# Close Encounters of the Advanced Persistent Kind

Leveraging Rootkits for Post-Exploitation

IBM.

Valentina Palmiotti

IBM X-Force Adversary Services

**Security Researcher**

**Weird Machine Mechanic**

@chompie1337

Ruben Boonen

IBM X-Force Adversary Services

**Just a Windows Dev**

@FuzzySec

# Introduction

# Kernel Rootkits

A sliding scale of BYOVD capabilities

Old Money

**Turla, Equation, Lamberts, ProjectSauron,..**

New Money

**BitPaymer, Trickbot, RobbinHood, BlackByte,..**

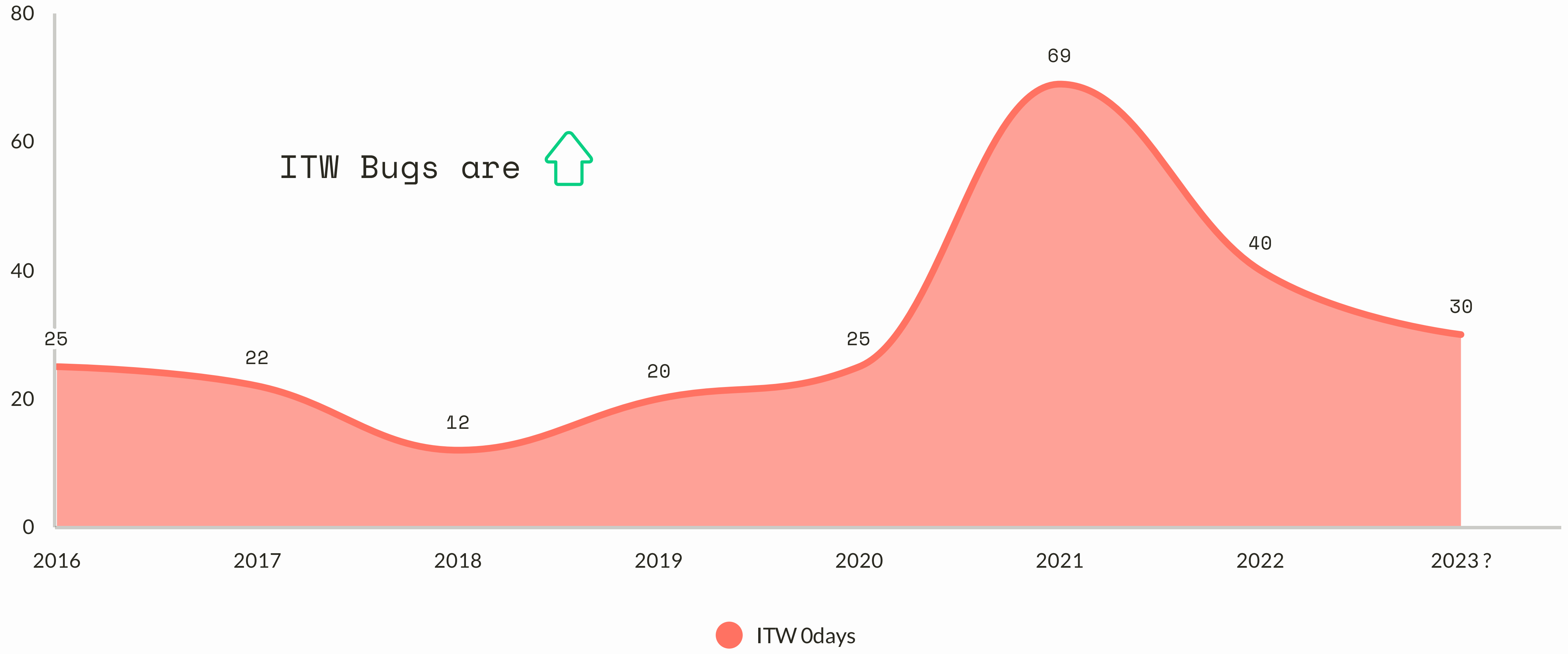
Degrees Of Exploitation

**Not all groups are created equal**

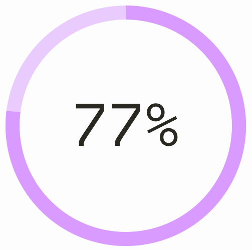
ITW 0day 2023

**30**

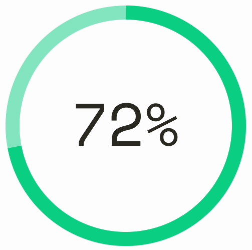
# Project Zero Trends



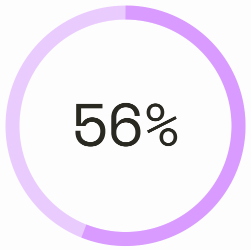
# Time-To-Patch (TTP)



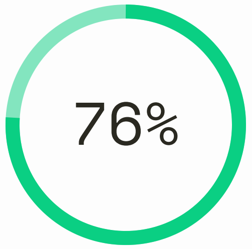
Not enough resources to keep up with the volume of patches



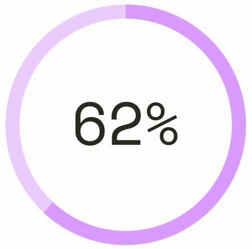
It is difficult to prioritize what needs to be patched



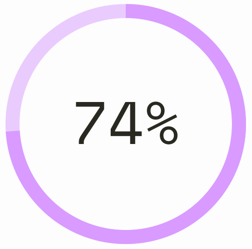
Emails & spread sheets are used to manage the process so things slip between the cracks



No common view of applications and assets across security and IT teams



We can't easily track whether vulnerabilities are being patched in a timely manner



Not able to take critical applications & systems off-line so they can be patched quickly

Human error

● High
**16 days**

● Medium/Low
**151 days**

*Ponemon Institute '18/'19, survey of 3000 IT security professionals*

Why operate in Ring0?

## Kernel vs User

### Mitigations & Restrictions

Some mitigations only exist in User Land

There are still actions that are disallowed by administrators in user space.

### Observability

User Land filesystem or memory activity may be observable from the Kernel

### EDR

EDR is generally specialized in User Land analysis not Kernel analysis

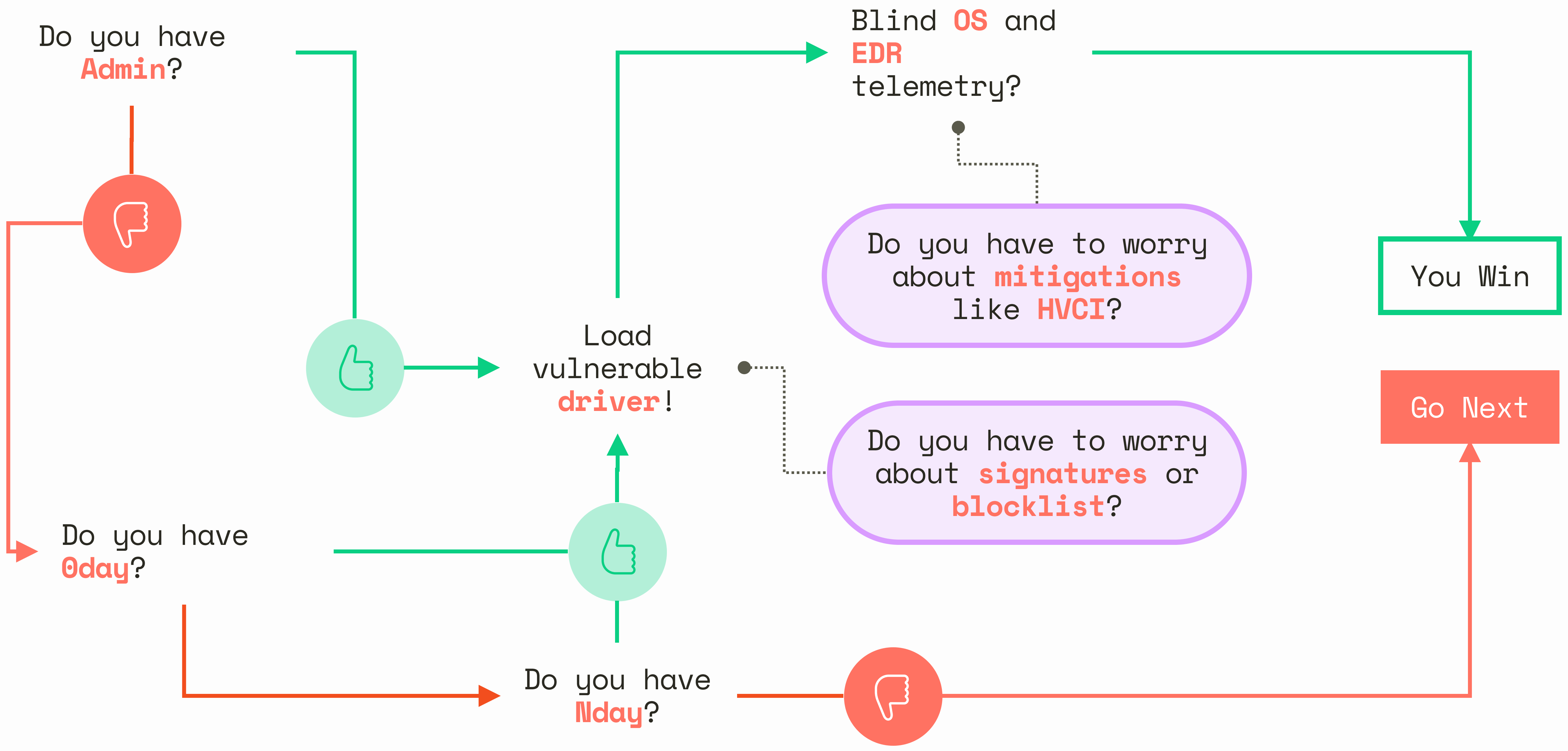
### Opportunity Cost

Kernel tradecraft may survive longer, this reduces the development efforts required to be effective operationally

