# 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]

Rooted 2025, Madrid

/Rooted CON

# WHOAMI

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

[in/vazquez-vazquez-alejandro]

# TABLE OF CONTENTS

## Rootkit Development

# SENSITIVE CONTENT

## Age Verification

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

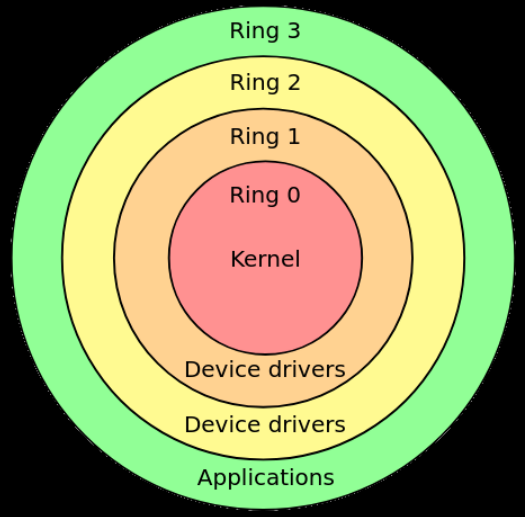
**Let me In**
**This is real stuff**

**No**
**I prefer OSINT**

# KERNEL (RING 0)

The kernel (Ring 0) of an operating system implements the core functionality that everything else in the operating system depends upon. The Microsoft Windows kernel provides basic low-level operations such as scheduling threads or routing hardware interrupts. It is the heart of the operating system and all tasks it performs must be fast and simple.

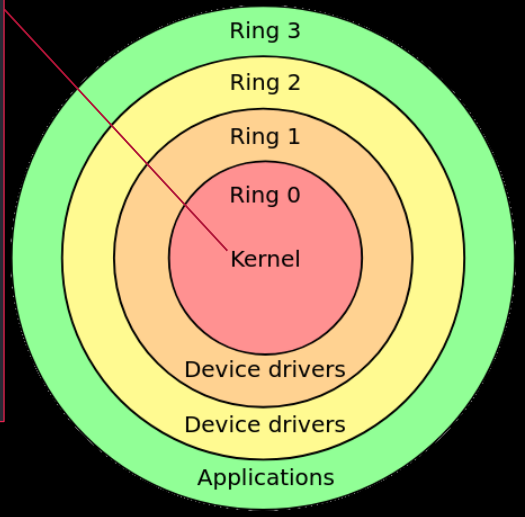
~ Microsoft



# KERNEL (RING 0)

```
//0x150 bytes (sizeof)
struct _DRIVER_OBJECT
{
    SHORT Type;                                                    //0x0
    SHORT Size;                                                    //0x2
    struct _DEVICE_OBJECT* DeviceObject;                           //0x8
    ULONG Flags;                                                   //0x10
    VOID* DriverStart;                                             //0x18
    ULONG DriverSize;                                              //0x20
    VOID* DriverSection;                                           //0x28
    struct _DRIVER_EXTENSION* DriverExtension;                     //0x30
    struct _UNICODE_STRING DriverName;                             //0x38
    struct _UNICODE_STRING* HardwareDatabase;                      //0x48
    struct _FAST_IO_DISPATCH* FastIoDispatch;                      //0x50
    LONG (*DriverInit)(struct _DRIVER_OBJECT* arg1, struct _UNICODE_STRING* arg2); //0x58
    VOID (*DriverStartIo)(struct _DEVICE_OBJECT* arg1, struct _IRP* arg2);  //0x60
    VOID (*DriverUnload)(struct _DRIVER_OBJECT* arg1);             //0x68
    LONG (*MajorFunction[28])(struct _DEVICE_OBJECT* arg1, struct _IRP* arg2); //0x70
};
```

ents the core functionality
ends upon. The Microsoft
ons such as scheduling
t of the operating system



# ROOTKIT

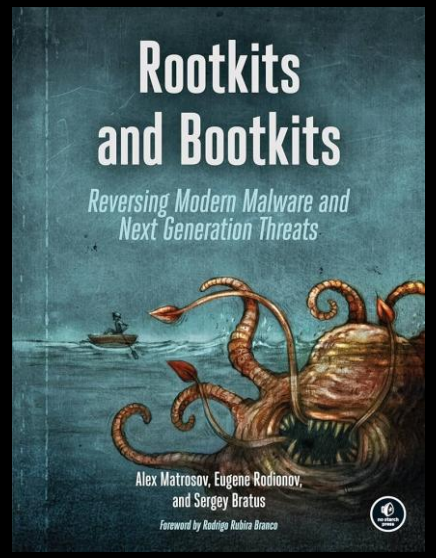
Rootkit: Sophisticated piece of malware that can add new code to the operating system or delete and edit operating system code. Rootkits may remain in place for years because they are hard to detect, due in part to their ability to block some antivirus software and malware scanner software.

~ Crowdstrike

# 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




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

# SECURITY MECHANISMS

[*Anti-Rootkit Installation*]

- Driver Signature Enforcement (DSE)

  Windows won't run drivers not certified by Microsoft

[*Anti-Rootkit Deep Funcionalities*]

- Kernel Patch Protection (PatchGuard)

  Feature of 64-bit editions of Microsoft Windows

  Prevents patching the kernel

# SECURITY MECHANISMS

*[Anti-Rootkit Installation]*

- Driver Signature Enforcemen
  Windows won't run drivers

*[Anti-Rootkit Deep Funcionalities]*

- Kernel Patch Protection (Pa
  Feature of 64-bit editions c
  Prevents patching the kern

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.

C:\Windows\System32>
```

1

2

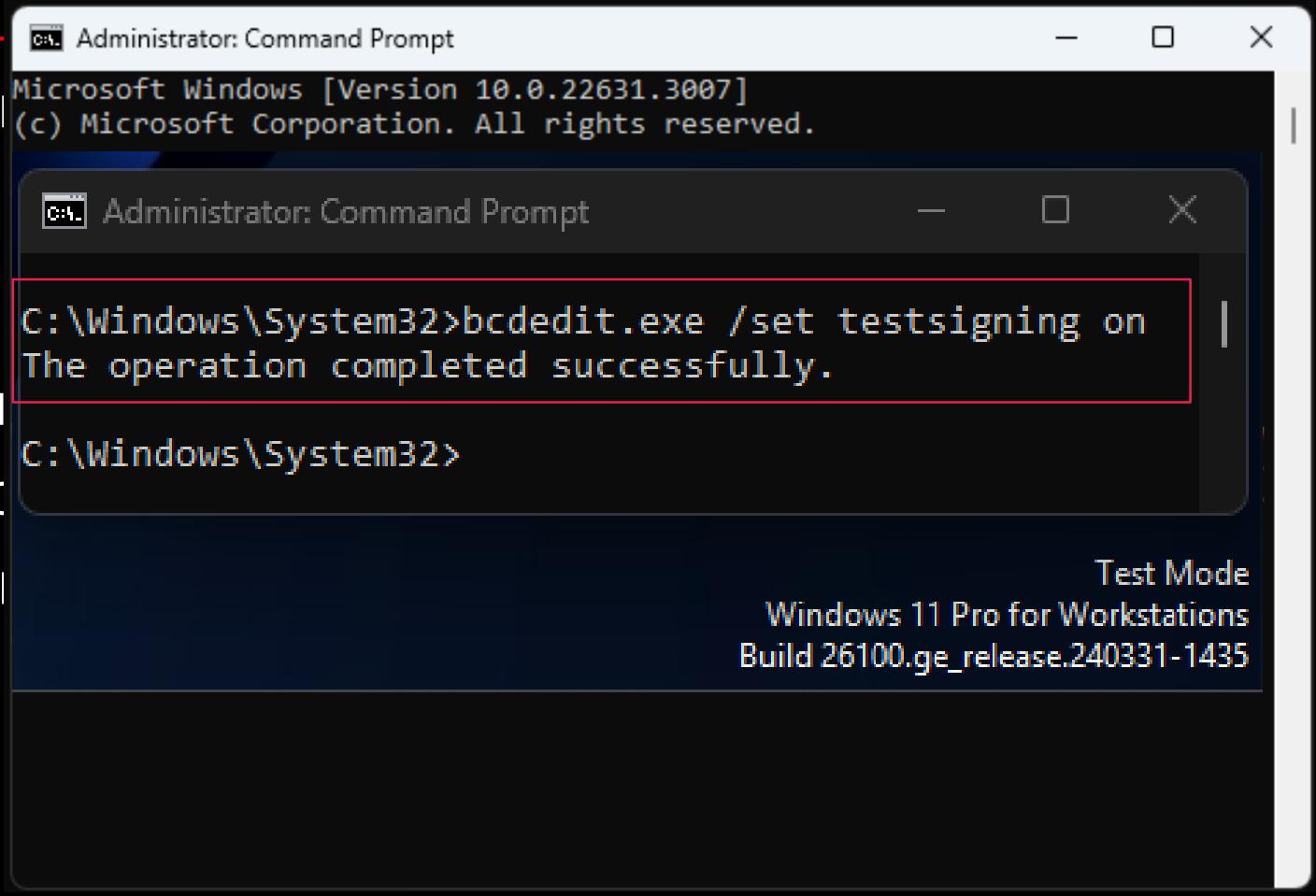
# SECURITY MECHANISMS

*[Anti-Rootkit Installation]*

- Driver Signature Enforcemen
  Windows won't run drivers

*[Anti-Rootkit Deep Funcionalities]*

- Kernel Patch Protection (Pa
  Feature of 64-bit editions c
  Prevents patching the kern



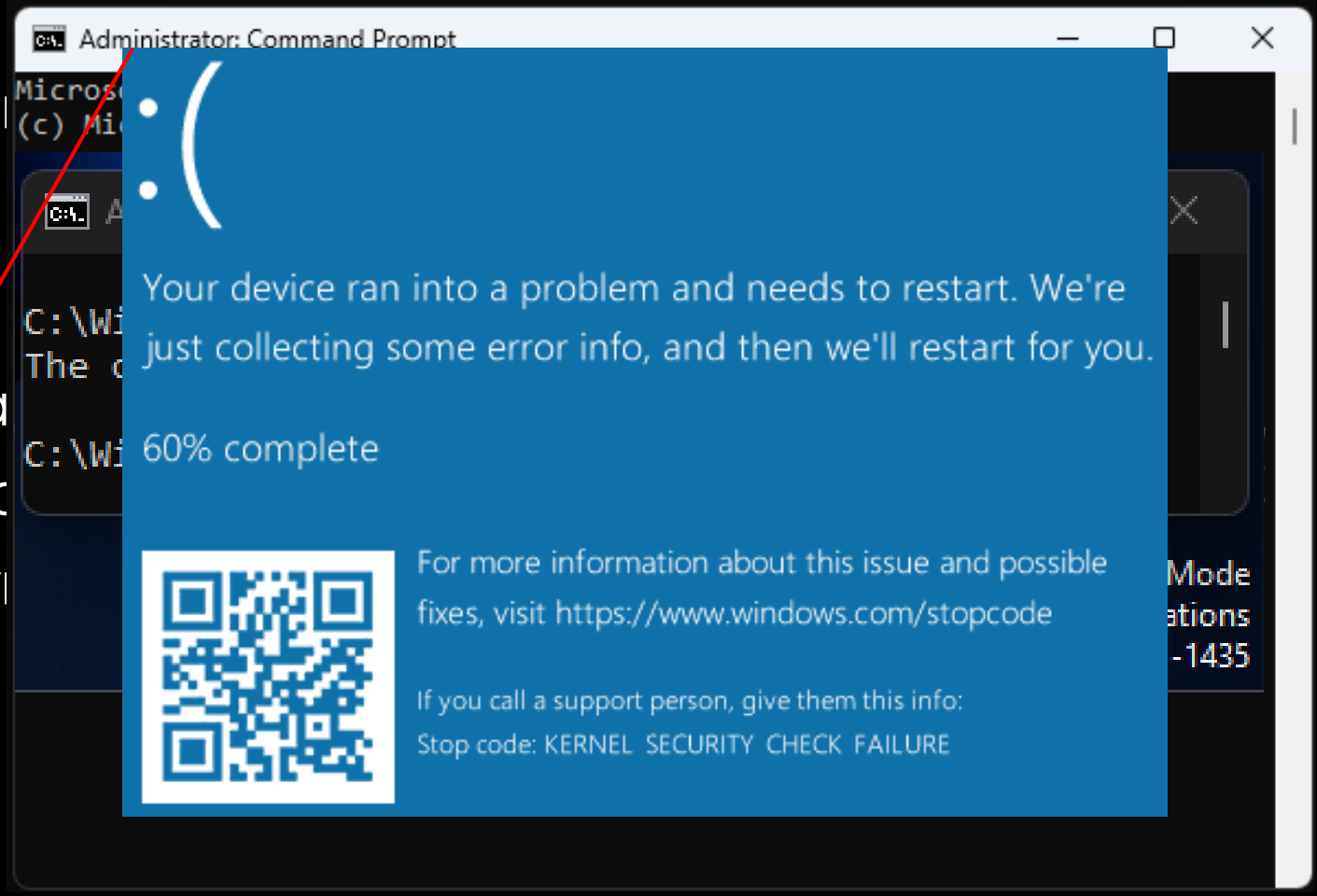
# SECURITY MECHANISMS

*[Anti-Rootkit Installation]*

- Driver Signature Enforcemen
  Windows won't run drivers

*[Anti-Rootkit Deep Funcionalities]*

- Kernel Patch Protection (Pa
  Feature of 64-bit editions c
  Prevents patching the kern



# CHECKPOINT

## Rootkit Development

1. Windows Kernel
2. Rootkit = Kernel Mode Driver
3. DSE, PatchGuard, …

# DEVELOPMENT ENVIRONMENT

Google

how to develop a Windows kernel mode driver

Learn / Windows / Windows Drivers /

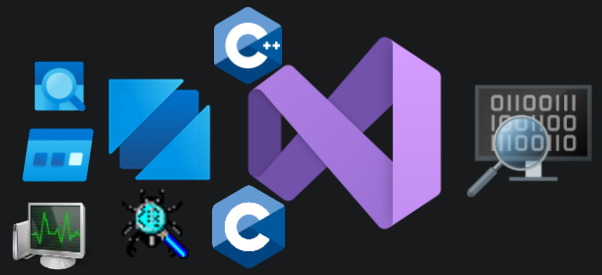
## Tutorial: Write a Hello World Windows Driver (Kernel-Mode Driver Framework)

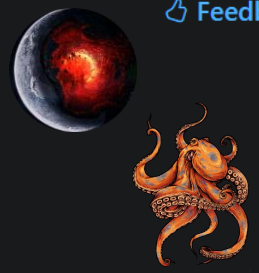
Article • 12/19/2024 • 9 contributors

Feedback

In this article







Google

how to develop a Windows kernel mode driver

Apriorit
https://www.apriorit.com · Traducir esta página

Writing a Windows Driver Model Driver: A Step-by-Step ...

18 mar 2024 — In this article, we provide a practical example of writing a Windows Driver Model (WDM) driver for encrypting a virtual disk that stores user data.

Red Team Notes
https://www.ired.team · Traducir esta página

Windows Kernel Drivers 101

7 mar 2020 — This living document captures some of the Kernel Driver and OS related concepts that I encounter as I study Windows kernel driver development.

221bluestreet.com
https://www.221bluestreet.com · Traducir esta página

Windows Kernel Drivers 101 - Creating a Simple Driver

29 oct 2022 — Short Introduction to Windows Software Kernel Driver with Code snippet and example for a basic driver and a User-Mode client.

GitHub
https://v3ded.github.io · Traducir esta página

Red Team Tactics: Writing Windows Kernel Drivers ...

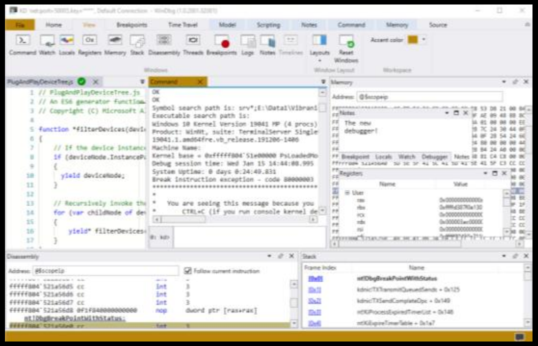
29 dic 2022 — In order to write kernel drivers, it is necessary to have a good understanding of the C programming language. If you are not familiar with C, it ...

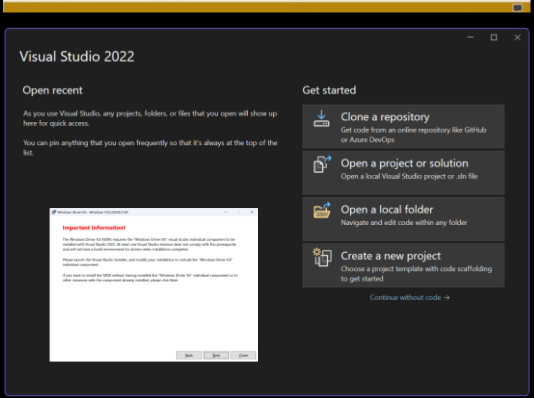
nixhacker
http://nixhacker.com · Traducir esta página

Developing and Installing your first Kernel driver in ...

10 jun 2020 — Select the Driver option in project type. Select the Kernel Mode Driver(KMDF). Enter the details on the next prompt. If you can't see the Driver ...

# DEVELOPMENT ENVIRONMENT





As you use Visual Studio, any projects, folders, or files that you open will show up here for quick access.

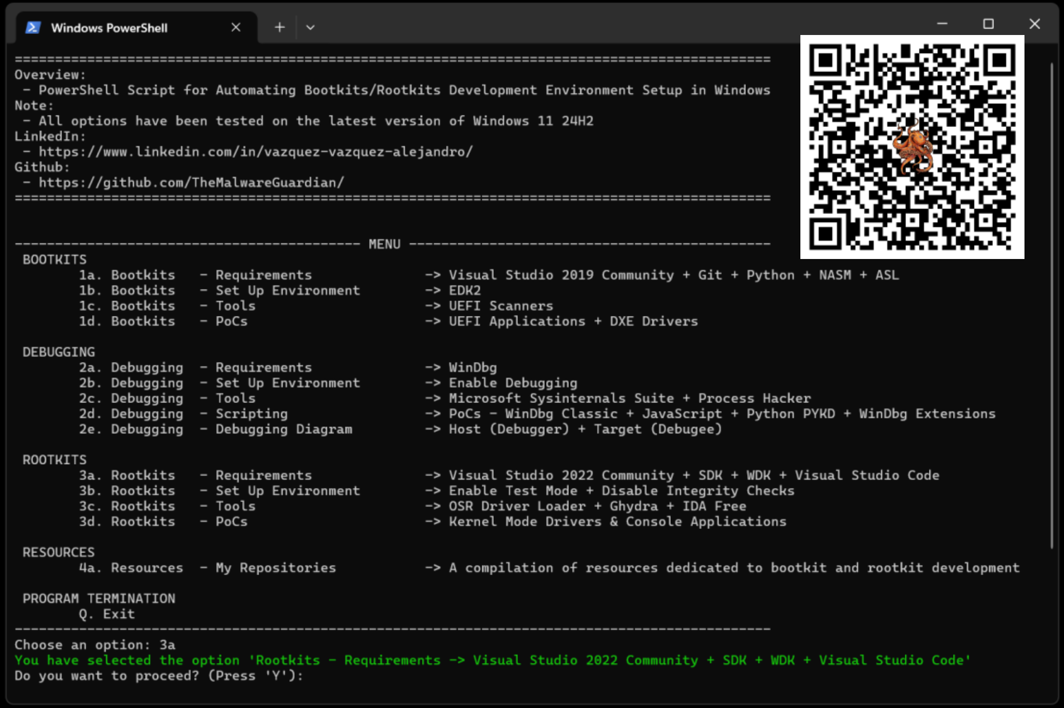
You can pin anything that you open frequently so that it's always at the top of the list.

Important Information!

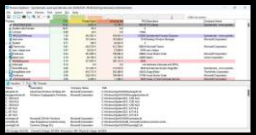
Take a look into the depths of Windows kernels and reveal more than 60000 undocumented structures

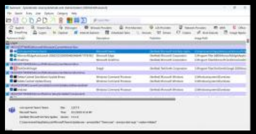
Awesome Bootkits & Rootkits Development

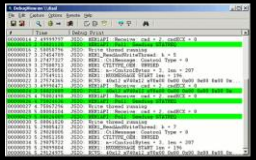
A curated compilation of extensive resources dedicated to bootkit and rootkit development.

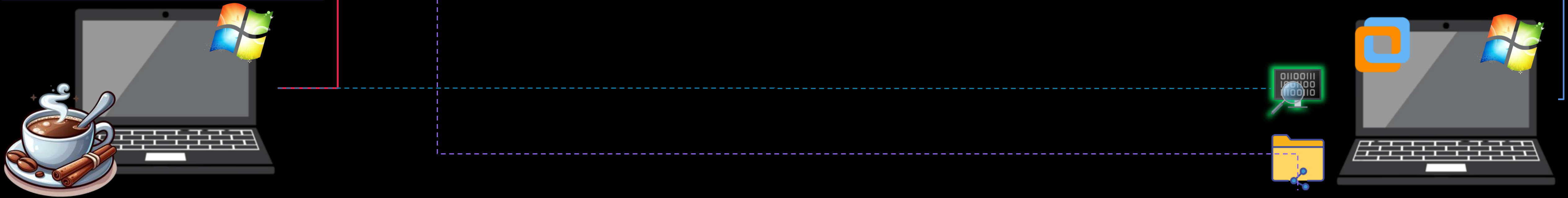


```
=============================================================================================
Overview:
 - PowerShell Script for Automating Bootkits/Rootkits Development Environment Setup in Windows
Note:
 - All options have been tested on the latest version of Windows 11 24H2
LinkedIn:
 - https://www.linkedin.com/in/vazquez-vazquez-alejandro/
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                     -> UEFI Scanners
       1d. Bootkits   - PoCs                      -> UEFI Applications + DXE 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: 3a
You have selected the option 'Rootkits - Requirements -> Visual Studio 2022 Community + SDK + WDK + Visual Studio Code'
Do you want to proceed? (Press 'Y'):
```









# KERNEL MODE DRIVER

Administrator: Command Prompt

```
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>
```

DebugView on \\MALWARE (local)

| # | 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 |

Create a new project

Recent project templates

Kernel Mode Driver, Empty (KMDF) C++

kernel

Clear all

C++ Windows All project types

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

Back Next

KMDF Driver1 — Driver.c

```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;
}
```

Output — Build

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


DEMO
LOADIND...

# 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

User Mode | Kernel Mode

## console_application.exe

```
// ----------------------
#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);
```

## IRP

```
typedef struct _IRP {}
```

## Simbolic Link

"\\DosDevices\\MyKernelDriver"

## Handle IRP

IRP_MJ_DEVICE_CONTROL

## Device Object

"\\Device\\MyKernelDriver"

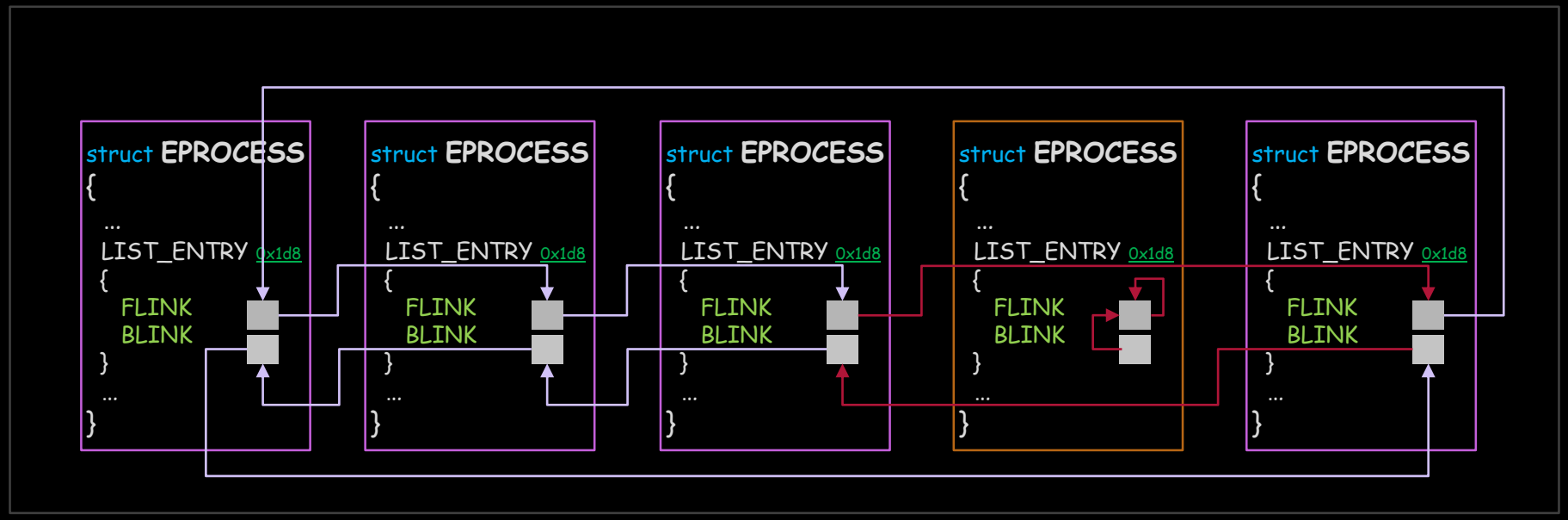
## kernel_mode_driver.sys

```
// ----------------------
#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;
}
```

II

# HIDE PROCCESSES

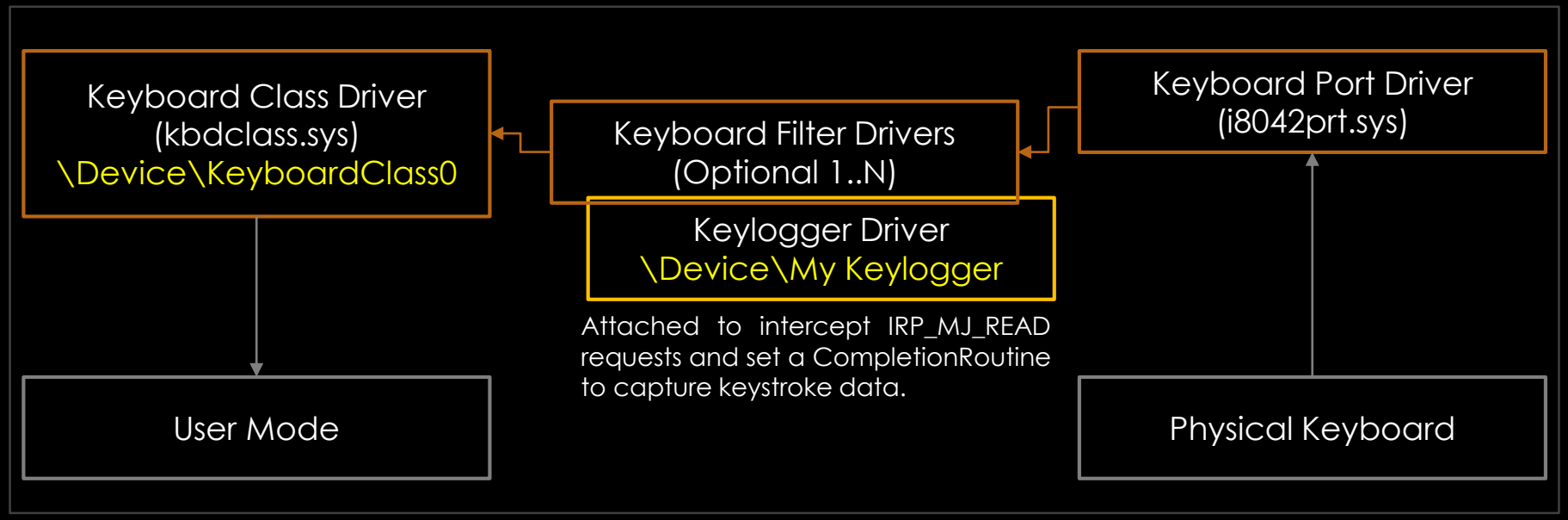
"Windows maintains a doubly linked list of active processes in (LIST_ENTRY) EPROCESS->ActiveProcessLinks. Unlink a process from the chain, and it disappears from user-mode enumeration."



III

# 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 (a callback executed after an IRP has been processed by lower drivers, allowing access to data before it reaches the next stage), we can capture raw keystroke data before it propagates to user-mode applications like text editors or browsers."



IV

# HIDE FOLDERS

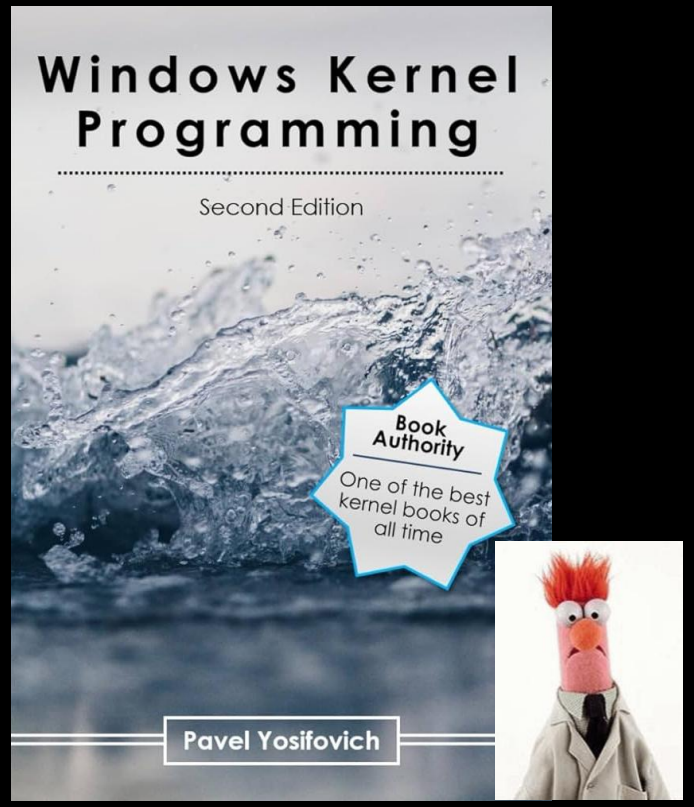
"MiniFilters attach to the file system stack to filter I/O operations.

Administrator: Command Prompt

```
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
```

Administrator: Command Prompt

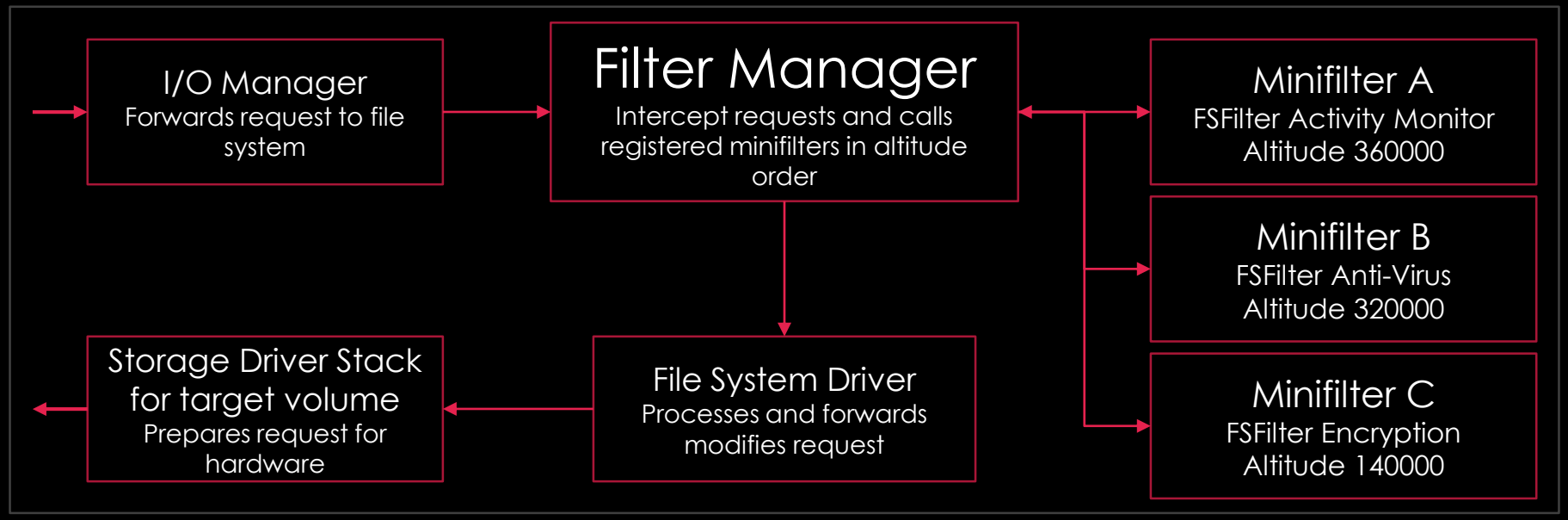
```
C:\Users\TheMalwareGuardian\Downloads>sc.exe create WindowsKernelMinifilter type=filesys
start=demand binpath="C:\Users\%USERNAME%\Downloads\KMDFDriver_Minifilter.sys"
[SC] CreateService SUCCESS

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>
```



IV

# HIDE FOLDERS

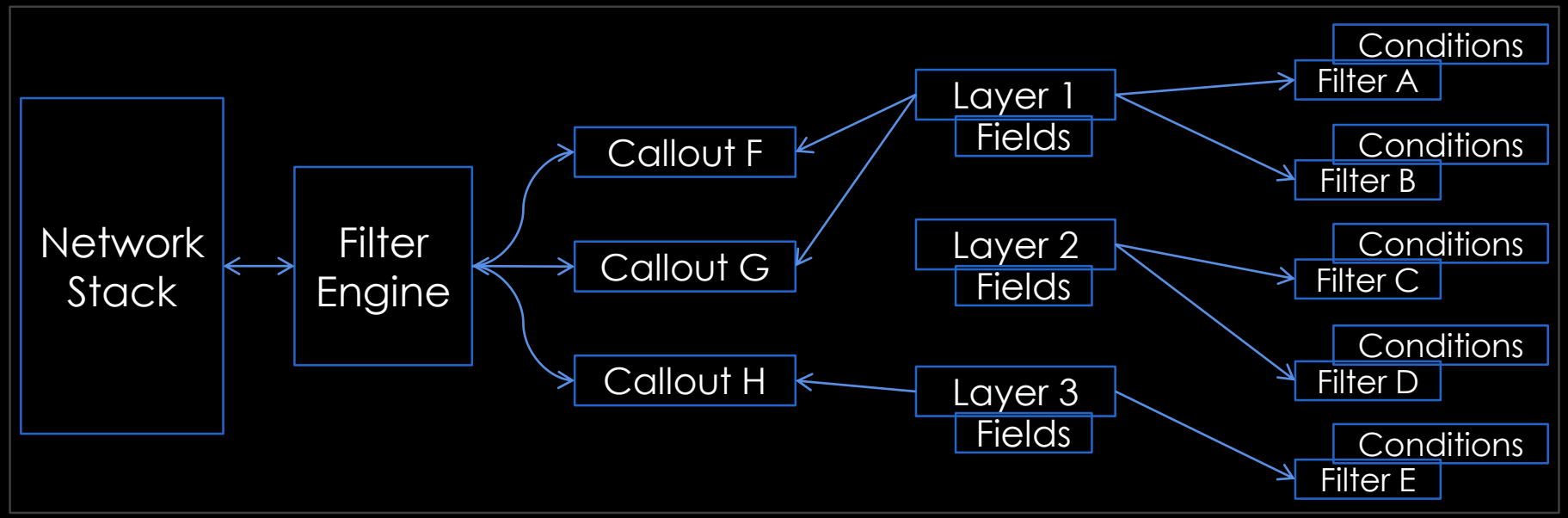
"MiniFilters attach to the file system stack to filter I/O operations. Using a PreOperation callback (triggered before the file system processes a request), access to files or directories can be explicitly denied by returning STATUS_ACCESS_DENIED or FLT_PREOP_COMPLETE. In the PostOperation callback (triggered 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."



V

# NETWORK CONTROL

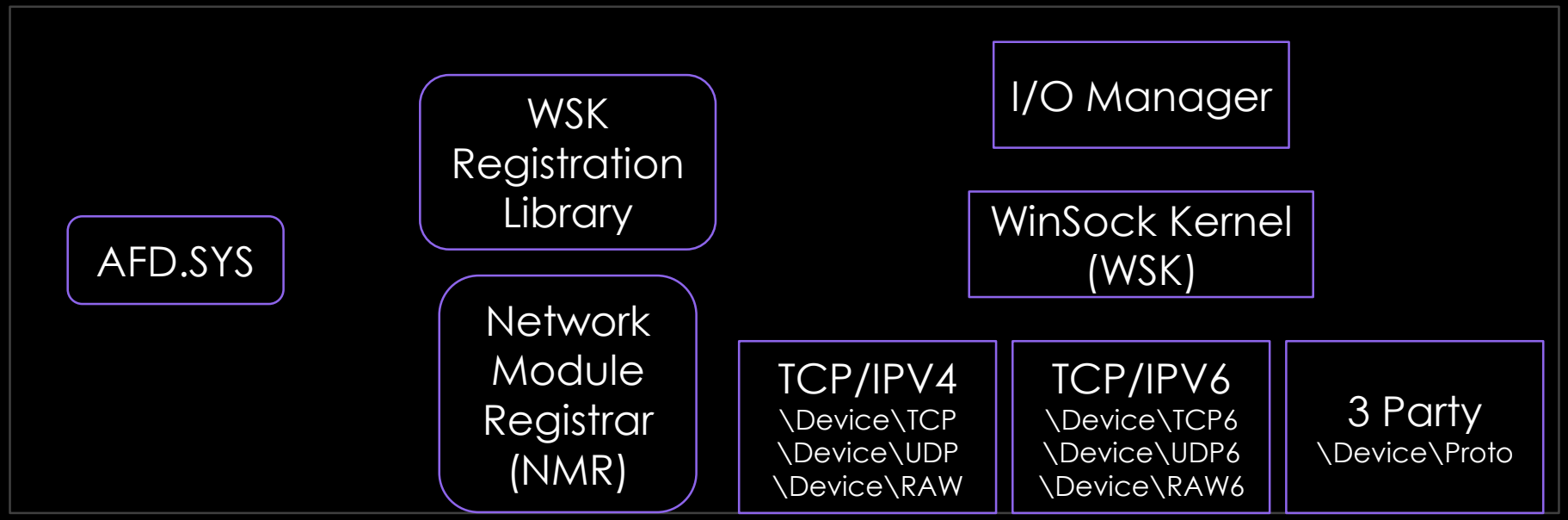
"Windows Filtering Platform (WFP) allows real-time inspection and control of network connections. By attaching filters (static rules applied at specific layers of the network stack to identify traffic based on attributes like IPs or ports) and callouts (custom drivers that execute dynamic logic on flagged 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."



VI

# 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."



# BENTICO

KernelRootkit001_HelloWorld

KernelRootkit002_Threading

KernelRootkit003_ZwFunctions

KernelRootkit004_Callbacks

KernelRootkit005_IOCTLs

KernelRootkit006_DKOM

KernelRootkit007_KeyboardFilter

KernelRootkit008_Minifilter

KernelRootkit009_FilterCommunicationPort

KernelRootkit010_WindowsFilteringPlatform

KernelRootkit011_WinSockKernel

# THE GATEWAY

Rootkit Installation

Kernel Mode Driver

# THE GATEWAY

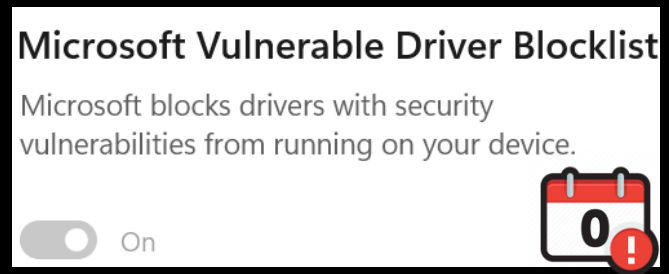
Rootkit Installation

Kernel Mode Driver ?

1. Vulnerable Kernel Driver

(BYOVD)
Bring Your Own Vulnerable Driver

Not Well Known



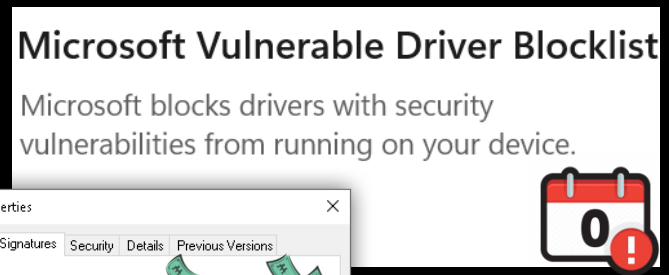
# THE GATEWAY

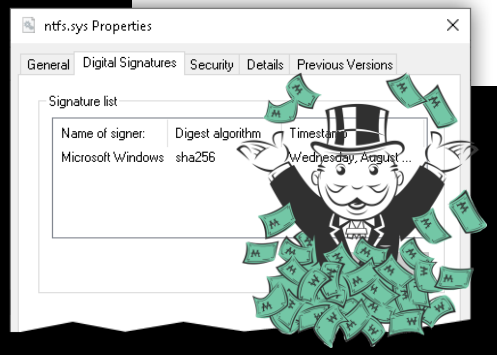
Rootkit Installation
Kernel Mode Driver ?

1. Vulnerable Kernel Driver (BYOVD)
   Bring Your Own Vulnerable Driver

Not Well Known

2. Leaked Certificate Driver Signing Policy



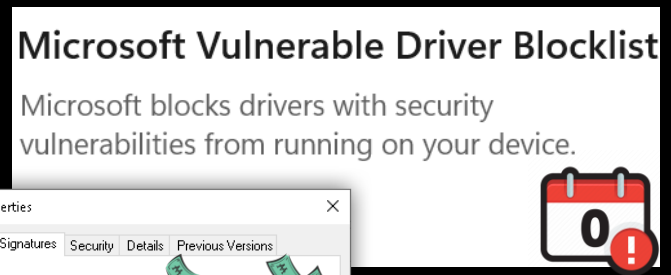



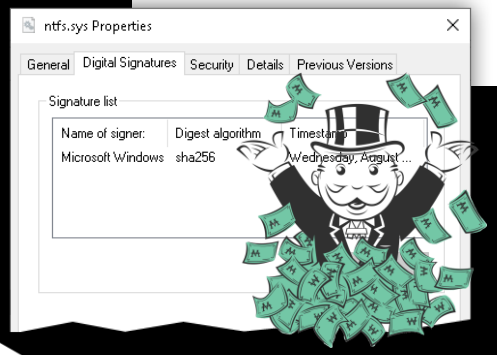
# THE GATEWAY

Rootkit Installation
Kernel Mode Driver

1. Vulnerable Kernel Driver — (BYOVD) Bring Your Own Vulnerable Driver

Not Well Known




2. Leaked Certificate — Driver Signing Policy

3. UEFI Bootkit — Infect a Computer's Boot process

SecureBoot

Physical Access

# B / ROOTKIS IN THE WILD

2022

MoonBounce

CosmicStrand

Fire Chili

2

2024

Bootkitty

Fudmodule

1

2021

3

ESPecter

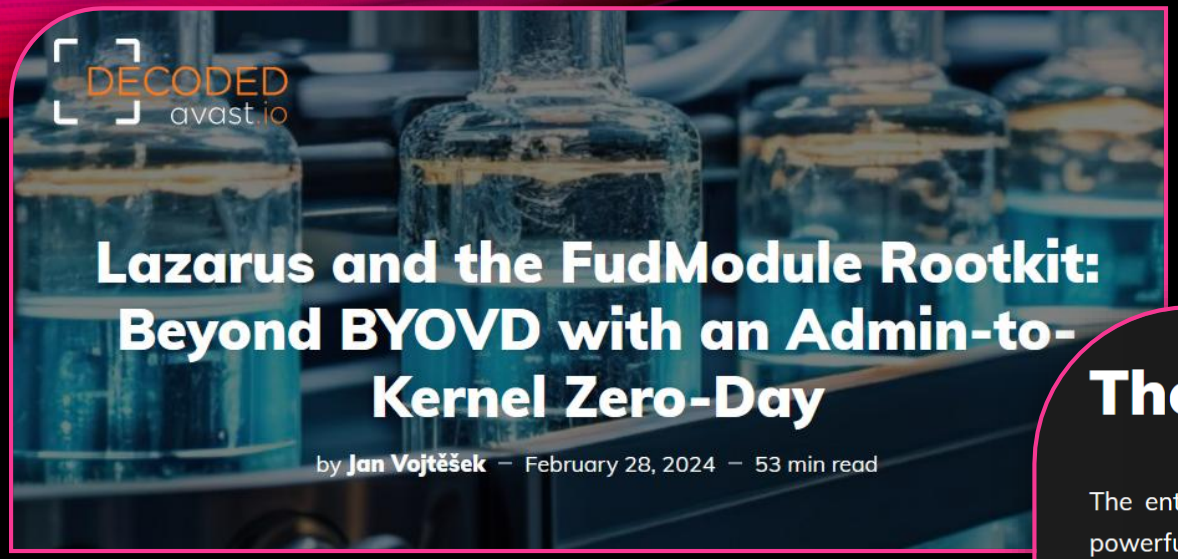
FinSpy

Moriya

2023

BlackLotus

# FUDMODULE




- Advanced Persistent Threat 38
- Bring Your Own Vulnerable Driver
- N-DAY, CVE
- Windows Kernel Mode Driver
- Minifilter Driver
- Windows Filtering Platform

## The FudModule Rootkit

The entire goal of the admin-to-kernel exploit was to corrupt the current thread's `PreviousMode`. This allows for a powerful kernel read/write primitive, where the affected user-mode thread can read and write arbitrary kernel memory using the `Nt(Read|Write)VirtualMemory` syscalls. Armed with this primitive, the FudModule rootkit employs direct kernel object manipulation (DKOM) techniques to disrupt various kernel security mechanisms. It's worth reiterating that FudModule is a data-only rootkit, meaning it executes entirely from user space and all the kernel tampering is performed through the read/write primitive.

The first variants of the FudModule rootkit were independently discovered by AhnLab and ESET research teams, with both publishing detailed analyses in September 2022. The rootkit was named after the `FudModule.dll` string used as the name in its export table. While this artifact is not present anymore, there is no doubt that what we found is an updated version of the same rootkit. AhnLab's report documented a sample from early 2022, which incorporated seven data-only rootkit techniques and was enabled through a BYOVD exploit for ene.sys. ESET's report examined a slightly earlier variant from late 2021, also featuring seven rootkit techniques but exploiting a different BYOVD vulnerability in dbutil_2_3.sys. In contrast, our discovery concerns a sample featuring nine rootkit techniques and exploiting a previously unknown admin-to-kernel vulnerability. Out of these nine techniques, four are new, three are improved, and two remain unchanged from the previous variants. This leaves two of the original seven techniques, which have been deprecated and are no longer present in the latest variant.

# FUDMODULE

## 0x01 – Registry Callbacks

The first rootkit technique is designed to address registry callbacks. This is a documented Windows mechanism which allows security solutions to monitor registry operations.

## 0x10 – Windows Filtering Platform

Windows Filtering Platform (WFP) is a documented set of APIs designed for host-based network traffic filtering. The WFP API offers capabilities for deep packet inspection as well as for modification or dropping of packets at various layers of the network stack.

Specifically, the kernel API allows for installing so-called callout drivers,

ead's `PreviousMode`. This allows for a
read and write arbitrary kernel memory
FudModule rootkit employs direct kernel
echanisms. It's worth reiterating that
nd all the kernel tampering is performed

Lab and ESET research teams, with both
the `FudModule.dll` string used as the
oubt that what we found is an updated
22, which incorporated seven data-only
report examined a slightly earlier variant
BYOVD vulnerability in dbutil_2_3.sys. In
exploiting a previously unknown admin-
roved, and two remain unchanged from
ave been deprecated and are no longer

FORTIGUARD LABS THREAT RESEARCH

# New Milestones for Deep Panda: Log4Shell and Digitally Signed Fire Chili Rootkits

By Rotem Sde-Or and Eliran Voronovitch | March 30, 2022

During the past month, FortiEDR detected a campaign by Deep Panda, a Chinese APT group. [redacted]

Following exploitation, Deep Panda deployed a backdoor on the infected machines. Following forensic leads from the backdoor led us to discover a novel kernel rootkit signed with a stolen digital certificate. We found that the same certificate was also used by another Chinese APT group, named Winnti, to sign some of their tools.

[redacted]

# FIRE CHILI

| IOCTL | Action | Description |
|---|---|---|
| 0xF3060000 | Hide file | Add a path to global file list |
| 0xF3060004 | Stop hiding file | Remove a path from global file list |
| 0xF3060008 | Hide\protect process | Add a file path or PID to global process list |
| 0xF306000C | Stop hiding\protecting process | Remove a file path or PID from global process list |
| 0xF3060010 | Hide registry key | Add a key to global registry list |
| 0xF3060014 | Stop hiding registry key | Remove a key from global registry list |
| 0xF3060018 | Hide network connections | Add a file path or port number to global network list |
| 0xF306001C | Stop hiding network connections | Remove a file path or port number from global network list |

ESET RESEARCH

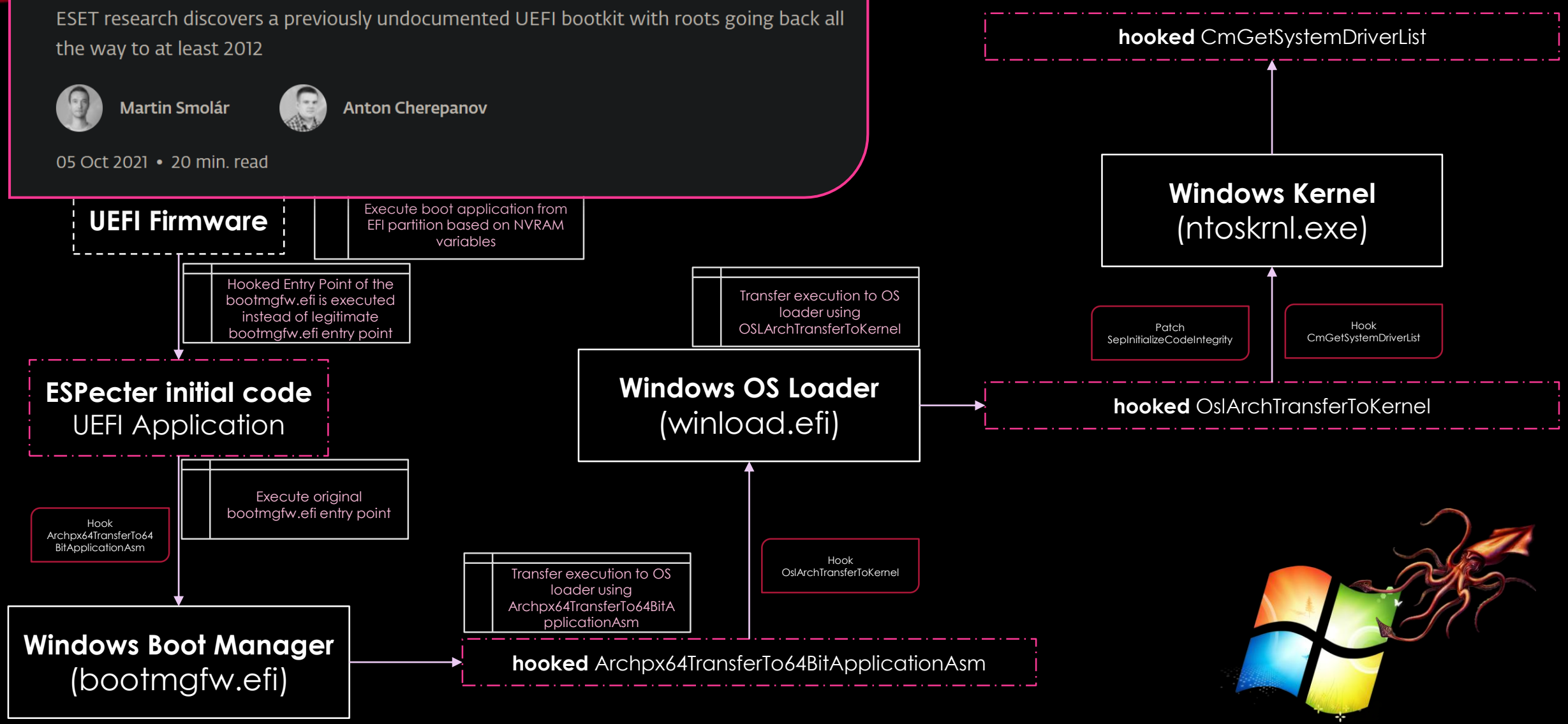
# UEFI threats moving to the ESP: Introducing ESPecter bootkit

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

# ESPECTER



# ESPECTER

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

**hooked** CmGetSystemDriverList

**Windows Kernel** (ntoskrnl.exe)

**UEFI Firmware**

Execute boot application from EFI partition based on NVRAM variables

**ESPecter initi**
UEFI Applic

### Kernel-mode driver

- Interception of keystrokes is done by setting up `CompletionRoutine` for `IRP_MJ_READ` requests for the keyboard driver object `\Device\KeyboardClass0`.

Execute original bootmgfw.efi entry point

Hook Archpx64TransferTo64BitApplicationAsm

Transfer execution to OS loader using Archpx64TransferTo64BitApplicationAsm

Hook OslArchTransferToKernel

**Windows Boot Manager** (bootmgfw.efi)

**hooked** Archpx64TransferTo64BitApplicationAsm

# 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)

mpus Internacional
BERSEGURIDAD

UNIVERSIDAD CATÓLICA SAN ANTONIO
IN LIBERTATEM VOCATI
AÑO MCMXCVI

ENIIT INNOVA IT BUSINESS SCHOOL

# THANK YOU ☺

Rootkits PoCs & Rooted 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: