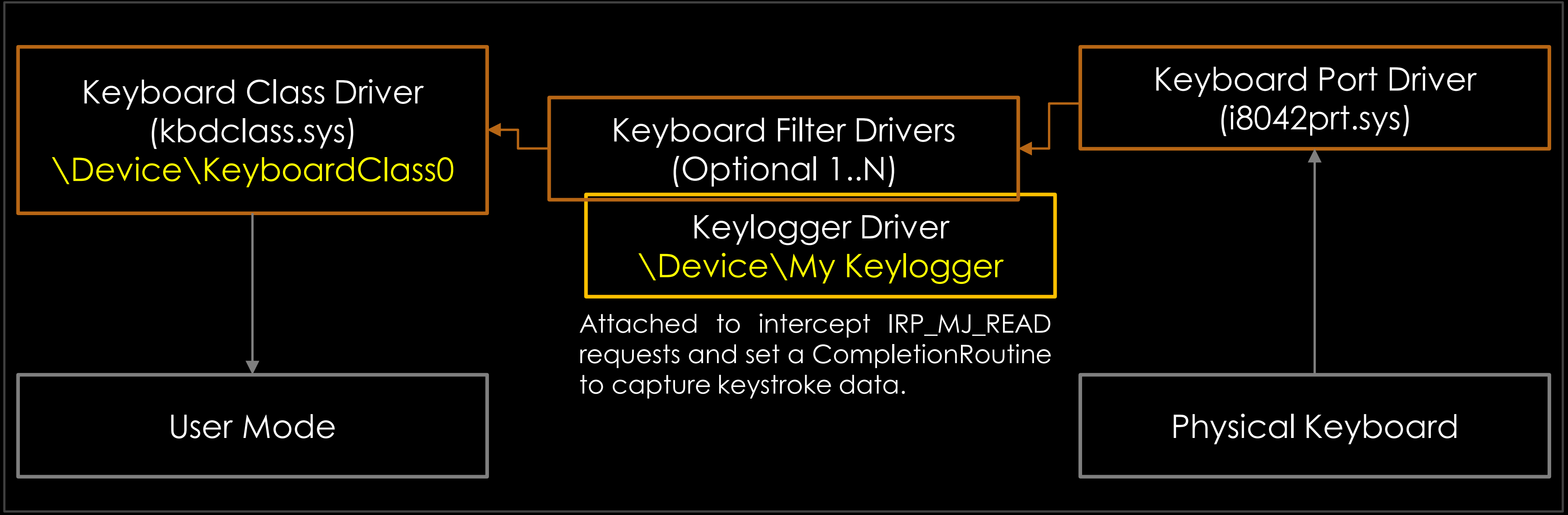
HIDE PROCCESSES

“Windows maintains a doubly linked list of active processes in (LIST_ENTRY) EPROCESS->ActiveProcessLinks. Unlink a process from the chain, and it disappears from user-mode enumeration.”



KEYLOGGER

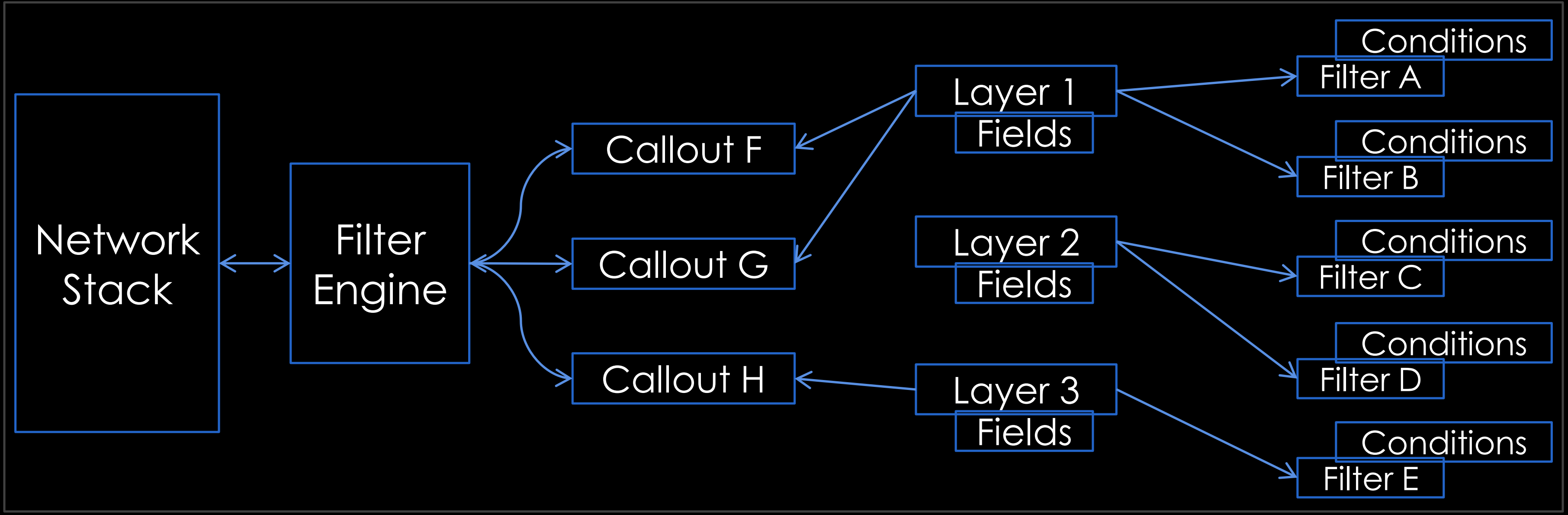
"Keystroke interception in kernel mode: The Windows keyboard driver stack routes all keystrokes through a device object called \Device\KeyboardClass0. By attaching a driver to this device and registering a CompletionRoutine, we can capture raw keystroke data before it propagates to user-mode applications like text editors or browsers."



IV

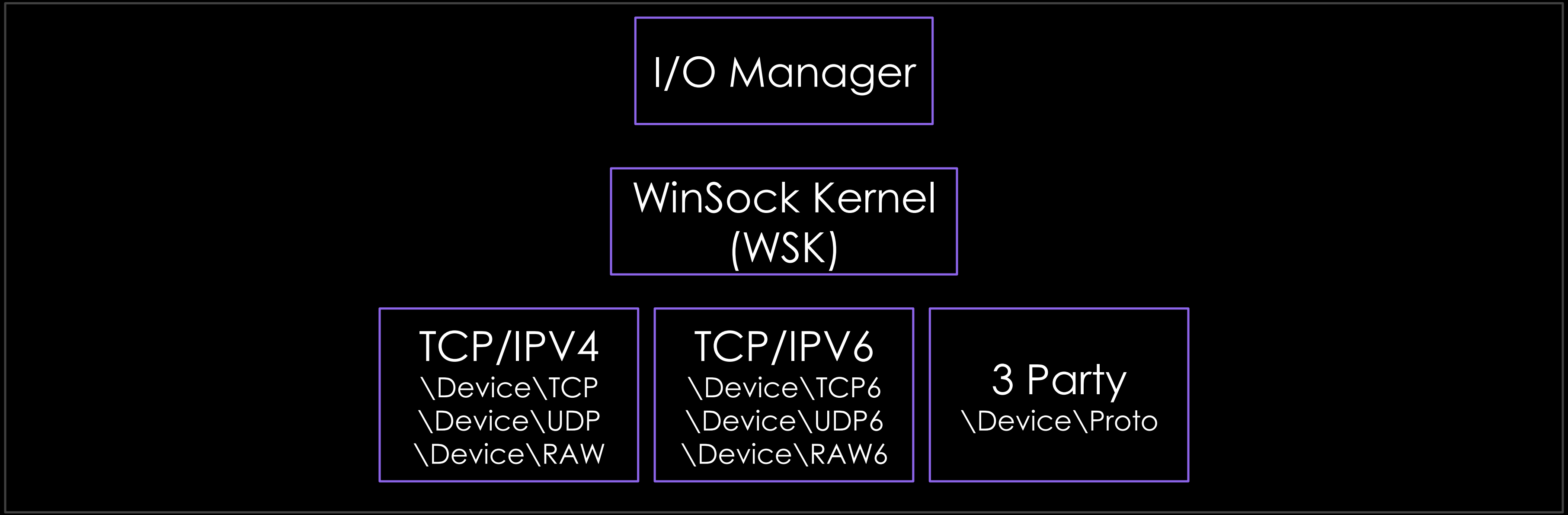
NETWORK CONTROL

"Windows Filtering Platform (WFP) allows real-time inspection and control of network connections. By attaching filters (static rules applied at specific layers of the network stack to identify traffic based on attributes like IPs or ports) and callouts (custom drivers that execute dynamic logic on flagged traffic), it's possible to classify traffic based on metadata such as the remote IP address and the associated process. Traffic that matches specific rules can be blocked, logged, or modified, enabling comprehensive network security policies."



