


Malware: Malware Analysis, Malware Development, ransomware,
APT/Cybercrime, Ransomware, Reverse Engineering

Ring0 Rootkit — 在 Windows Kernel 與病毒共存

Zeze



Zeze

- HITCON Speaker and Staff
- Member of NTU DCNS Lab
- TeamT5 Intern
- Member of BambooFox,  CTF Team
- Windows Security Enthusiast

Outline

01

Background

About antivirus

02

PatchGuard

Detect kernel patch

03

ObRegisterCallbacks

Monitor thread,
process, and desktop
handle operations

04

Infinity Hook

Use kernel hook
bypassing PatchGuard
to implement rootkit

05

KDU

Load drivers without
signature

06

Coexist With Virus

Inject into antivirus
- TamperAV



Kingsoft Antivirus



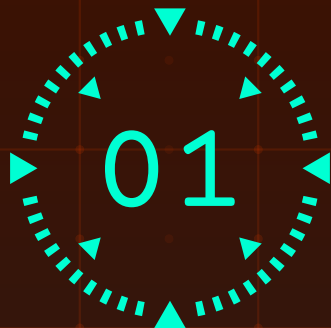
360 Total Security



Jiangmin Antivirus

Target Antivirus

The PoC is tested on Kingsoft Antivirus, 360 Total Security, and Jiangmin Antivirus in Windows 11 21H2.



Background

About antivirus



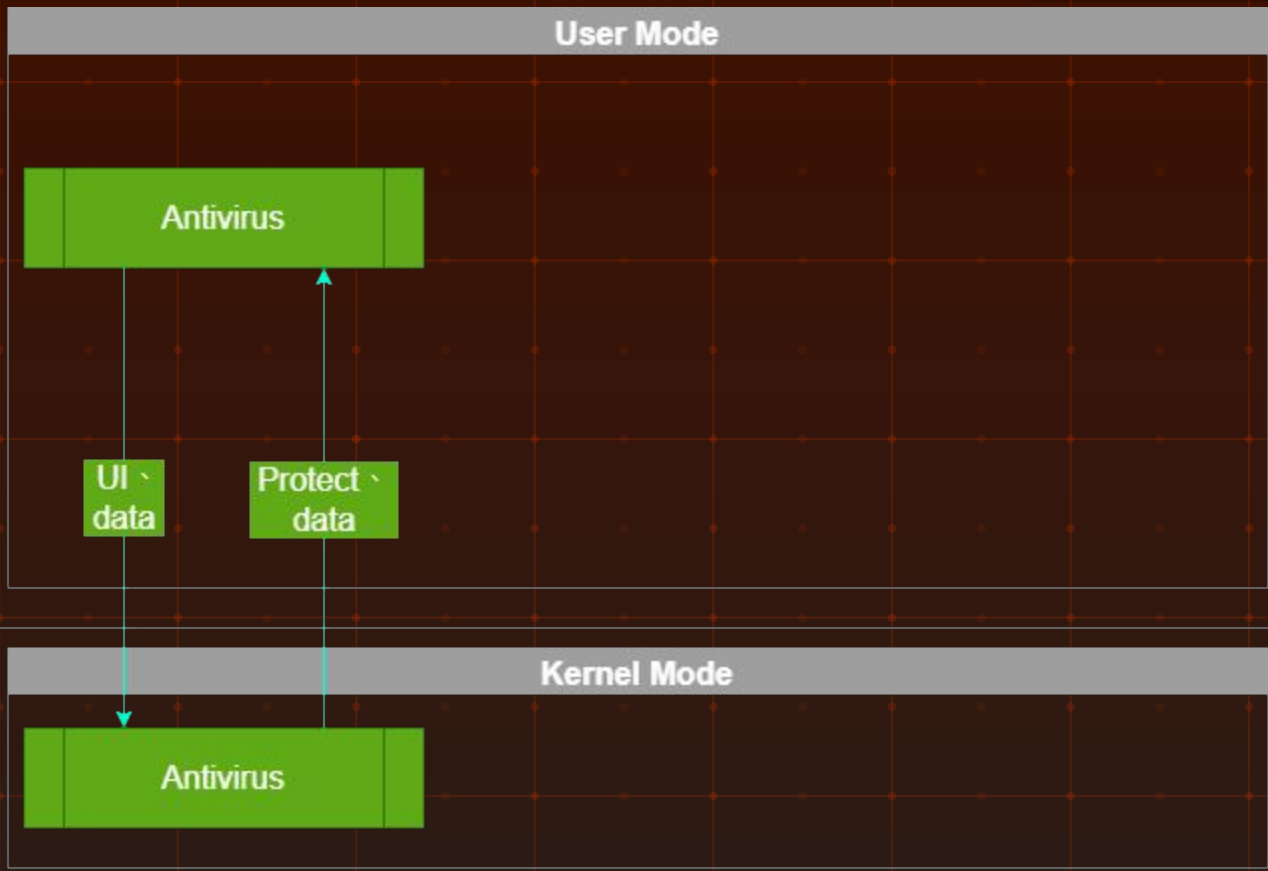
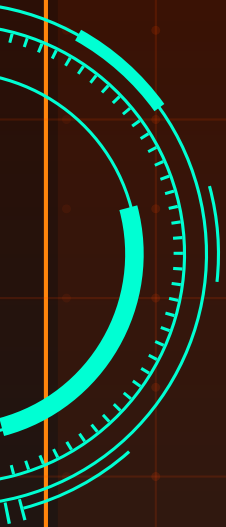
Antivirus

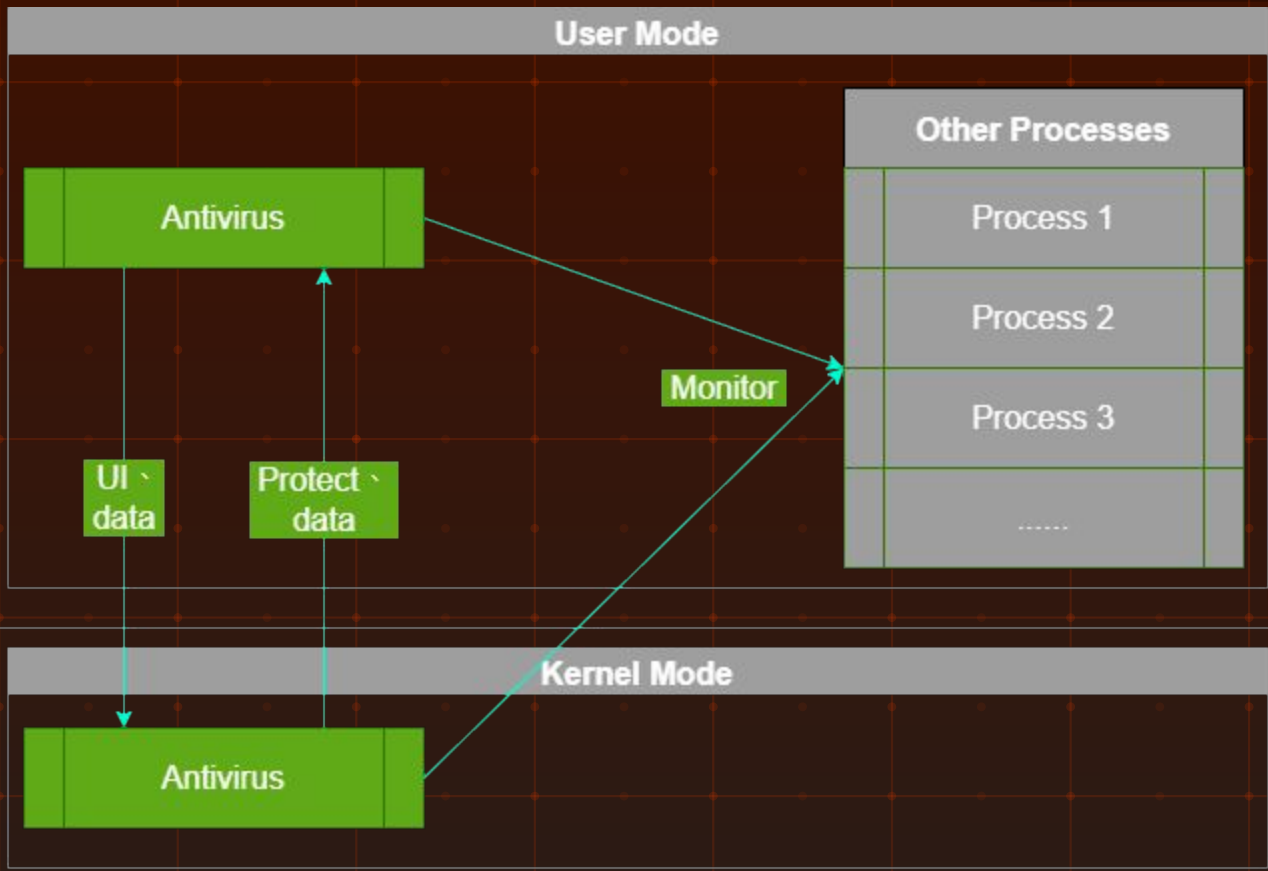
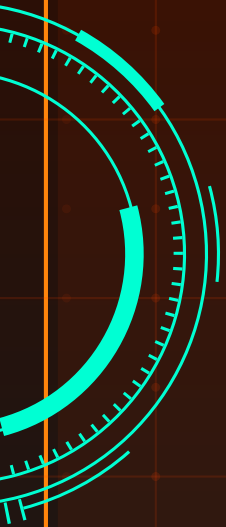
- Isolate malicious files
- Block suspicious operations
- anti-ransomware
-

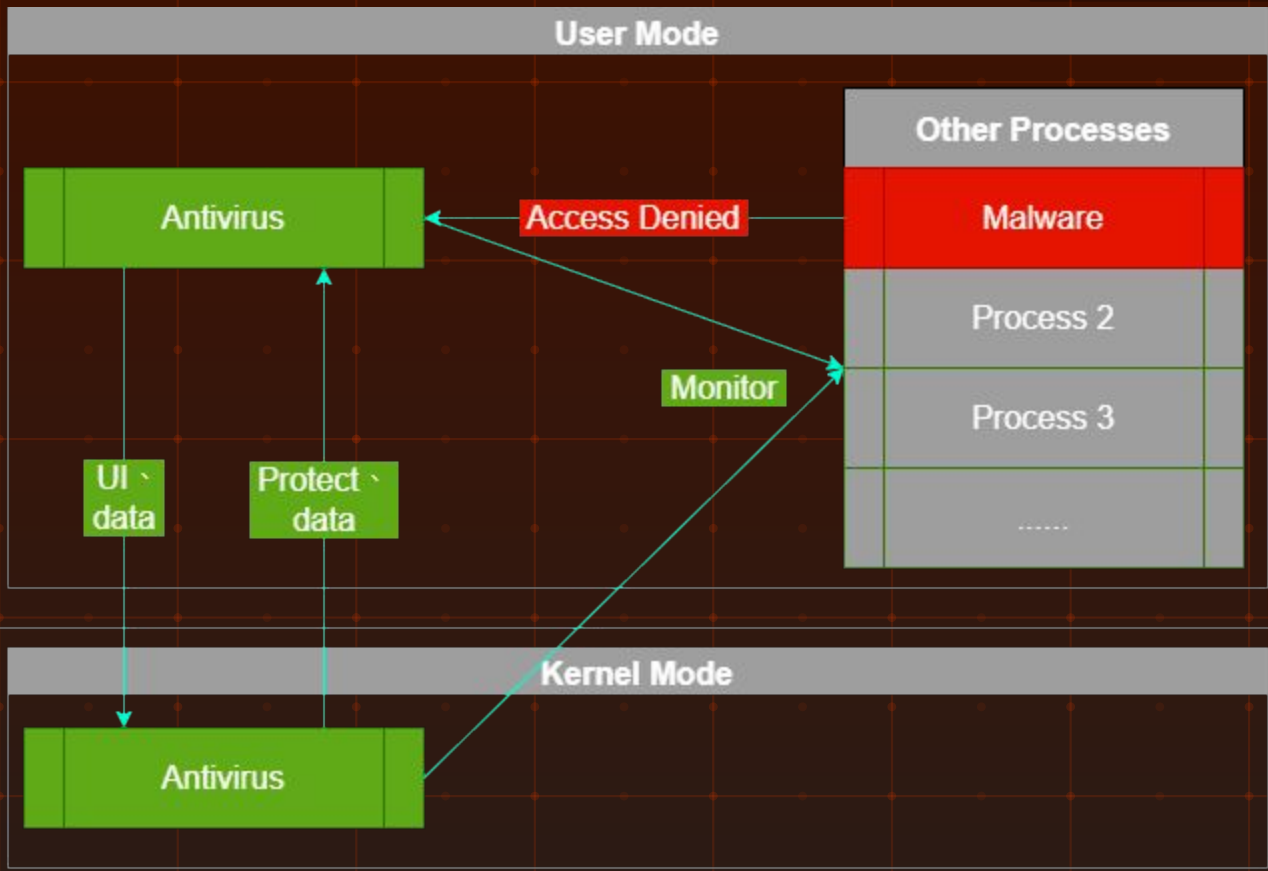
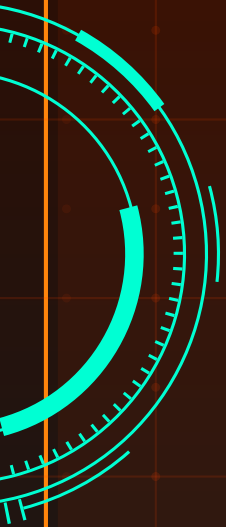
Antivirus Ring0 X Ring3

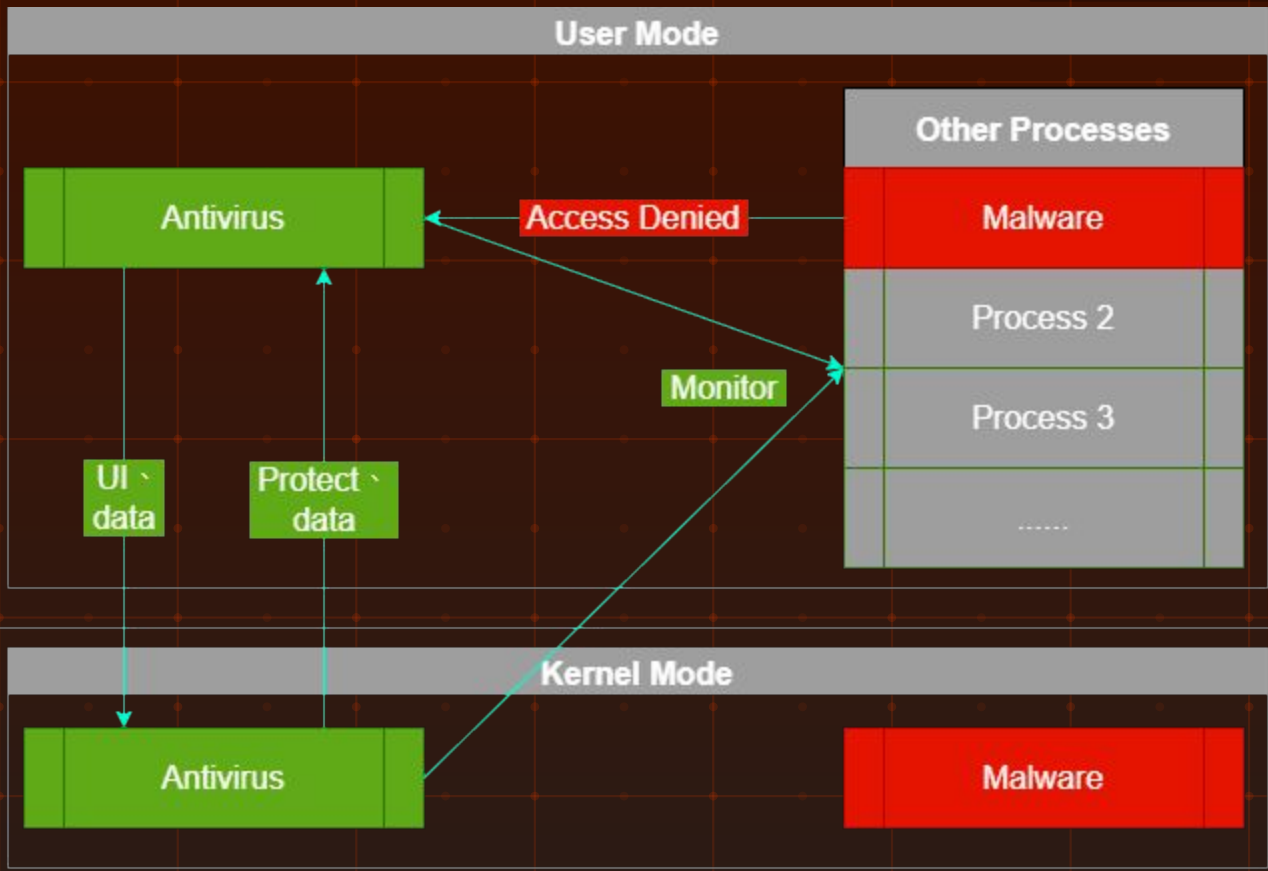
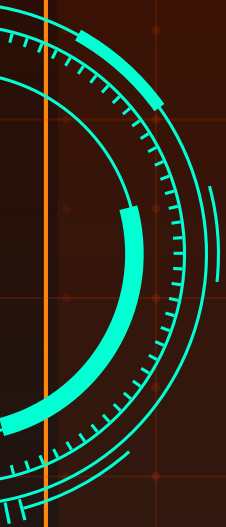
What is the relationship between user mode and kernel mode antivirus?

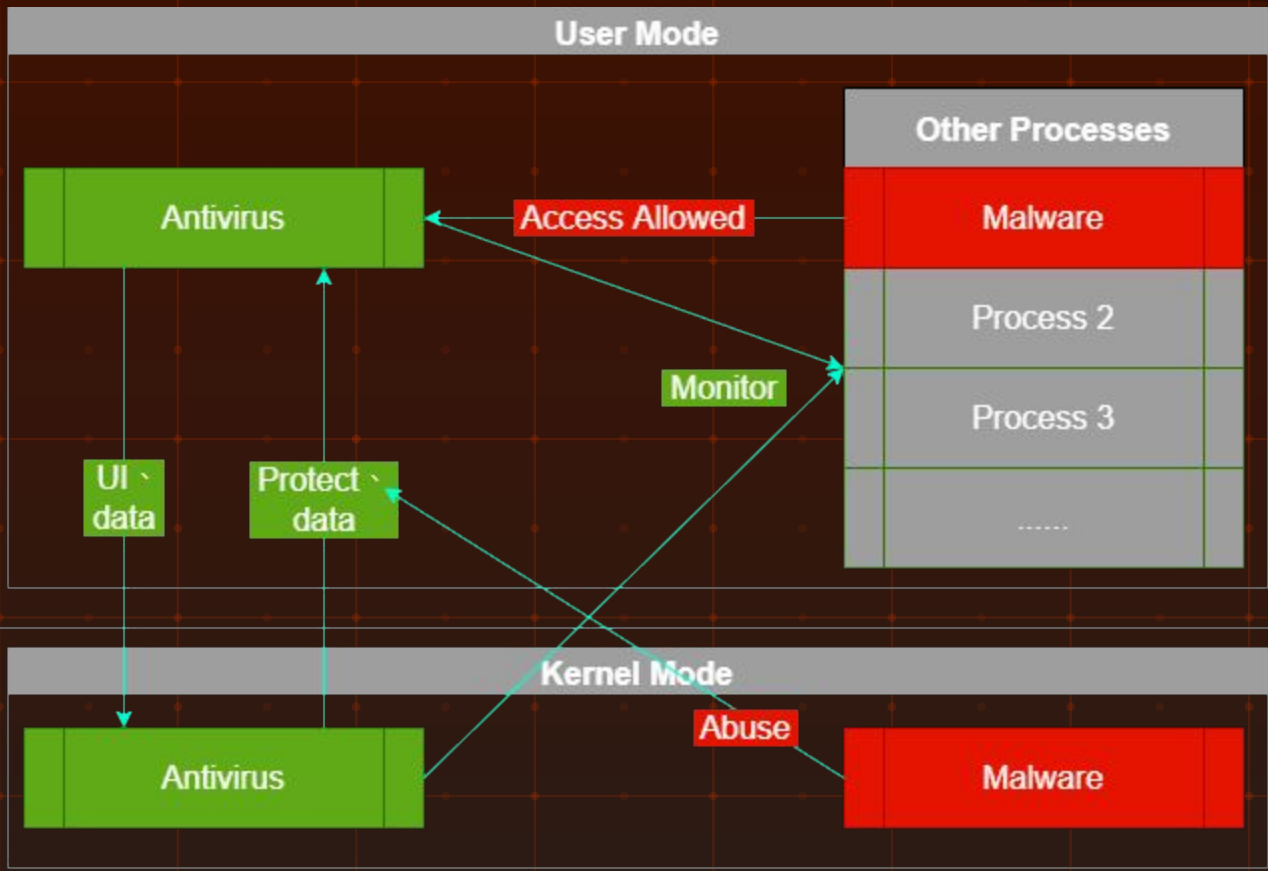
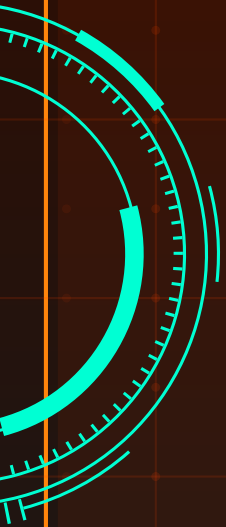
How does a malware tamper antivirus?













PatchGuard

Detect kernel patch

Kernel Patch Protection

Also known as PatchGuard.

Detect kernel patch in x64 Windows from Windows XP and Windows Server 2003.

Protected Areas

- SSDT(System Service Descriptor Table)
- IDT(Interrupt Descriptor Table)
- GDT(Global Descriptor Table)
- System images Processor
- MSR(Model Specific Register)
- Kernel Stack not allocated by Kernel
- Kernel itself, HAL, NDIS kernel Library

Protected Areas

- SSDT(System Service Descriptor Table)
- IDT(Interrupt Descriptor Table)
- GDT(Global Descriptor Table)
- System images Processor
- MSR(Model Specific Register)
- Kernel Stack not allocated by Kernel
- Kernel itself, HAL, NDIS kernel Library



```
bcdedit /set testsining on
```

BCDEdit.exe

Disable PatchGuard by enabling test mode, and we can disable integrity checks to load driver without signatures.

Bypass PatchGuard



Leverage Unprotected

Leverage the area that is not under the protection of PatchGuard.



Restore After Patch

Patch and restore the protected kernel areas fast.

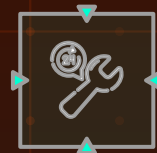


Bypass PatchGuard



Leverage Unprotected

Leverage the area that is not under the protection of PatchGuard.

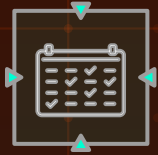


Restore After Patch

Patch and restore the protected kernel areas fast.



Bypass PatchGuard



Leverage Unprotected

Leverage the area that is not under the protection of PatchGuard.



Restore After Patch

Patch and restore the protected kernel areas fast.



Valid Alternative of Kernel Hook



Kernel
Hook

ObRegisterCallbacks

Monitor thread, process, desktop handle operations

NotifyRoutine

driver-supplied callback routine

MiniFilter

functionality in file system filter drivers

ETW

trace and log events

.....



Valid Alternative of Kernel Hook



Kernel
Hook

ObRegisterCallbacks

Monitor thread, process, desktop handle operations

NotifyRoutine

driver-supplied callback routine

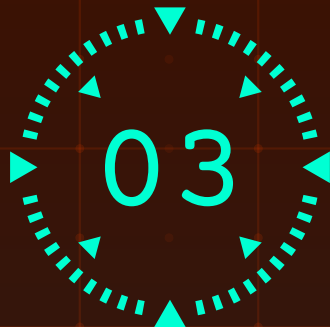
MiniFilter

functionality in file system filter drivers

ETW

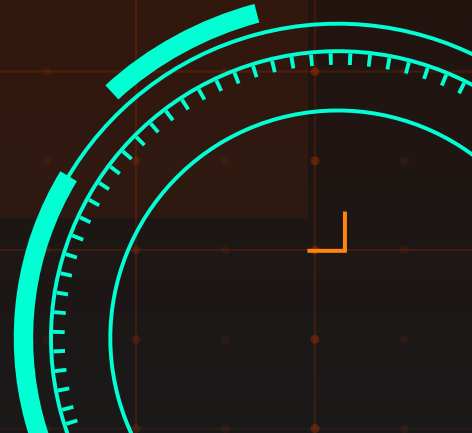
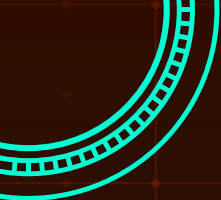
trace and log events

.....



ObRegisterCallbacks

Monitor thread, process, and desktop handle
operations



The ObRegisterCallbacks routine registers a list of callback routines for thread, process, and desktop handle operations.

—MSDN



```
NTSTATUS ObRegisterCallbacks(  
    [in] POB_CALLBACK_REGISTRATION CallbackRegistration,  
    [out] PVOID *RegistrationHandle  
);
```

ObRegisterCallbacks

We can use ObRegistersCallbacks to register a list of callback routines with parameters CallbackRegistration and RegistrationHandle.



```
NTSTATUS ObRegisterCallbacks(the config of callbacks  
    [in] POB_CALLBACK_REGISTRATION CallbackRegistration,  
    [out] PVOID *RegistrationHandle  
);
```

ObRegisterCallbacks

We can use ObRegistersCallbacks to register a list of callback routines with parameters CallbackRegistration and RegistrationHandle.



```
NTSTATUS ObRegisterCallbacks(  
    [in] POB_CALLBACK_REGISTRATION CallbackRegistration,  
    [out] PVOID *RegistrationHandle  
);
```

```
typedef struct _OB_CALLBACK_REGISTRATION {  
    USHORT Version;  
    USHORT OperationRegistrationCount;  
    UNICODE_STRING Altitude;  
    PVOID RegistrationContext;  
    OB_OPERATION_REGISTRATION *OperationRegistration;  
} OB_CALLBACK_REGISTRATION, *POB_CALLBACK_REGISTRATION;
```

version of ObRegisterCallbacks
number of entries
load order group
value is driver-defined
an array of OB_OPERATION_REGISTRATION

```
NTSTATUS ObRegisterCallbacks(  
    [in] POB_CALLBACK_REGISTRATION CallbackRegistration,  
    [out] PVOID *RegistrationHandle  
);
```

```
typedef struct _OB_CALLBACK_REGISTRATION {  
    USHORT Version;  
    USHORT OperationRegistrationCount;  
    UNICODE_STRING Altitude;  
    PVOID RegistrationContext;  
    OB_OPERATION_REGISTRATION *OperationRegistration;  
} OB_CALLBACK_REGISTRATION, *POB_CALLBACK_REGISTRATION;
```

an array of OB_OPERATION_REGISTRATION

```
NTSTATUS ObRegisterCallbacks(  
    [in] POB_CALLBACK_REGISTRATION CallbackRegistration,  
    [out] PVOID *RegistrationHandle  
);
```

```
typedef struct _OB_CALLBACK_REGISTRATION {  
    USHORT Version;  
    USHORT OperationRegistrationCount;  
    UNICODE_STRING Altitude;  
    PVOID RegistrationContext;  
    OB_OPERATION_REGISTRATION *OperationRegistration;  
} OB_CALLBACK_REGISTRATION, *POB_CALLBACK_REGISTRATION;
```

```
typedef struct _OB_OPERATION_REGISTRATION {  
    POBJECT_TYPE *ObjectType;  
    OB_OPERATION Operations;  
    POB_PRE_OPERATION_CALLBACK PreOperation;  
    POB_POST_OPERATION_CALLBACK PostOperation;  
} OB_OPERATION_REGISTRATION, *POB_OPERATION_REGISTRATION;
```

thread/process/desktop object
Flags: handle opened/duplicated
ObjectPreCallback routine
ObjectPostCallback routine

```
CBOperationRegistrations[0].ObjectType = PsProcessType;
CBOperationRegistrations[0].Operations |= OB_OPERATION_HANDLE_CREATE;
CBOperationRegistrations[0].Operations |= OB_OPERATION_HANDLE_DUPLICATE;
CBOperationRegistrations[0].PreOperation = CBTdPreOperationCallback;
CBOperationRegistrations[0].PostOperation = CBTdPostOperationCallback;
```

```
OB_PREOP_CALLBACK_STATUS CBTdPreOperationCallback (
    _In_ PVOID RegistrationContext,
    _Inout_ POB_PRE_OPERATION_INFORMATION PreInfo
);
```

Implement block logic here!

```
VOID CBTdPostOperationCallback (
    _In_ PVOID RegistrationContext,
    _In_ POB_POST_OPERATION_INFORMATION PostInfo
);
```

ObRegisterCallbacks

Not only antivirus use ObRegisterCallbacks, anti-cheat may also take advantage of this mechanism to protect itself.

Bypass ObRegisterCallbacks



ObUnRegisterCallbacks

A builtin method to unregister callbacks of the given handle.



Patch

Patch the code of callbacks to make it lose effectiveness.

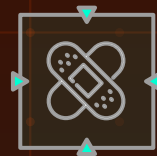


Bypass ObRegisterCallbacks



ObUnRegisterCallbacks

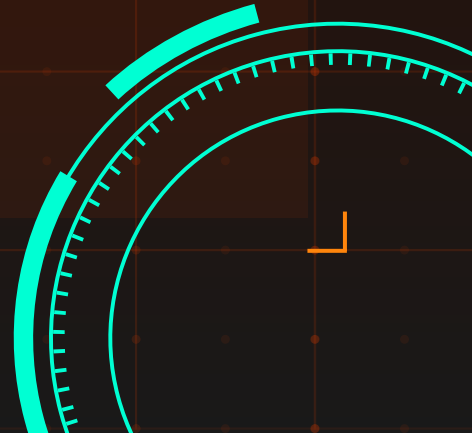
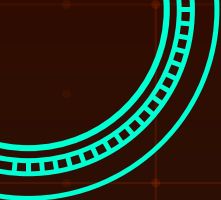
A builtin method to unregister callbacks of the given handle.



Patch

Patch the code of callbacks to make it lose effectiveness.





The ObUnRegisterCallbacks routine unregisters a set of callback routines that were registered with the ObRegisterCallbacks routine.

—MSDN

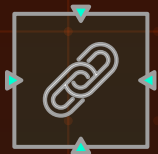


```
void ObUnRegisterCallbacks(  
    [in] PVOID RegistrationHandle  
);
```

ObUnRegisterCallbacks

We can use ObUnRegistersCallbacks to unregister a callback routines with a parameter RegistrationHandle.

Bypass ObRegisterCallbacks



ObUnRegisterCallbacks

A builtin method to unregister callbacks of the given handle.



Patch

Patch the code of callbacks to make it lose effectiveness.



函數名	當前函數地址	Hook	原始函數地址	Object類型	Object地址	當前函數地址所在模塊
	0xFFFFF80F59C07440	Callback_Object	-	PowerState(0xFFFFF868381A7A...	0xFFFFF868382C4610	C:\Windows\System32\drivers\ltpic.sys
OpenProcedure	0xFFFFF80F58503E0E	Callback_Object	-	LLTDCallback&rsprnd\000600800...	0xFFFFF868381A70CB	C:\Windows\System32\drivers\rsprnd.sys
CloseProcedure	0xFFFFF80F5926A3F0	-	-	PovObject	0xFFFFF868382B608D	C:\Windows\System32\drivers\pov.sys
DeleteProcedure	0xFFFFF80F5926A200	-	-	PovObject	0xFFFFF868382B608D	C:\Windows\System32\drivers\pov.sys
	0xFFFFF80F5926A230	-	-	PovObject	0xFFFFF868382B608D	C:\Windows\System32\drivers\pov.sys
	0xFFFFF80F592D6E80	Callback_Object	-	PowerState(0xFFFFF868381A7A...	0xFFFFF86838284050	C:\Windows\System32\drivers\pcol.sys
	0xFFFFF80F58B964E0	Callback_Object	-	PowerState(0xFFFFF868381A7A...	0xFFFFF868381AA540	C:\Windows\System32\drivers\ntosetl.sys
DeleteProcedure	0xFFFFF80F5996A420	-	-	IndexState	0xFFFFF868382BA730	C:\Windows\System32\drivers\index.sys
	0xFFFFF80F59A33700	Callback_Object	-	ProcessorAdd(0xFFFFF868381A...	0xFFFFF868382BA6920	C:\Windows\System32\drivers\index.sys
	0xFFFFF80F58503090	Callback_Object	-	ProcessorAdd(0xFFFFF868381A...	0xFFFFF868381ACD730	C:\Windows\System32\drivers\http.sys
CloseProcedure	0xFFFFF80F58A52810	-	-	FilterConnectorPort	0xFFFFF868381B8930	C:\Windows\System32\drivers\fltmgr.sys
DeleteProcedure	0xFFFFF80F58A52890	-	-	FilterConnectorPort	0xFFFFF868381B8930	C:\Windows\System32\drivers\fltmgr.sys
CloseProcedure	0xFFFFF80F58A52680	-	-	FilterCommunicationPort	0xFFFFF868381B89CD0	C:\Windows\System32\drivers\fltmgr.sys
DeleteProcedure	0xFFFFF80F58A526E0	-	-	FilterCommunicationPort	0xFFFFF868381B89CD0	C:\Windows\System32\drivers\fltmgr.sys
OpenProcedure	0xFFFFF80F5A4996F0	-	-	DxgkSharedResource	0xFFFFF8683820AB70	C:\Windows\System32\drivers\dxgkrnl.sys
OpenProcedure	0xFFFFF80F5A4996F0	-	-	DxgkSharedResource	0xFFFFF8683820AB70	C:\Windows\System32\drivers\dxgkrnl.sys
OpenProcedure	0xFFFFF80F5A4996F0	-	-	DxgkSharedBundleObject	0xFFFFF8683820C6560	C:\Windows\System32\drivers\dxgkrnl.sys
DeleteProcedure	0xFFFFF80F5A563AA0	-	-	DxgkSharedBundleObject	0xFFFFF8683820C6560	C:\Windows\System32\drivers\dxgkrnl.sys
OpenProcedure	0xFFFFF80F5A4996F0	-	-	DxgkSharedProtectedSessionO...	0xFFFFF8683820C5C00	C:\Windows\System32\drivers\dxgkrnl.sys
DeleteProcedure	0xFFFFF80F5A384F00	-	-	DxgkSharedProtectedSessionO...	0xFFFFF8683820C5C00	C:\Windows\System32\drivers\dxgkrnl.sys
OpenProcedure	0xFFFFF80F5A4996F0	-	-	DxgkSharedSyncObject	0xFFFFF8683820ABC10	C:\Windows\System32\drivers\dxgkrnl.sys
DeleteProcedure	0xFFFFF80F5A453450	-	-	DxgkSharedSyncObject	0xFFFFF8683820ABC10	C:\Windows\System32\drivers\dxgkrnl.sys
OpenProcedure	0xFFFFF80F5A4996F0	-	-	DxgkDisplayManagerObject	0xFFFFF8683820AB950	C:\Windows\System32\drivers\dxgkrnl.sys
DeleteProcedure	0xFFFFF80F5A380800	-	-	DxgkDisplayManagerObject	0xFFFFF8683820AB950	C:\Windows\System32\drivers\dxgkrnl.sys
OpenProcedure	0xFFFFF80F5A4996F0	-	-	-	0xFFFFF8683820ABAB0	C:\Windows\System32\drivers\dxgkrnl.sys
CloseProcedure	0xFFFFF80F5A58AD00	-	-	-	0xFFFFF8683820ABAB0	C:\Windows\System32\drivers\dxgkrnl.sys
DeleteProcedure	0xFFFFF80F5A58AE20	-	-	-	0xFFFFF8683820ABAB0	C:\Windows\System32\drivers\dxgkrnl.sys
	0xFFFFF80F5A1943D0	Callback_Object	-	ProcessorAdd(0xFFFFF868381A...	0xFFFFF86838291EFD0	C:\Windows\System32\drivers\acpi.sys
	0xFFFFF80F5A159800	Callback_Object	-	PowerState(0xFFFFF868381A7A...	0xFFFFF86838291EFD0	C:\Windows\System32\drivers\acpi.sys
PostOperation	0xFFFFF80286FD8730	ObjectType_Callback	-	Process	0xFFFFF868381A801A0	C:\Windows\System32\DRIVERS\360Safe.dll.sys
PreOperation	0xFFFFF80286FD8600	ObjectType_Callback	-	Process	0xFFFFF868381A801A0	C:\Windows\System32\DRIVERS\360Safe.dll.sys
PostOperation	0xFFFFF80286FD8730	ObjectType_Callback	-	Thread	0xFFFFF868381A80900	C:\Windows\System32\DRIVERS\360Safe.dll.sys
PreOperation	0xFFFFF80286FD8600	ObjectType_Callback	-	Thread	0xFFFFF868381A80900	C:\Windows\System32\DRIVERS\360Safe.dll.sys
PostOperation	0xFFFFF80286F681C	ObjectType_Callback	-	Process	0xFFFFF868381A801A0	C:\Windows\System32\DRIVERS\360Box64.sys
PreOperation	0xFFFFF80286F6810	ObjectType_Callback	-	Process	0xFFFFF868381A801A0	C:\Windows\System32\DRIVERS\360Box64.sys
PostOperation	0xFFFFF80286F681C	ObjectType_Callback	-	Thread	0xFFFFF868381A80900	C:\Windows\System32\DRIVERS\360Box64.sys
PreOperation	0xFFFFF80286F6810	ObjectType_Callback	-	Thread	0xFFFFF868381A80900	C:\Windows\System32\DRIVERS\360Box64.sys

Patch Callbacks

Instead of unregistering callbacks, we can also patch them to make them ineffective.

函數名	當前函數地址	Hook	原始函數地址	Object類型	Object地址	當前函數地址所在模塊
	0xFFFFF80F59C074A0	Callback_Object	-	PowerState(0xFFFFF80F59C074A0)	0xFFFFF80F59C074A0	C:\Windows\System32\drivers\tdtcp.sys
	0xFFFFF80F5B503BE0	Callback_Object	-	LTDCallbackRspndr(0xFFFFF80F5B503BE0)	0xFFFFF80F5B503BE0	C:\Windows\System32\drivers\rspsndr.sys
OpenProcedure	0xFFFFF80F5926A1F0	-	-	PcwObject	0xFFFFF80F5926A1F0	C:\Windows\System32\drivers\pcw.sys
CloseProcedure	0xFFFFF80F5926A200	-	-	PcwObject	0xFFFFF80F5926A200	C:\Windows\System32\drivers\pcw.sys
DeleteProcedure	0xFFFFF80F5926A230	-	-	PcwObject	0xFFFFF80F5926A230	C:\Windows\System32\drivers\pcw.sys
	0xFFFFF80F592D6EB0	Callback_Object	-	PowerState(0xFFFFF80F592D6EB0)	0xFFFFF80F592D6EB0	C:\Windows\System32\drivers\pci.sys
	0xFFFFF80F58B964E0	Callback_Object	-	PowerState(0xFFFFF80F58B964E0)	0xFFFFF80F58B964E0	C:\Windows\System32\drivers\ntosext.sys
DeleteProcedure	0xFFFFF80F5996EA20	-	-	NdisCmState	0xFFFFF80F5996EA20	C:\Windows\System32\drivers\ndis.sys
	0xFFFFF80F59A337D0	Callback_Object	-	ProcessorAdd(0xFFFFF80F59A337D0)	0xFFFFF80F59A337D0	C:\Windows\System32\drivers\ndis.sys
	0xFFFFF80F5B5B3090	Callback_Object	-	ProcessorAdd(0xFFFFF80F5B5B3090)	0xFFFFF80F5B5B3090	C:\Windows\System32\drivers\HTTP.sys
CloseProcedure	0xFFFFF80F58A52810	-	-	FilterConnectionPort	0xFFFFF80F58A52810	C:\Windows\System32\drivers\FLTMRG.SYS
DeleteProcedure	0xFFFFF80F58A52890	-	-	FilterConnectionPort	0xFFFFF80F58A52890	C:\Windows\System32\drivers\FLTMRG.SYS
CloseProcedure	0xFFFFF80F58A526B0	-	-	FilterCommunicationPort	0xFFFFF80F58A526B0	C:\Windows\System32\drivers\FLTMRG.SYS
DeleteProcedure	0xFFFFF80F58A526E0	-	-	FilterCommunicationPort	0xFFFFF80F58A526E0	C:\Windows\System32\drivers\FLTMRG.SYS
OpenProcedure	0xFFFFF80F5A499BF0	-	-	DxgkSharedResource	0xFFFFF80F5A499BF0	C:\Windows\System32\drivers\dxgkrnl.sys
DeleteProcedure	0xFFFFF80F5A44C960	-	-	DxgkSharedResource	0xFFFFF80F5A44C960	C:\Windows\System32\drivers\dxgkrnl.sys
OpenProcedure	0xFFFFF80F5A499BF0	-	-	DxgkSharedBundleObject	0xFFFFF80F5A499BF0	C:\Windows\System32\drivers\dxgkrnl.sys
DeleteProcedure	0xFFFFF80F5A563AA0	-	-	DxgkSharedBundleObject	0xFFFFF80F5A563AA0	C:\Windows\System32\drivers\dxgkrnl.sys
OpenProcedure	0xFFFFF80F5A499BF0	-	-	DxgkSharedProtectedSessionObject	0xFFFFF80F5A499BF0	C:\Windows\System32\drivers\dxgkrnl.sys
DeleteProcedure	0xFFFFF80F5A563AF0	-	-	DxgkSharedProtectedSessionObject	0xFFFFF80F5A563AF0	C:\Windows\System32\drivers\dxgkrnl.sys
OpenProcedure	0xFFFFF80F5A499BF0	-	-	DxgkSharedSyncObject	0xFFFFF80F5A499BF0	C:\Windows\System32\drivers\dxgkrnl.sys
DeleteProcedure	0xFFFFF80F5A453450	-	-	DxgkSharedSyncObject	0xFFFFF80F5A453450	C:\Windows\System32\drivers\dxgkrnl.sys
OpenProcedure	0xFFFFF80F5A499BF0	-	-	DxgkDisplayManagerObject	0xFFFFF80F5A499BF0	C:\Windows\System32\drivers\dxgkrnl.sys
DeleteProcedure	0xFFFFF80F5A58DB00	-	-	DxgkDisplayManagerObject	0xFFFFF80F5A58DB00	C:\Windows\System32\drivers\dxgkrnl.sys
OpenProcedure	0xFFFFF80F5A499BF0	-	-	DxgkDisplayManagerObject	0xFFFFF80F5A499BF0	C:\Windows\System32\drivers\dxgkrnl.sys
CloseProcedure	0xFFFFF80F5A58ADD0	-	-	DxgkDisplayManagerObject	0xFFFFF80F5A58ADD0	C:\Windows\System32\drivers\dxgkrnl.sys
DeleteProcedure	0xFFFFF80F5A58AE20	-	-	DxgkDisplayManagerObject	0xFFFFF80F5A58AE20	C:\Windows\System32\drivers\dxgkrnl.sys
	0xFFFFF80F5A1943D0	Callback_Object	-	ProcessorAdd(0xFFFFF80F5A1943D0)	0xFFFFF80F5A1943D0	C:\Windows\System32\drivers\ACPI.sys
	0xFFFFF80F5A15B980	Callback_Object	-	PowerState(0xFFFFF80F5A15B980)	0xFFFFF80F5A15B980	C:\Windows\System32\drivers\ACPI.sys
PostOperation	0xFFFFF80286FDB730	ObjectType_Callback	-	Process	0xFFFFF80286FDB730	C:\Windows\System32\DRIVERS\360FsFlt.sys
PreOperation	0xFFFFF80286FDB6D0	ObjectType_Callback	-	Process	0xFFFFF80286FDB6D0	C:\Windows\System32\DRIVERS\360FsFlt.sys
PostOperation	0xFFFFF80286FDB730	ObjectType_Callback	-	Thread	0xFFFFF80286FDB730	C:\Windows\System32\DRIVERS\360FsFlt.sys
PreOperation	0xFFFFF80286FDB6D0	ObjectType_Callback	-	Thread	0xFFFFF80286FDB6D0	C:\Windows\System32\DRIVERS\360FsFlt.sys
PostOperation	0xFFFFF80286F6BB1C	ObjectType_Callback	-	Process	0xFFFFF80286F6BB1C	C:\Windows\System32\DRIVERS\360Box64.sys
PreOperation	0xFFFFF80286F6BB1C	ObjectType_Callback	-	Process	0xFFFFF80286F6BB1C	C:\Windows\System32\DRIVERS\360Box64.sys
PostOperation	0xFFFFF80286F6BB1C	ObjectType_Callback	-	Thread	0xFFFFF80286F6BB1C	C:\Windows\System32\DRIVERS\360Box64.sys
PreOperation	0xFFFFF80286F6BB10	ObjectType_Callback	-	Thread	0xFFFFF80286F6BB10	C:\Windows\System32\DRIVERS\360Box64.sys

Callback List

Object Type函數：356, 被掛鉤函數：10

反彙編器

地址：

FF80286FDB6D0

大小(字節)：

000000C8

反彙編

Assembly of callback

地址	二進制	彙編
FFFFF80286...	48:895C24 08	mov qword ptr [rsp+8], rbx
FFFFF80286...	57	push rdi
FFFFF80286...	48:83EC 20	sub rsp, 20
FFFFF80286...	33FF	xor edi, edi
FFFFF80286...	F605 FD0C0500 10	test byte ptr [rip+50CFD], 10
FFFFF80286...	48:8BDA	mov rbx, rdx
FFFFF80286...	74 07	je FFFFF80286FDB6EF
FFFFF80286...	E8 1F030200	call FFFFF80286FFBA0C
FFFFF80286...	8BF8	mov edi, eax
FFFFF80286...	F643 04 01	test byte ptr [rbx+4], 1
FFFFF80286...	75 28	jne FFFFF80286FDB71D
FFFFF80286...	48:8B0D B41E0400	mov rcx, qword ptr [rip+41EB4]
FFFFF80286...	48:8B11	mov rdx, qword ptr [rcx]
FFFFF80286...	48:3953 10	cmp qword ptr [rbx+10], rdx
FFFFF80286...	75 18	jne FFFFF80286FDB71D
FFFFF80286...	833B 01	cmp dword ptr [rbx], 00000001
FFFFF80286...	75 13	jne FFFFF80286FDB71D

Patch Callbacks



Original

```
mov qword ptr [rsp+8], rbx
push rdi
sub rsp, 20
xor edi, edi
.....
```


Patch Callbacks



Patched

```
xor rax, rax  
ret  
sub rsp, 20  
xor edi, edi  
.....
```

Patch Callbacks

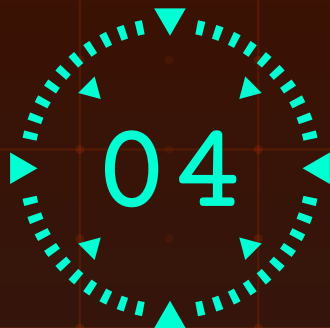


Patched

```
xor rax, rax  
ret  
sub rsp, 20  
xor edi, edi  
.....
```

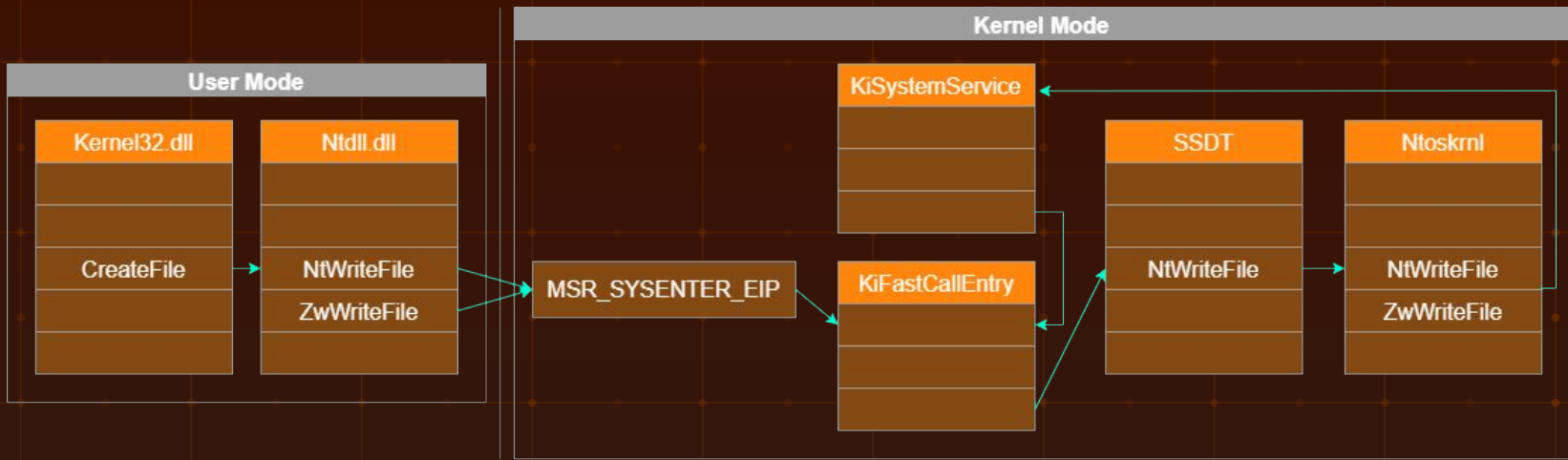
return 0;





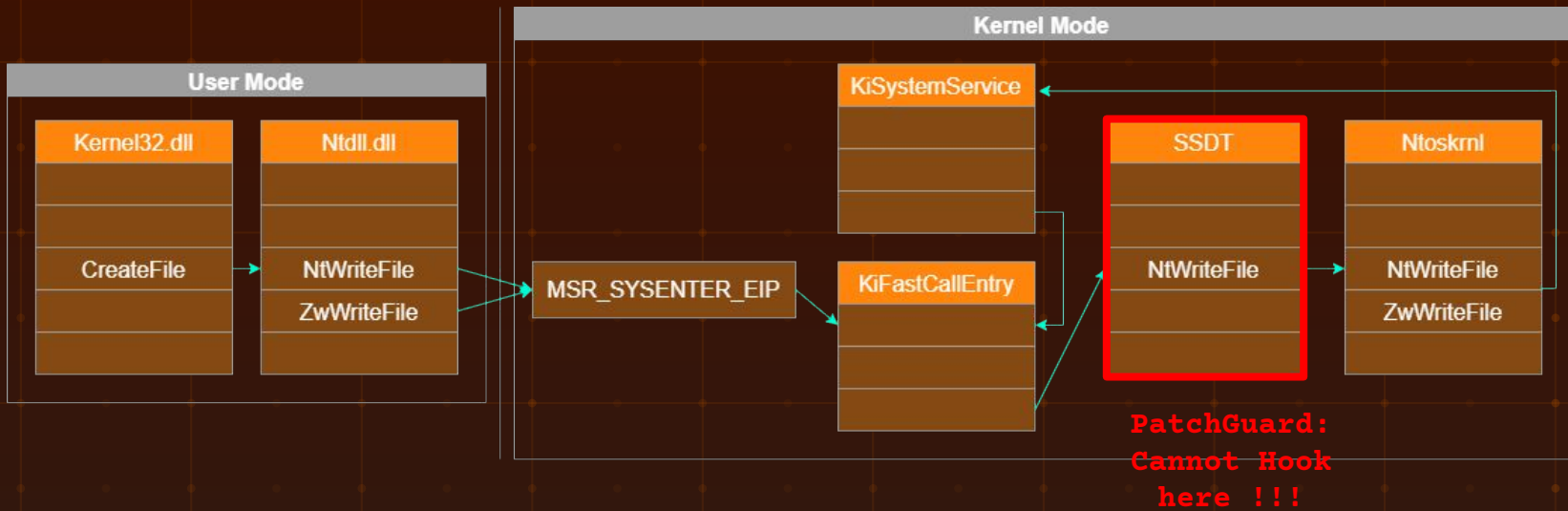
Infinity Hook

Use kernel hook bypassing PatchGuard to implement
rootkit



Kernel Hook

In x86 Windows, we can hook in kernel mode such as SSDT, but it is forbidden by PatchGuard in x64 after Windows XP and Windows Server 2003.



Kernel Hook

In x86 Windows, we can hook in kernel mode such as SSDT, but it is forbidden by PatchGuard in x64 after Windows XP and Windows Server 2003.





Make hook great again!

—InfinityHook

InfinityHook

Steps

Find ETW_DEBUGGER_DATA

Find ETW_DEBUGGER_DATA with signature.

Get WMI_LOGGER_CONTEXT

Get WMI_LOGGER_CONTEXT after ETW_DEBUGGER_DATA.

Find SSDT

Find the pointer of SSDT.

Hook GetCpuClock

Hook GetCpuClock after WMI_LOGGER_CONTEXT.

Find Syscall

Find address of syscall from stack.

Hook Syscall

Finally hook in kernel and do whatever we want.

InfinityHook

Steps

Find ETW_DEBUGGER_DATA

Find ETW_DEBUGGER_DATA with signature.

Get WMI_LOGGER_CONTEXT

Get WMI_LOGGER_CONTEXT after ETW_DEBUGGER_DATA.

Find SSDT

Find the pointer of SSDT.

Hook GetCpuClock

Hook GetCpuClock after WMI_LOGGER_CONTEXT.

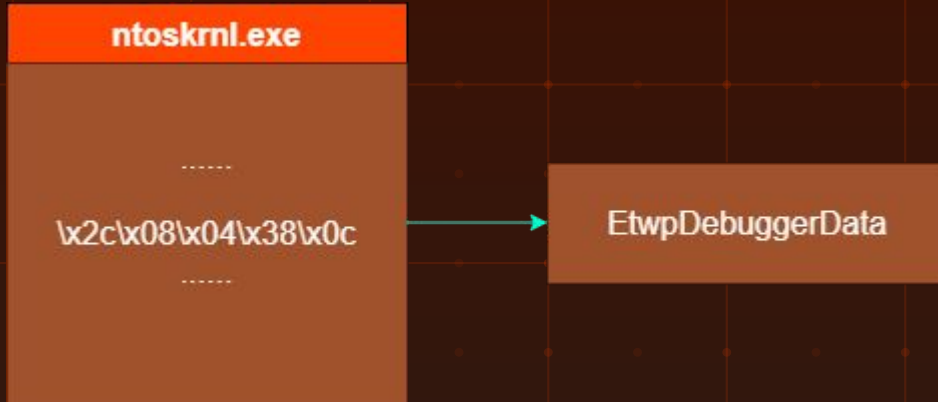
Find Syscall

Find address of syscall from stack.

Hook Syscall

Finally hook in kernel and do whatever we want.

Find ETW_DEBUGGER_DATA



InfinityHook

Steps

Find ETW_DEBUGGER_DATA

Find ETW_DEBUGGER_DATA with signature.

Get WMI_LOGGER_CONTEXT

Get WMI_LOGGER_CONTEXT after ETW_DEBUGGER_DATA.

Find SSDT

Find the pointer of SSDT.

Hook GetCpuClock

Hook GetCpuClock after WMI_LOGGER_CONTEXT.

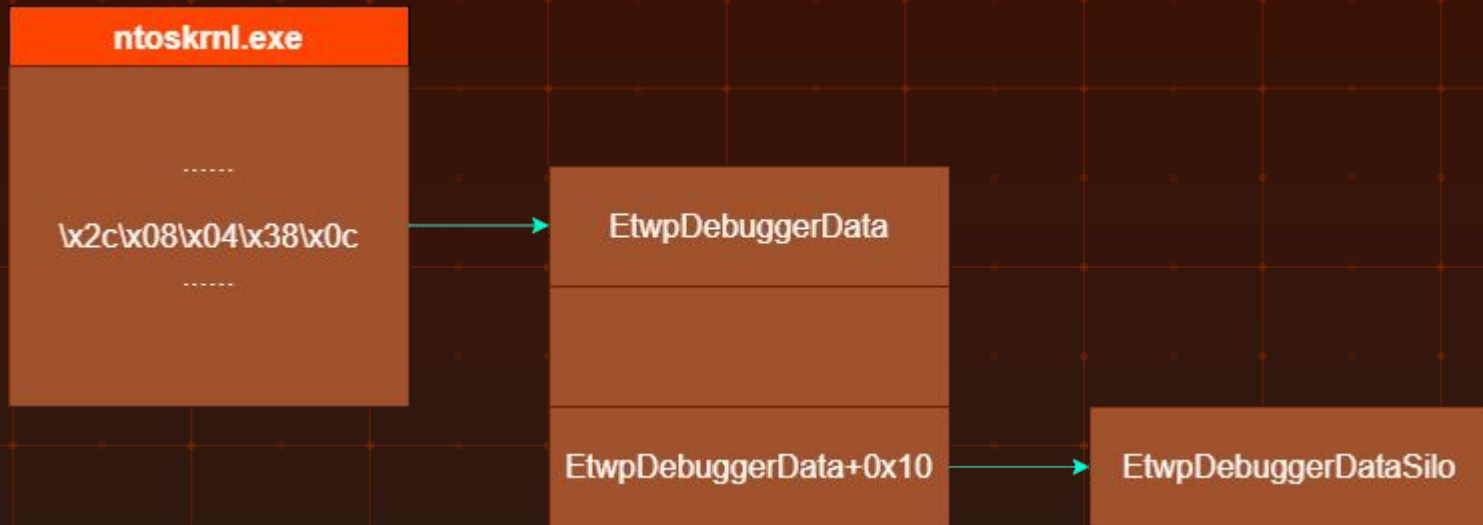
Find Syscall

Find address of syscall from stack.

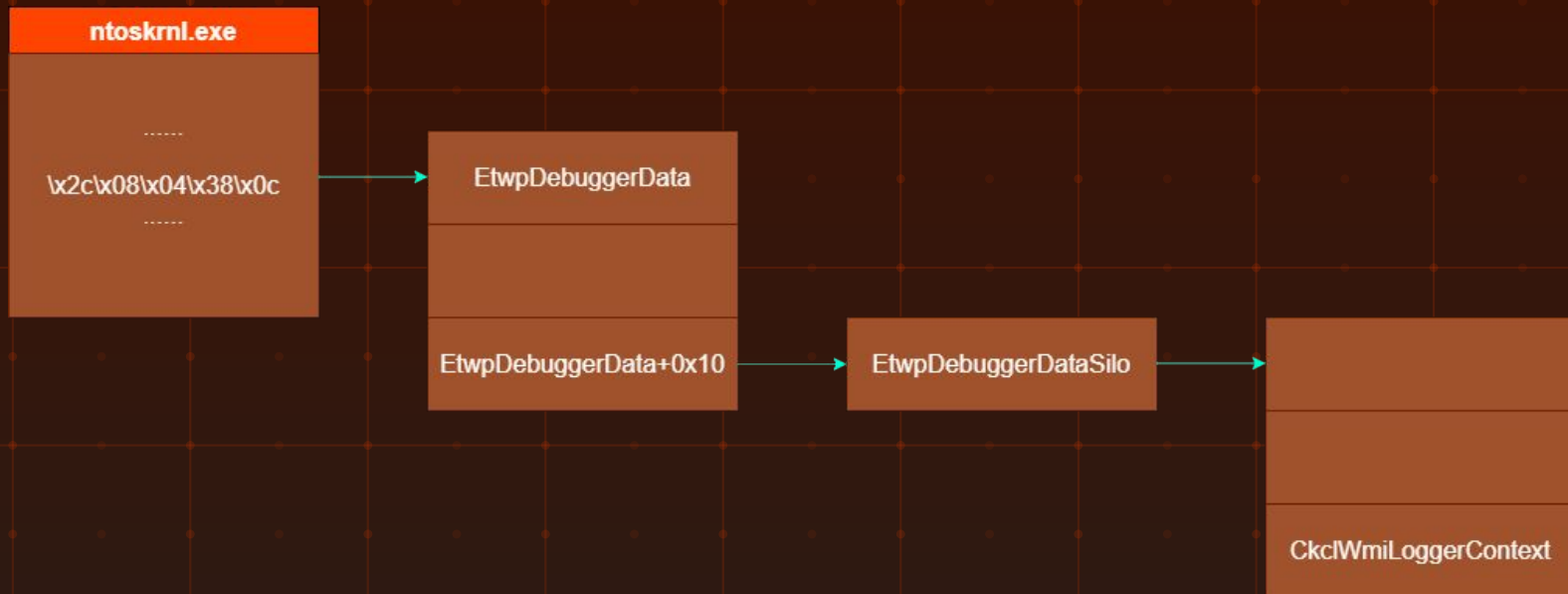
Hook Syscall

Finally hook in kernel and do whatever we want.

Get WMI_LOGGER_CONTEXT



Get WMI_LOGGER_CONTEXT



InfinityHook

Steps

Find ETW_DEBUGGER_DATA

Find ETW_DEBUGGER_DATA with signature.

Get WMI_LOGGER_CONTEXT

Get WMI_LOGGER_CONTEXT after ETW_DEBUGGER_DATA.

Find SSDT

Find the pointer of SSDT.

Hook GetCpuClock

Hook GetCpuClock after WMI_LOGGER_CONTEXT.

Find Syscall

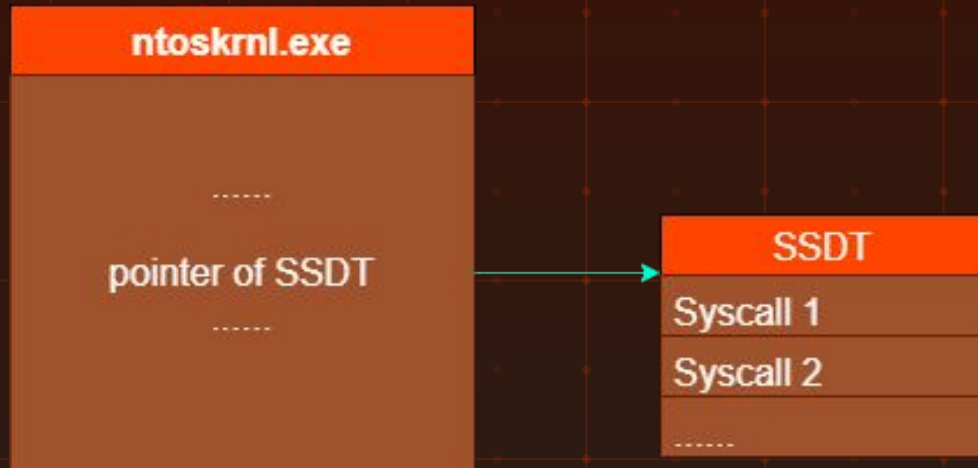
Find address of syscall from stack.

Hook Syscall

Finally hook in kernel and do whatever we want.

Find SSDT

Another long story...



InfinityHook

Steps

Find ETW_DEBUGGER_DATA

Find ETW_DEBUGGER_DATA with signature.

Get WMI_LOGGER_CONTEXT

Get WMI_LOGGER_CONTEXT after ETW_DEBUGGER_DATA.

Find SSDT

Find the pointer of SSDT.

Hook GetCpuClock

Hook GetCpuClock after WMI_LOGGER_CONTEXT.

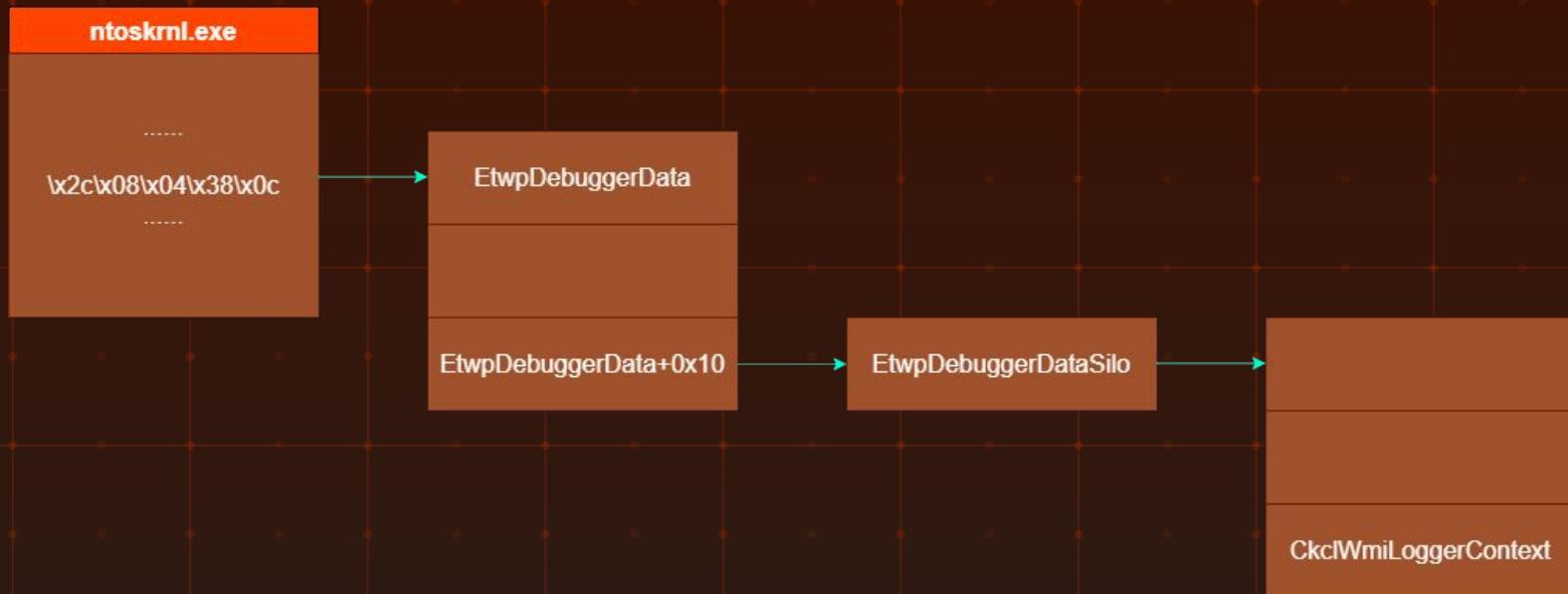
Find Syscall

Find address of syscall from stack.

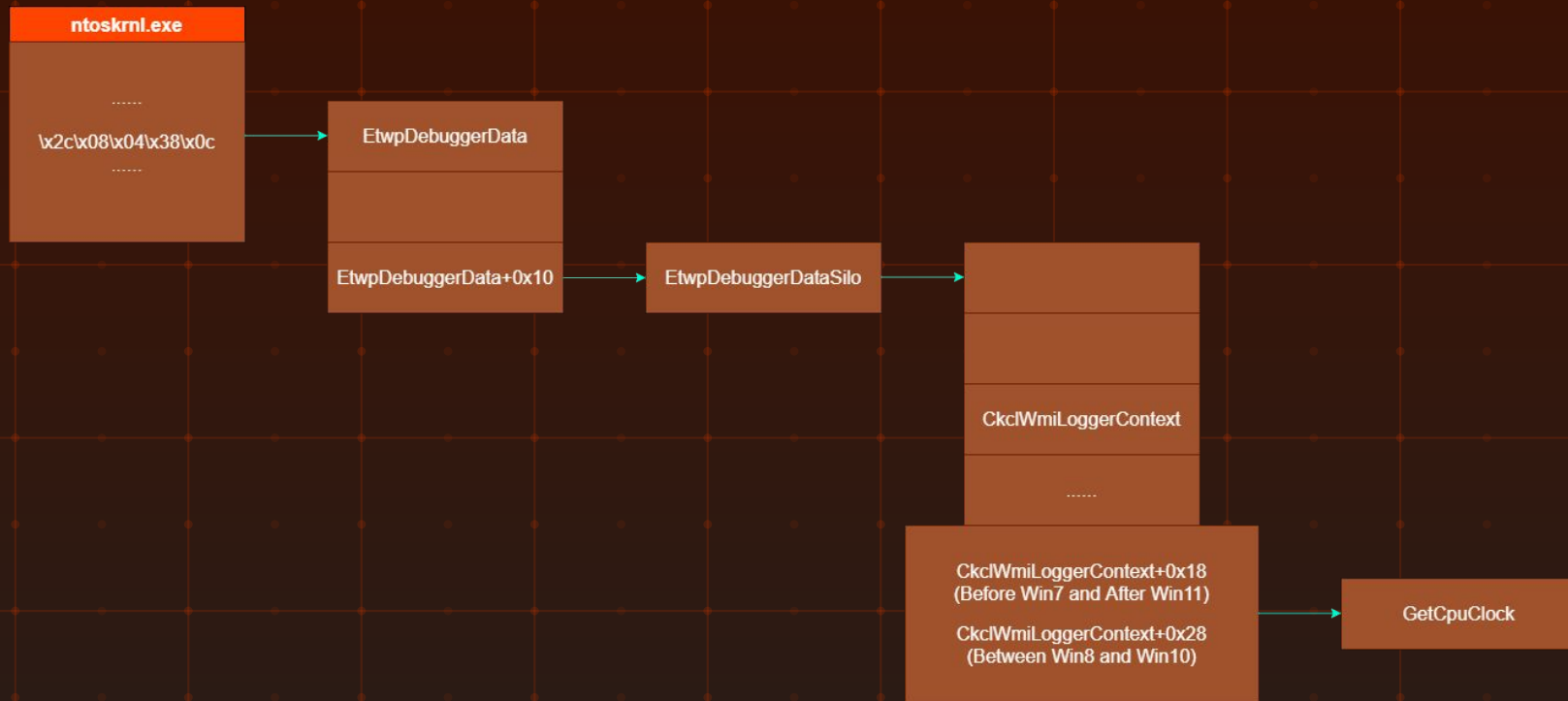
Hook Syscall

Finally hook in kernel and do whatever we want.

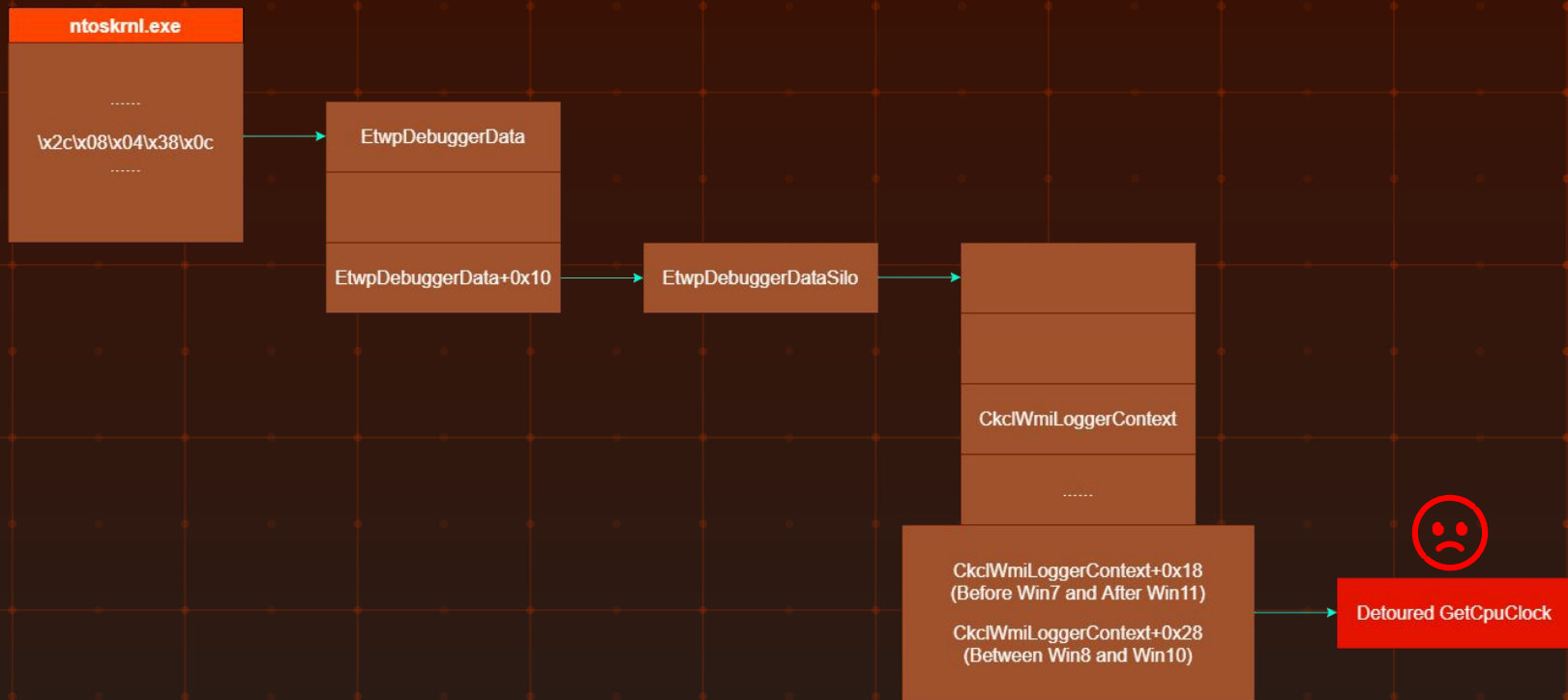
Hook GetCpuClock



Hook GetCpuClock



Hook GetCpuClock



InfinityHook

Steps

Find ETW_DEBUGGER_DATA

Find ETW_DEBUGGER_DATA with signature.

Get WMI_LOGGER_CONTEXT

Get WMI_LOGGER_CONTEXT after ETW_DEBUGGER_DATA.

Find SSDT

Find the pointer of SSDT.

Hook GetCpuClock

Hook GetCpuClock after WMI_LOGGER_CONTEXT.

Find Syscall

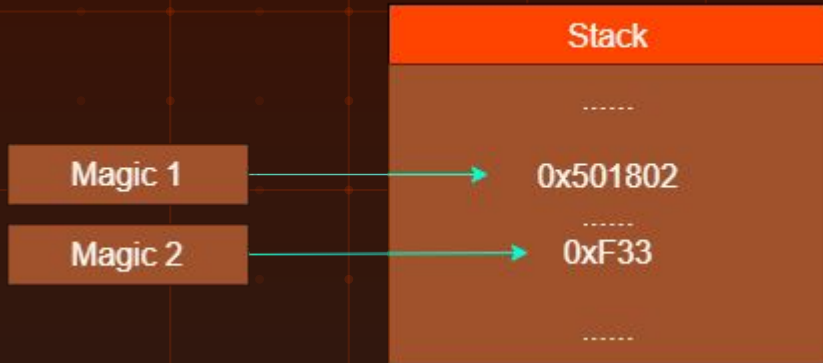
Find address of syscall from stack.

Hook Syscall

Finally hook in kernel and do whatever we want.

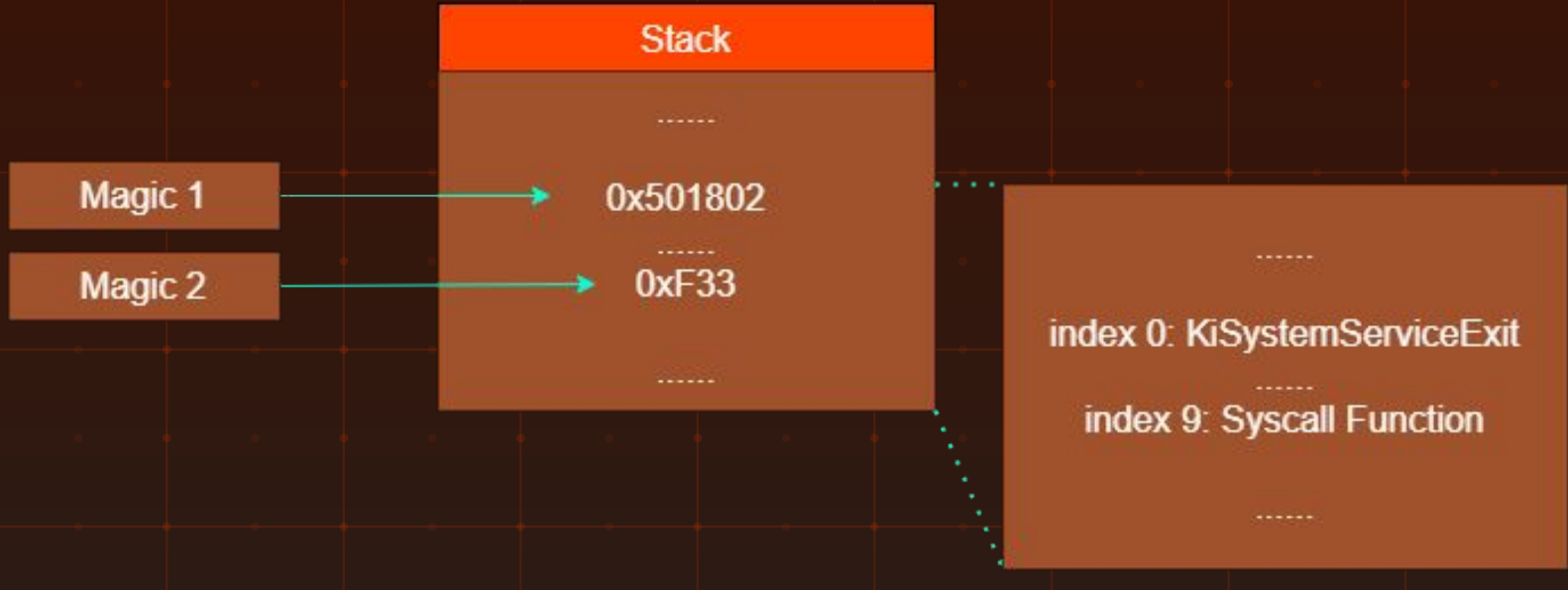
Find Syscall

DetouredGetCpuClock



Find Syscall

DetouredGetCpuClock



InfinityHook

Steps

Find ETW_DEBUGGER_DATA

Find ETW_DEBUGGER_DATA with signature.

Get WMI_LOGGER_CONTEXT

Get WMI_LOGGER_CONTEXT after ETW_DEBUGGER_DATA.

Find SSDT

Find the pointer of SSDT.

Hook GetCpuClock

Hook GetCpuClock after WMI_LOGGER_CONTEXT.

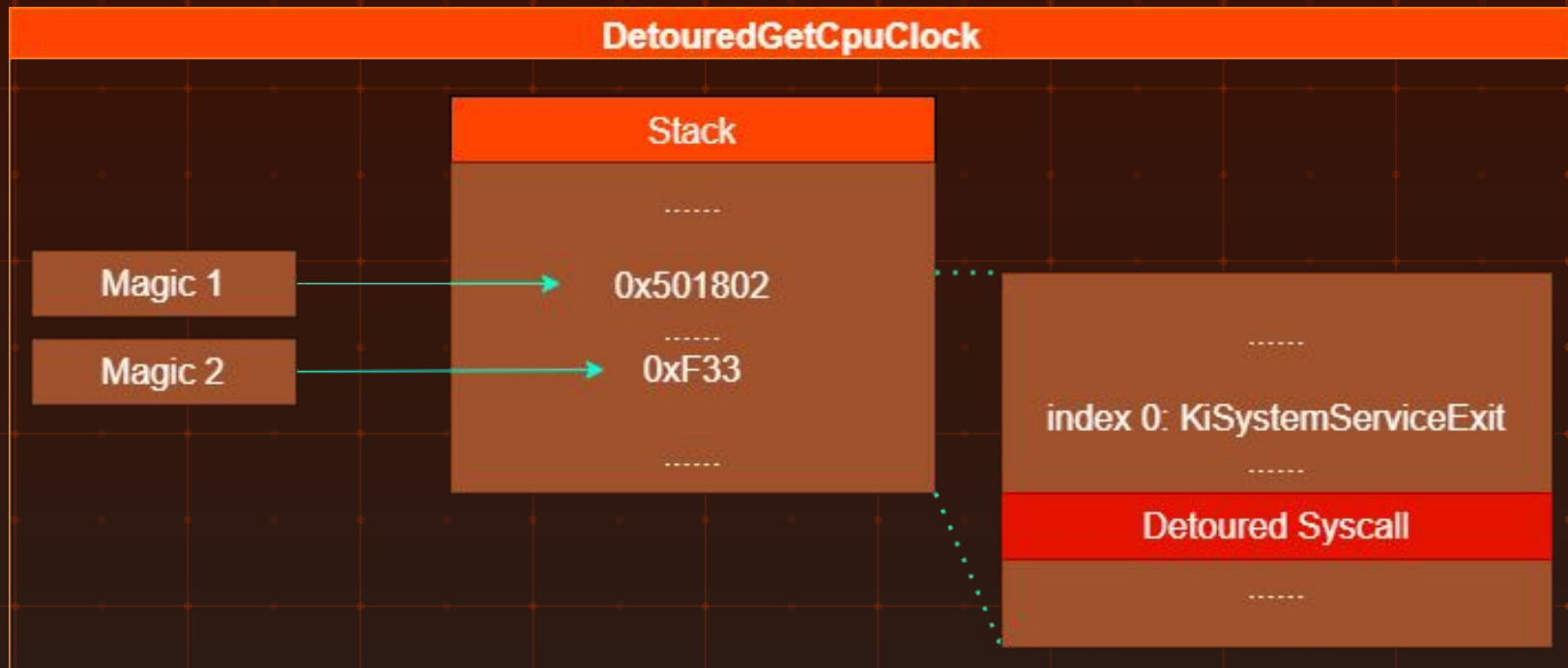
Find Syscall

Find address of syscall from stack.

Hook Syscall

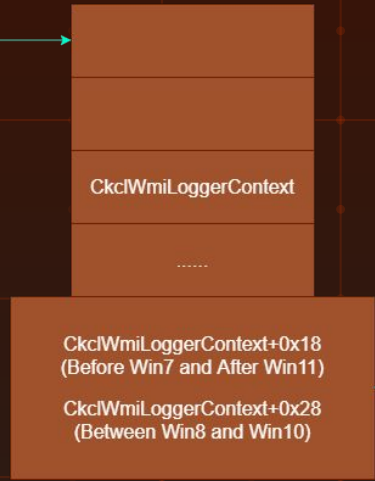
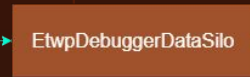
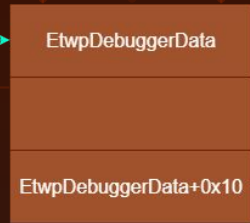
Finally hook in kernel and do whatever we want.

Hook Syscall

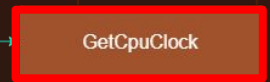


But...

InfinityHook supports up to Windows 10 1909 (OS build 18363), because `GetCpuClock` is not a function pointer anymore.



cannot hook !





Make InfinityHook great again!

—InfinityHookPro

What happened to GetCpuClock?

After Windows 10 1909 (OS build 18363), GetCpuClock becomes an index.

Index	Function
0	RtlGetSystemTimePrecise
1	KeQueryPerformanceCounter
2	HalpTimerQueryHostPerformanceCounter
3	rdtsc

Hv1GetQpcBias

In HalpTimerQueryHostPerformanceCounter, Hv1GetQpcBias can be hooked without detected by PatchGuard.

Index	Function
0	RtlGetSystemTimePrecise
1	KeQueryPerformanceCounter
2	HalpTimerQueryHostPerformanceCounter
3	rdtsc

InfinityHookPro

Steps

Find ETW_DEBUGGER_DATA

Find ETW_DEBUGGER_DATA with signature.

Get WMI_LOGGER_CONTEXT

Get WMI_LOGGER_CONTEXT after ETW_DEBUGGER_DATA.

Find SSDT

Find the pointer of SSDT.

Hook HvlGetQpcBias

Find and hook HvlGetQpcBias.

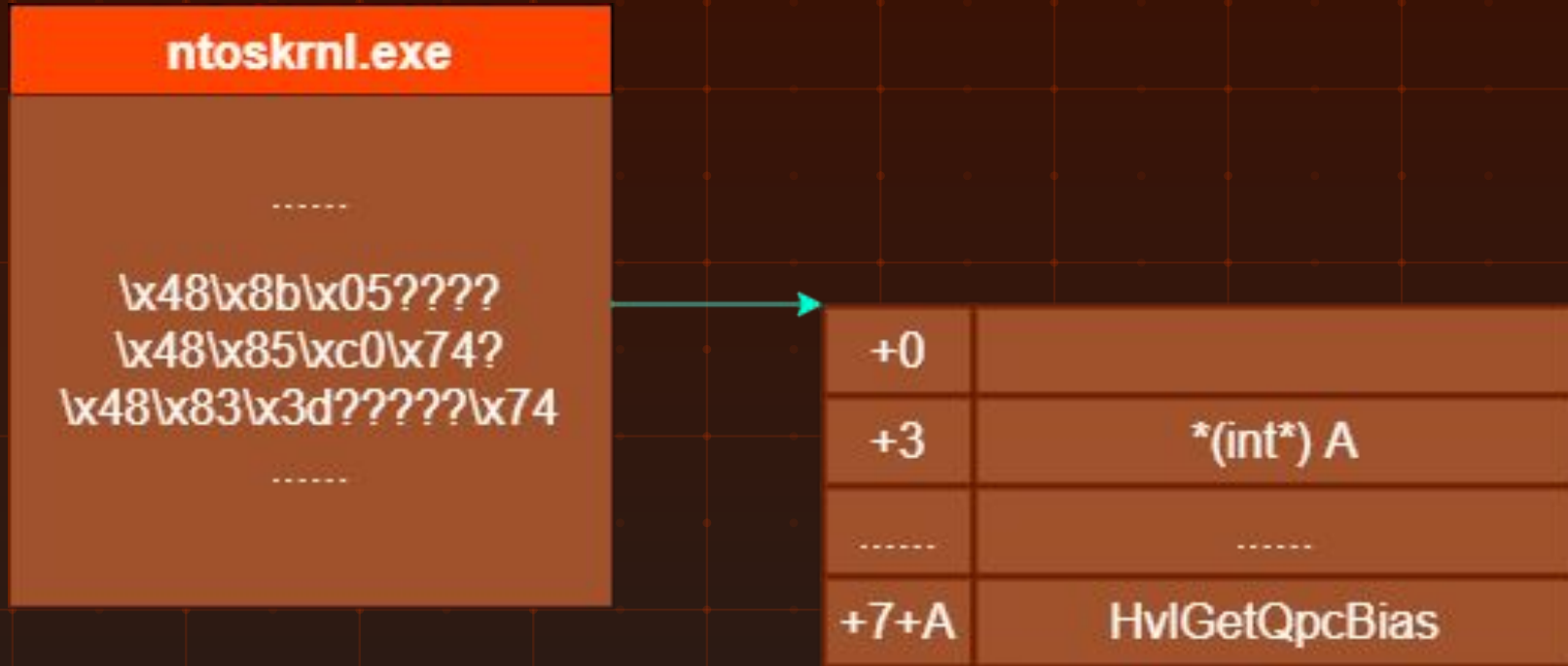
Find Syscall

Find address of syscall from stack.

Hook Syscall

Finally hook in kernel and do whatever we want.

Hook Hv1GetQpcBias



Hook Hv1GetQpcBias

ntoskrnl.exe

.....
\\x48\\x8b\\x05????
\\x48\\x85\\xc0\\x74?
\\x48\\x83\\x3d?????\\x74
.....

+0

+3

+7+A

(int) A

Detoured Hv1GetQpcBias



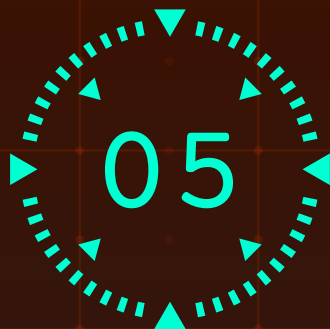
Cont.

Then similar to InfinityHook to find syscall from stack and hook it.

Kernel Rootkit

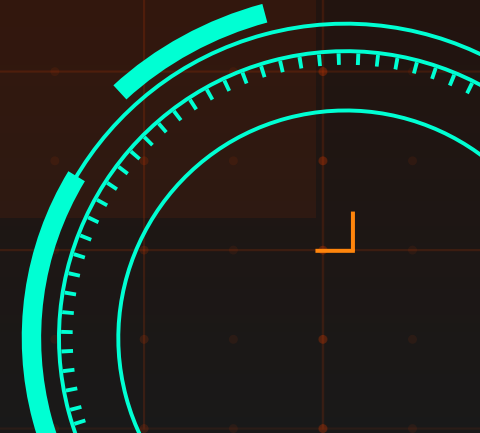
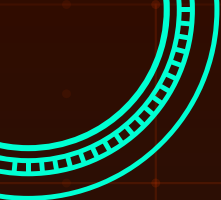
With InfinityHookPro, we can equally hook SSDT to implement kernel rootkit.

Target Syscall	Effect
NtCreateFile	Deny access to files.
NtQueryDirectoryFile	Hide files.
NtQuerySystemInformation	Hide processes.
.....



KDU

Load drivers without signature



The kernel-mode code signing policy for 64-bit versions of Windows Vista and later versions of Windows specifies that a kernel-mode driver must be signed for the driver to load.

—MSDN

Valid Methods To Load Driver



Buy Certificate

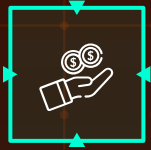
Buy certificate from Microsoft.



Test Mode

Turn on testing mode to load testsigning drivers.

Valid Methods To Load Driver



Buy Certificate

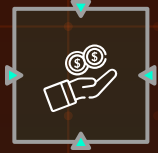
Buy certificate from Microsoft.



Test Mode

Turn on testing mode to load test signing drivers.

Valid Methods To Load Driver



Buy Certificate

Buy certificate from Microsoft.



Test Mode

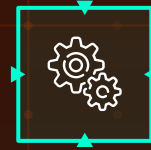
Turn on testing mode to load test signing drivers.

Bypass Digital Signature



Leaked Certificate

Use leaked certificate to sign the driver.



Kernel Driver Utility

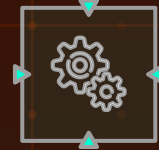
Leverage vulnerable driver to disable DSE and load an unsigned driver.

Bypass Digital Signature



Leaked Certificate

Use leaked certificate to sign the driver.



Kernel Driver Utility

Leverage vulnerable driver to disable DSE and load an unsigned driver.





Bill Demirkapi
@BillDemirkapi

As part of the #NvidiaLeaks, two code signing certificates have been compromised. Although they have expired, Windows still allows them to be used for driver signing purposes. See the talk I gave at BH/DC for more context on leaked certificates: youtu.be/1H9tEfjFXs?t=...

VIDIA Corporation	VIDIA Corporation
VeriSign Class 3 Code Signing	VeriSign Class 3 Code Signing
1/2011 to 9/1/2014	/27/2015 to 7/26/2018
ate key that corresponds to	ate key that corresponds to



Florian Roth
@cyb3rops

That escalated quickly #Lapsus #Nvidia #LeakedCertificate

Mimikatz
virustotal.com/gui/file/9d123...

KDU
virustotal.com/gui/file/0e163...

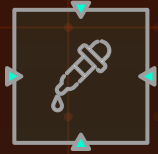
The image shows two VirusTotal analysis results side-by-side. The left result is for 'mimikatz.exe' (SHA256: 9d123...) and shows a single signature from 'NVIDIA Corporation' with a description of 'VeriSign Class 3 Code Signing 2010 CA'. The right result is for 'KDU' (SHA256: 0e163...) and shows a single signature from 'VeriSign Class 3 Code Signing 2010 CA' with a description of 'VeriSign Class 3 Code Signing 2010 CA'.

Leaked Certificate

Use leaked certificate to sign the driver.



Bypass Digital Signature



Leaked Certificate

Use leaked certificate to sign the driver.



Kernel Driver Utility

Leverage vulnerable driver to disable DSE and load an unsigned driver.



Driver Signature Enforcement

ntoskrnl.exe
(before Windows8 build 9600)

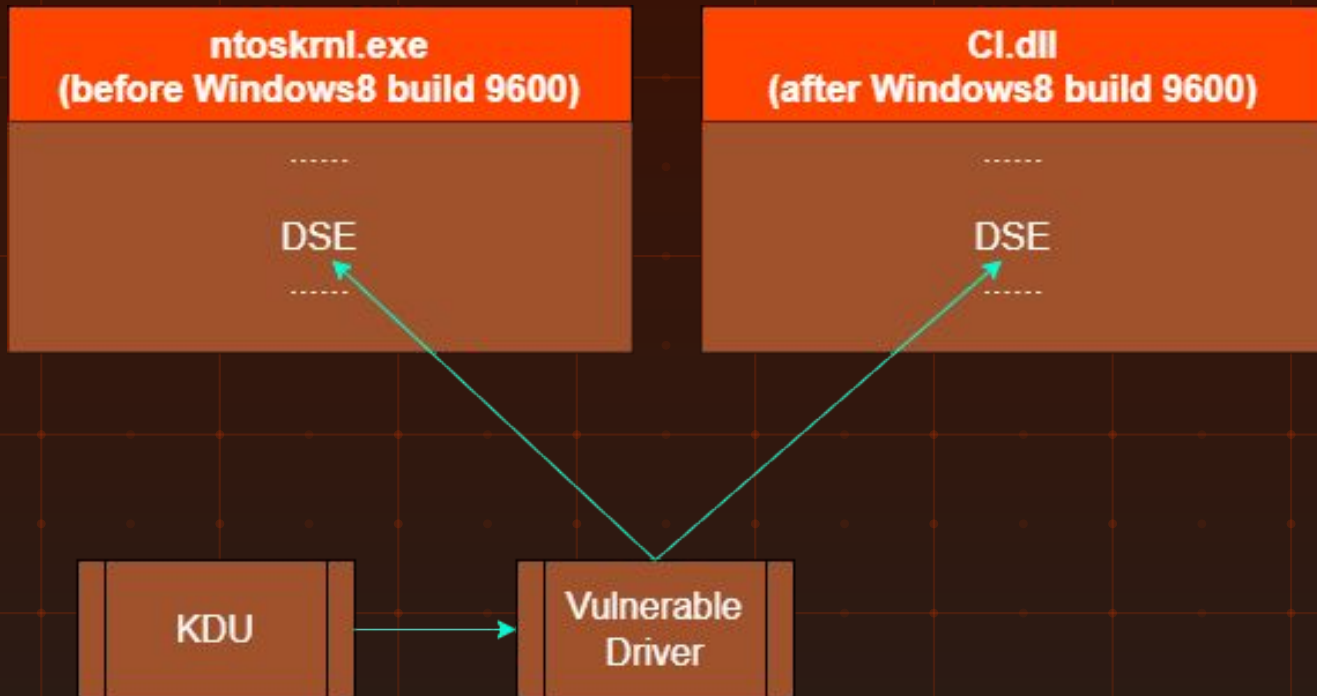
DSE

Cl.dll
(after Windows8 build 9600)

DSE

DSE	Effect
0	DSE Disabled
6	DSE Enabled
8	Test Mode

Kernel Driver Utility



Evade PatchGuard

Remember to restore DSE flag to original value, or ...



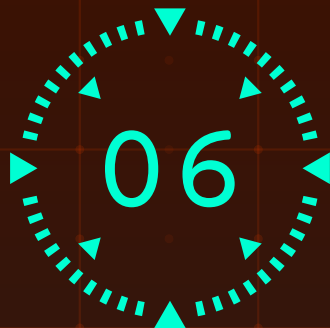
Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you.

100% complete



For more information about this issue and possible fixes, visit <https://www.windows.com/stopcode>

If you call a support person, give them this info:
Stop code: CRITICAL_STRUCTURE_CORRUPTION
What failed: Ci.dll



Coexist With Virus

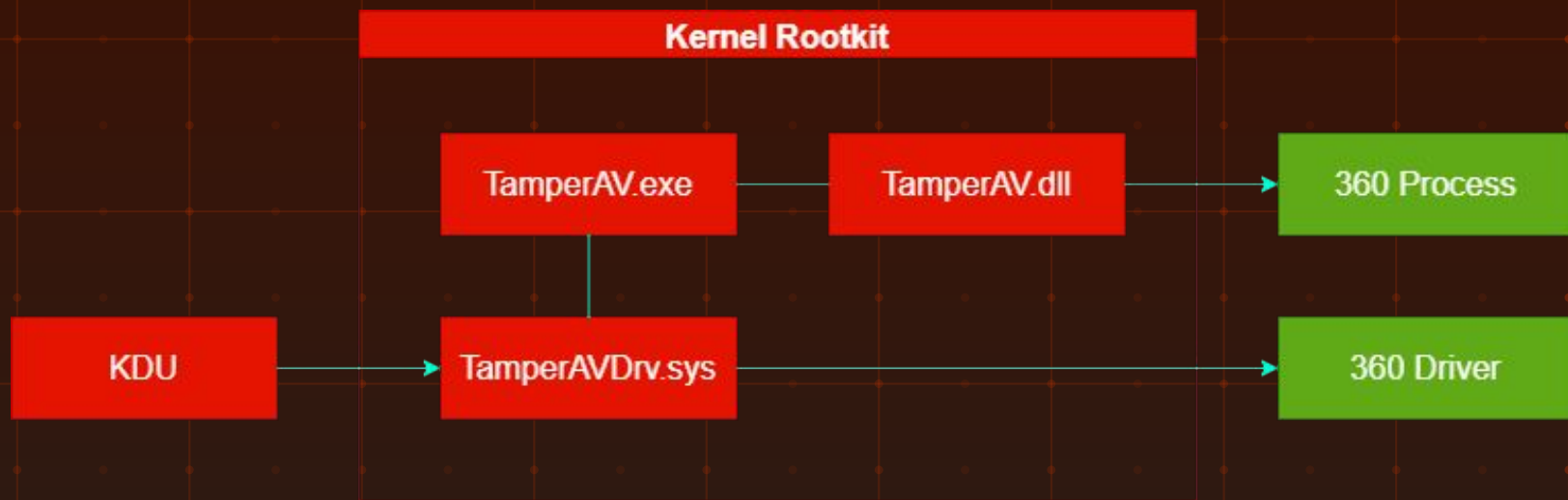
Inject into antivirus - TamperAV

Recall

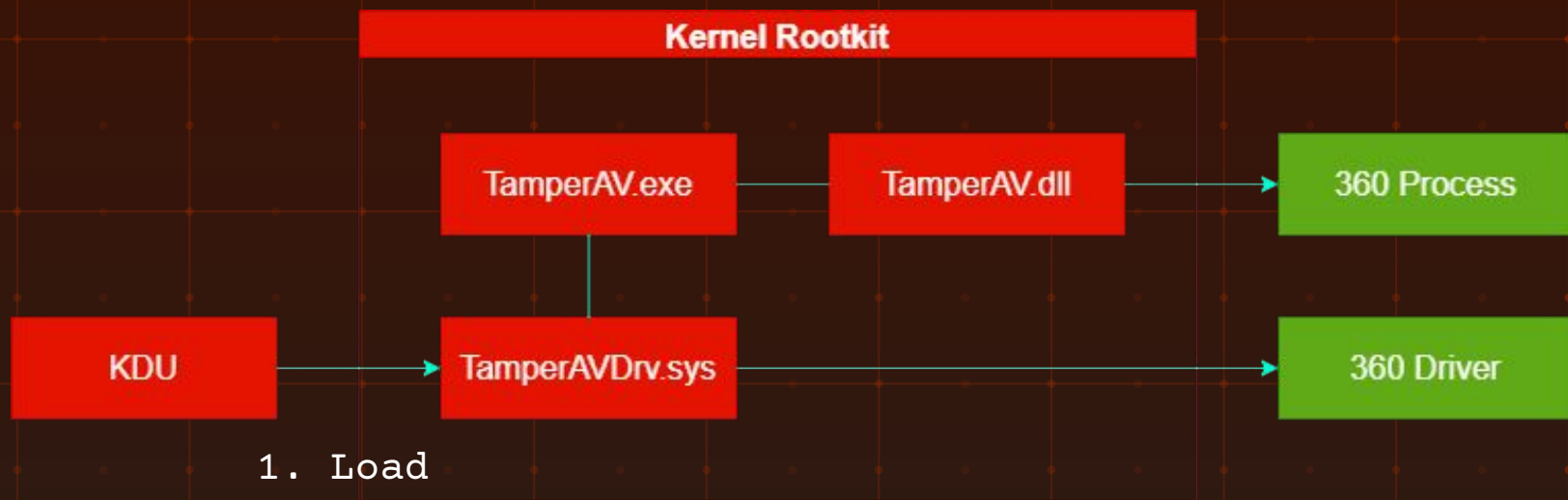
Now we can

- patch ObRegisterCallbacks
- hook in Kernel to implement rootkit
- load driver without digital signature

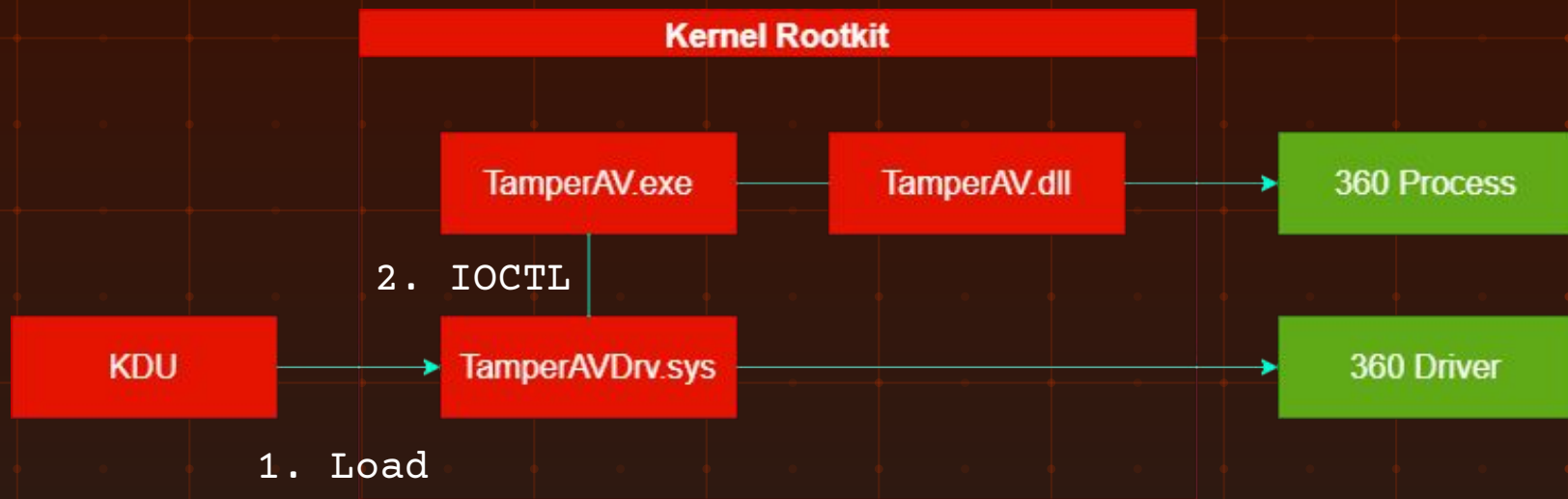
TamperAV



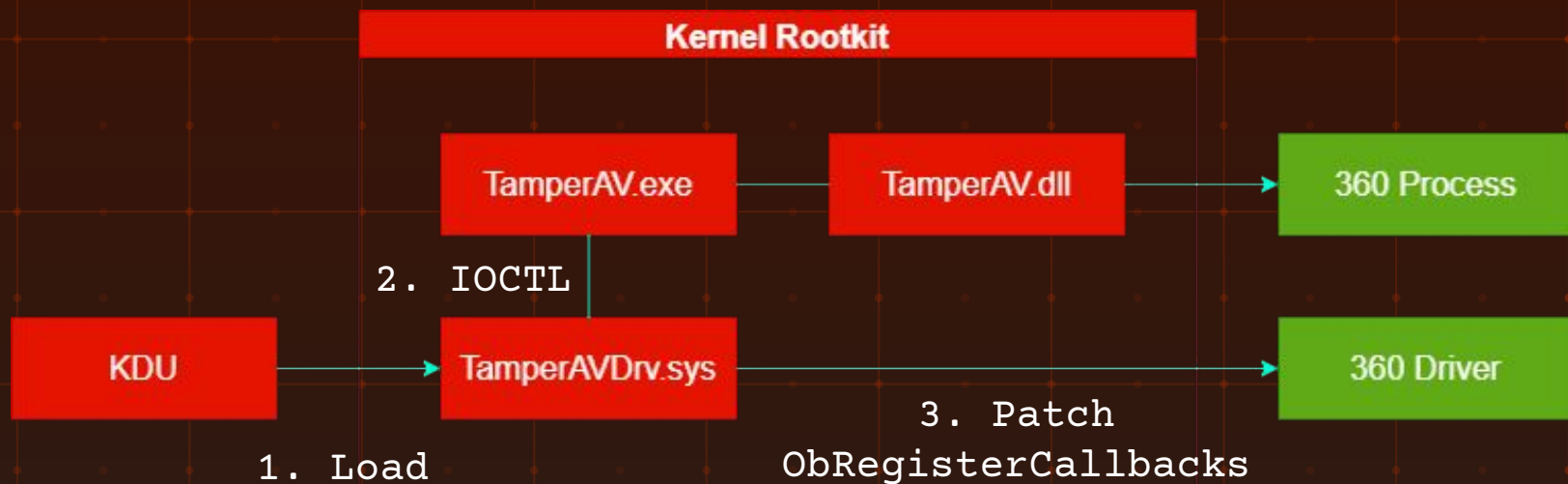
TamperAV



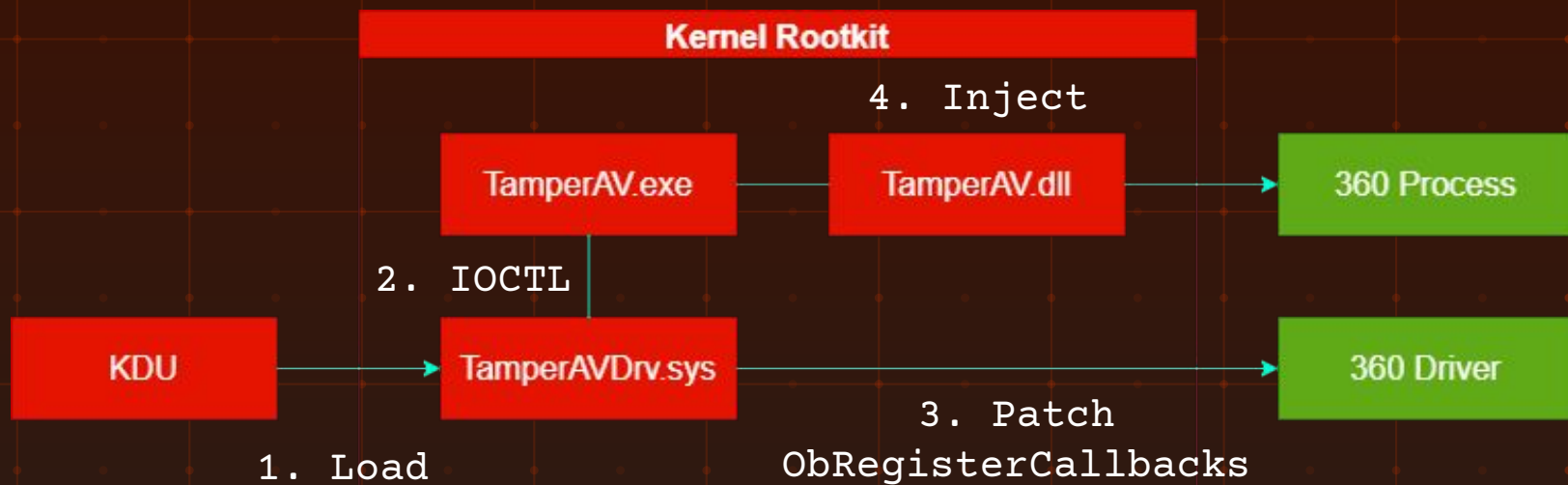
TamperAV



TamperAV

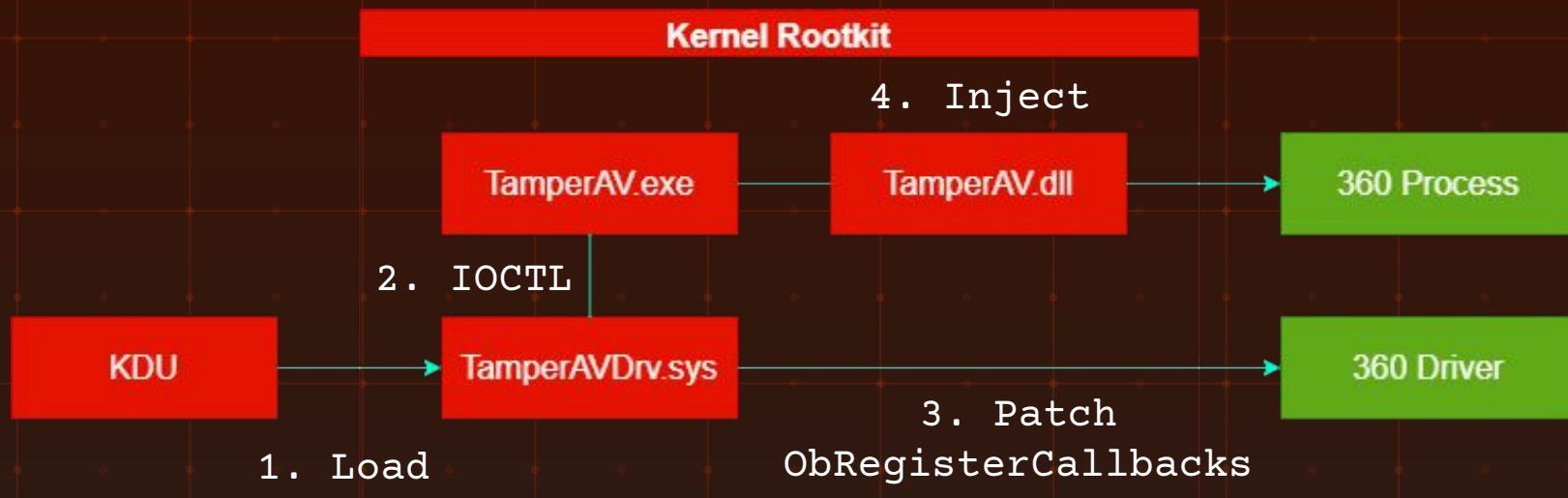


TamperAV



TamperAV

5. Hide and deny access to TamperAV files.





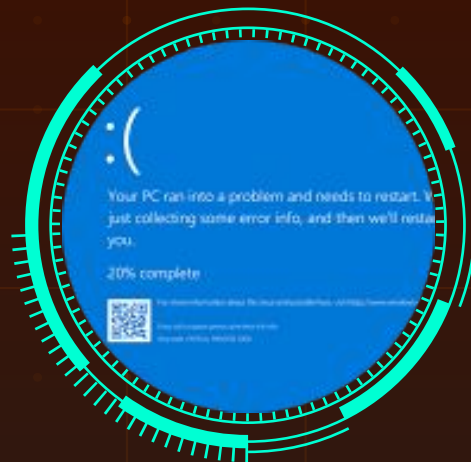
DEMO

Special Thanks



Kenny

TeamT5 Kernel Security
Lab leader and mentor.



Windows BSOD

Crash caused by
unstability of system

THANKS

Any questions?

zezectf@gmail.com

+886 989325139



CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon** and infographics & images by **Freepik**