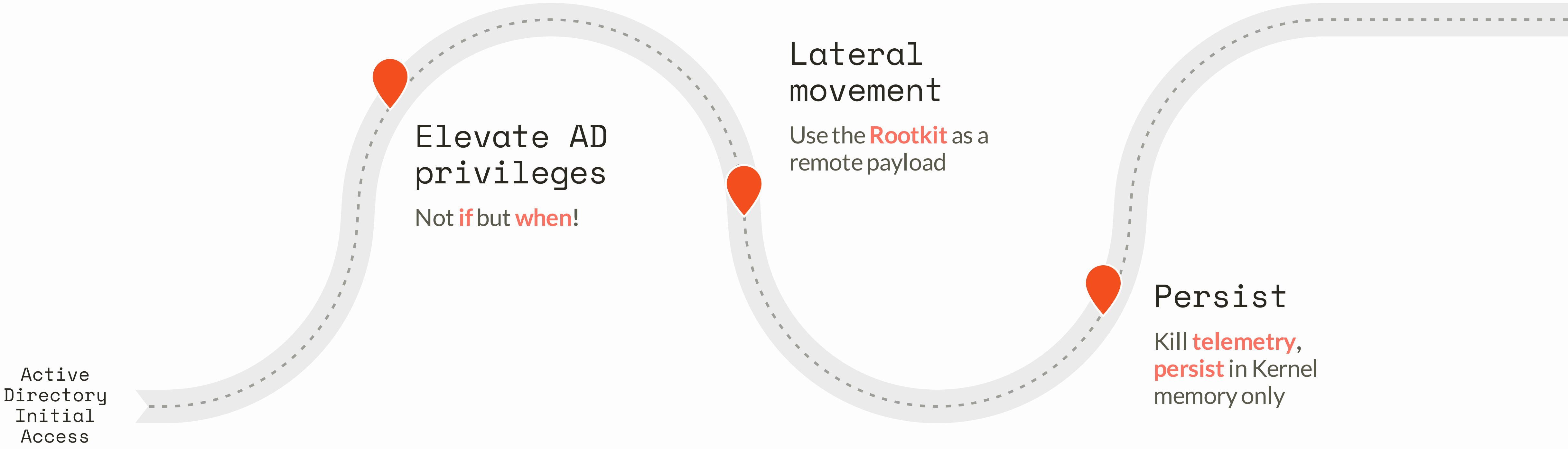
### Complexity

Kernel sensors are complex and often undocumented, this makes any potential detections brittle, such detections may also have an unacceptable performance cost

# Kernel full-chain lifecycle


Do you have Admin?
Load vulnerable driver!
Blind OS and EDR telemetry?
Do you have to worry about mitigations like HVCI?
You Win
Go Next
Do you have to worry about signatures or blocklist?
Do you have 0day?
Do you have Nday?

# Why care about Admin to Kernel?



### Elevate AD privileges

Not **if** but **when**!

### Lateral movement

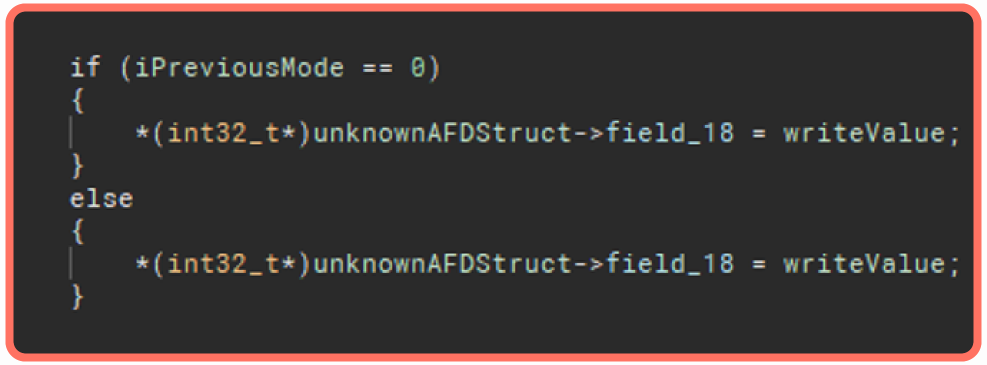
Use the **Rootkit** as a remote payload

### Persist

Kill **telemetry**, **persist** in Kernel memory only

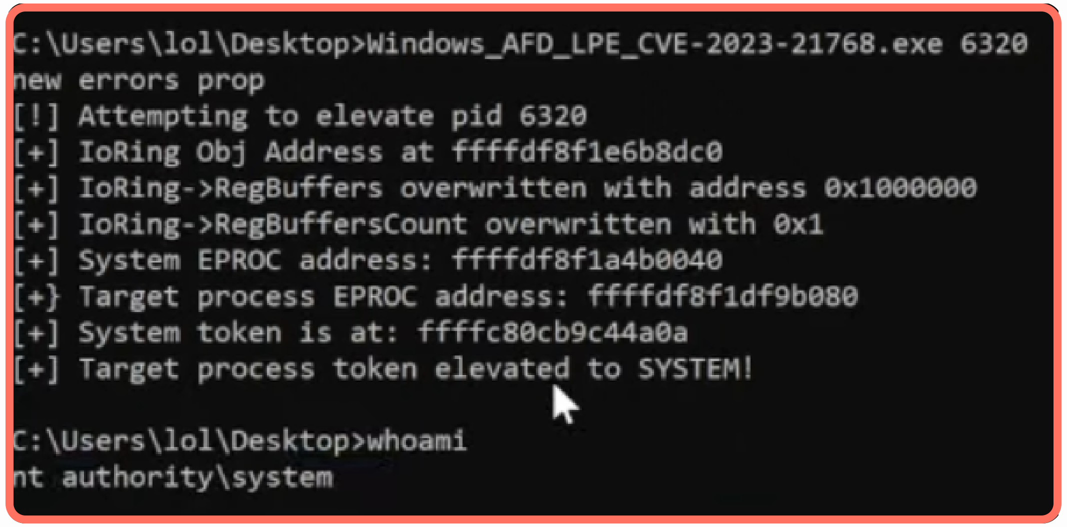
## Can I get some insufficient user-mode checks?

Ancillary Function Driver



```
if (iPreviousMode == 0)
{
    *(int32_t*)unknownAFDStruct->field_18 = writeValue;
}
else
{
    *(int32_t*)unknownAFDStruct->field_18 = writeValue;
}
```

Put an **IORING** on it!



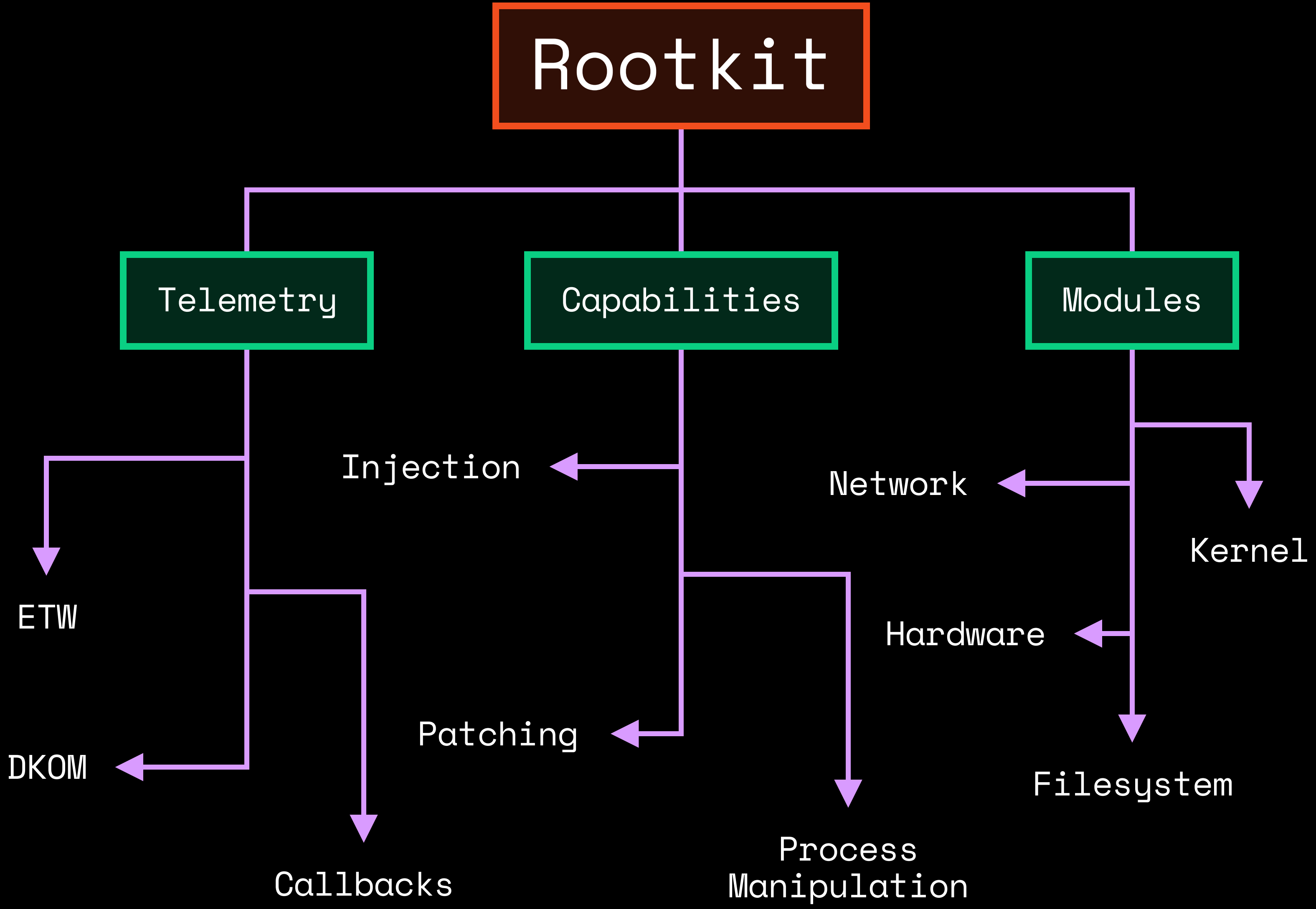
```
C:\Users\lol\Desktop>Windows_AFD_LPE_CVE-2023-21768.exe 6320
new errors prop
[!] Attempting to elevate pid 6320
[+] IoRing Obj Address at ffffdf8f1e6b8dc0
[+] IoRing->RegBuffers overwritten with address 0x1000000
[+] IoRing->RegBuffersCount overwritten with 0x1
[+] System EPROC address: ffffdf8f1a4b0040
[+} Target process EPROC address: ffffdf8f1df9b080
[+] System token is at: ffffc80cb9c44a0a
[+] Target process token elevated to SYSTEM!

C:\Users\lol\Desktop>whoami
nt authority\system
```

Case Study => **CVE-2023-21768**

24 hours of research and development for weaponized exploit

## Driving operations from Ring0

Shaping detections & exercising capabilities



Tradecraft

# Callbacks & ETW

- AV / EDR products need OS telemetry
  - Kernel mitigations have **narrowed vendor capabilities** to implement custom collection routines
  - Microsoft provides a number of **built-in mechanisms** to collect information on endpoint activity

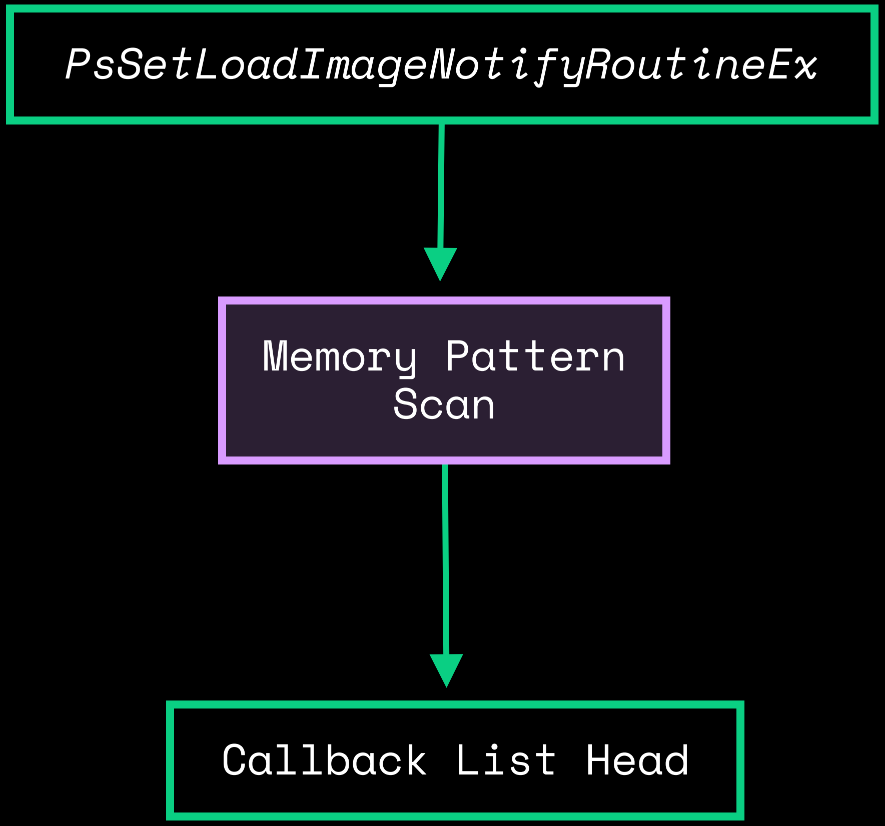
- Surfacing alerts
  - **Native** and **3rd party** products ingest telemetry and use **magic heuristics** to **surface alerts**

- Subverting alerts
  - By manipulating data structures in Kernel memory we can hide, reduce or eliminate specific telemetry
  - There are **OpSec considerations**, better to reduce or redirect telemetry than to eliminate

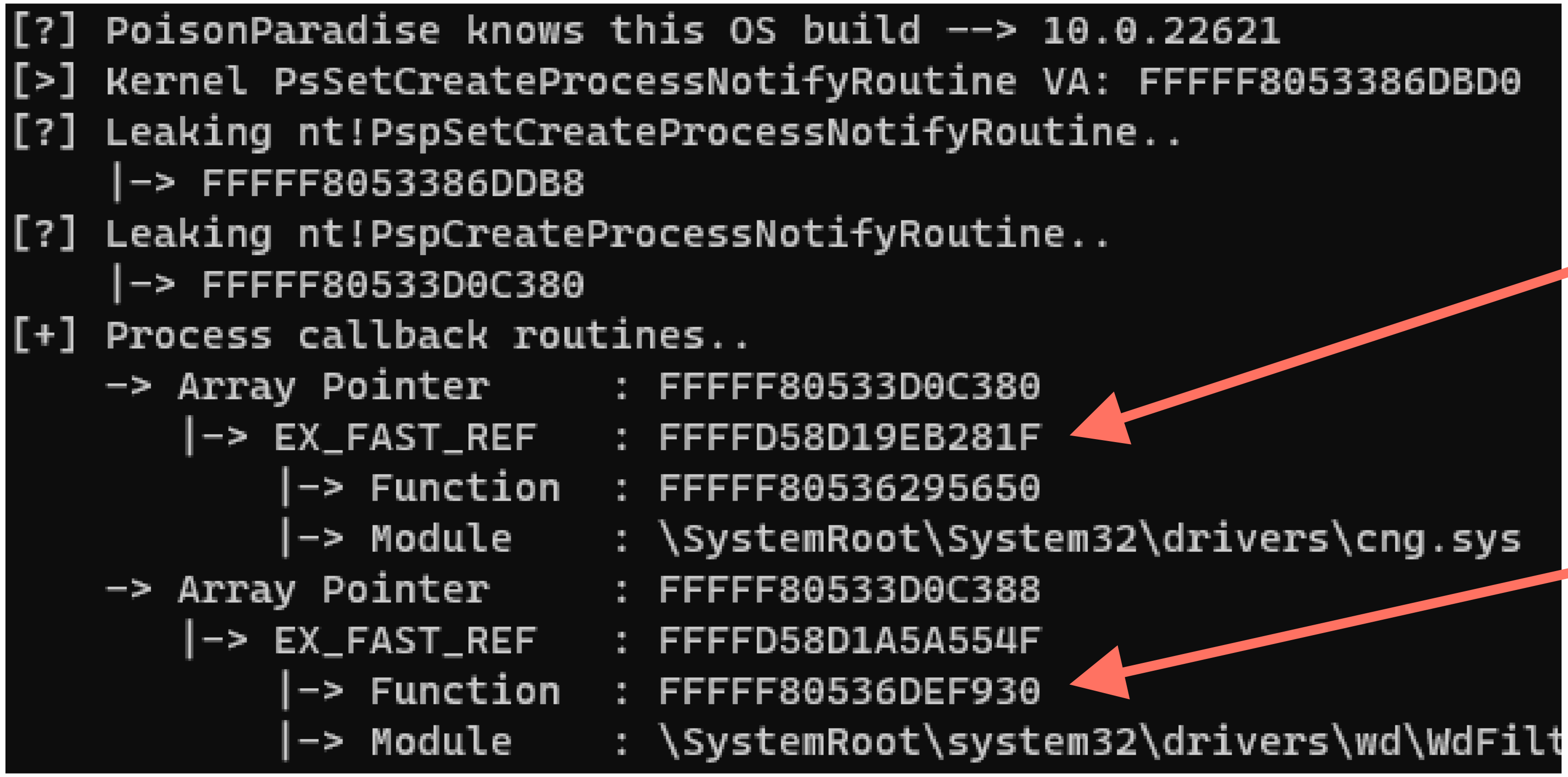
# Kernel Callbacks

Registering for event notifications

- Image, Process, Thread
  *PsSet.....NotifyRoutine(Ex)*
- Registry
  *ObGetObjectType -> CallbackListHead*
- Object
  *CmUnRegisterCallback -> CallbackListHead*



# Callback Tampering



```
[?] PoisonParadise knows this OS build --> 10.0.22621
[>] Kernel PsSetCreateProcessNotifyRoutine VA: FFFFF8053386DBD0
[?] Leaking nt!PspSetCreateProcessNotifyRoutine..
    |-> FFFFF8053386DDB8
[?] Leaking nt!PspCreateProcessNotifyRoutine..
    |-> FFFFF80533D0C380
[+] Process callback routines..
    -> Array Pointer      : FFFFF80533D0C380
       |-> EX_FAST_REF    : FFFFD58D19EB281F
           |-> Function   : FFFFF80536295650
           |-> Module     : \SystemRoot\System32\drivers\cng.sys
    -> Array Pointer      : FFFFF80533D0C388
       |-> EX_FAST_REF    : FFFFD58D1A5A554F
           |-> Function   : FFFFF80536DEF930
           |-> Module     : \SystemRoot\system32\drivers\wd\WdFilt
```

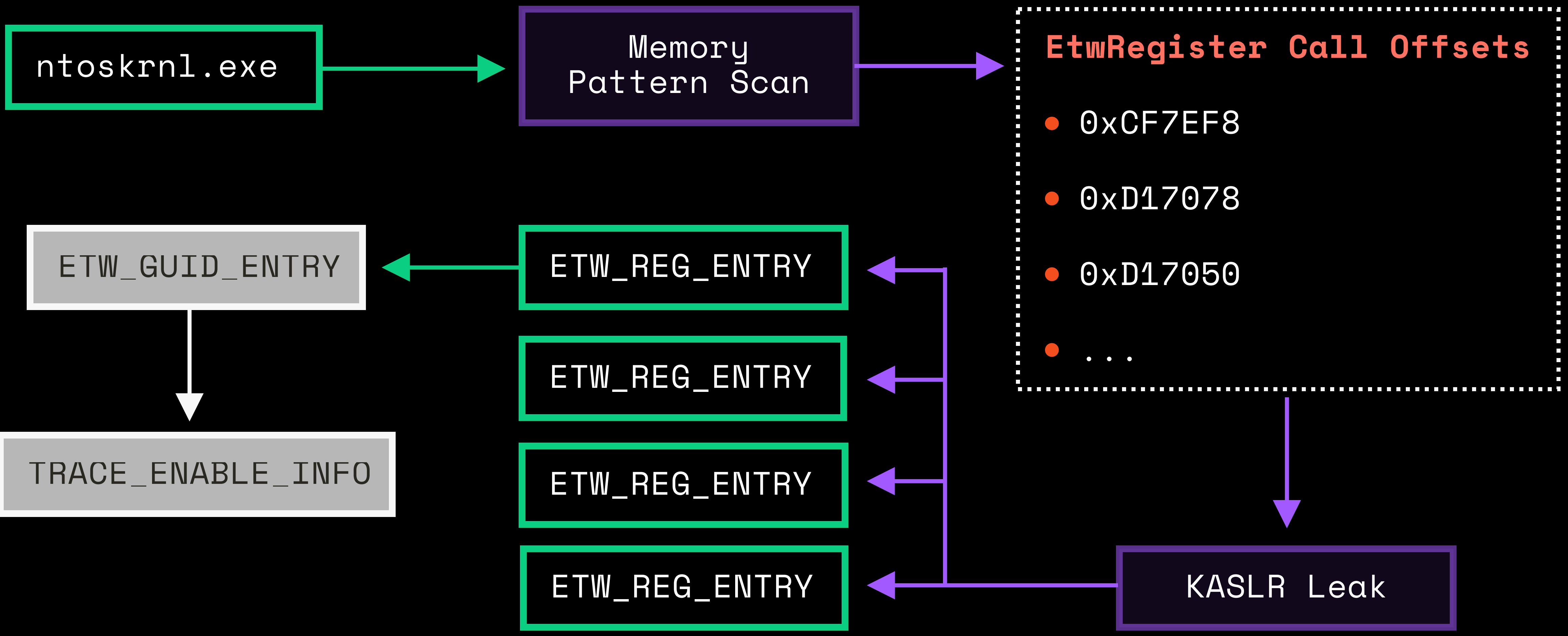
Object Pointers

Function Pointers

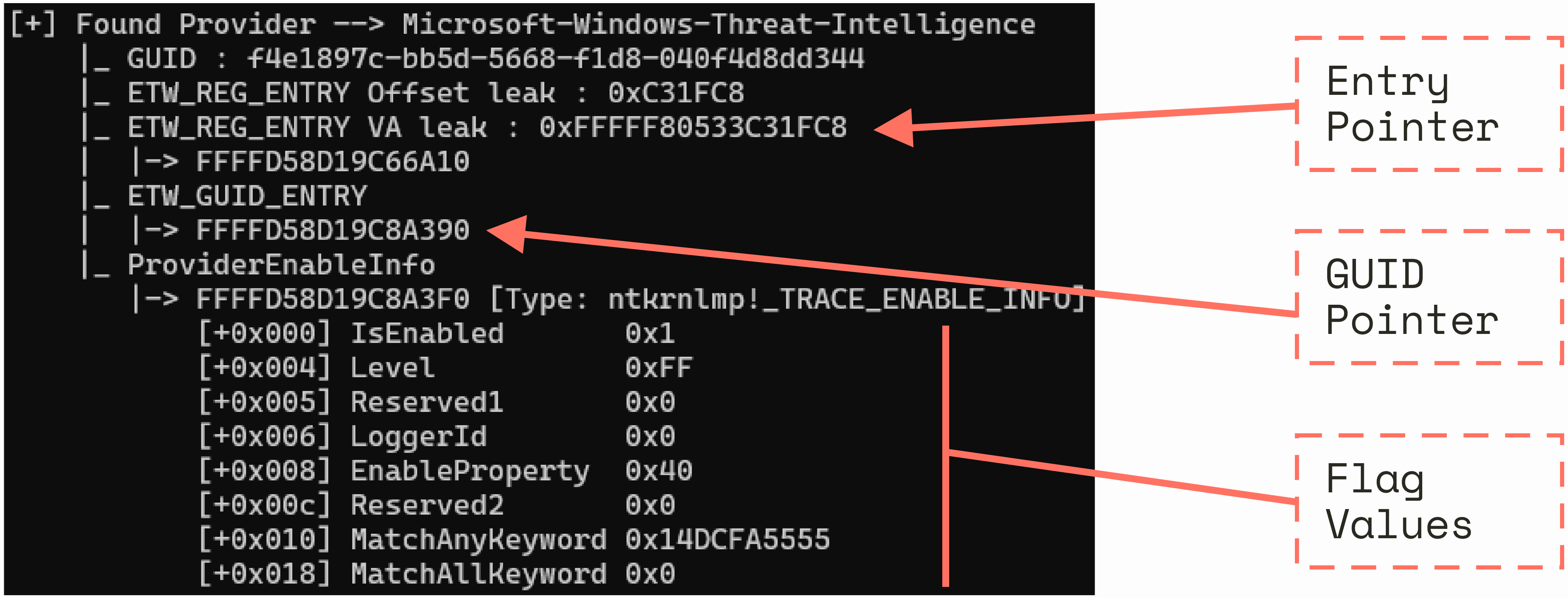
**REDIRECT DETECTION EVENTS** TO A **DIFFERENT MODULE** OR A **DIFFERENT FUNCTION** OR **MANIPULATE THE ARRAY** ITSELF!

# Kernel ETW

Registering for event notifications



# ETW Tampering



```
[+] Found Provider --> Microsoft-Windows-Threat-Intelligence
    |_ GUID : f4e1897c-bb5d-5668-f1d8-040f4d8dd344
    |_ ETW_REG_ENTRY Offset leak : 0xC31FC8
    |_ ETW_REG_ENTRY VA leak : 0xFFFFF80533C31FC8
    |   |-> FFFFD58D19C66A10
    |_ ETW_GUID_ENTRY
    |   |-> FFFFD58D19C8A390
    |_ ProviderEnableInfo
        |-> FFFFD58D19C8A3F0 [Type: ntkrnlmp!_TRACE_ENABLE_INFO]
            [+0x000] IsEnabled        0x1
            [+0x004] Level            0xFF
            [+0x005] Reserved1        0x0
            [+0x006] LoggerId         0x0
            [+0x008] EnableProperty   0x40
            [+0x00c] Reserved2        0x0
            [+0x010] MatchAnyKeyword 0x14DCFA5555
            [+0x018] MatchAllKeyword 0x0
```

**KILL DETECTION EVENTS** BY ZEROING OUT **THE REGISTRATION**,
**THE GUID** OR BY **MODIFYING THE CAPTURE FILTER**!

## Demo

(With **0day**!)

## X-Force disclosure policy

- The Adversary Services team at X-Force has vulnerability research as one of its operating objectives
- X-Force takes responsible disclosure seriously. We follow a defined internal review processed followed by a coordinated disclosure to vendors
- We do not have details we can present on this vulnerability yet
- A blog post will be released once a patch is available

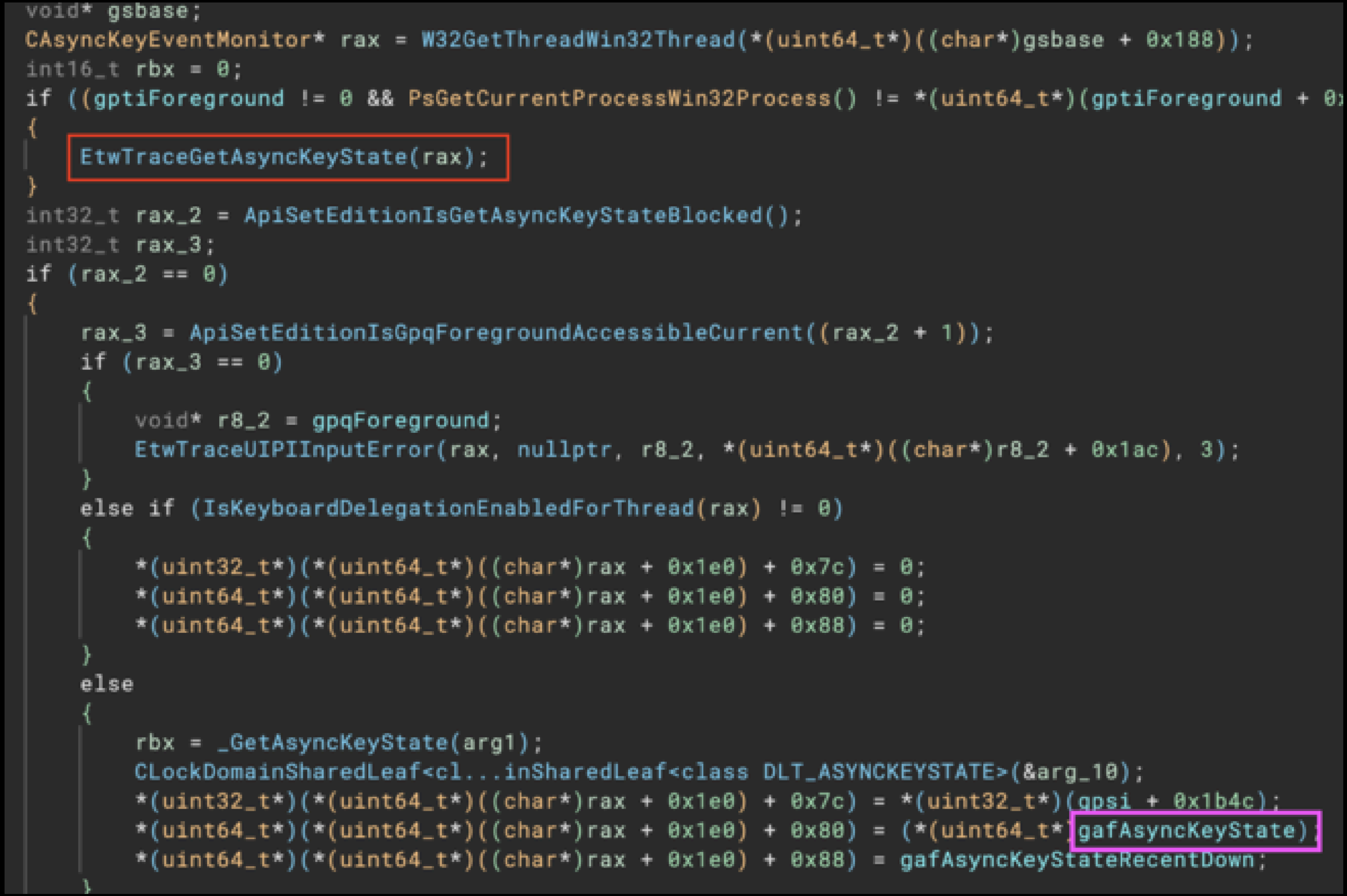
# Keylogging

Current known methods of keylogging can all be detected

- **User Mode** - Low Level Hooks
  - Detected by querying installed **keyboard hooks**
- **User Mode** - Polling keystrokes via NT system call (GetAsyncKeyState)
  - Detected by monitoring **WinAPI functions** or **system calls** (via hooking, call stack unwinding, ETW, etc.)
- **Kernel Mode** - Keyboard Filter Driver
  - Detected by enumerating **keyboard devices** and devices attached to them - can't be hidden otherwise they are unlinked from the I/O IRP stack

# Keylogging

Reverse engineering a Ring3 implementation



```
void* gsbase;
CAsyncKeyEventMonitor* rax = W32GetThreadWin32Thread(*(uint64_t*)((char*)gsbase + 0x188));
int16_t rbx = 0;
if ((gptiForeground != 0 && PsGetCurrentProcessWin32Process() != *(uint64_t*)(gptiForeground + 0x
{
    EtwTraceGetAsyncKeyState(rax);
}
int32_t rax_2 = ApiSetEditionIsGetAsyncKeyStateBlocked();
int32_t rax_3;
if (rax_2 == 0)
{
    rax_3 = ApiSetEditionIsGpqForegroundAccessibleCurrent((rax_2 + 1));
    if (rax_3 == 0)
    {
        void* r8_2 = gpqForeground;
        EtwTraceUIPIInputError(rax, nullptr, r8_2, *(uint64_t*)((char*)r8_2 + 0x1ac), 3);
    }
    else if (IsKeyboardDelegationEnabledForThread(rax) != 0)
    {
        *(uint32_t*)(*(uint64_t*)((char*)rax + 0x1e0) + 0x7c) = 0;
        *(uint64_t*)(*(uint64_t*)((char*)rax + 0x1e0) + 0x80) = 0;
        *(uint64_t*)(*(uint64_t*)((char*)rax + 0x1e0) + 0x88) = 0;
    }
    else
    {
        rbx = _GetAsyncKeyState(arg1);
        CLockDomainSharedLeaf<cl...inSharedLeaf<class DLT_ASYNCKEYSTATE>(&arg_10);
        *(uint32_t*)(*(uint64_t*)((char*)rax + 0x1e0) + 0x7c) = *(uint32_t*)(gpsi + 0x1b4c);
        *(uint64_t*)(*(uint64_t*)((char*)rax + 0x1e0) + 0x80) = (*(uint64_t*)gafAsyncKeyState);
        *(uint64_t*)(*(uint64_t*)((char*)rax + 0x1e0) + 0x88) = gafAsyncKeyStateRecentDown;
    }
```

$$\begin{tikzcd}\boxed{\text{NtUserGetAsyncKeyState}} \arrow[d] \\ \boxed{\text{gafAsyncKeyState}}\end{tikzcd}$$

global af
(sort of)

# Keylogging

An undetectable method, simple to implement

1 | Locate **gafAsyncKeyState**

Exported by win32kbase on Windows 10, stored in win32ksgd -> gSessionGlobalSlots on Windows 11

2 | **win32kbase**/**win32ksgd** is a session driver, must be attached to the process running in the correct session

3 | Map the physical page of the keystate array to a usermode virtual address

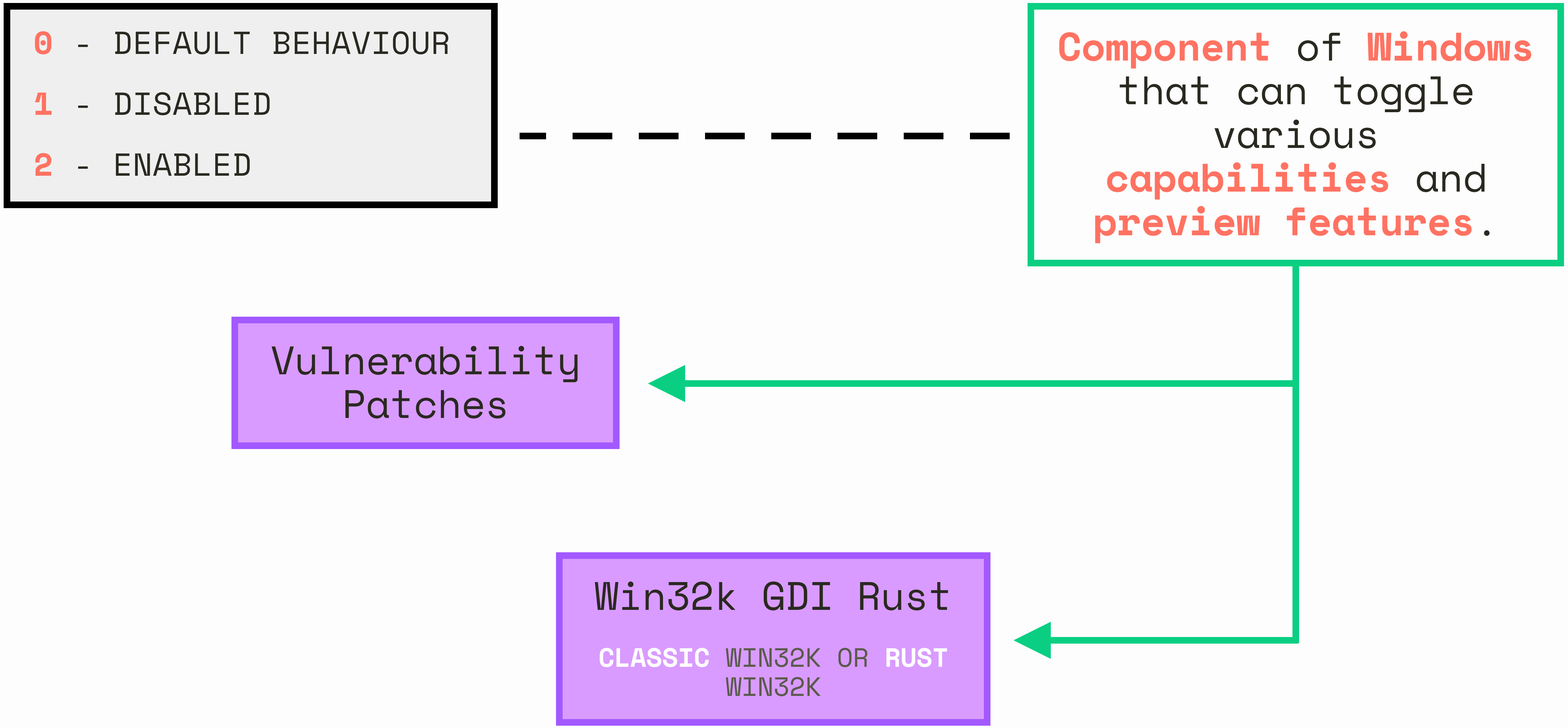
Create a MDL -> MmProbeAndLockPages -> MmMapLockedPagesSpecifyCache

4 | Poll keystrokes in Ring3 without calling into the kernel

Avoids costly Kernel context switches

Almost impossible to detect

# Feature Flags



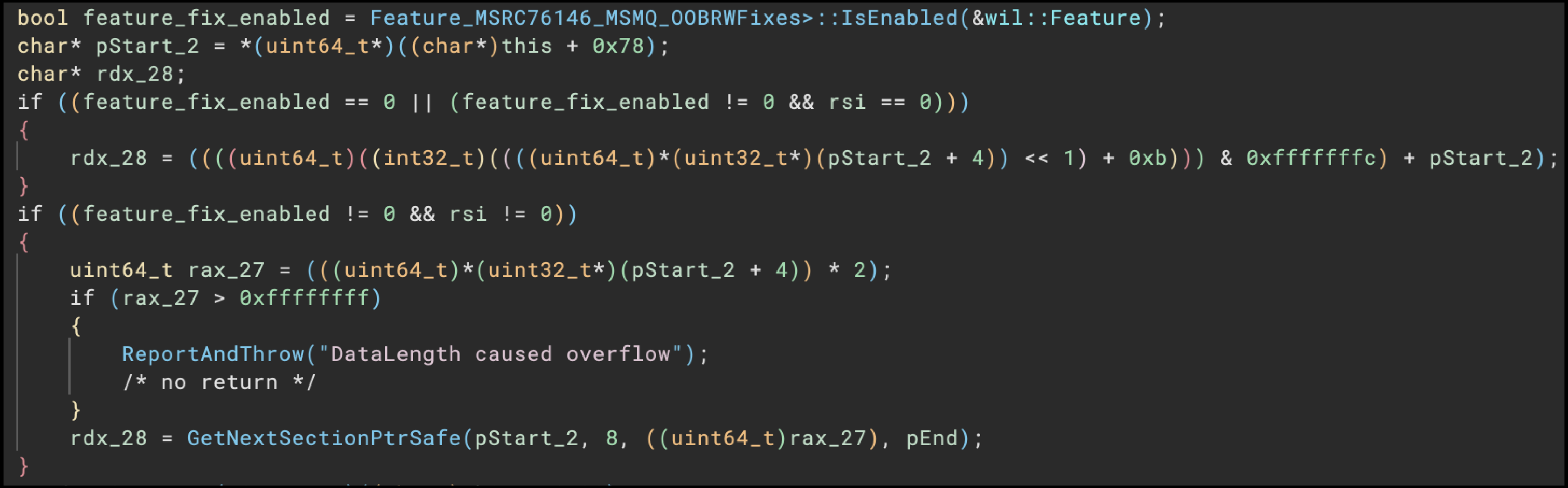
0 - DEFAULT BEHAVIOUR

1 - DISABLED

2 - ENABLED

**Component** of **Windows** that can toggle various **capabilities** and **preview features**.

# Implementation



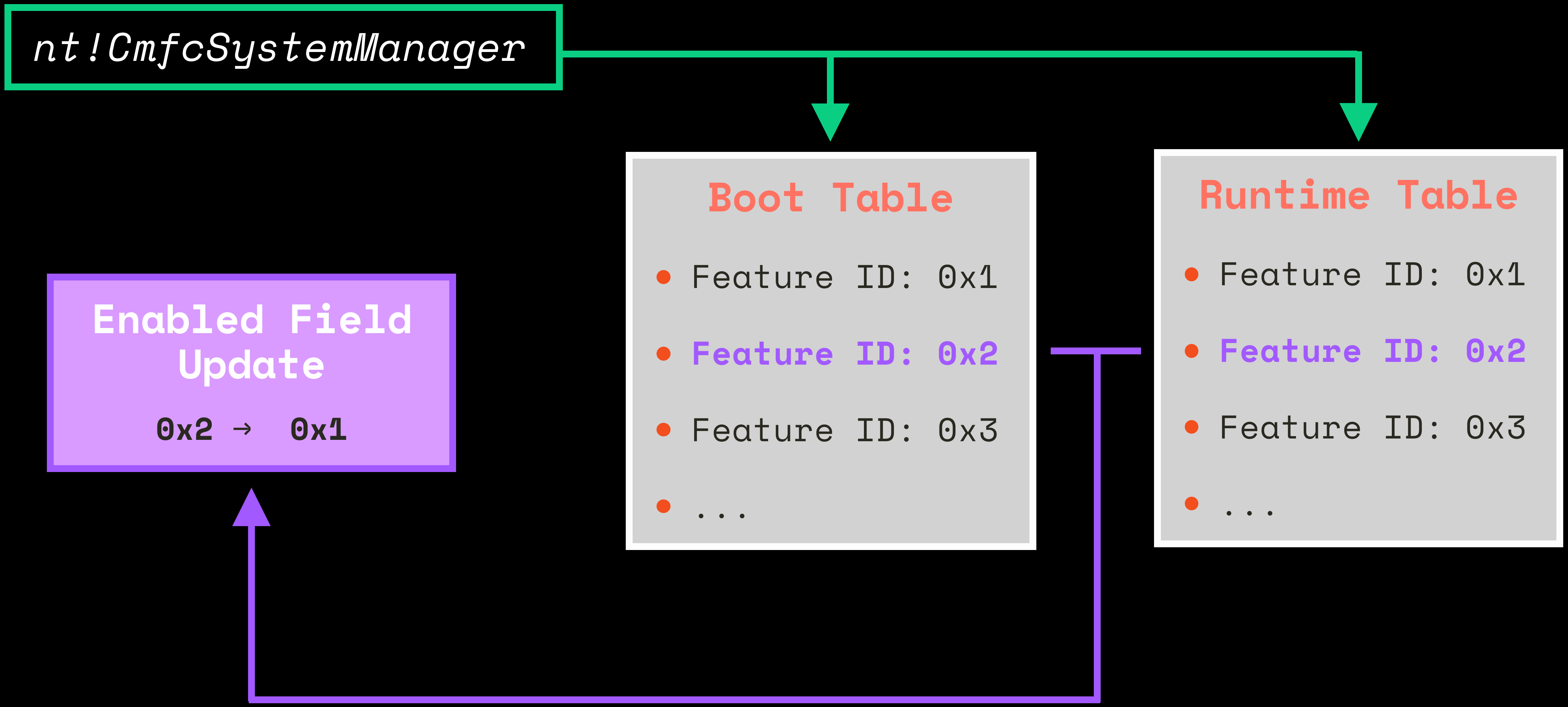
```
bool feature_fix_enabled = Feature_MSRC76146_MSMQ_OOBRWFixes>::IsEnabled(&wil::Feature);
char* pStart_2 = *(uint64_t*)((char*)this + 0x78);
char* rdx_28;
if ((feature_fix_enabled == 0 || (feature_fix_enabled != 0 && rsi == 0)))
{
    rdx_28 = ((((uint64_t)((int32_t)((((uint64_t)*(uint32_t*)(pStart_2 + 4)) << 1) + 0xb))) & 0xfffffffc) + pStart_2);
}
if ((feature_fix_enabled != 0 && rsi != 0))
{
    uint64_t rax_27 = (((uint64_t)*(uint32_t*)(pStart_2 + 4)) * 2);
    if (rax_27 > 0xffffffff)
    {
        ReportAndThrow("DataLength caused overflow");
        /* no return */
    }
    rdx_28 = GetNextSectionPtrSafe(pStart_2, 8, ((uint64_t)rax_27), pEnd);
}
```

# Feature Flag Manipulation

- Can be set in **User Mode** using undocumented WinAPI's
  - RtlSetFeatureConfigurations
  - Requires elevated access
  - **Restrictions** on what features can be toggled (Security and Image Override features)
- ViVe, mach2 - open source tools to manipulate feature configurations using WinAPI

- Restrictions can be bypassed in **Kernel Mode** using **DKOM**
  - Overwrite enabled flag value in feature table
  - Toggle patches and security features
  - Use of features is increasing over time and is all over Windows
- Take Care
  - Changing global features can cause **unexpected behaviour** in applications
  - Some applications cache features configurations when accessed, and require a **refresh** or application **restart** to take effect

# Feature Flag Manipulation



# Network Filtering

I don't always shape traffic, but when I do, I do it in the Kernel!

- **WinDivert** is opensource and offers robust capabilities
  - Used in enterprise projects like **Suricata**
- Network & Socket related manipulations **not visible in Ring3**
  - Rules based traffic shaping
  - Filter on port, source, destination, PID, content
  - Drop, redirect, inject
- Many possible **use-cases**!
  - Drop/intercept/manipulate EDR cloud telemetry
  - Traffic relay (SMB anyone?)
  - Covert persistence
- What about the **driver**?
  - Patch Kernel CI
  - Reflectively load the driver
  - Sign the driver

## Demo

# Userland Puppeteering

There are **Kernel** to **Userland** operations which may be useful in a variety of situations

- Handle duplication
  - Process and object handles
- Kill/Start a process
- Thread suspension
- Process adjustment
  - Token substitution
  - Token permission change
  - Protected status change
- Shellcode injection

## Virtualization Based Security (VBS)

HVCI, KDP, HYPERGUARD
WHAT IT IS AND WHAT IT ISN'T

## Mitigiations for Kernel Compromise

Specifically intended to weaken kernel exploits and rootkits

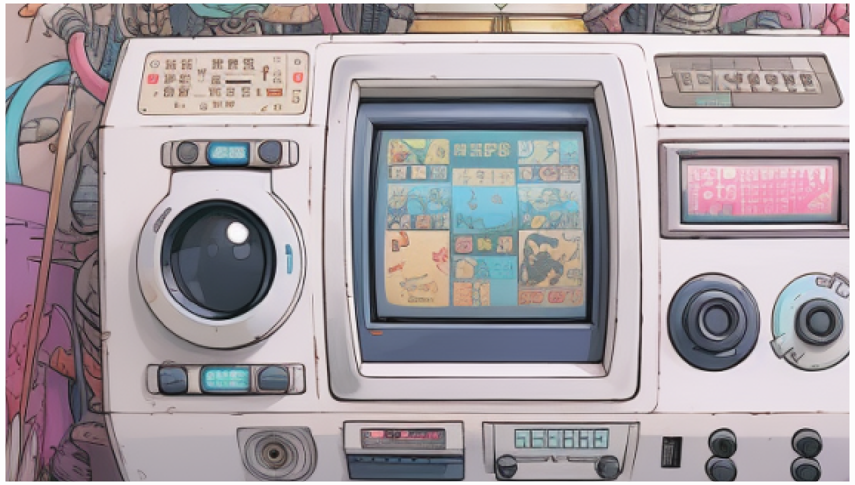
- **Virtualization Based Security (VBS)**

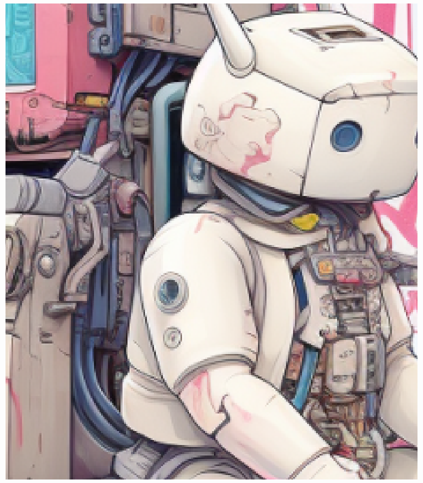
  Hardware virtualization and the Windows hypervisor to create an isolated virtual environment that becomes the root of trust of the OS that assumes the kernel can be compromised.

  - Kernel Data Protection (**KDP**)

    Protects important kernel structures
  - Hypervisor-Protected Code Integrity (**HVCI**)

    Prevents the execution of unsigned code in the kernel
  - Secure Kernel Guard (**HyperGuard**)
    Patch guard but in the trusted hypervisor





## Virtualization Based Security

How much does the Hypervisor help to prevent these attacks?

(non-exhaustive listing)

| | Default | VBS (KDP, HVCI, ...) |
|---|---|---|
| DKOM/Data Only Techniques | ✓ | ✓ |
| Bypassing Driver Signing Enforcement | ✓ | ✗ |
| Loading (signed) Kernel modules with RWX sections | ✓ | ✗ |
| Calling Arbitrary Kernel Functions | ✓ | ✓ |
| Registering Kernel Callbacks | ✓ | ✓ |
| PTE manipulation, executable bit | ✓ | ✗ |
| PTE manipulation, R/W bit | ✓ | ✓ |

# VBS Bypasses

Modern Techniques to Bypass Virtualization Based Security

- **Thread Context Manipulation**

  Putting a thread into alertable state and modifying its context to resume execution at a chosen address.

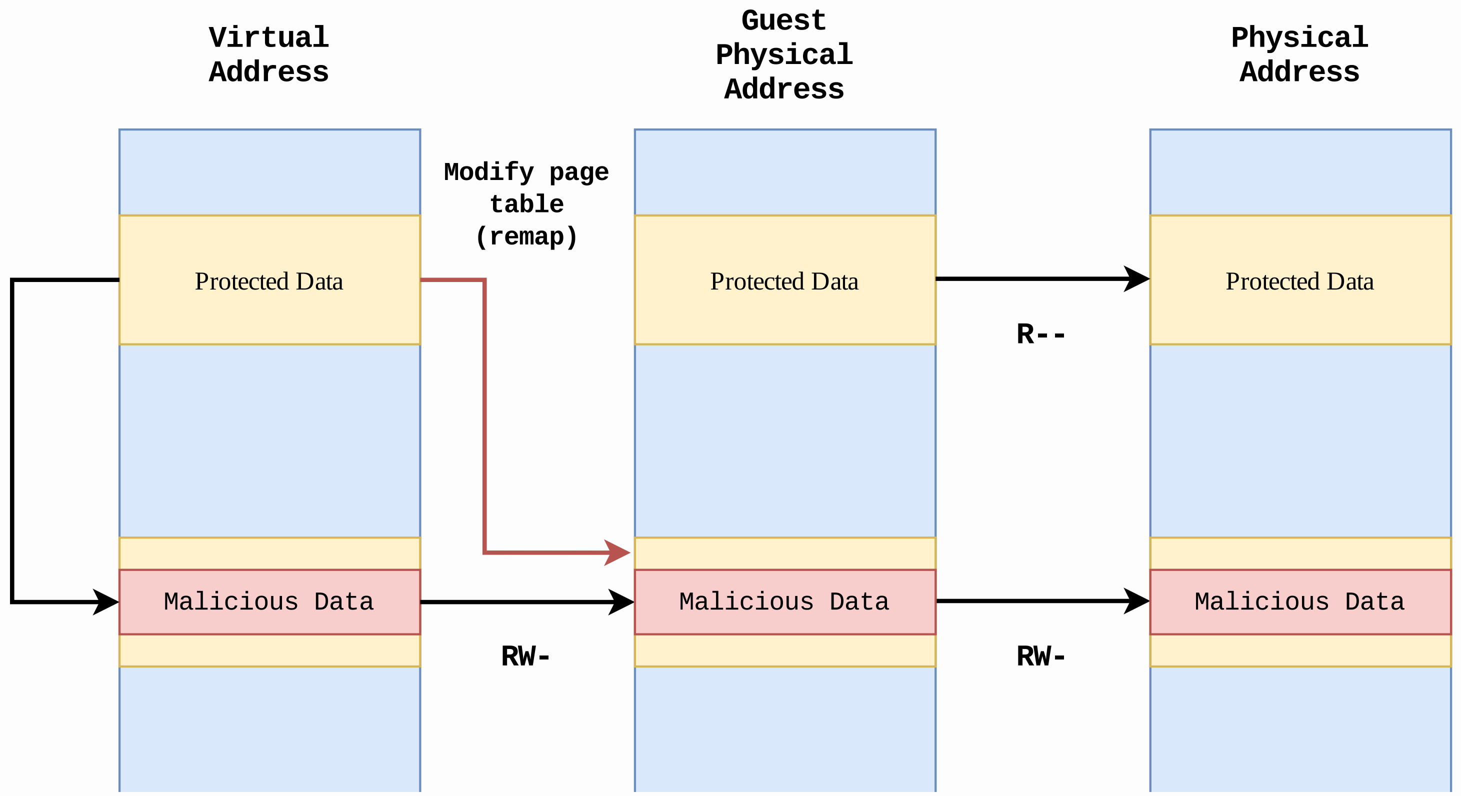
- **Page Swapping attacks**

  *KDP does NOT protect how the virtual address that maps a protected region is TRANSLATED.*

  Any protected region can be remapped.

# Catch All VBS Bypass

Page Swapping



## Example

- Change Page Table for user space process SSDT to point to a writable physical page
- Modify SSDT system call pointer to arbitrary kernel address. Parameters all controllable from user space (remember, Windows doesn't have SMAP!)
- Success! No need to load a driver. (Except if you absolutely *need* chained calls)

## Hardware Mitigations

Hardware enforced mitigations can block these attacks

- Kernel Code Enforcement Technology (**kCET**)
  - Intel processor feature, adds a second stack for return address integrity
  - Kills ROP attacks (like thread context manipulation)

- Intel **VT-rp**
  - Hypervisor Linear Address Translation (HLAT) - sensitive data pages can't be remapped
  - Blocks page swap/remapping attacks
  - **Not implemented by Windows! (yet)**

The most powerful mitigations **require specific hardware** and are **not enabled by default -** or **not even implemented**!

# Hardening Advice

VBS configuration is unnecessarily hard! Does anyone understand this really?

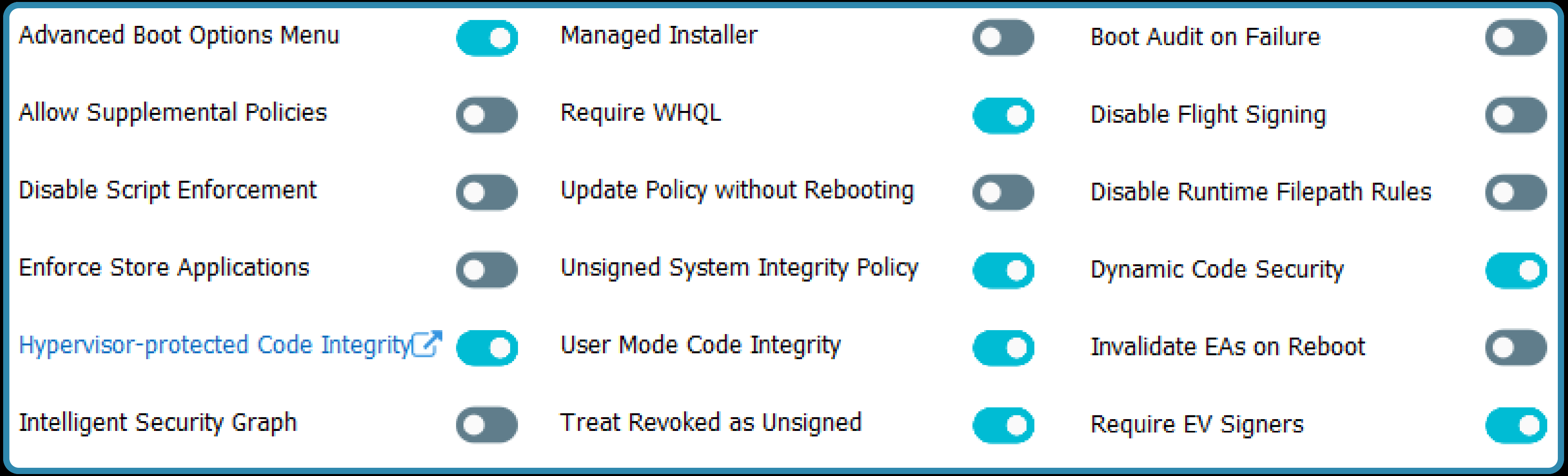
- Configuration
  - Create a policy in the **Windows Defender Application Control** (WDAC) Wizard
  - Customize **Driver Blocklist**
  - In Group Policy, enable **VBS** and **WDAC**
  - Windows Security -> **Core Isolation** & **Memory Integrity**
  - Configure the **file-path** in the **WDAC policy setting**
  - Force Group Policy updates to synchronize
  - Reboot

- **Not** a **user-friendly** experience, the policy wizard is however an improvement
  - Pre-Wizard process involves **PowerShell** & **XML**
  - Even harder for **home users**, why though?

WDAC Wizard

# Hardening Configuration

- **Wizard** - https://webapp-wdac-wizard.azurewebsites.net/
- **GPO** - https://learn.microsoft.com/en-us/windows/security/hardware-security/enable-virtualization-based-protection-of-code-integrity



| | | | | | |
|---|---|---|---|---|---|
| Advanced Boot Options Menu | [✓] | Managed Installer | [ ] | Boot Audit on Failure | [ ] |
| Allow Supplemental Policies | [ ] | Require WHQL | [✓] | Disable Flight Signing | [ ] |
| Disable Script Enforcement | [ ] | Update Policy without Rebooting | [ ] | Disable Runtime Filepath Rules | [ ] |
| Enforce Store Applications | [ ] | Unsigned System Integrity Policy | [✓] | Dynamic Code Security | [✓] |
| Hypervisor-protected Code Integrity | [✓] | User Mode Code Integrity | [✓] | Invalidate EAs on Reboot | [ ] |
| Intelligent Security Graph | [ ] | Treat Revoked as Unsigned | [✓] | Require EV Signers | [✓] |

# References

- https://securityintelligence.com/posts/patch-tuesday-exploit-wednesday-pwning-windows-ancillary-function-driver-winsock/
- https://securityintelligence.com/posts/direct-kernel-object-manipulation-attacks-etw-providers/
- https://googleprojectzero.blogspot.com/p/0day.html
- https://www.virusbulletin.com/uploads/pdf/conference/vb2022/papers/VB2022-Lazarus-and-BYOVD-evil-to-the-Windows-core.pdf
- https://i.blackhat.com/EU-21/Wednesday/EU-21-Teodorescu-Veni-No-Vidi-No-Vici-Attacks-On-ETW-Blind-EDRs.pdf
- https://windows-internals.com/one-i-o-ring-to-rule-them-all-a-full-read-write-exploit-primitive-on-windows-11/
- https://connormcgarr.github.io/hvci/
- https://www.servicenow.com/content/dam/servicenow-assets/public/en-us/doc-type/resource-center/analyst-report/ponemon-state-of-vulnerability-response.pdf
- https://reqrypt.org/windivert.html
- https://datafarm-cybersecurity.medium.com/code-execution-against-windows-hvci-f617570e9df0
- https://tandasat.github.io/blog/2023/07/05/intel-vt-rp-part-1.html
- https://github.com/riverar/mach2
- https://github.com/thebookisclosed/ViVe/tree/f9a6fbc4d763665eef521273b9e4f2b3242b1d82

## Questions?