ConfidEncE 2011
24-25 may 2011, Krakow

Defeating x64: The Evolution of the TDL Rootkit

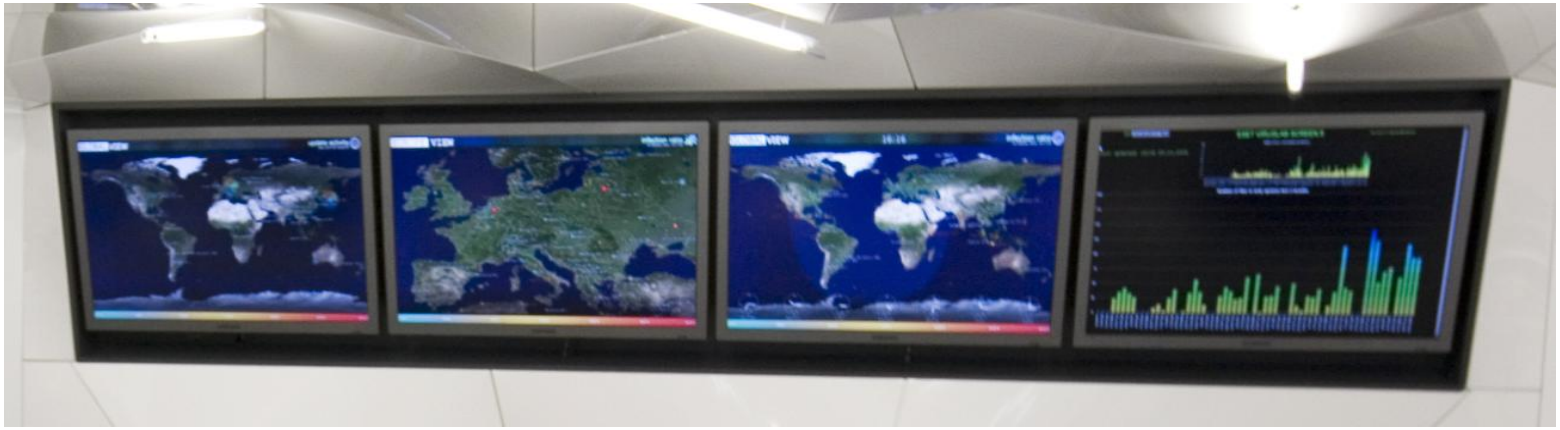
Aleksandr Matrosov

Eugene Rodionov

eset

Who we are?

- **Malware researchers at ESET**
  - **rootkits analysis**
  - **developing cleaning tools**
  - **tracking new rootkit techniques**
  - **research cybercrime groups**



http://www.joineset.com/

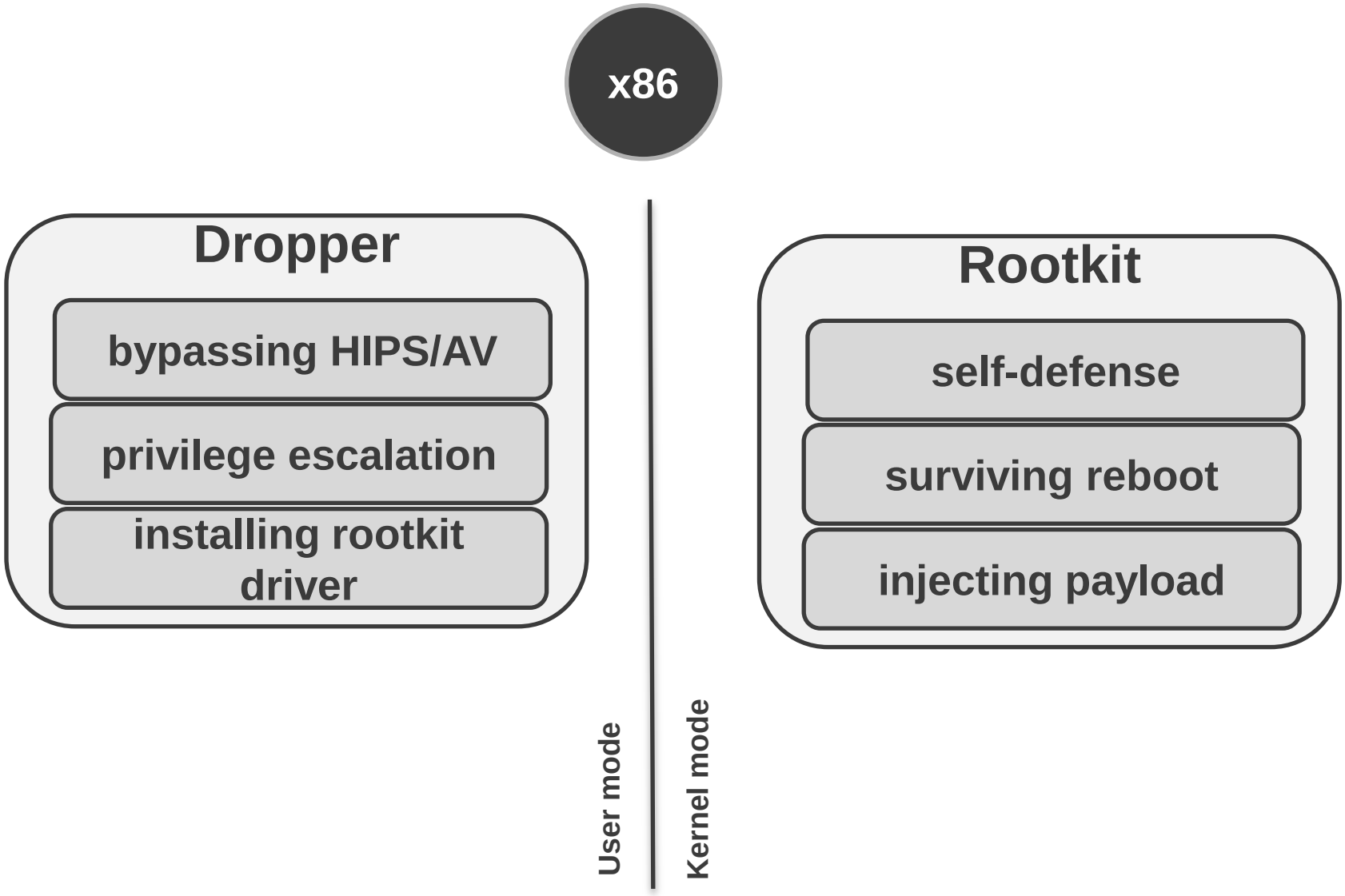
Agenda

- ✓ Evolution of TDL rootkits
- ✓ Installation on x86 vs. x64
- ✓ TDL bootkit or how to bypass driver signature check
- ✓ How to debug bootkit with Bochs emulator
- ✓ Kernel-mode hooks
- ✓ TDL hidden file system layout
- ✓ Payload injection

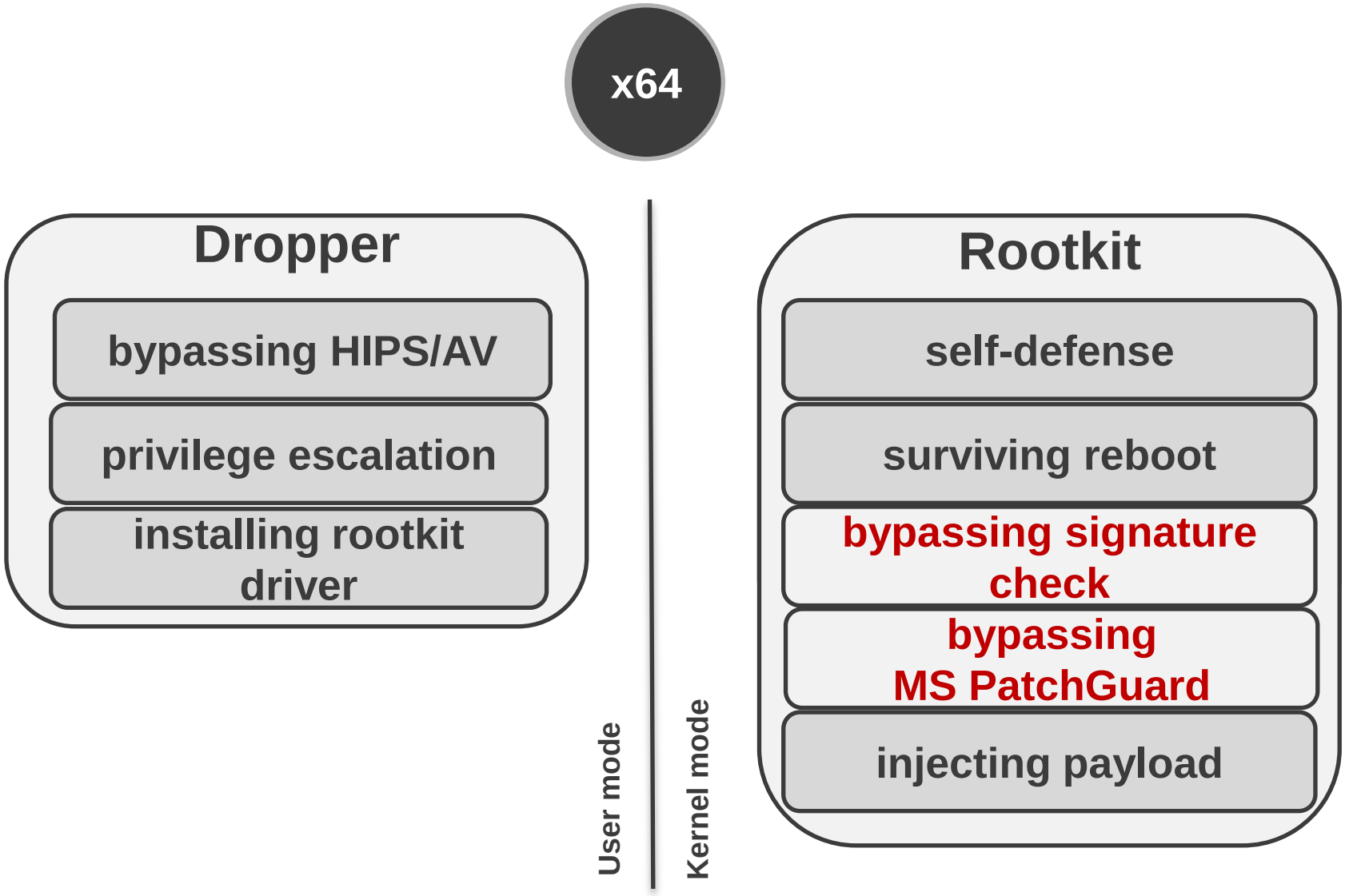
- ✓ TdlFsReader as a forensic tool

Evolution of rootkits

Evolution of rootkits features



Evolution of rootkits features



Obstacles for 64-bit rootkits

- **Kernel-Mode Code Signing Policy**
  - ✓ **It is “difficult” to load unsigned kernel-mode driver**
- **Kernel-Mode Patch Protection (Patch Guard):**
  - ✓ **SSDT (System Service Dispatch Table)**
  - ✓ **IDT (Interrupt Descriptor Table)**
  - ✓ **GDT (Global Descriptor Table)**
  - ✓ **MSRs (Model Specific Registers)**

Evolution of TDL rootkits

	TDL3/TDL3+	TDL4
Kernel-mode code representation	Base independent piece of code in hidden file system	PE image in the hidden file system
Surviving after reboot	Infecting disk miniport/random kernel-mode driver	Infecting MBR of the disk
Self-defense	Kernel-mode hooks, registry monitoring	Kernel-mode hooks, MBR monitoring
Injecting payload into processes in the system	tdlcmd.dll	cmd.dll/cmd64.dll
x64 support	☒	☑
Complexity	■■■□□	■■■■□

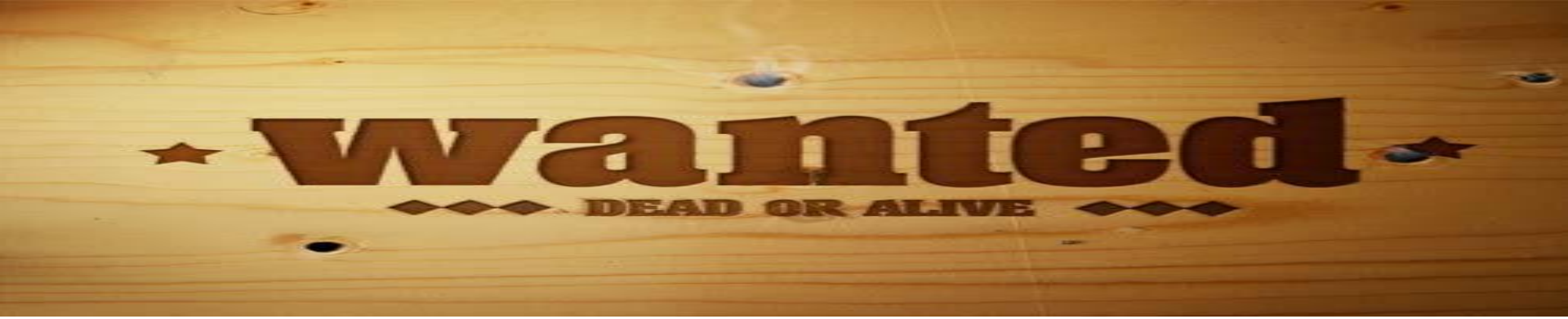
Evolution of TDL rootkits

	TDL3/TDL3+	TDL4
Bypassing HIPS	AddPrintProcessor AddPrintProvidor	AddPrintProvidor, ZwConnectPort
Privilege Escalation	☒	MS10-092
Installation mechanism	by loading kernel-mode driver	by loading kernel-mode driver overwriting MBR of the disk
Number of installed modules	4	10

Installation on x86 vs. x64

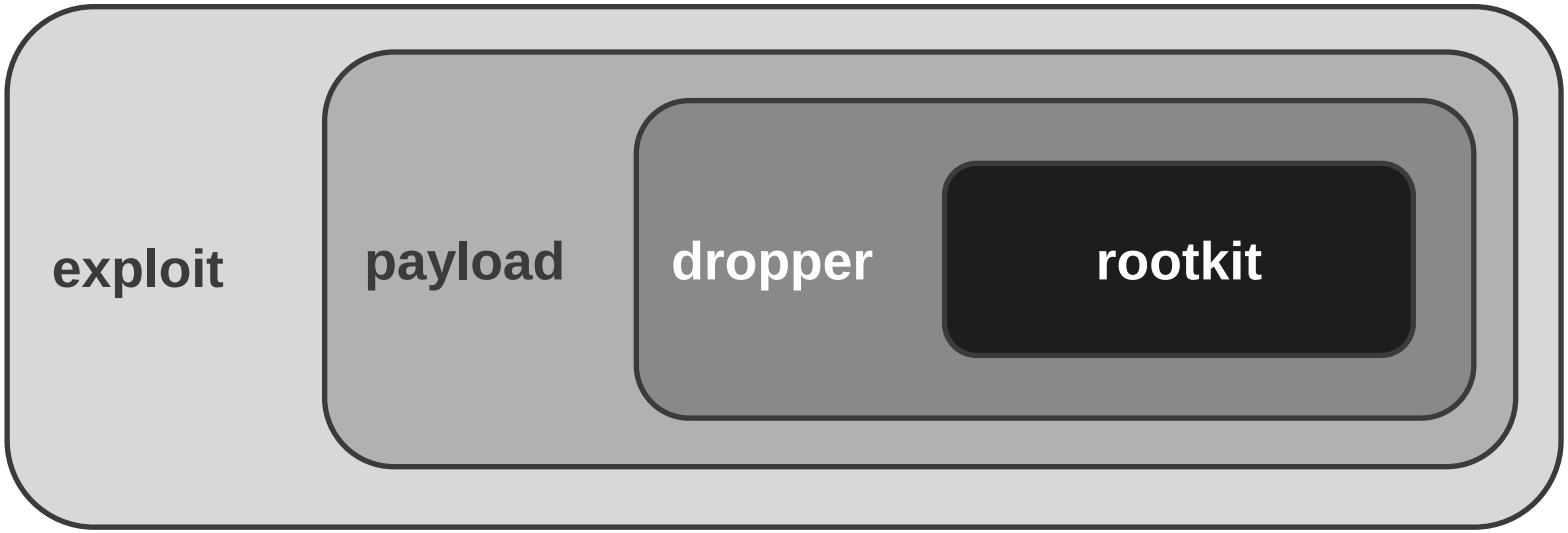
wanted

DEAD OR ALIVE


wanted
DEAD OR ALIVE


DOGMA
Главная
Регистрация
FAQ
Топ10
Саппорт
Rus | Eng
GangstaBucks.com
Statistic
Home
Conditions
Registration
Tariffs
Contacts
Наши преимущества
- Лучший выхлоп среди аналогичных решений
- Стабильные выплаты
- Надежность сотрудничества
- Индивидуальный подход
- Дружественный саппорт
- Активное совершенствование конвертации
Дополнительная информация
Успешно конвертируем следующие страны: US, CA, AU, GB, DE, FR. Увеличена долгосрочность работы и выхлоп с каждого инсталла. Мы готовы предложить индивидуальные рейты и условия оплаты постоянным партнерам. Вы можете использовать собственные лендинги для слива веб траффика.
Стандартные условия
Вы получаете 60% от общего дохода с инсталлов.
Вы получаете 3% от дохода привлеченных Вами мастеров.
Стабильные выплаты 2 раза в месяц, 1-го и 16-го числа.
качества работы партнёрки, проводится техническое переоснащение текущих серверов, а так же добавление новых. В следствии этого некоторое время возможна разбежность баланса с текущим заработком. Данная погрешность будет ликвидирована по окончанию технических работ. Ожидаемый срок завершения до 1.04.2010
20-01-2010
возросшие инсталы
В данный момент так как мы исправили проблему отстука к нам возвращается часть старых инсталов и поэтому, если вы видите что у вас прибавились инсталы сверх нормы - это доходят старые сделанные вами когдато инсталы.

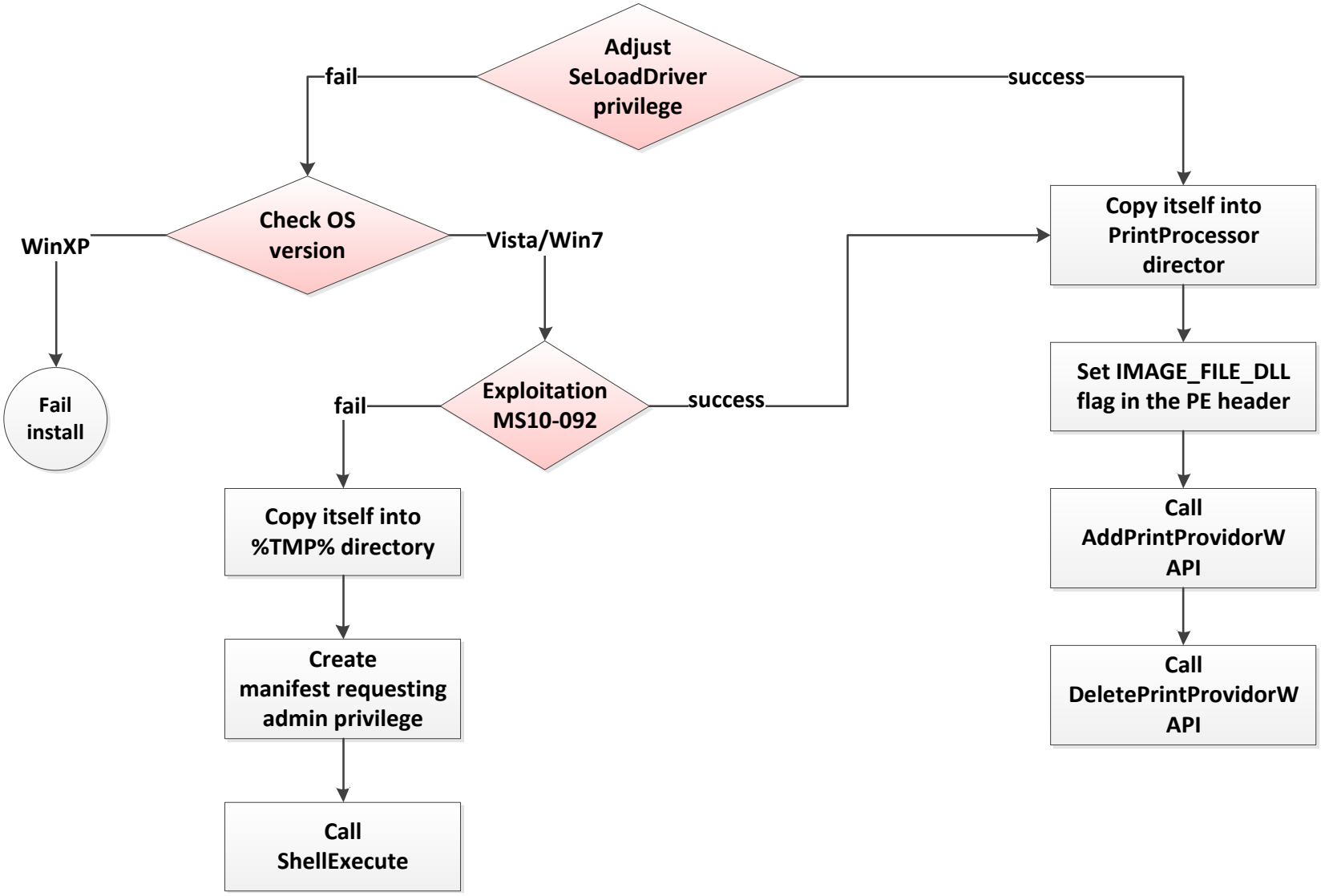
Installation stages



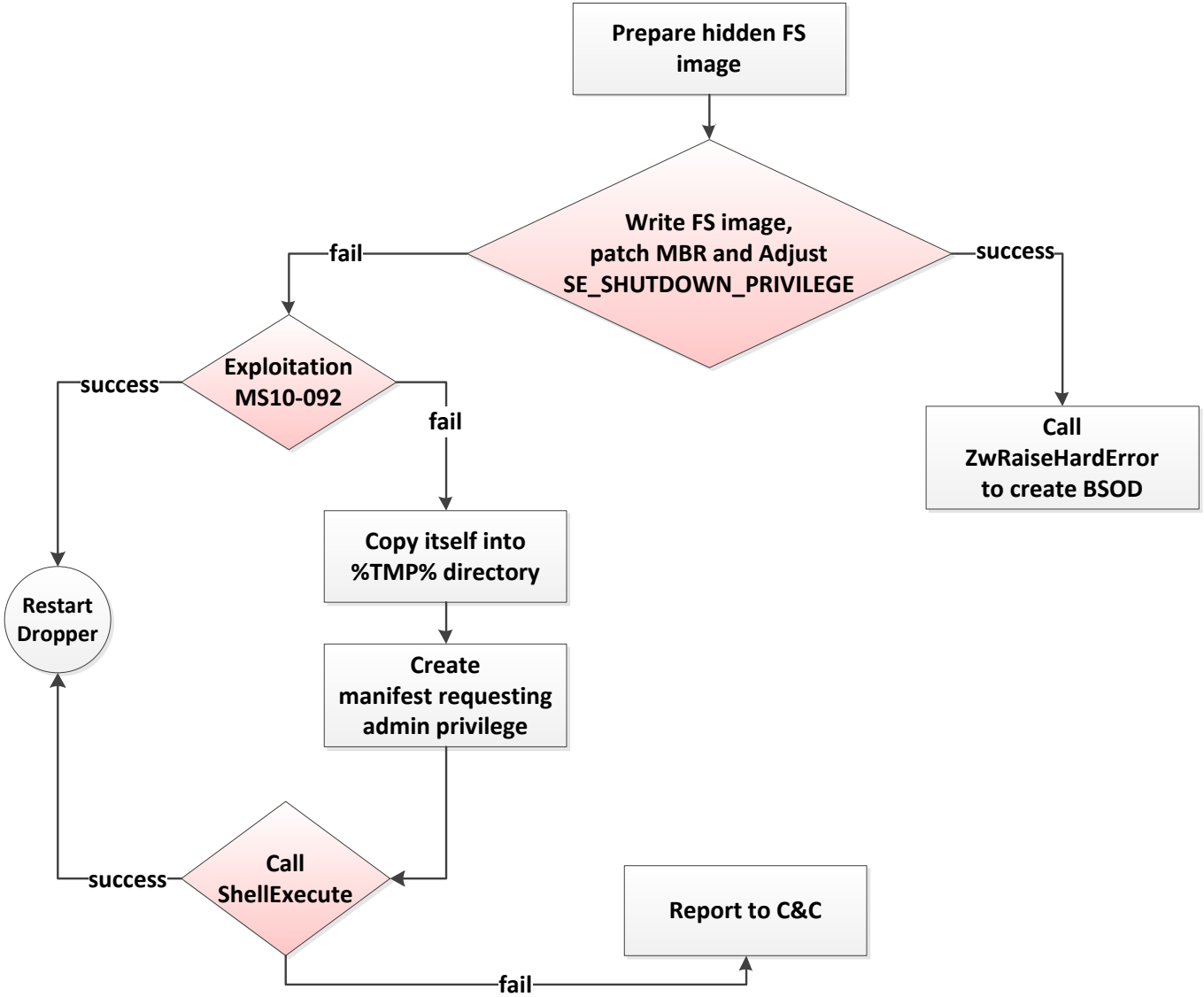
Dropped modules

Dropped modules	Description
mbr	original contents of the infected hard drive boot sector
ldr16	16-bit real-mode loader code
ldr32	fake kdcom.dll for x86 systems
ldr64	fake kdcom.dll for x64 systems
drv32	the main bootkit driver for x86 systems
drv64	the main bootkit driver for x64 systems
cmd.dll	payload to inject into 32-bit processes
cmd64.dll	payload to inject into 64-bit processes
cfg.ini	configuration information
bckfg.tmp	encrypted list of C&C URLs

Installation on x86



Installation on x64



TDL bootkit or how to bypass driver signature check

Types of integrity checks

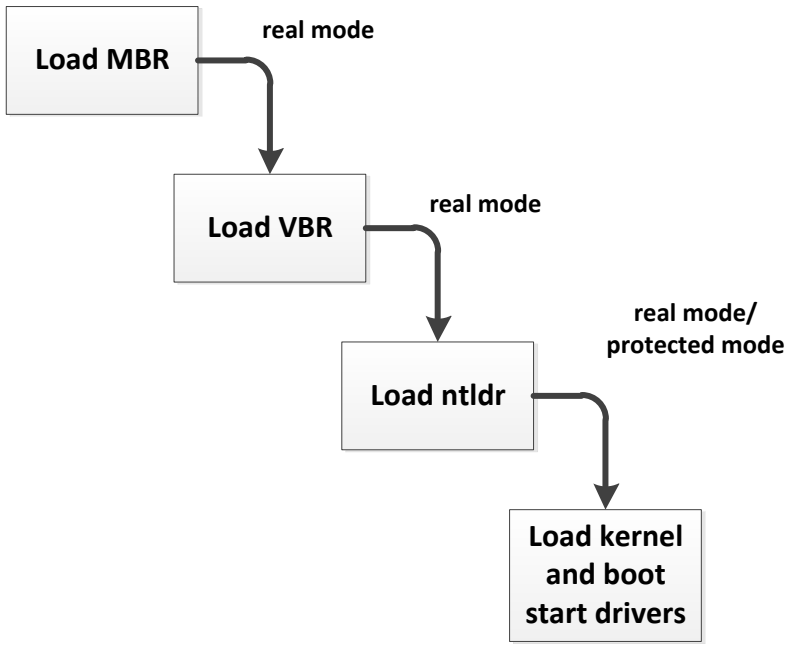
- PnP Device Installation Signing Requirements
- Kernel-Mode Code Signing Policy
  - ✓Enforced on 64-bit version of Windows Vista and later versions

	64-bit Windows Vista and later	32-bit Windows Vista and later
Boot-start driver	☑	☑
Non boot-start PnP driver	☑	☒
Non boot-start, non-PnP driver	☑	☒ (except stream protected media drivers)

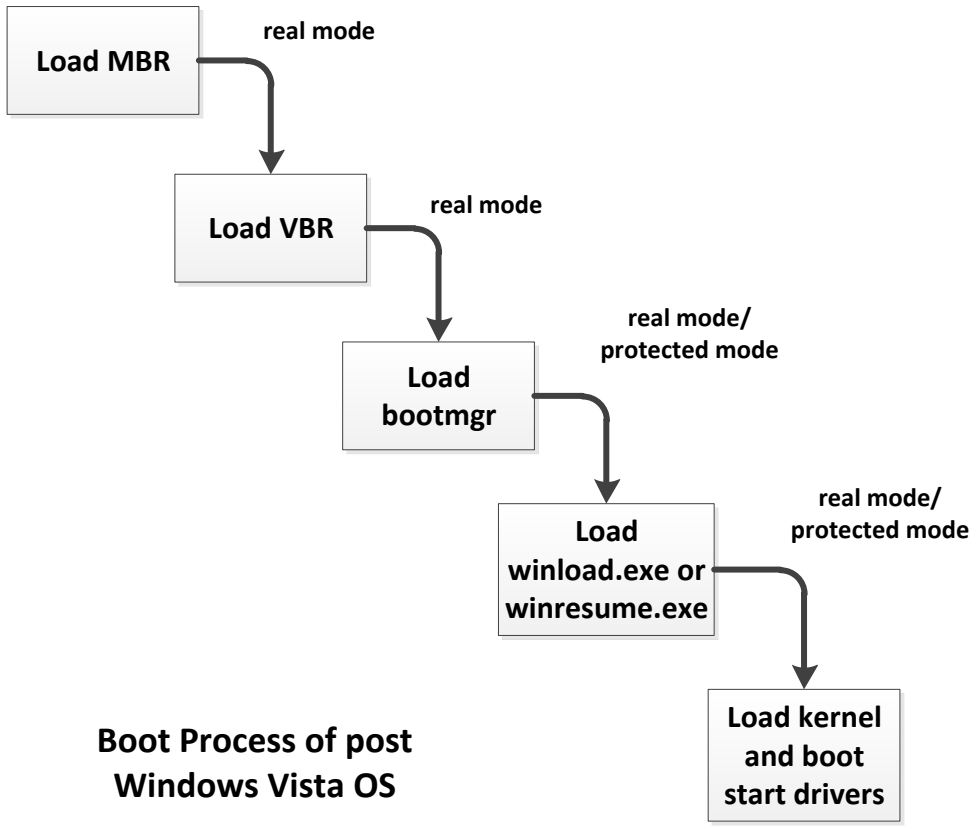
Subverting KMCSP

- Abusing vulnerable signed legitimate kernel-mode driver
- Switching off kernel-mode code signing checks by altering BCD data:
  - ✓abusing WinPe Mode
  - ✓disabling signing check
  - ✓patching Bootmgr and OS loader

Boot process of Windows OS



Boot Process of pre Windows Vista OS

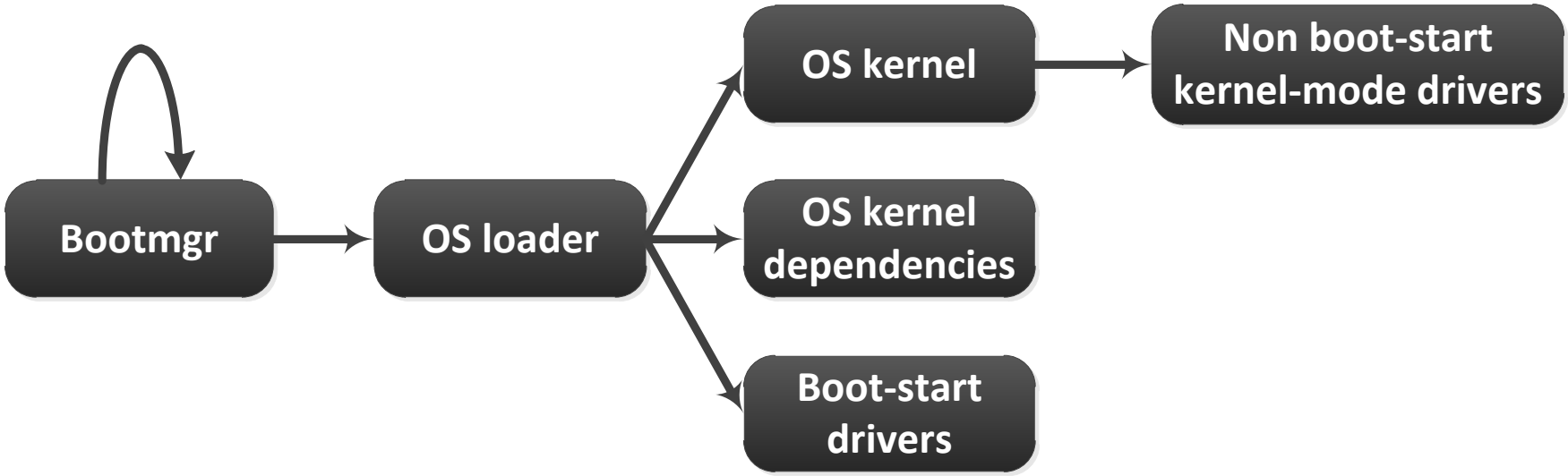


Boot Process of post Windows Vista OS

MBR – Master Boot Record

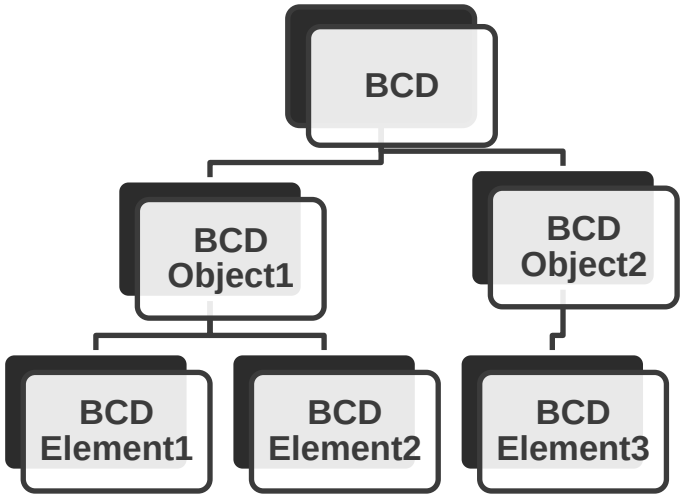
VBR – Volume Boot Record

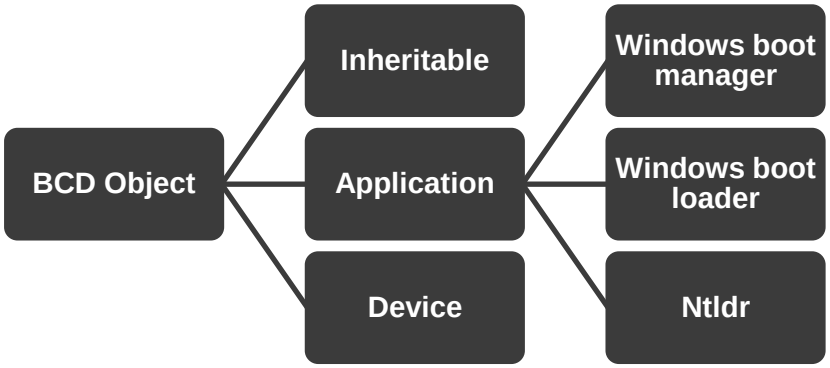
Code integrity check



Boot Configuration Data (BCD)

- HKEY_LOCAL_MACHINE
  - BCD00000000
    - Description
    - Objects
      - {0ce4991b-e6b3-4b16-b23c-5e0d9250e5d9}
      - {1afa9c49-16ab-4a5c-901b-212802da9460}
      - {4636856e-540f-4170-a130-a84776f4c654}
      - {5189b25c-5558-4bf2-bca4-289b11bd29e2}
      - {6567fef2-0e15-11e0-a000-cb5e17119bb9}
      - {6567fef3-0e15-11e0-a000-cb5e17119bb9}
      - {6567fef4-0e15-11e0-a000-cb5e17119bb9}
      - {6567fef5-0e15-11e0-a000-cb5e17119bb9}
      - {6efb52bf-1766-41db-a6b3-0ee5eff72bd7}
      - {7ea2e1ac-2e61-4728-aaa3-896d9d0a9f0e}
      - {7ff607e0-4395-11db-b0de-0800200c9a66}
      - {9dea862c-5cdd-4e70-acc1-f32b344d4795} **Windows Boot Manager**
        - Description
        - Elements
          - 11000001 **BcdLibraryDevice_ApplicationDevice**
          - 12000004 **BcdLibraryString_Description**
          - 12000005 **BcdLibraryString_PreferredLocale**
          - 14000006 **BcdLibraryObjectList_InheritedObjects**
          - 23000003 **BcdBootMgrObject_DefaultObject**
          - 23000006 **BcdBootMgrObject_ResumeObject**
          - 24000001 **BcdBootMgrObjectList_DisplayOrder**
          - 24000010 **BcdBootMgrObjectList_ToolsDisplayOrder**
          - 25000004 **BcdBootMgrInteger_Timeout**
      - {b2721d73-1db4-4c62-bf78-c548a880142d} **Windows Memory Tester**





BCD Elements determining KMCSP
(before KB2506014)

BCD option	Description
BcdLibraryBoolean_DisableIntegrityCheck (0x16000020)	disables kernel-mode code integrity checks
BcdOSLoaderBoolean_WinPEMode (0x26000022)	instructs kernel to be loaded in preinstallation mode, disabling kernel-mode code integrity checks as a byproduct
BcdLibraryBoolean_AllowPrereleaseSignatures (0x16000049)	enables test signing

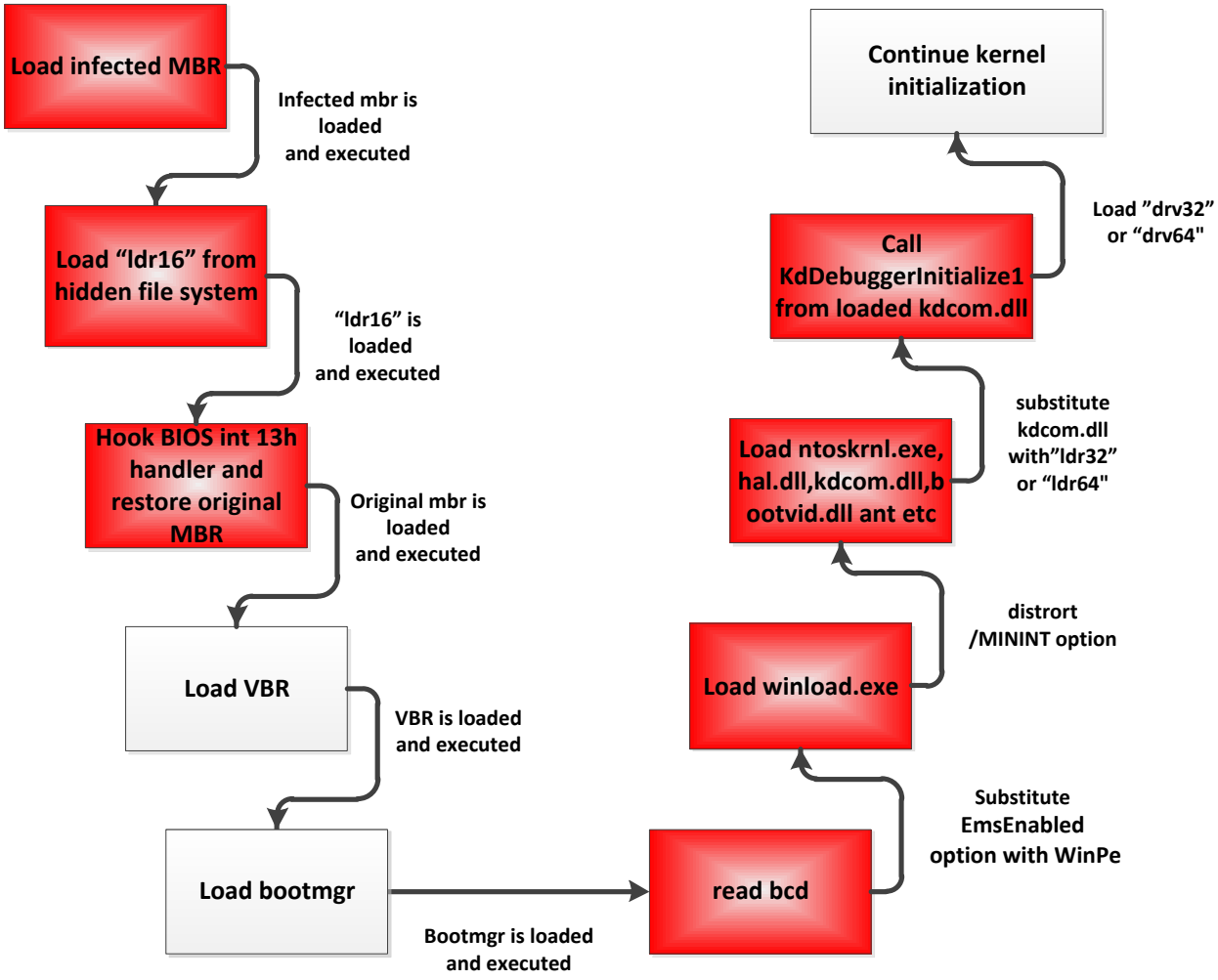
```
; BYTE __stdcall BlImgQueryCodeIntegrityBootOptions(BYTE *a2, BYTE *pOption, BYTE *AllowPreReleaseSign)
_BlImgQueryCodeIntegrityBootOptions@12 proc near
                mov     edi, edi
                push    ebp
                mov     ebp, esp
                push    ecx
                push    esi
                mov     esi, [edx+14h]
                lea     eax, [ebp+var_1]
                push    eax
                push    BcdLibraryBoolean_DisableIntegrityChecks
                push    esi
                call    _BlGetBootOptionBoolean@12 ; BlGetBootOptionBoolean(x,x,x)
                test    eax, eax
                jge     short loc_428742
                mov     [ebp+var_1], 0

loc_428742:                             ; CODE XREF: BlImgQueryCodeIntegrityBootOptions(x,x,x)+1B↑j
                test    byte ptr [edx], 4
                jz      short loc_428764
                cmp     [ebp+var_1], 0
                jnz     short loc_428764
                lea     eax, [ebp+var_1]
                push    eax
                push    BcdLibraryBoolean_WinPEEnabled
                push    esi
                call    _BlGetBootOptionBoolean@12 ; BlGetBootOptionBoolean(x,x,x)
                test    eax, eax
                jge     short loc_428764
                mov     [ebp+var_1], 0

loc_428764:                             ; CODE XREF: BlImgQueryCodeIntegrityBootOptions(x,x,x)+24↑j
                                        ; BlImgQueryCodeIntegrityBootOptions(x,x,x)+2A↑j ...
                mov     eax, [ebp+pOption]
                mov     cl, [ebp+var_1]
                mov     [eax], cl
                lea     eax, [ebp+var_1]
                push    eax
                push    BcdLibraryBoolean_AllowPrereleaseSignatures
                push    esi
                call    _BlGetBootOptionBoolean@12 ; BlGetBootOptionBoolean(x,x,x)
```

Abusing Win PE mode: TDL4 modules

Module name	Description
mbr (infected)	infected MBR loads *ldr16* module and restores original MBR in memory
ldr16	hooks 13h interrupt to disable KMCSP and substitute *kdcom.dll* with *ldr32* or *ldr64*
ldr32	reads TDL4's kernel-mode driver from hidden file system and maps it into kernel-mode address space
ldr64	implementation of *ldr32* module functionality for 64-bit OS

int 13h – service provided by BIOS to communicate to IDE HDD controller

Abusing Win PE mode: workflow



MS Patch (KB2506014)

- ***BcdOsLoaderBoolean_WinPEMode*** **option no longer influences kernel-mode code signing policy**
- **Size of the export directory of *kdcom.dll* has been changed**

MS Patch (KB2506014)

- ***BcdOsLoa*** **…on no longer influences …icy**
- **Size of the … has been changed**

```
BlImgQueryCodeIntegrityBootOptions proc near
    mov     [rsp+arg_8], rbx
    push    rdi
    sub     rsp, 20h
    mov     r11, [rcx+18h]
    mov     rbx, r8
    mov     r10, rdx
    lea     r8, [rsp+28h+arg_0]
    mov     edx, BcdLibraryBoolean_DisableIntegrityCheck
    mov     rcx, r11
    call    BlGetBootOptionBoolean
    movzx   r9d, [rsp+28h+arg_0]
    xor     edi, edi
    cmp     eax, edi
    lea     r8, [rsp+28h+arg_0]
    mov     edx, BcdLibraryBoolean_AllowPrereleaseSignatures
    cmovl   r9d, edi
    mov     rcx, r11
    mov     [rsp+28h+arg_0], r9b
    mov     [r10], r9b
    call    BlGetBootOptionBoolean
    movzx   ecx, [rsp+28h+arg_0]
    cmp     eax, edi
    cmovl   ecx, edi
    mov     [rbx], cl
    mov     rbx, [rsp+28h+arg_8]
    add     rsp, 20h
    pop     rdi
    retn
BlImgQueryCodeIntegrityBootOptions endp
```

MS Patch (KB2506014)

- ***BcdOsLoaderBoolean_WinPEMode*** **option no longer influences kernel-mode code signing policy**
- **Size of the export directory of *kdcom.dll* has been changed**

Ordinal	Function RVA	Name Ordinal	Name RVA	Name
N/A	00001E3C	00001E7A	00001E60	00001EE6
(nFunctions)	Dword	Word	Dword	szAnsi
00000001	00001014	0000	0000608C	KdD0Transition
00000002	00001014	0001	0000609B	KdD3Transition
00000003	00001020	0002	000060AA	KdDebuggerInitialize0
00000004	00001104	0003	000060C0	KdDebuggerInitialize1
00000005	00001228	0004	000060D6	KdReceivePacket
00000006	00001008	0005	000060E6	KdReserved0
00000007	00001158	0006	000060F2	KdRestore
00000008	00001144	0007	000060FC	KdSave
00000009	00001608	0008	00006103	KdSendPacket

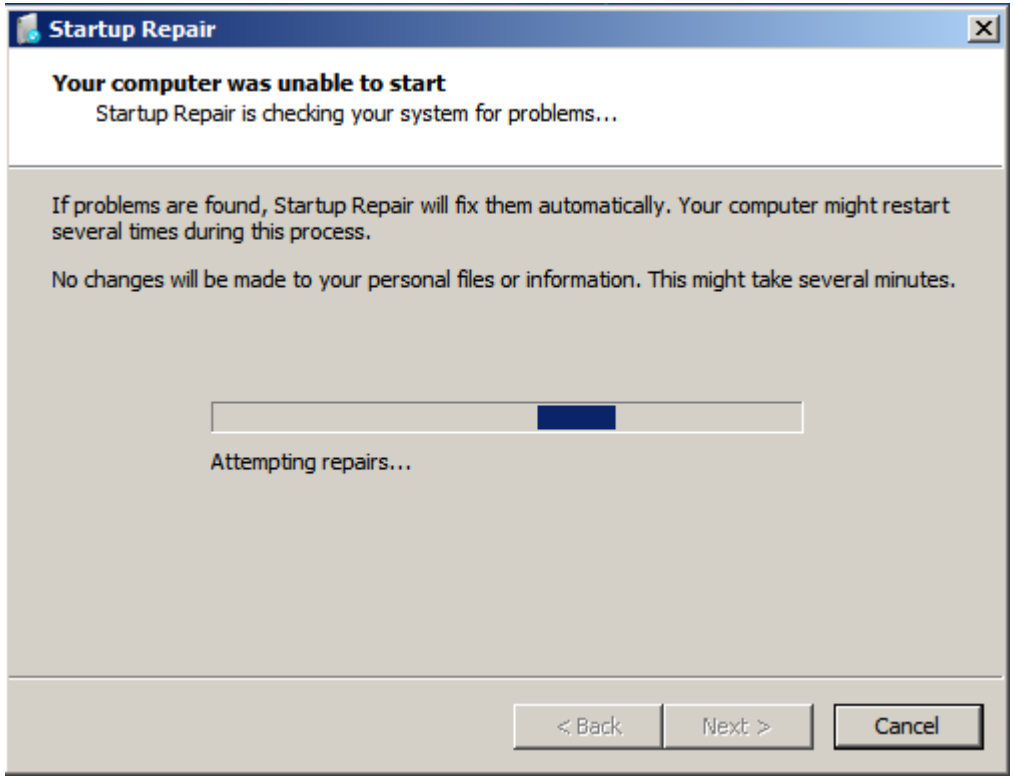
Bypassing KMCSP: another attempt

Patch Bootmgr and OS loader (*winload.exe*) to disable KMCSP:

```
                        ; __int32 __stdcall I_CheckImageHashInCatalog(struct _CRYPTOAPI_BLOB *, unsigned __int8 *const )
                        ?I_CheckImageHashInCatalog@@YAJPEAU_CRYPTOAPI_BLOB@@QEAE@Z proc near
                                                ; CODE XREF: MinCrypL_CheckImageHash+2C↑p
                                                ; MinCrypL_CheckImageHash+52↑p
                                                ; DATA XREF: ...

                        var_88          = dword ptr -88h
                        var_80          = qword ptr -80h
                        var_78          = byte ptr -78h
                        var_68          = dword ptr -68h
                        var_28          = dword ptr -28h
                        Source2         = qword ptr -20h
                        var_18          = byte ptr -18h
                        arg_0           = dword ptr  8
                        arg_8           = qword ptr  10h
                        arg_10          = qword ptr  18h

48 89 5C 24 10                          mov     [rsp+arg_8], rbx
48 89 6C 24 18                          mov     [rsp+arg_10], rbp
56                                      push    rsi
57                                      push    rdi
41 54                                   push    r12
48 81 EC 90 00 00 00                    sub     rsp, 90h
8B 19                                   mov     ebx, [rcx]
48 8B 69 08                             mov     rbp, [rcx+8]
4C 8B E2                                mov     r12, rdx
85 DB                                   test    ebx, ebx
BF 28 04 00 C0                          mov     edi, 0C0000428h ; STATUS_INVALID_IMAGE_HASH
```

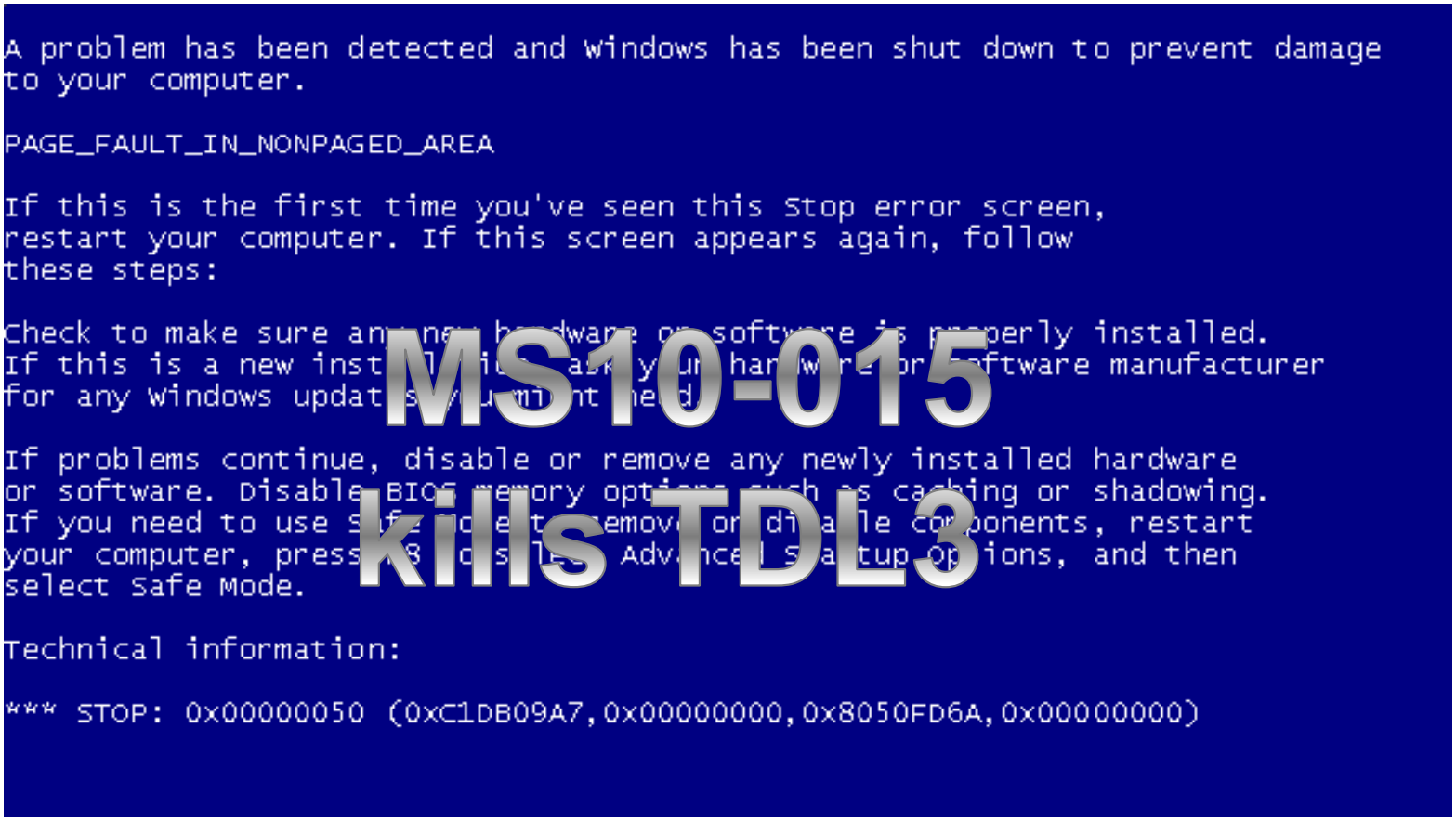
Bypassing KMCSP: Result

Bootmgr fails to verify OS loader's integrity



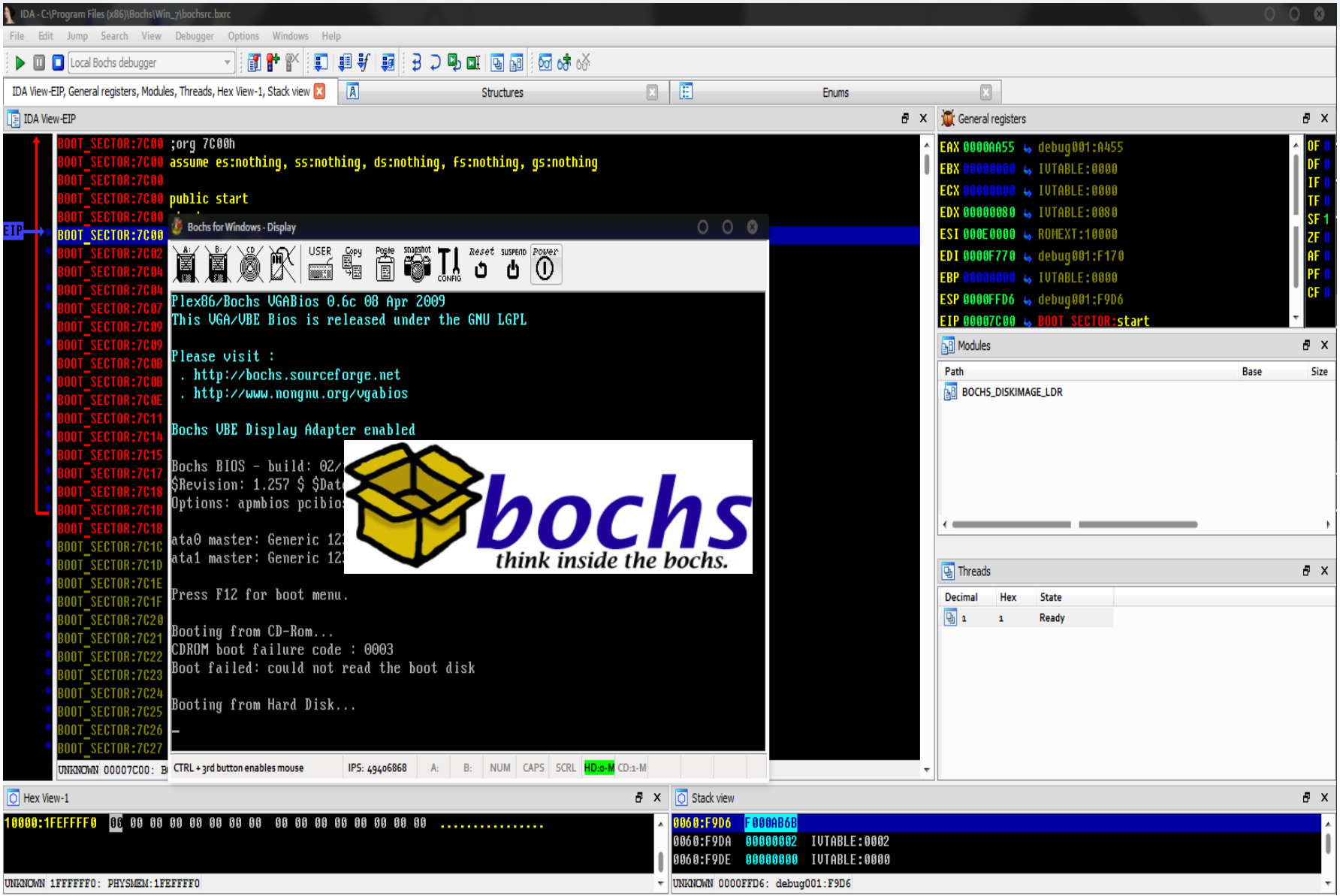
Bypassing KMCSP: Result

Bootmgr fails to verify OS loader's integrity



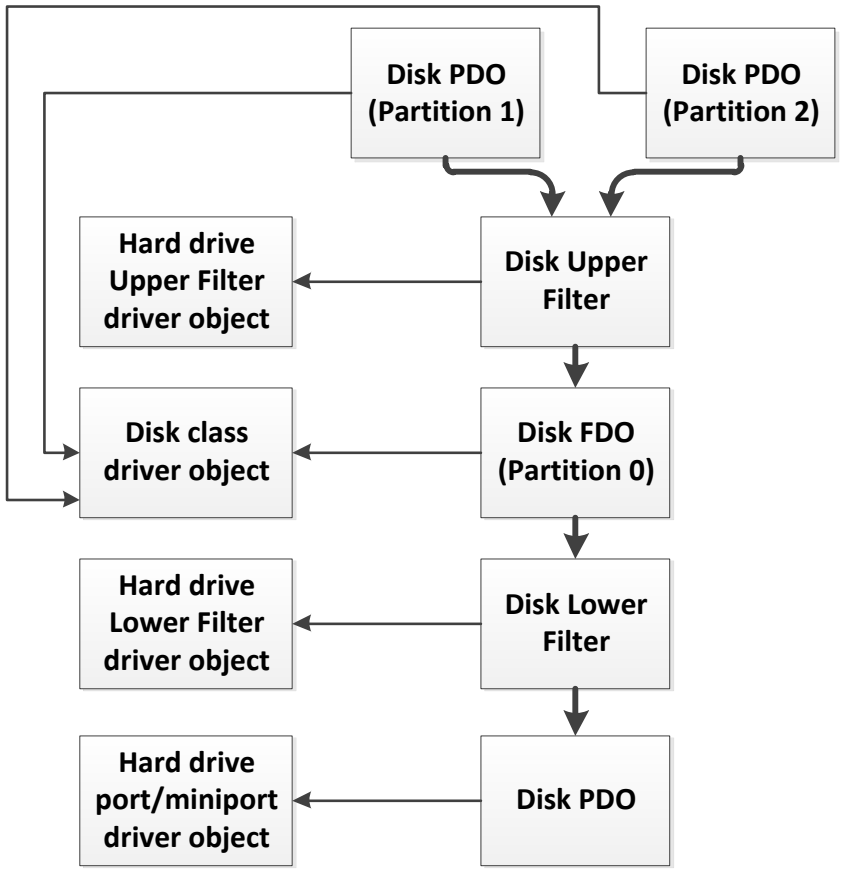
Debugging the bootkit with Bochs

Bochs support starting from IDA 5.5



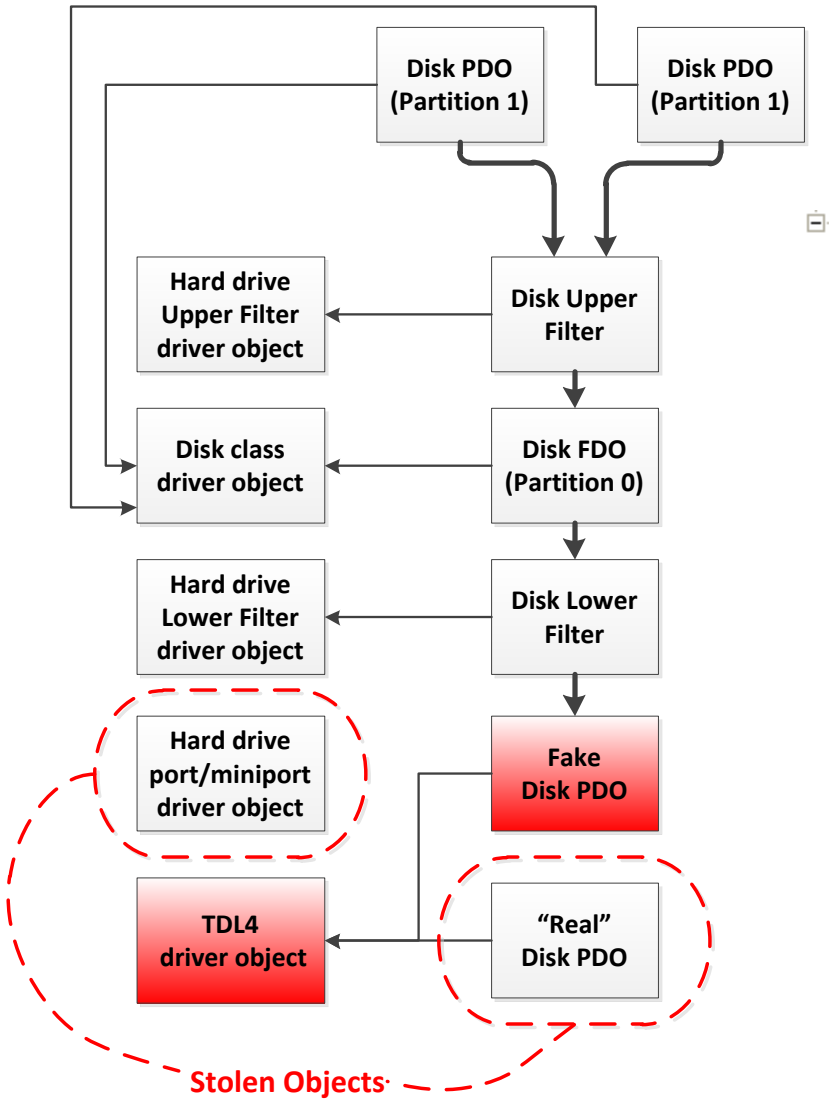
Kernel-mode hooks

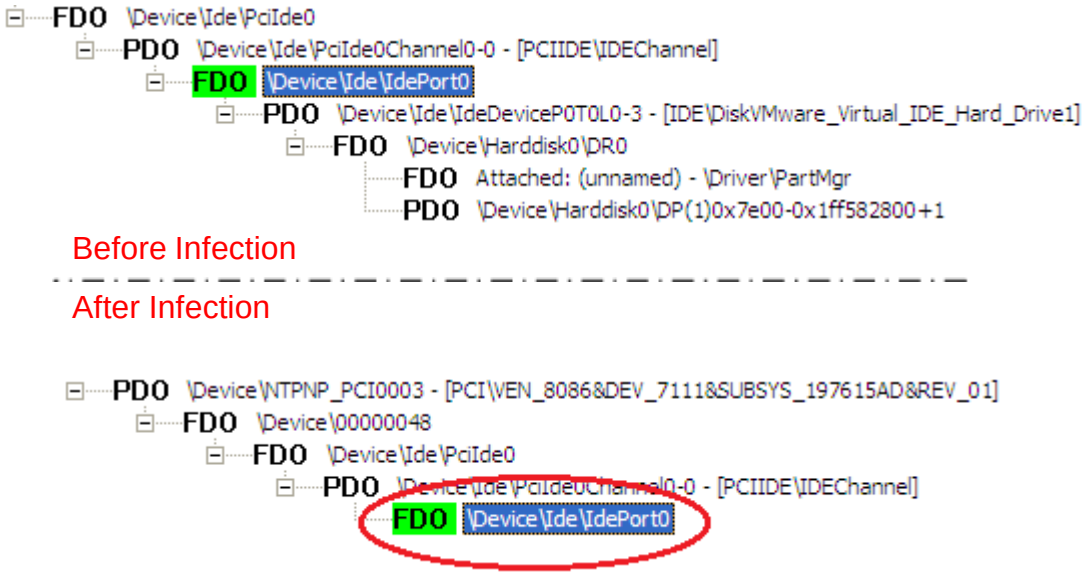
Storage Device Stack



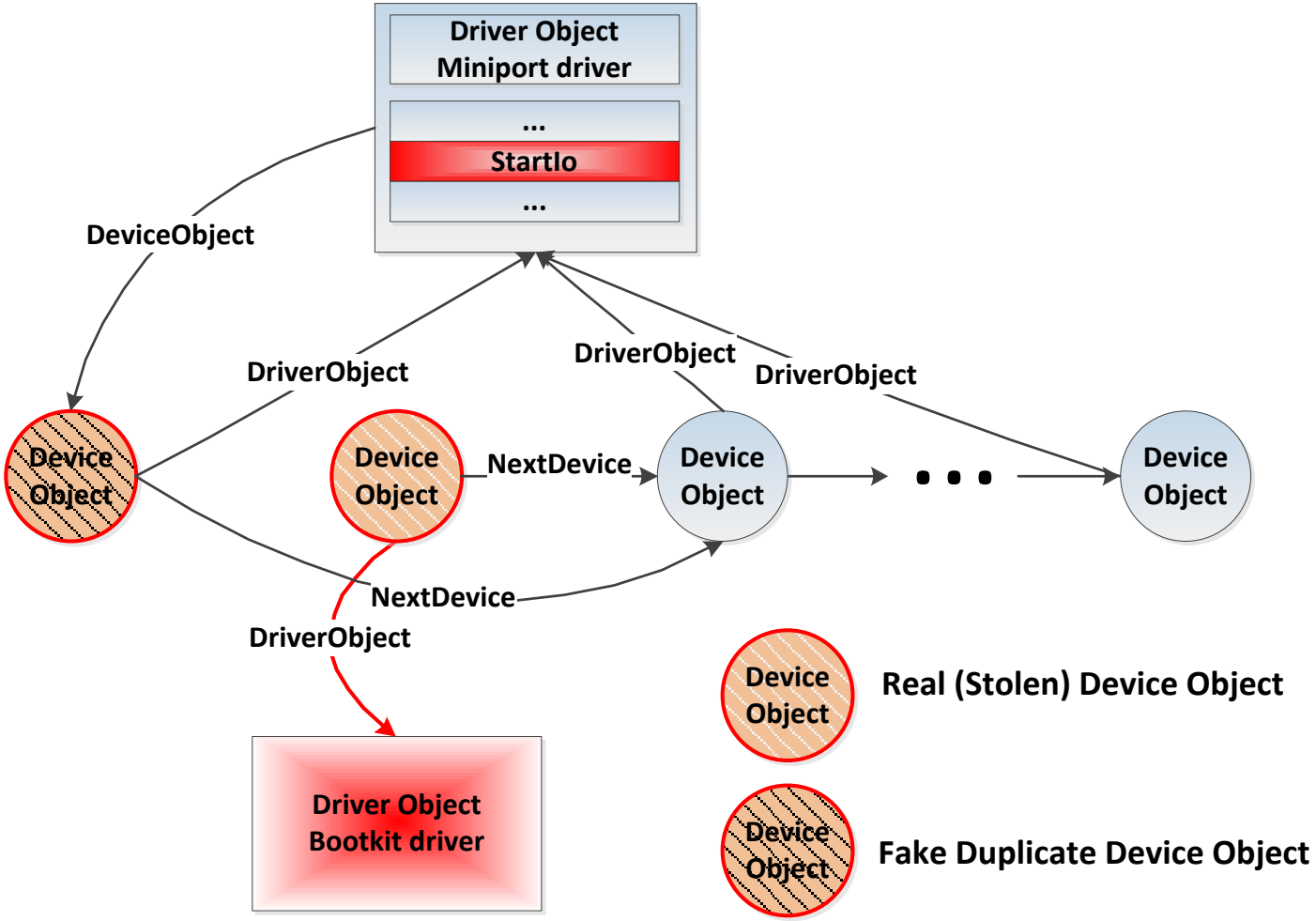
PDO \Device\NTPNP_PCI0003 - [PCI\VEN_8086&DEV_7111&SUBSYS_197615AD&REV_01]
FDO \Device\00000047
FDO \Device\Ide\PciIde0
PDO \Device\Ide\PciIde0Channel0-0 - [PCIIDE\IDEChannel]
FDO \Device\Ide\IdePort0
PDO \Device\Ide\IdeDeviceP0T0L0-3 - [IDE\DiskVMware_Virtual_IDE_Hard_Drive_____00000001] Disk PDO
FDO \Device\Harddisk0\DR0 Disk FDO
FDO Attached: (unnamed) - \Driver\PartMgr Disk Upper Filter
PDO \Device\Harddisk0\DP(1)0x7e00-0x1ff582800+1 Disk PDO (Partition 1)

Stealing Miniport Driver Object

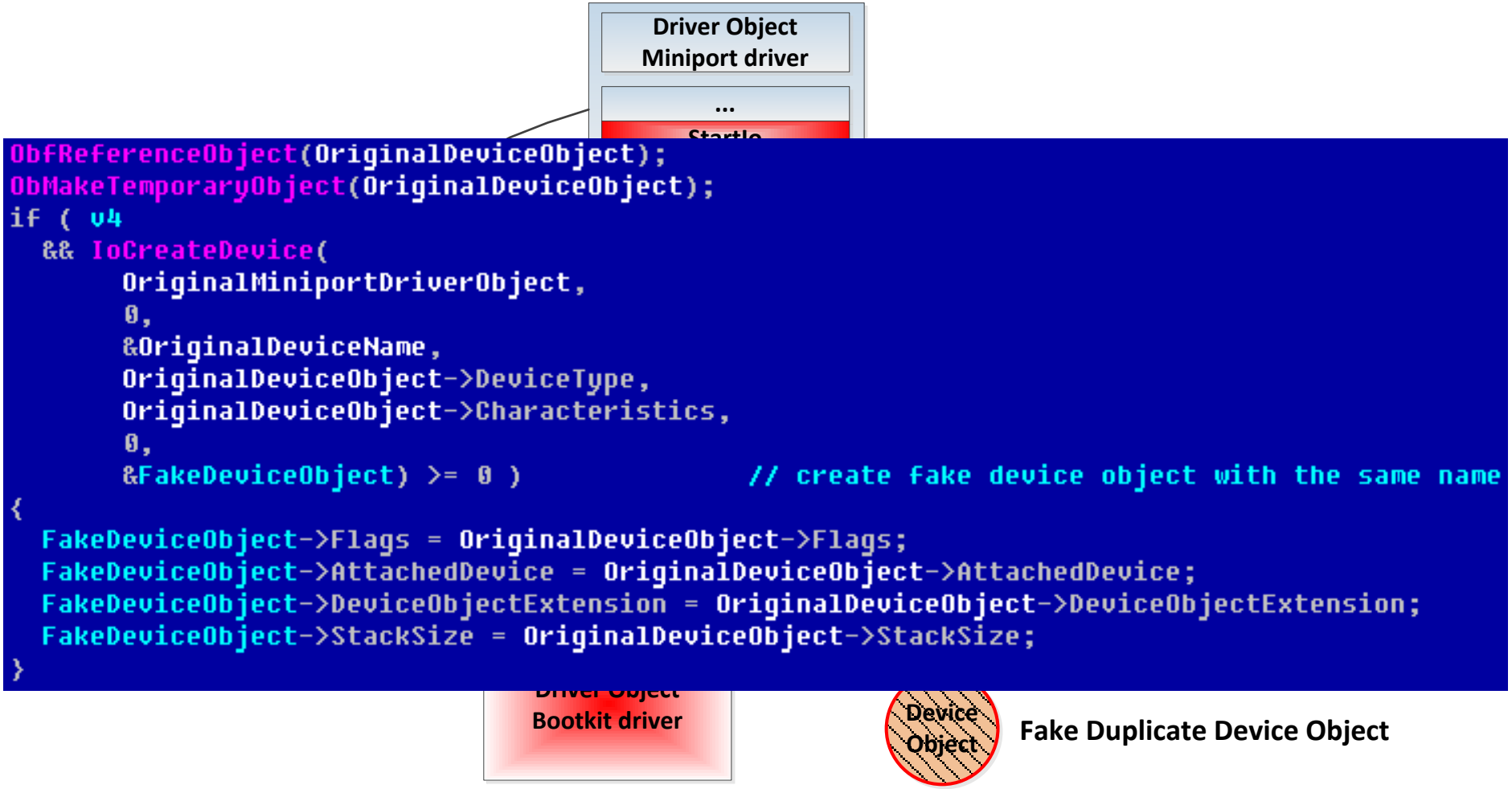




Stealing Miniport Device Object

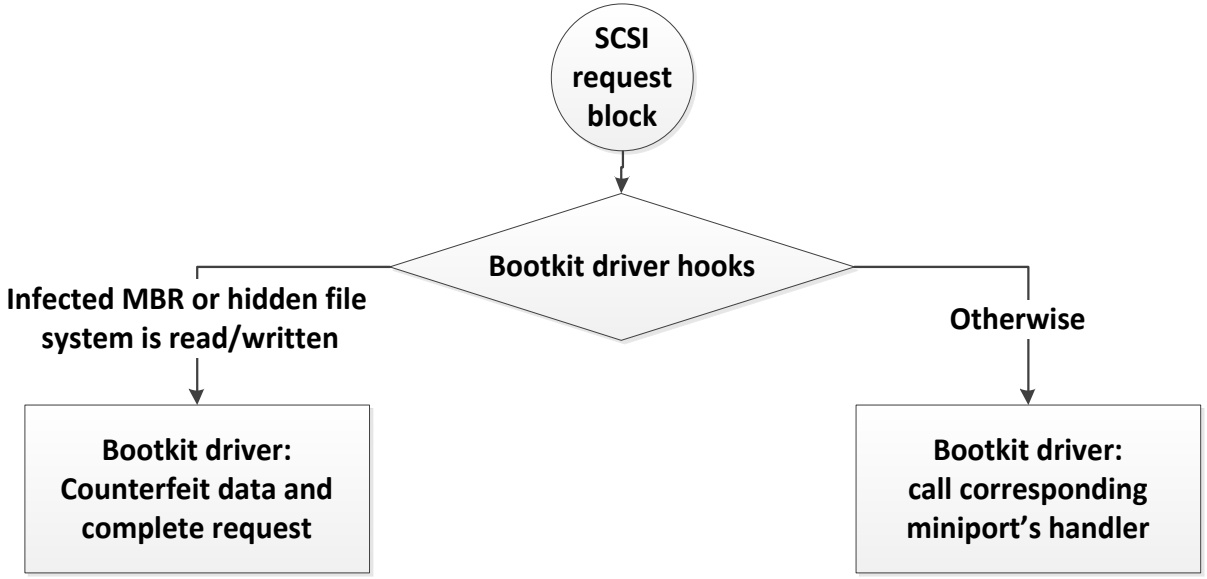


Stealing Miniport Device Object



Filtering Disk Read/Write Requests

- **Filtered requests:**
  - ✓ IOCTL_ATA_PASS_THROUGH_DIRECT
  - ✓ IOCTL_ATA_PASS_THROUGH;
  - ✓ IRP_MJ_INTERNAL_DEVICE_CONTROL
- **To protect:**
  - ✓ Infected MBR;
  - ✓ Hidden file system from being read or overwritten

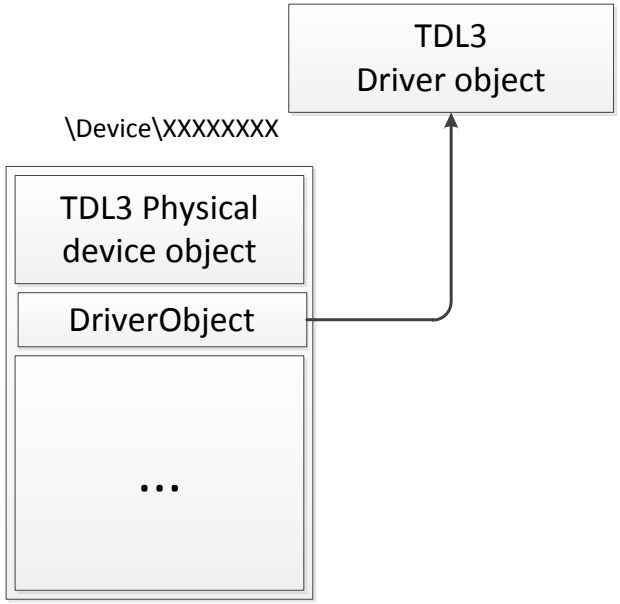


TDL hidden file system

TDL’s hidden storage

- Reserve space in the end of the hard drive (not visible at file system level analysis)
- Encrypted contents (stream cipher: RC4, XOR-ing)
- Implemented as a hidden volume in the system
- Can be accessed by standard APIs (CreateFile, ReadFile, WriteFile, SetFilePointer, CloseHandle)

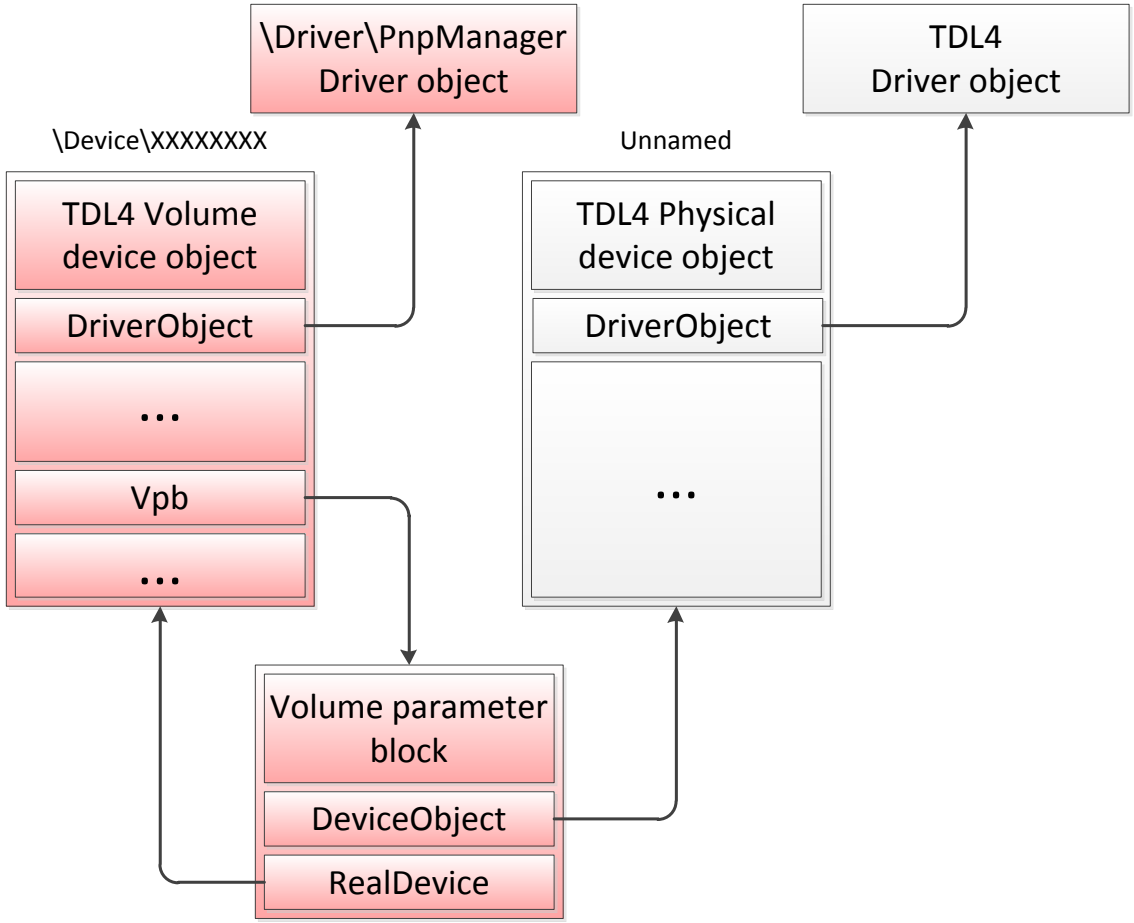
TDL3/TDL3+ Rootkit Device Stack



XXXXXXXX – random 8-character ASCII string

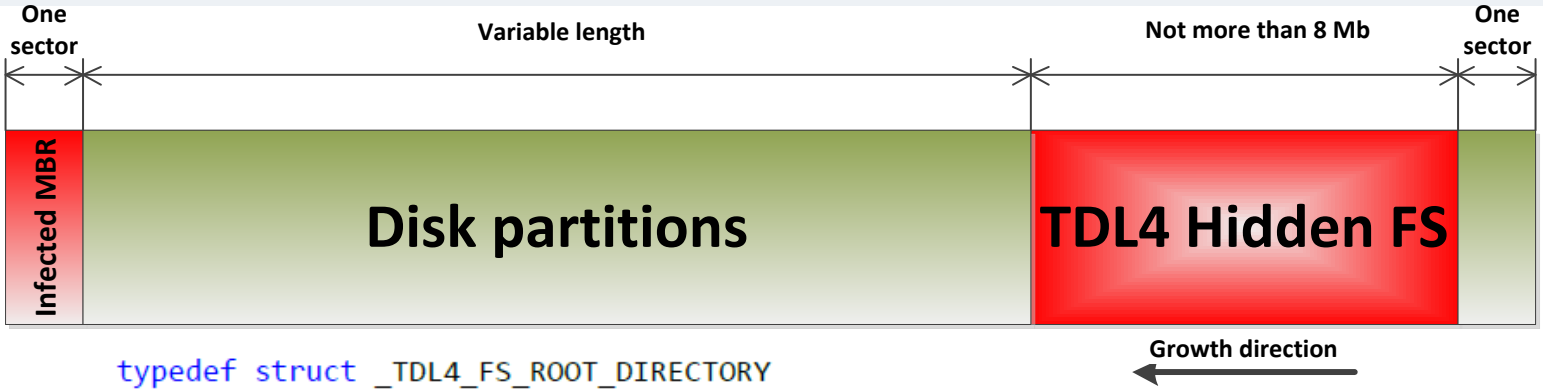
\\?\globalroot\device\XXXXXXXX\YYYYYYYY\file_name - for user-mode components
\device\XXXXXXXX\YYYYYYYY\file_name – for kernel-mode components

TDL4 Device Stack



XXXXXXXX – random 32-bit hexadecimal integer

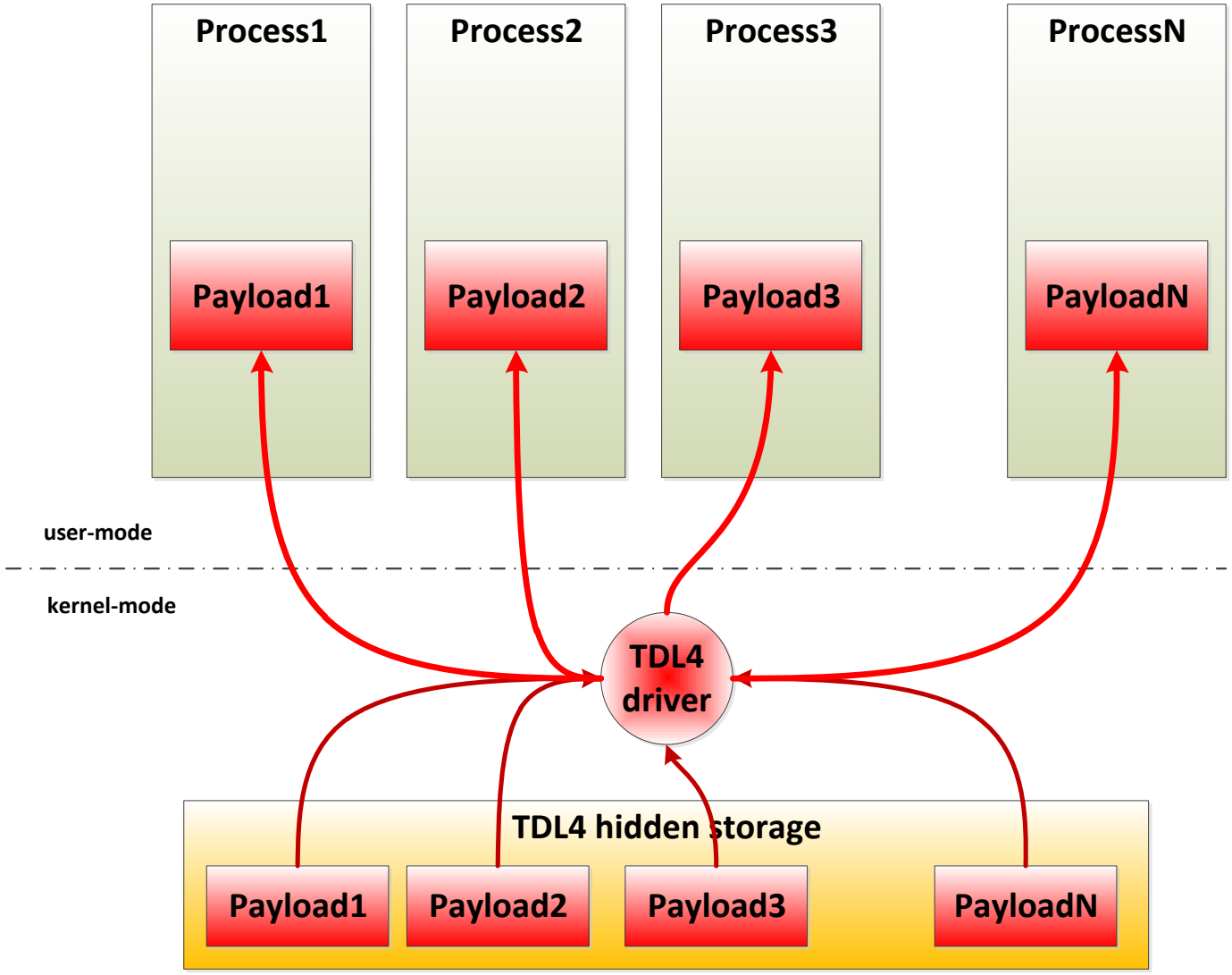
TDL4 File System Layout



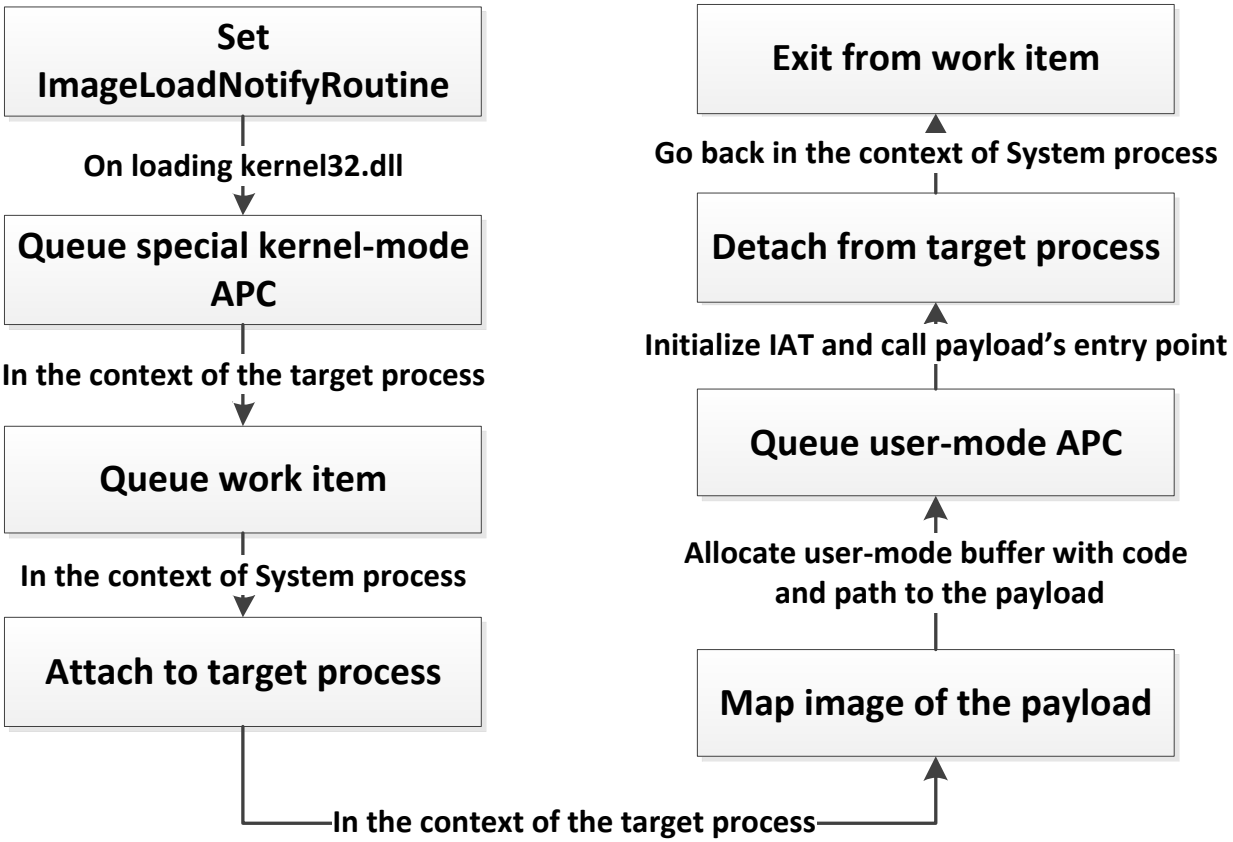

```
typedef struct _TDL4_FS_ROOT_DIRECTORY
{
        // Signature of the block
        // DC - root directory
        WORD Signature;
        // Set to zero
        DWORD Reserved;
        // Array of entries corresponding to files in FS
        TDL4_FS_FILE_ENTRY FileTable[15];
}TDL4_FS_ROOT_DIRECTORY, *PTDL4_FS_ROOT_DIRECTORY;
typedef struct _TDL4_FS_FILE_ENTRY
{
        // File name - null terminated string
        char FileName[16];
        // Offset from beginning of the file system to file
        DWORD FileBlockOffset;
        // Reserved
        DWORD dwFileSize;
        // Time and Date of file creation
        FILETIME CreateTime;
}TDL4_FS_FILE_ENTRY, *PTDL4_FS_FILE_ENTRY;
```

Payload injection

Injecting payload into target process



Injecting workflow



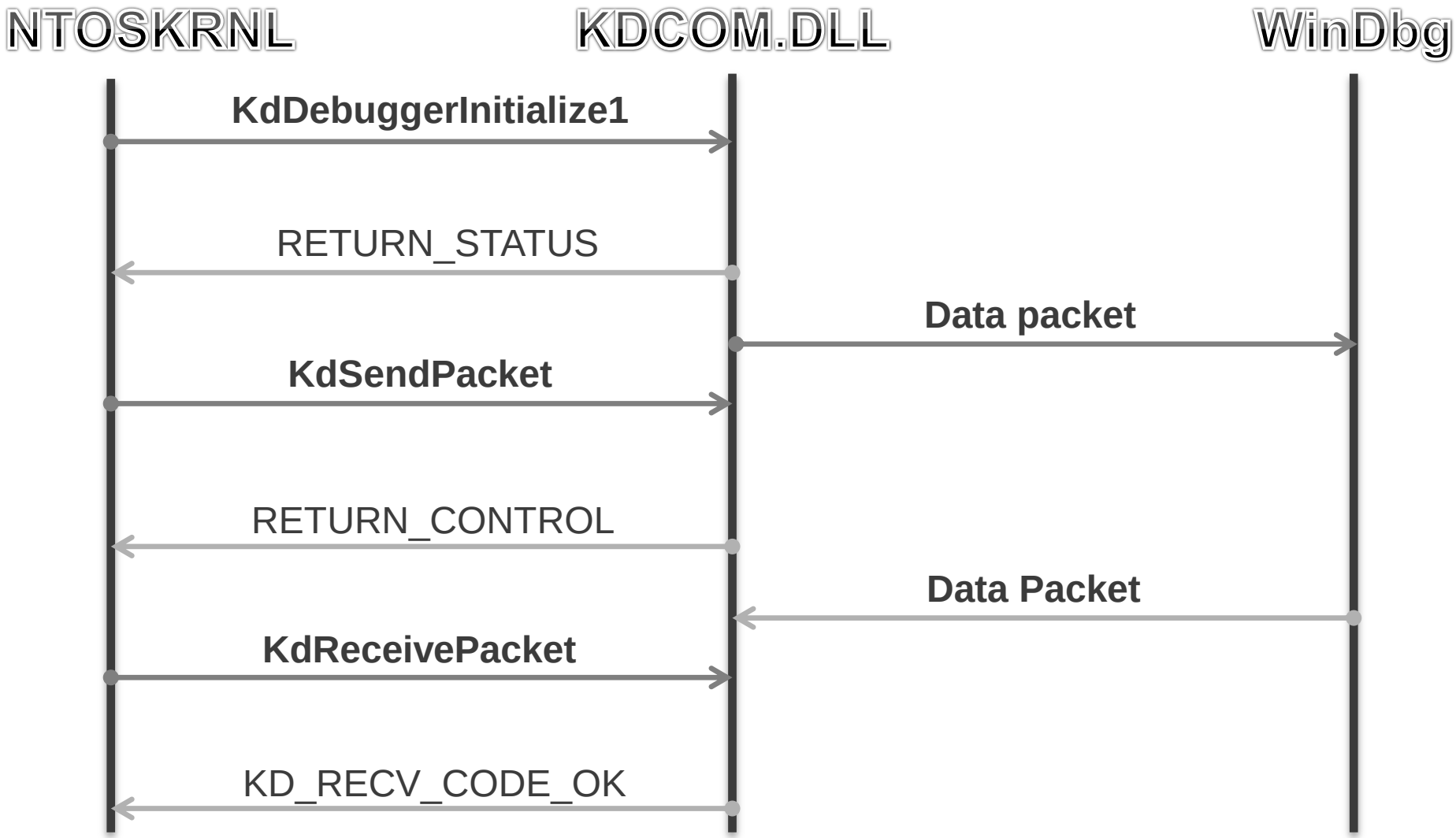
TDL4 cleaning approach

How to remove TDL

- **Defeat self defense:**
  - ✓**Disable WORK_ITEM checking infected MBR and kernel-mode hooks**
  - ✓**Remove hooks of storage miniport device object**
- **Restore original MBR**

Debugging bootkit with WinDbg

WinDbg and kdcom.dll



TDL4 and kdcom.dll

original routine

```
; __stdcall KdDebuggerInitialize1(x)
                public _KdDebuggerInitialize1@4
_KdDebuggerInitialize1@4 proc near      ; DATA XREF: .edata:off_80011328↓o
                call    _KdCompInitialize1@0 ; KdCompInitialize1()
                xor     eax, eax
                retn    4
_KdDebuggerInitialize1@4 endp
```

modified routine

```
                public KdDebuggerInitialize1
KdDebuggerInitialize1 proc near         ; DATA XREF: .text:off_10001058↑o
                push    offset NotifyRoutine ; NotifyRoutine
                call    PsSetCreateThreadNotifyRoutine
                retn    4
KdDebuggerInitialize1 endp
```

```
; void __stdcall NotifyRoutine(HANDLE, HANDLE, BOOLEAN)
NotifyRoutine   proc near               ; DATA XREF: CallbackRoutine+1CE↑o
                                        ; KdDebuggerInitialize1↓o
                cmp     dword_100017F0, 0
                jnz     short locret_1000179D
                push    offset DriverEntry
                push    0
                call    IoCreateDriver
                xor     ecx, ecx
                test    eax, eax
                setns   cl
                mov     dword_100017F0, ecx

locret_1000179D:                        ; CODE XREF: NotifyRoutine+7↑j
                retn    0Ch
NotifyRoutine   endp
```

TDL4 and kdcom.dll

original export table

Name	Address	Ordinal
KdD0Transition	80010386	1
KdD3Transition	80010386	2
KdDebuggerInitialize0	800103A6	3
KdDebuggerInitialize1	8001044C	4
KdReceivePacket	80010F4C	5
KdRestore	80010460	6
KdSave	80010456	7
KdSendPacket	800111B2	8
HalInitSystem(x,x)	80010CE6	

modified export table

Name	Address	Ordinal
KdD0Transition	1000171A	1
KdD3Transition	10001724	2
KdDebuggerInitialize0	100017A0	3
KdDebuggerInitialize1	100017AC	4
KdReceivePacket	100017DC	5
KdRestore	100017C6	6
KdSave	100017BA	7
KdSendPacket	100017D2	8
DriverEntry	1000172E	

```
; void __stdcall NotifyRoutine(HANDLE, HANDLE, BOOLEAN)
NotifyRoutine   proc near               ; DATA XREF: CallbackRoutine+1CE↑o
                                        ; KdDebuggerInitialize1↓o
                cmp     dword_100017F0, 0
                jnz     short locret_1000179D
                push    offset DriverEntry
                push    0
                call    IoCreateDriver
                xor     ecx, ecx
                test    eax, eax
                setns   cl
                mov     dword_100017F0, ecx

locret_1000179D:                        ; CODE XREF: NotifyRoutine+7↑j
                retn    0Ch
NotifyRoutine   endp
```

How to debug TDL4 with WinDbg

- **Patch *ldr16* to disable *kdcom.dll* substitution**
- **Reboot the system and attach to it with WinDbg**
- **Manually load *drv32/drv64***

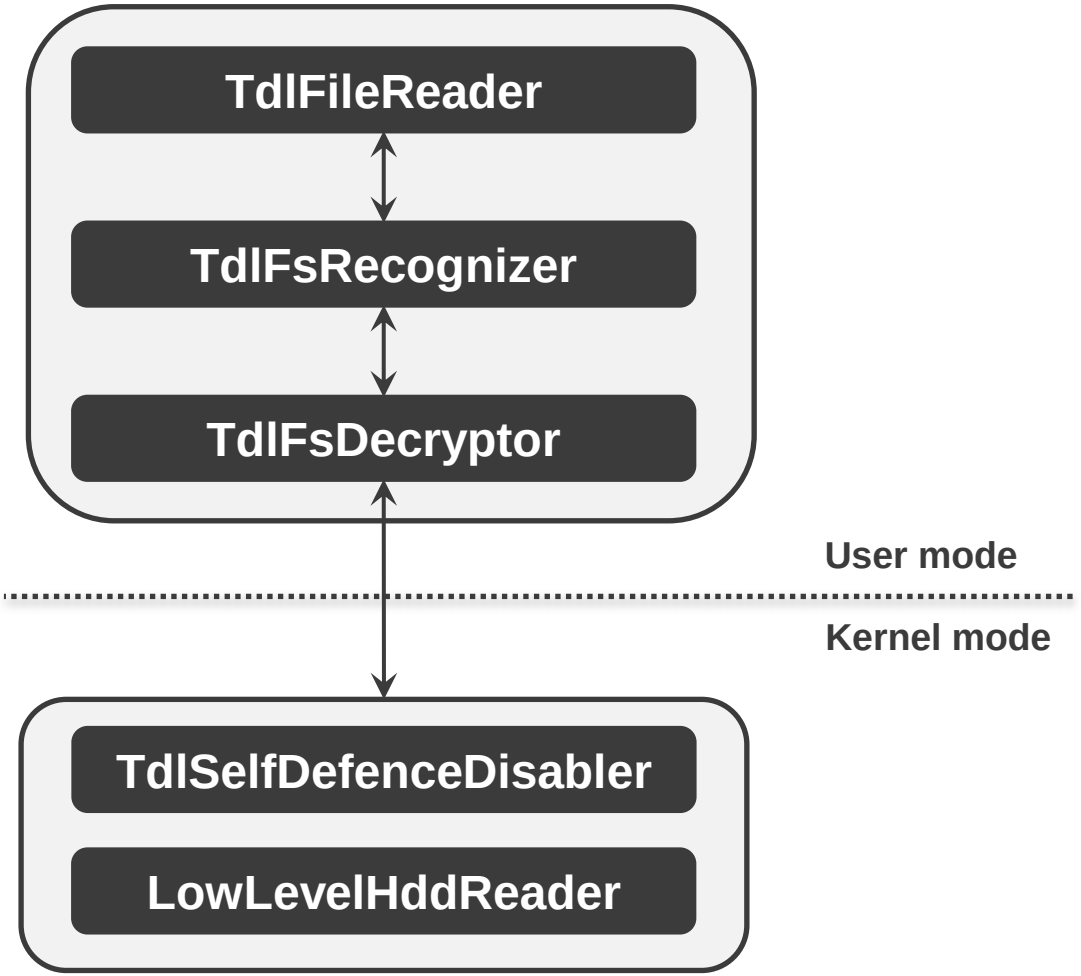
“TDL4 Analysis Paper: a brief introduction and How to Debug It”, Andrea Allievi
http://www.aall86.altervista.org/TDLRootkit/TDL4_Analysis_Paper.pdf

TdlFsReader as a forensic tool

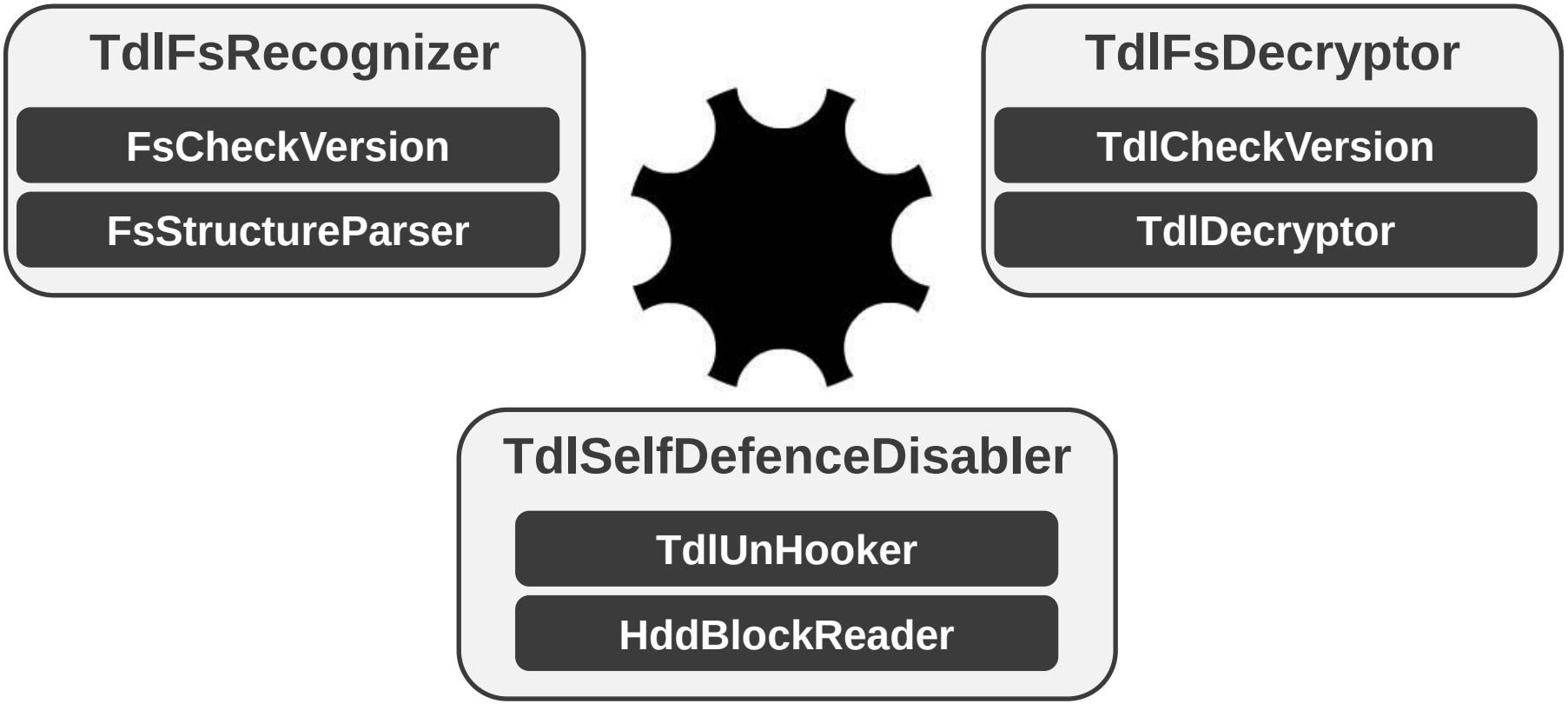
TdlFsReader as a forensic tool

```
C:\>TdlFsReader.exe
Contents of TDL file system:
         cfg.ini MD5: B8C8B1B5C01EBF2F48760F2E06C402E6
             mbr MD5: AF1EC9B9C5CE1D74D3D9CA3BBE0FA941
       bckfg.tmp MD5: 6AD76461EEB59A1D77529B595D3672ED
         cmd.dll MD5: 4DADED6C7EFF7230D68AE48B02A847FA
           ldr16 MD5: 4FB9748189F6688ADA9A51A5901406FA
           ldr32 MD5: C078E5EA19F853AC0830D2F6088F7161
           ldr64 MD5: ADEB890D564913F11301C648A5FB6220
           drv64 MD5: 87A462D034192EDD60ACE835E91B930B
       cmd64.dll MD5: BC3B9FB8EAFD440D43B76DC12FB445C7
           drv32 MD5: 1EF0E0C765DA7F727E1EB8FF38D02FF1

C:\>_
```

TdlFsReader architecture



TdlFsReader architecture



Conclusion

- Win64/Olmarik (TDL4) is the first widely spread rootkit targeting Win x64
- Return to old-school techniques of infecting MBR
- The only possible way of debugging bootkit is to use emulators (Bochs, QEMU)
- TDL4 is highly resistant to forensic analysis
- TdlFsReader will be shared amongst malware researchers

References

- ✓“The Evolution of TDL: Conquering x64”

http://www.eset.com/us/resources/white-papers/The_Evolution_of_TDL.pdf

- ✓“Rooting about in TDSS”

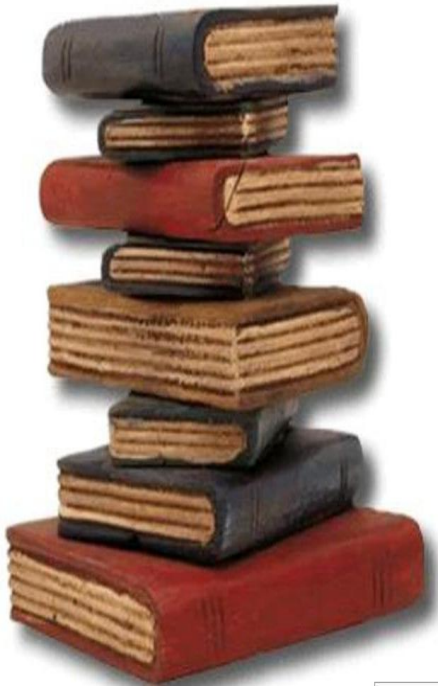
http://www.eset.com/us/resources/white-papers/Rooting-about-in-TDSS.pdf

- ✓“TDL3: The Rootkit of All Evil?”

http://www.eset.com/us/resources/white-papers/TDL3-Analysis.pdf

- ✓Follow ESET Threat Blog

http://blog.eset.com



Questions

Thank you for your attention ;)

Aleksandr Matrosov
matrosov@eset.sk
@matrosov

Eugene Rodionov
rodionov@eset.sk
@vxradius