NETWORK REQUESTS

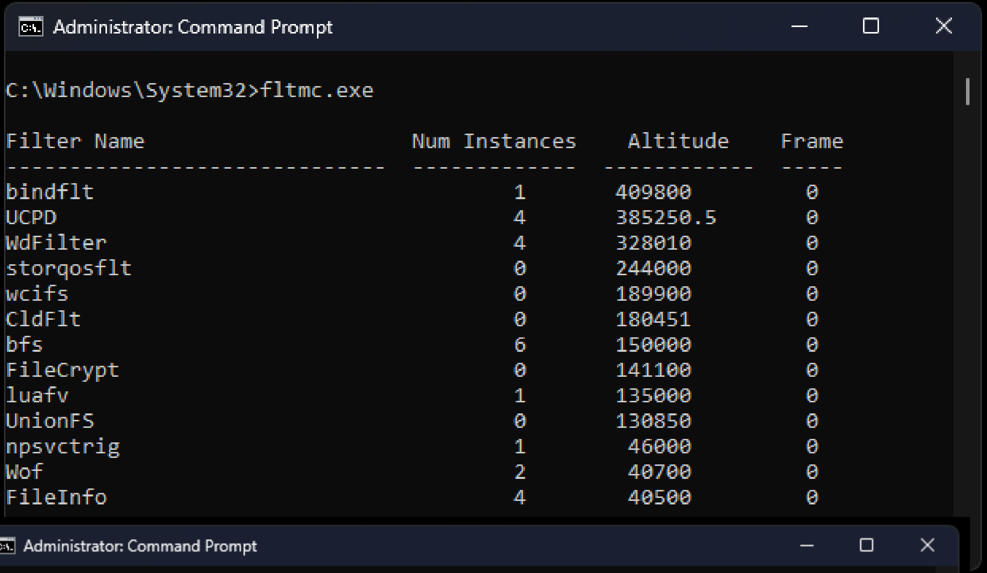
"WinSock Kernel (WSK) allows kernel-mode programs to perform complex network operations, such as establishing connections, binding sockets, and transferring data. With support for asynchronous communication using IRPs, WSK enables efficient and controlled interaction with network protocols, ensuring low-latency communication and making it a robust solution for implementing kernel-level networking features."



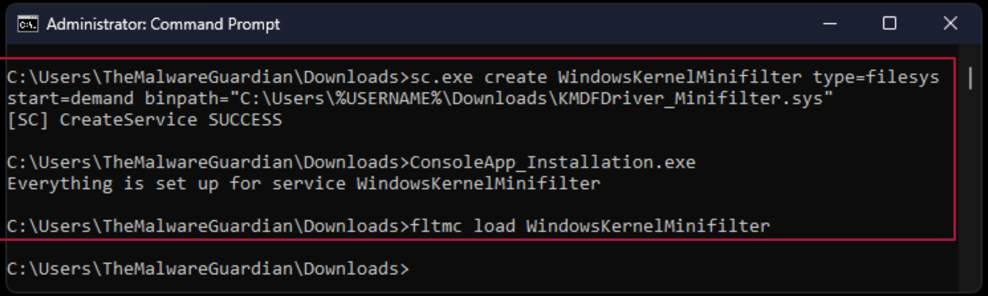
VI

HIDE FOLDERS

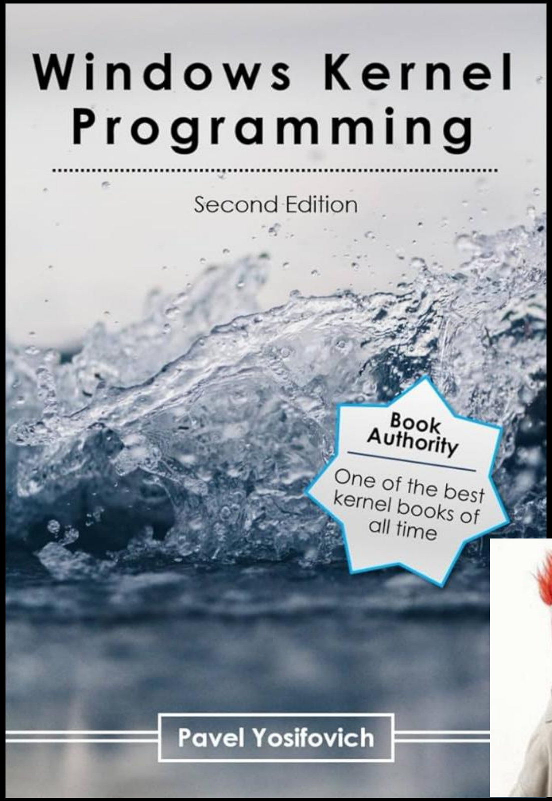
“MiniFilters attach to the file system stack to filter I/O operations.



```
C:\Windows\System32>fltmc.exe

Filter Name                     Num Instances    Altitude    Frame
------------------------------  -------------  ------------  -----
bindflt                                 1       409800         0
UCPD                                    4       385250.5       0
WdFilter                                4       328010         0
storqosflt                              0       244000         0
wcifs                                   0       189900         0
CldFlt                                  0       180451         0
bfs                                     6       150000         0
FileCrypt                               0       141100         0
luafv                                   1       135000         0
UnionFS                                 0       130850         0
npsvctrig                               1        46000         0
Wof                                     2        40700         0
FileInfo                                4        40500         0
```

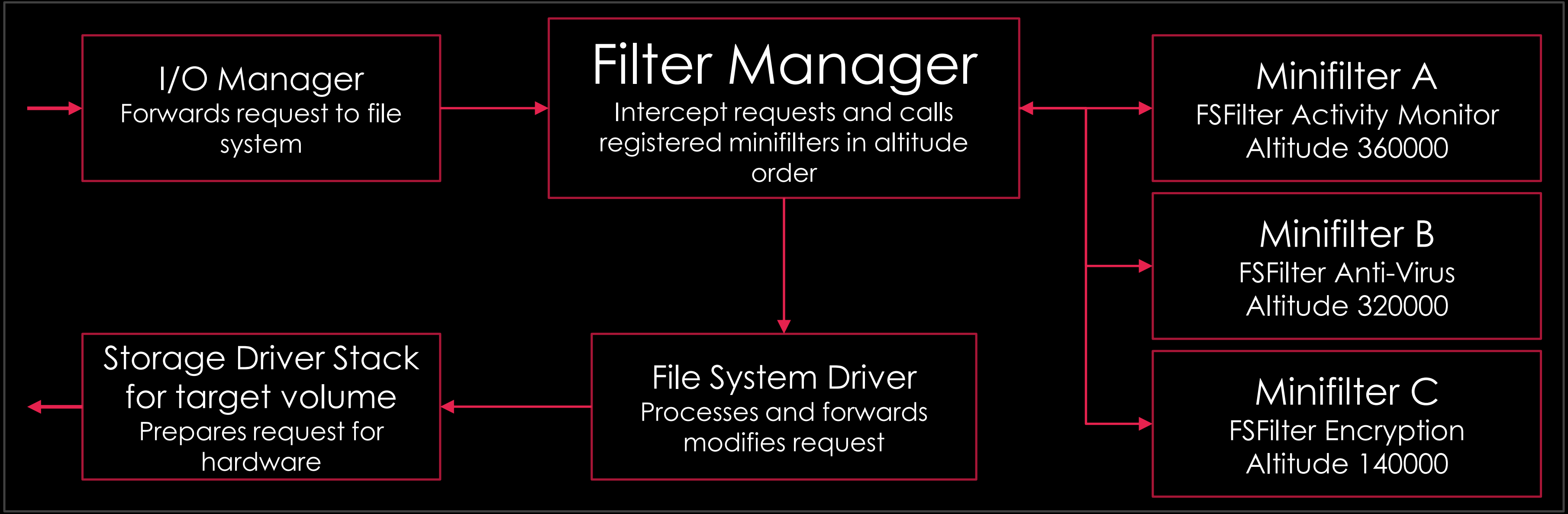


```
C:\Users\TheMalwareGuardian\Downloads>sc.exe create WindowsKernelMinifilter type=filesys
start=demand binpath="C:\Users\%USERNAME%\Downloads\KMDFDriver_Minifilter.sys"
[SC] CreateService SUCCESS

C:\Users\TheMalwareGuardian\Downloads>ConsoleApp_Installation.exe
Everything is set up for service WindowsKernelMinifilter

C:\Users\TheMalwareGuardian\Downloads>fltmc load WindowsKernelMinifilter

C:\Users\TheMalwareGuardian\Downloads>
```


Windows Kernel Programming
Second Edition
Book Authority
One of the best kernel books of all time
Pavel Yosifovich

HIDE FOLDERS

“MiniFilters attach to the file system stack to filter I/O operations. Using a PreOperation callback (triggered before the file system processes a request), access to files or directories can be explicitly denied by returning STATUS_ACCESS_DENIED or FLT_PREOP_COMPLETE. In the PostOperation callback (triggered after the request finishes), the DirectoryBuffer - which temporarily holds the directory listing - can be modified to remove specific entries, effectively making files and folders invisible to user-mode applications like File Explorer.”


I/O Manager
Forwards request to file system
Filter Manager
Intercept requests and calls registered minifilters in altitude order
Minifilter A
FSFilter Activity Monitor
Altitude 360000
Minifilter B
FSFilter Anti-Virus
Altitude 320000
Minifilter C
FSFilter Encryption
Altitude 140000
File System Driver
Processes and forwards modifies request
Storage Driver Stack for target volume
Prepares request for hardware

OUR MALICIOUS DRIVER

Rootkit Development

1. User Mode - Kernel Mode Communication ntddk.h	Toolkit Communication
2. Direct Kernel Object Modification ntddk.h	Hide Processes DKOM
3. Keyboard and Mouse Filter ntddk.h	Keylogger Keyboard Filter
4. Windows Filtering Platform fwpmk.h, fwpsk.h, fwpmu.h	Network Control WFP
5. Windows Kernel Sockets wsk.h	Network Requests WSK
6. File System Minifilter Driver fltKernel.h	Hide Folders Minifilter

THE GATEWAY

Rootkit Installation

Kernel Mode Driver

THE GATEWAY

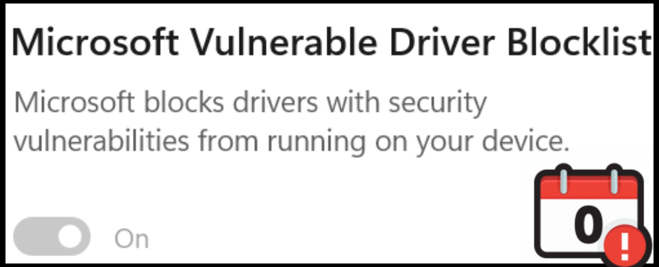
Rootkit Installation

Kernel Mode Driver ?

1. Vulnerable Kernel Driver (BYOVD)

Bring Your Own Vulnerable Driver

Not Well Known



Microsoft Vulnerable Driver Blocklist

Microsoft blocks drivers with security vulnerabilities from running on your device.

On

THE GATEWAY

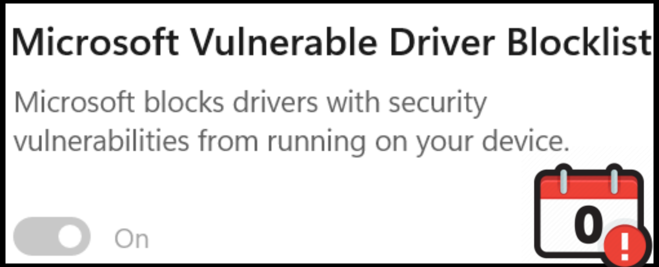
Rootkit Installation

Kernel Mode Driver ?

1. Vulnerable Kernel Driver

(BYOVD)
Bring Your Own Vulnerable Driver

Not Well Known



Microsoft Vulnerable Driver Blocklist

Microsoft blocks drivers with security vulnerabilities from running on your device.

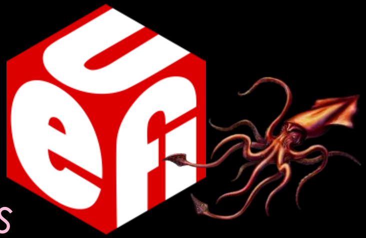
On

2. UEFI Bootkit

Infect a Computer's Boot process

SecureBoot

Physical Access



B / ROOTKIS IN THE WILD

2022

MoonBounce

CosmicStrand

Fire Chili

2024

Bootkitty

Fudmodule

2021

ESPecter

FinSpy

Moriya

2023

BlackLotus

REVERSE ENGINEERING / MALWARE ANALYSIS / BUG HUNTING

Module 10 – Windows Reverse Engineering

- Windows architecture (User mode and Kernel mode)
- Windows protections (DSE, KPP, VBS, CFG)
- Malware hunting with SysInternals tools
- Windows kernel opaque structures (EPROCESS, ETHREAD)
- Windows kernel debugging
- WinDbg scripting (Commands, Javascript, PyKd)
- Rootkit hooking techniques (IDT, SSDT)
- Rootkit development (Kernel Mode Drivers)
- Bootkit development (UEFI Applications)
- Bootkit analysis (ESPecter, BlackLotus)
- Kernel exploitation (Vulnerable drivers, Write-What-Where)

THANK YOU ☺

Rootkits PoCs & ViCONgal 2025 PPT:

github.com/TheMalwareGuardian/**Bentico**

Every resource you need to develop Rootkits:

github.com/TheMalwareGuardian/**Awesome-Bootkits-Rootkits-Development**

Automate Bootkits/Rootkits Development

github.com/TheMalwareGuardian/**Bootkits-Rootkits-Development-Environment**

Contact:

www.linkedin.com/in/**vazquez-vazquez-alejandro**

Agradecimientos: