DEVCORE

Proxying to Kernel : Streaming vulnerabilities from the Windows Kernel

Angelboy

angelboy@devco.re

HEXACON2024 | 2024.10.05

Who am I

DEVCORE

- Angelboy (@scwuaptx)
- Senior Security of DEVCORE
- MSRC 2024 MVR Top 100
- Speaker at
  - CODE BLUE, HITCON, HITB GSEC
- Master of Pwn of Pwn2Own Toronto 2022

Looking at historical vulnerabilities is indispensable

Pwn2Own Vancouver 2024

Target	Prize	Master of Pwn Points
Ubuntu Desktop	$20,000	2
Microsoft Windows 11	$30,000	3
Apple macOS	$40,000	4

In-the-wild

- Win32k
  - GDI (Graphics Device Interface) and UI functions
    - Windows drawing, font management ...
  - Complexity of Code
  - It has been a popular target for attackers over the past decade.

DEVCORE

In-the-wild

- CLFS
  - Common Log File System
  - Handles log-based transaction processing
    - Complexity of Code
  - It has been a popular target for attackers over the past six years.

DEVCORE

In-the-wild

- MSKSSRV
  - Microsoft Kernel Streaming Service
  - Handles synchronization of multimedia streams
  - Very small

DEVCORE

In-the-wild

- MSKSSRV
  - Microsoft Kernel Streaming Service
  - Handles synchronization of multimedia streams
  - Very small
  - Last year it became a very popular target, with 2 ITW exploits in just a few month.

DEVCORE

In-the-wild

- ~~Win32k~~
- ~~CLFS~~
- MSKSSRV
- …

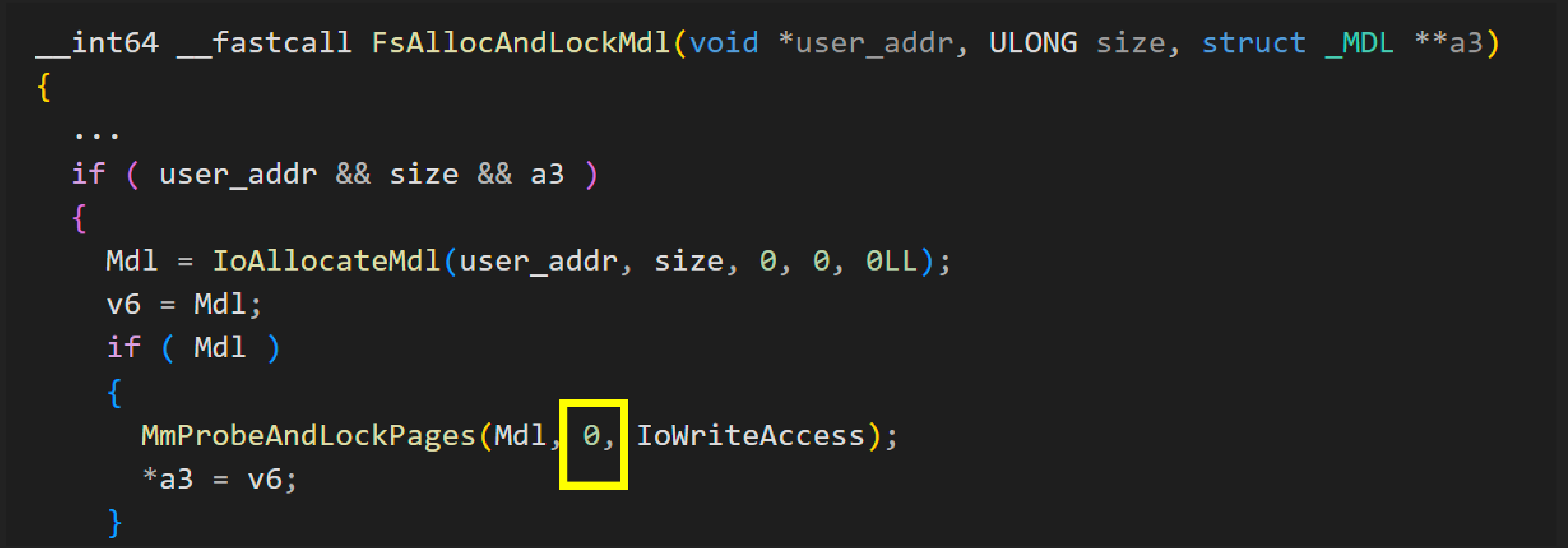
DEVCORE

Let's take a look at MSKSSRV

DEVCORE

MSKSSRV

- CVE-2023-29360 – logical bug (found by @masthoon)
  - MmProbeAndLockPages invalid AccessMode
    - No check if access mode is KernelMode (0)



```
__int64 __fastcall FsAllocAndLockMdl(void *user_addr, ULONG size, struct _MDL **a3)
{
  ...
  if ( user_addr && size && a3 )
  {
    Mdl = IoAllocateMdl(user_addr, size, 0, 0, 0LL);
    v6 = Mdl;
    if ( Mdl )
    {
      MmProbeAndLockPages(Mdl, 0, IoWriteAccess);
      *a3 = v6;
    }
```

DEVCORE

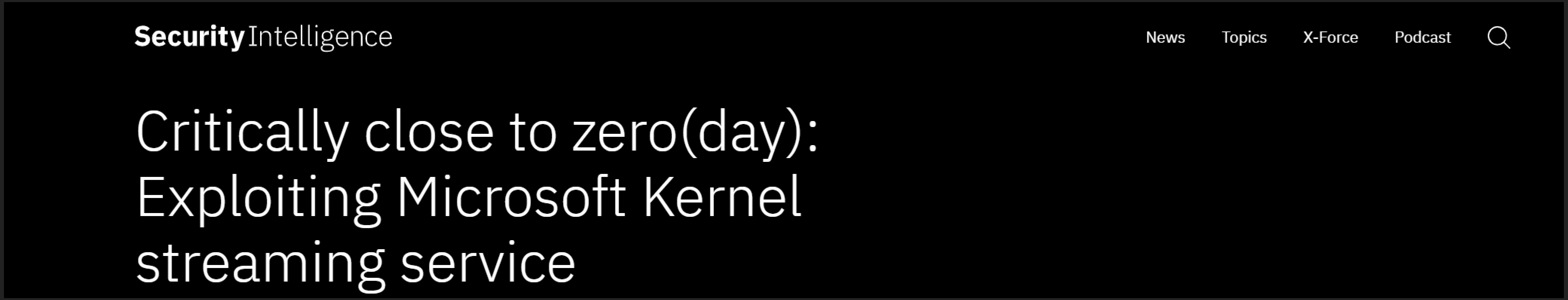
MSKSSRV

- CVE-2023-29360 – logical bug (found by @masthoon)
  - MmProbeAndLockPages invalid AccessMode
    - No check if access mode is KernelMode (0)
      - Mapping arbitrary kernel memory to user space
        - Arbitrary memory writing

DEVCORE

MSKSSRV

- CVE-2023-36802 – Type Confusion
  - No any check for FileObject->FsContext2
    - Context Object & Stream Object type confusion

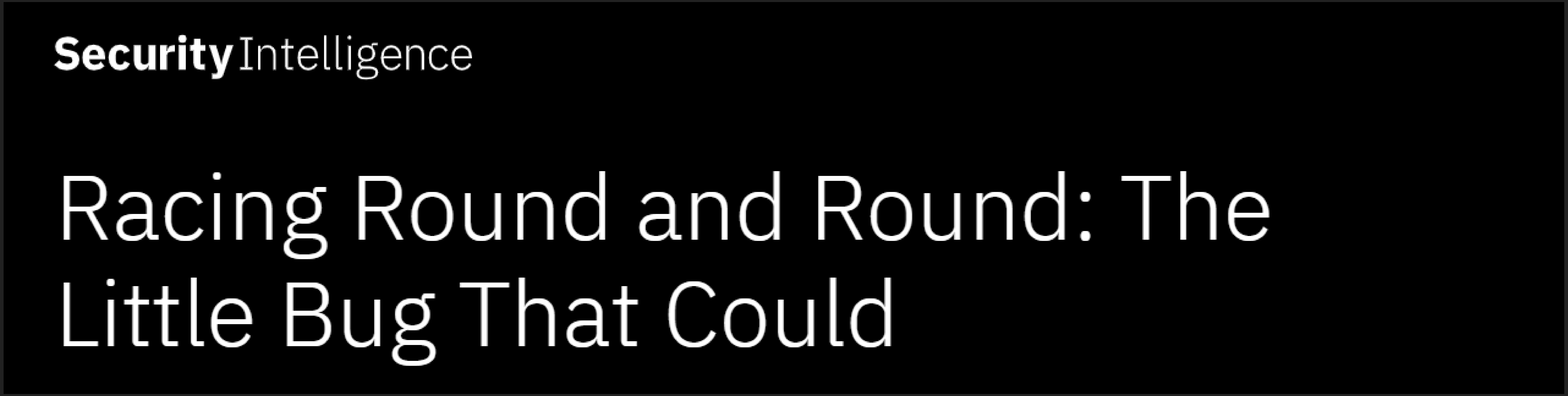


Critically close to zero(day): Exploiting Microsoft Kernel streaming service

DEVCORE

MSKSSRV

- CVE-2024-30089 (found by chompie)



Security Intelligence

Racing Round and Round: The Little Bug That Could

DEVCORE

But is that the end of it ?

Actually …

DEVCORE

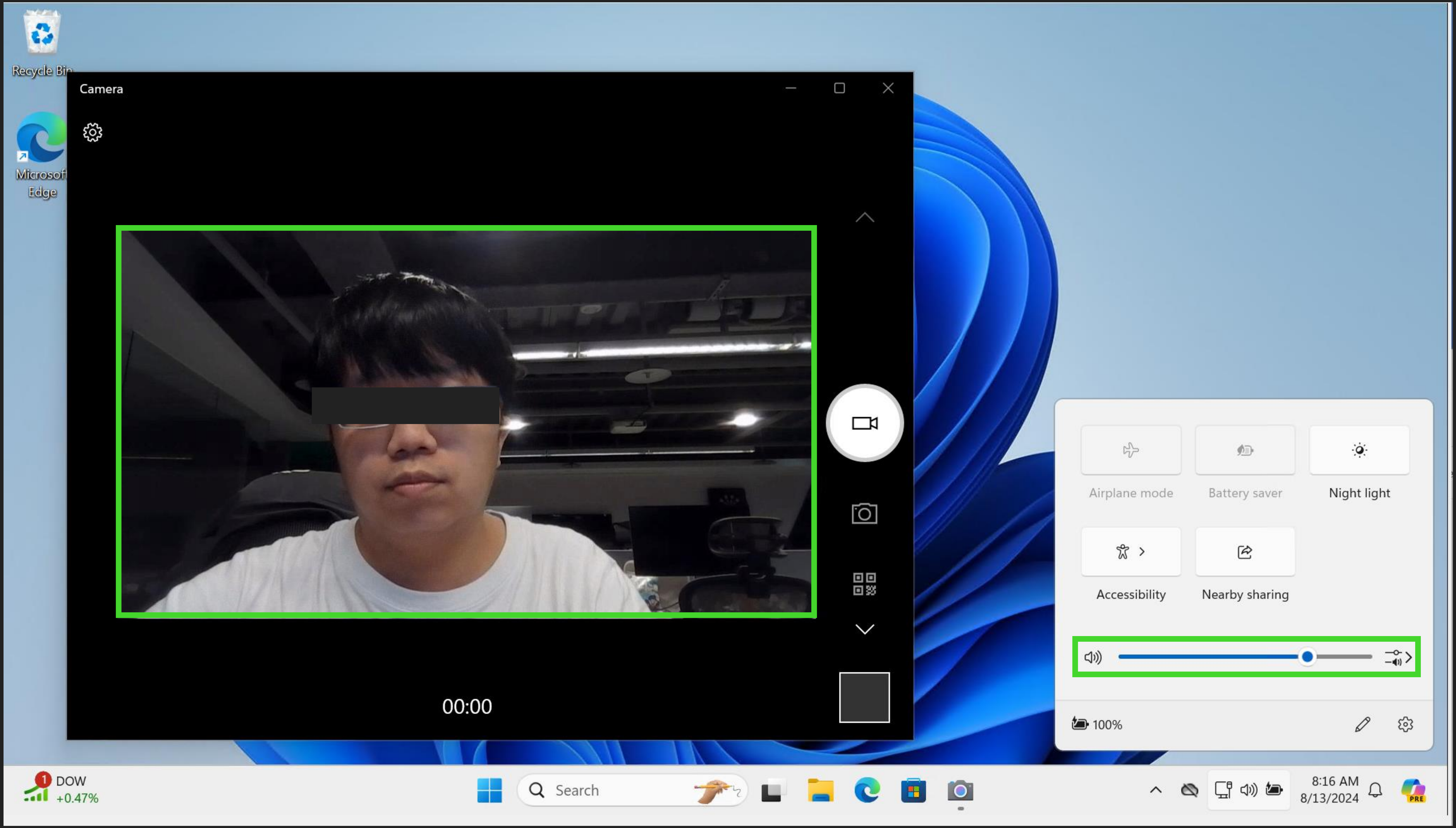
MSKSSRV
ksthunk.sys
ks.sys
portcls.sys
mspclock.sys
HdAudio.sys

DEVCORE

CVE-2024-38054
CVE-2024-30084
CVE-2024-35250
CVE-2024-30090
CVE-2024-38057

Brief overview of Kernel Streaming

DEVCORE



[redacted]

DEVCORE

What is Kernel Streaming ?

- Microsoft-provided services that support kernel-mode processing of streamed data
  - Low Latency
  - Efficient Data Processing
  - Unified Interface
  - High Extensibility

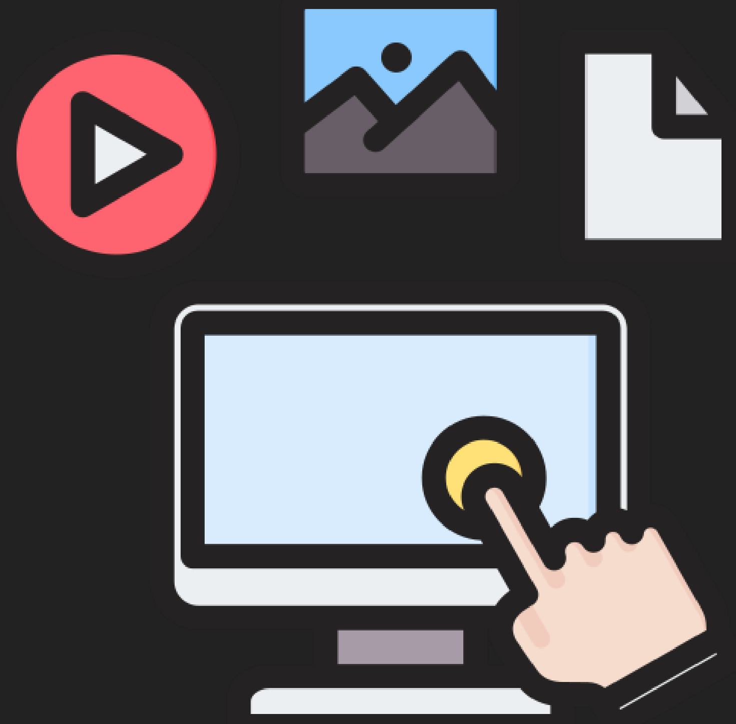
DEVCORE

What is kernel streaming ?

- Microsoft provides 3 multimedia class driver models
  - Port class
    - Audio device
  - AVStream
    - integrated audio/video streaming
  - Stream class

DEVCORE

How to interact with Device?



DEVCORE

Enumerate Device

DEVCORE

Enumerate KS Device

- You can use SetupDiGetClassDevs with class GUID to emulate device

```
\\?\hdaudio#subfunc_01&ven_8086&dev_2812&nid_0001&subsys_00000000&rev_1000#6&2f1f346a&0&0002&0000001d#{6994ad04-93ef-11d0-a3cc-00a0c9223196}\ehdmiouttopo
```

DEVCORE

Enumerate KS Device

- KsOpenDefaultDevice
  - Opens a handle to the first device that is listed in the specified Plug and Play (PnP) category

```
hr = KsOpenDefaultDevice(KSCATEGORY_VIDEO_CAMERA,
    GENERIC_READ | GENERIC_WRITE, &g_hDevice);
```

DEVCORE

KS Object

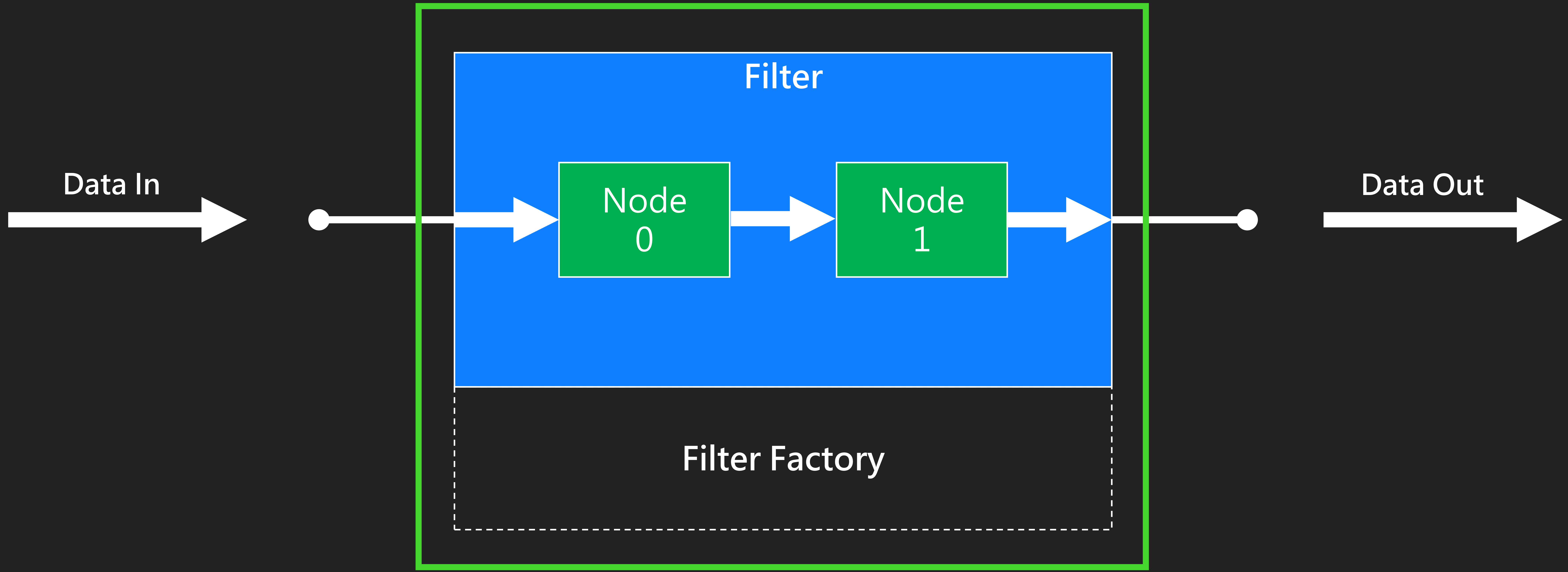
DEVCORE

KS Object

- After we open these Devices, Kernel Streaming will establish some Kernel Streaming related instance
  - KS Filter
  - KS Pin
  - …
- Encapsulate hardware function

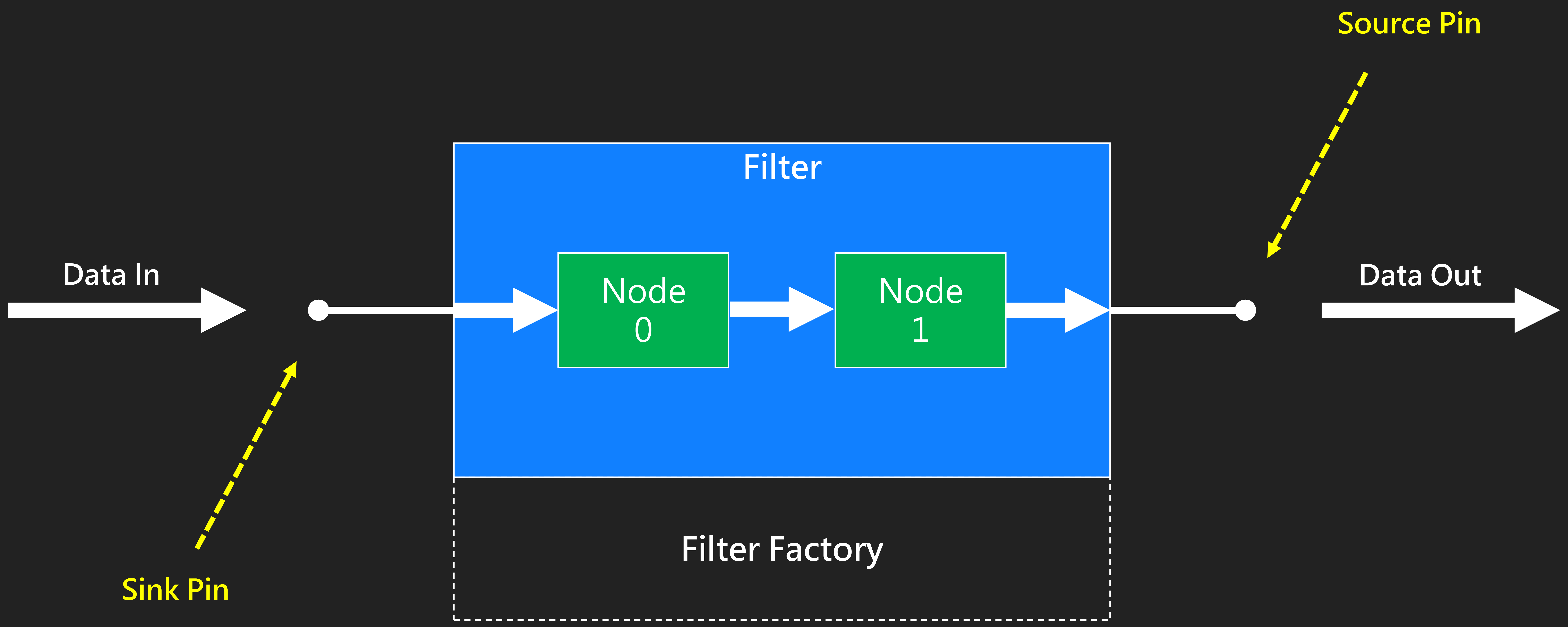
DEVCORE

KS Filter



https://learn.microsoft.com/en-us/windows-hardware/drivers/audio/audio-filters

KS Pin



DEVCORE

https://learn.microsoft.com/en-us/windows-hardware/drivers/audio/audio-filters

KS Property

- A Property represents a capability or control-state setting that belongs to a kernel streaming object
- Client can set or get property to KS Object with GUID
    - Device State
    - Data format
    - Volume Level

DEVCORE

KS Property

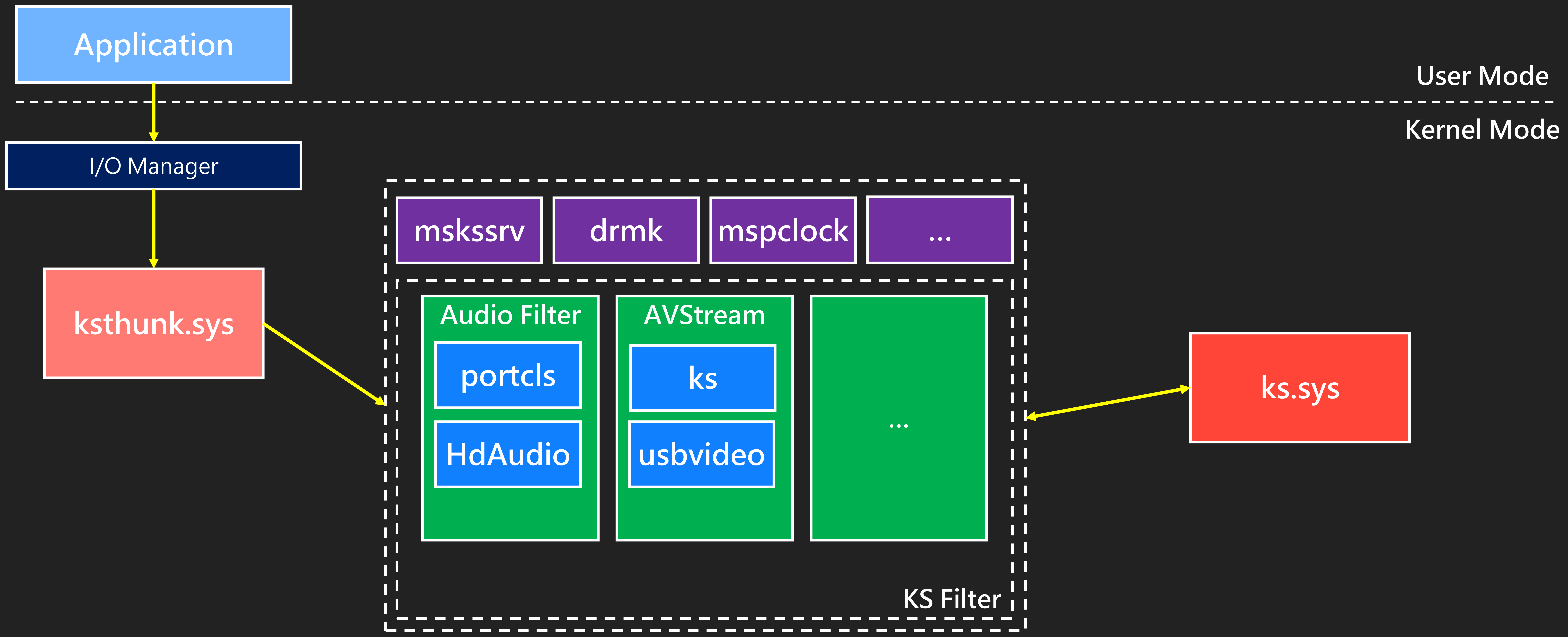
- Device State is a KS property
- Through IOCTL_KS_PROPERTY to get or set it

```
    DeviceIoControl(hPin, IOCTL_KS_PROPERTY, &pinProp, sizeof(pinProp),
        &state, sizeof(state), &cbReturned, NULL);
```

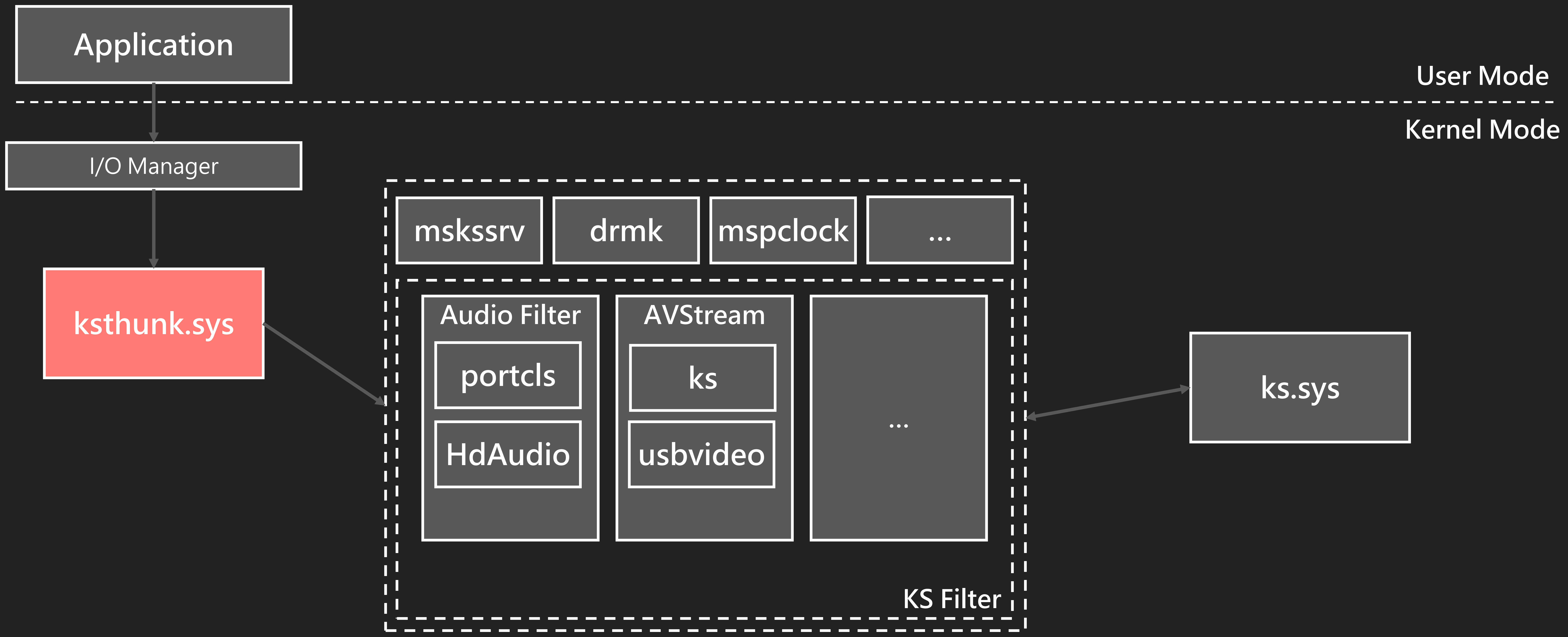
DEVCORE

Kernel Streaming Architecture

Kernel Streaming Architecture



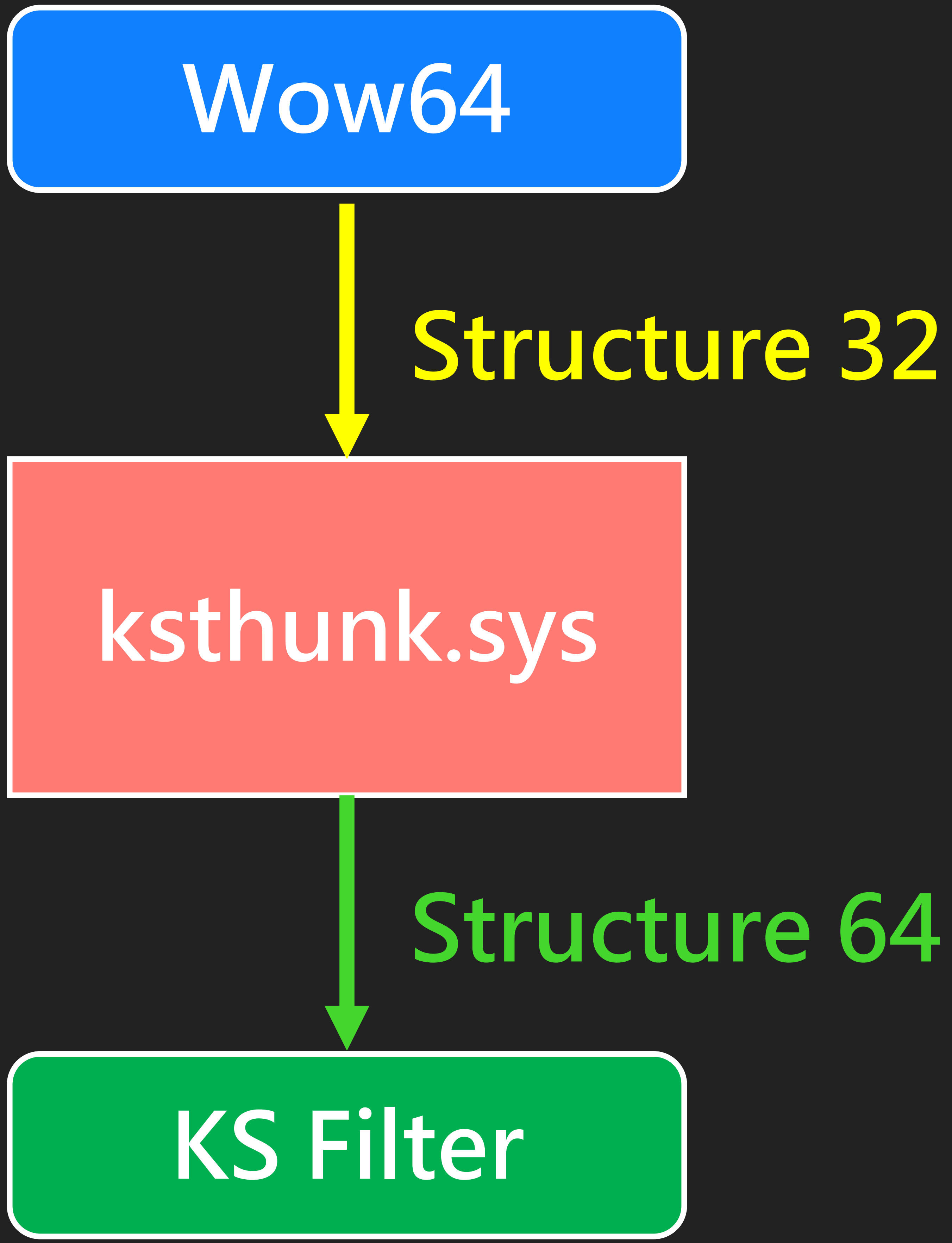
Kernel Streaming Architecture



DEVCORE

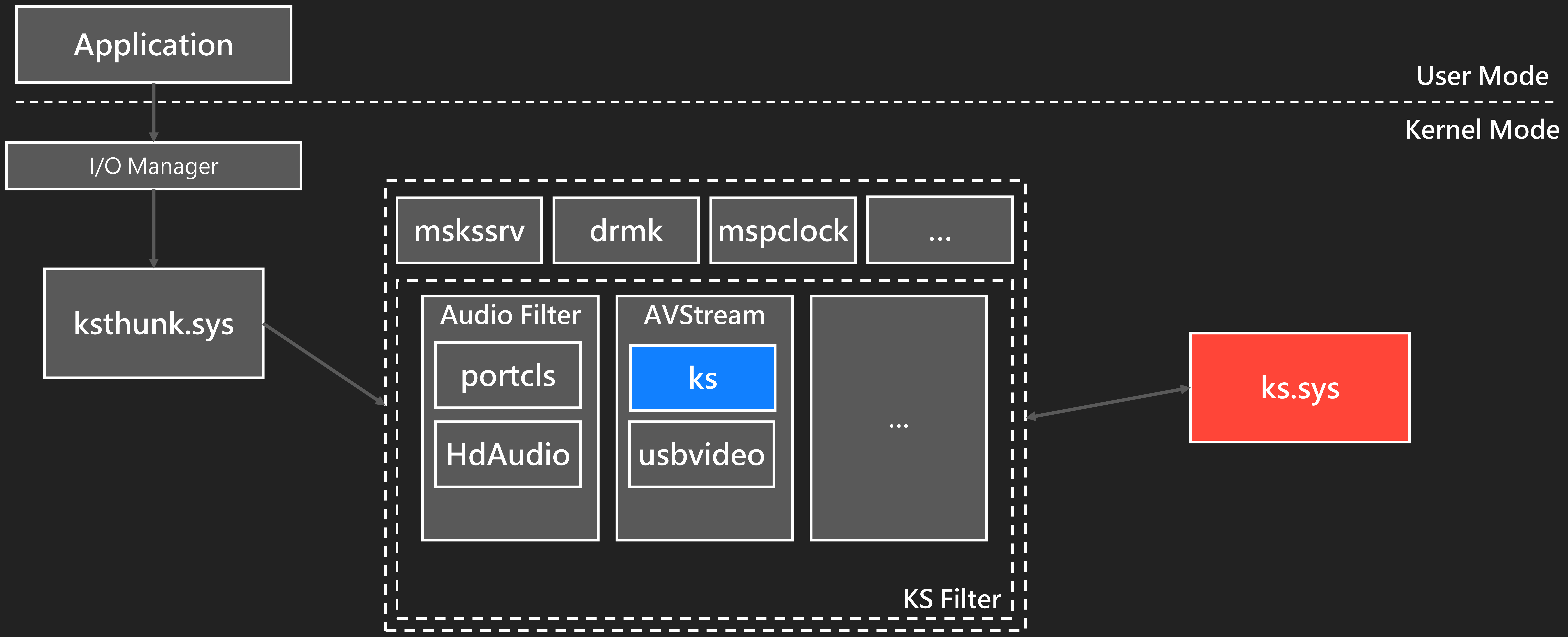
ksthunk

- Kernel Streaming WOW Thunk Service Driver
- Entry point of Kernel Streaming
- For backward compatibility
  - If the request process is WoW64
    - Transfer 32-bits to 64-bit request



DEVCORE

Kernel Streaming Architecture



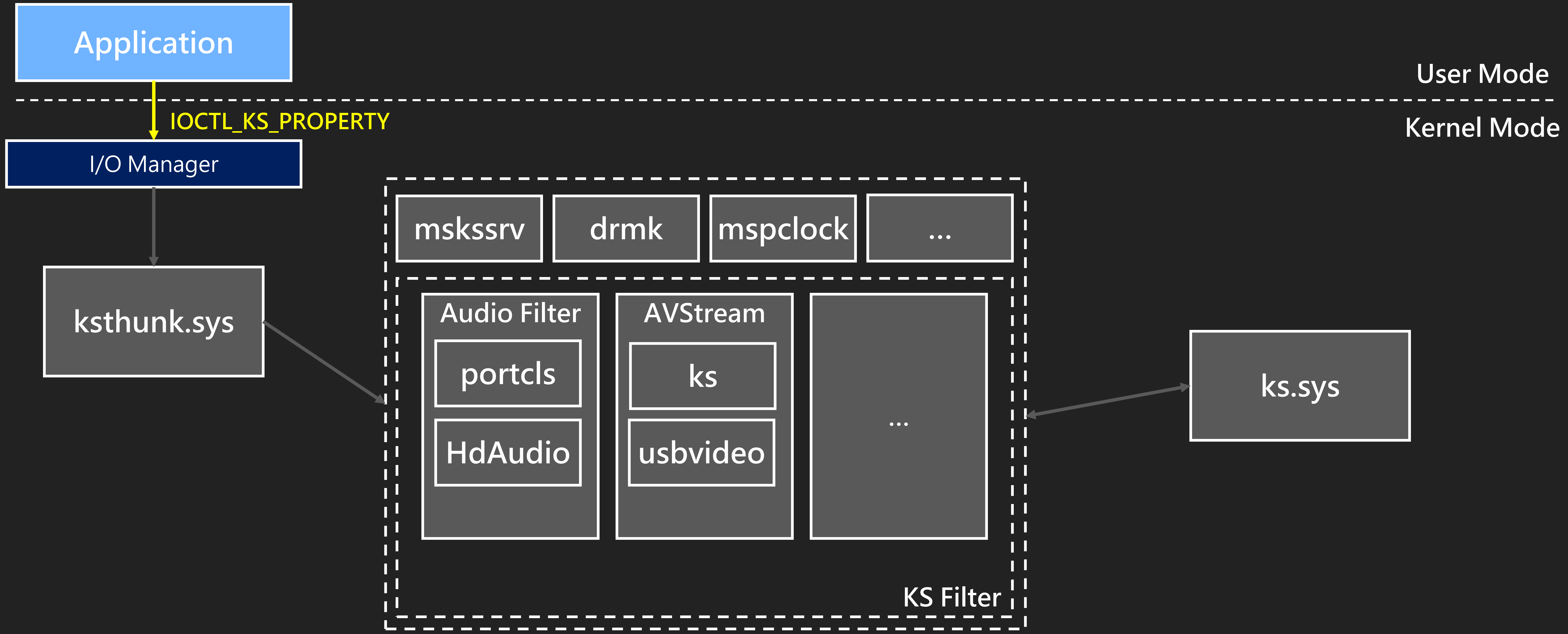
DEVCORE

ks.sys

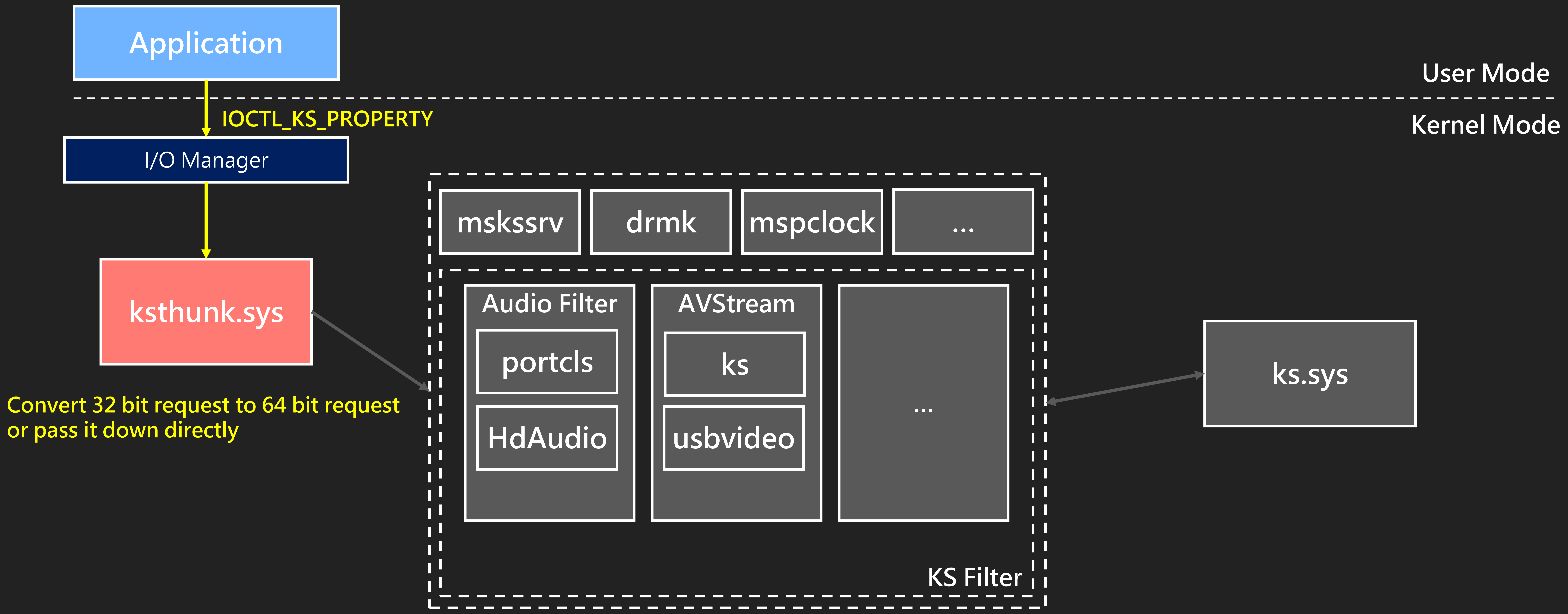
- Kernel CSA Library
- One of the main components of Kernel Streaming
- Provide interface for Kernel Stream
  - Property
  - Event
  - …

DEVCORE

The work flow of set pin state

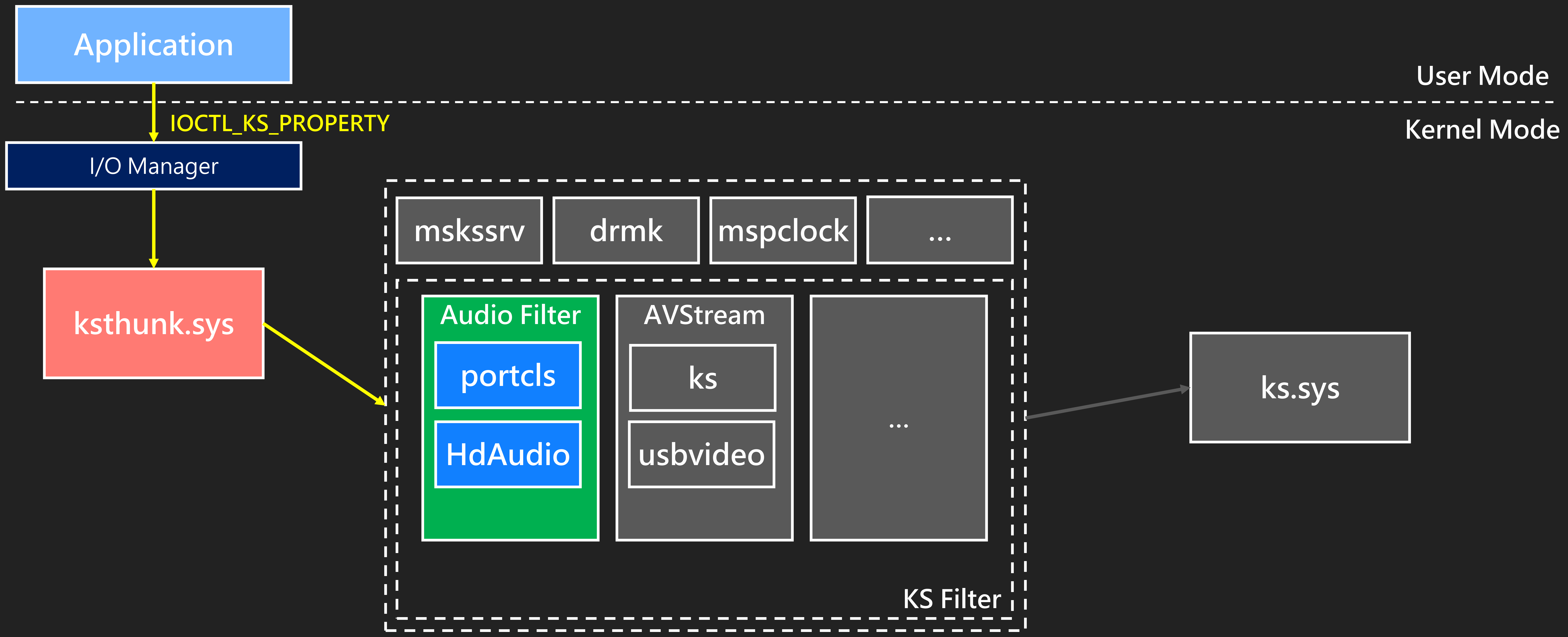


The work flow of set pin state



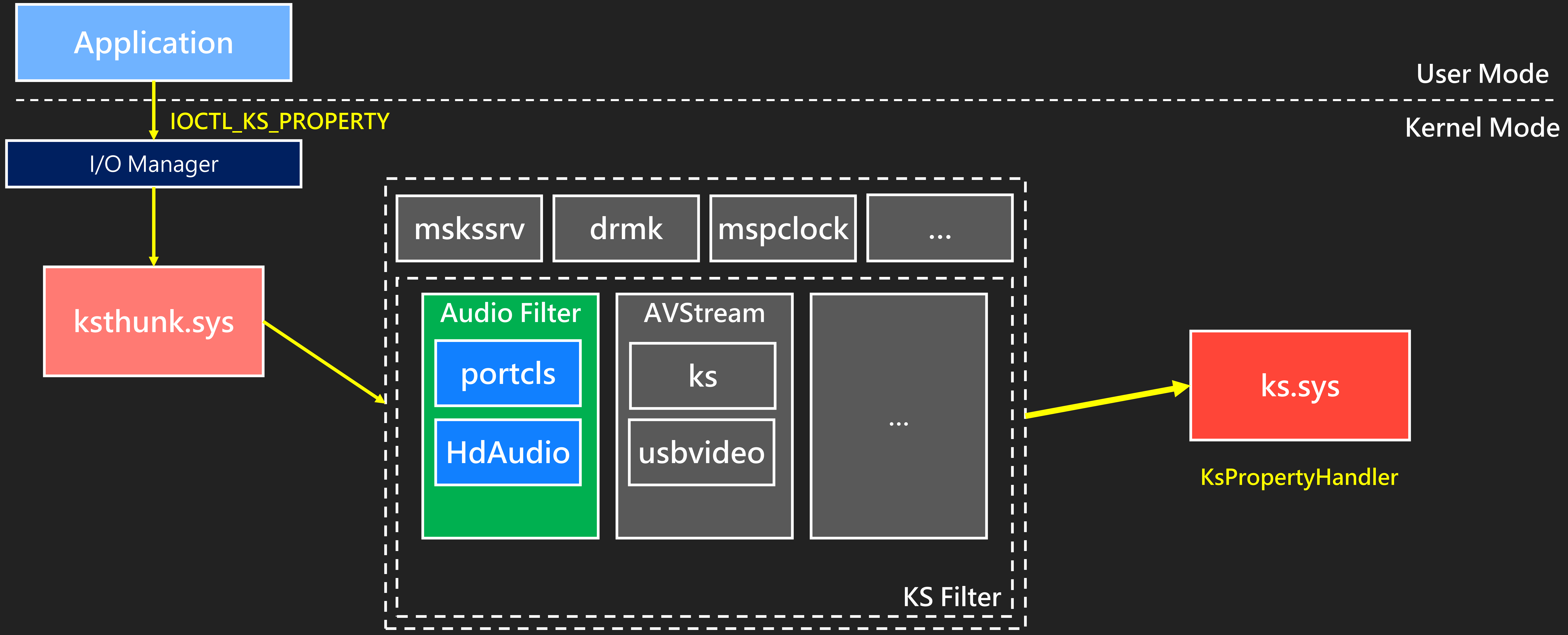
DEVCORE

The work flow of set pin state



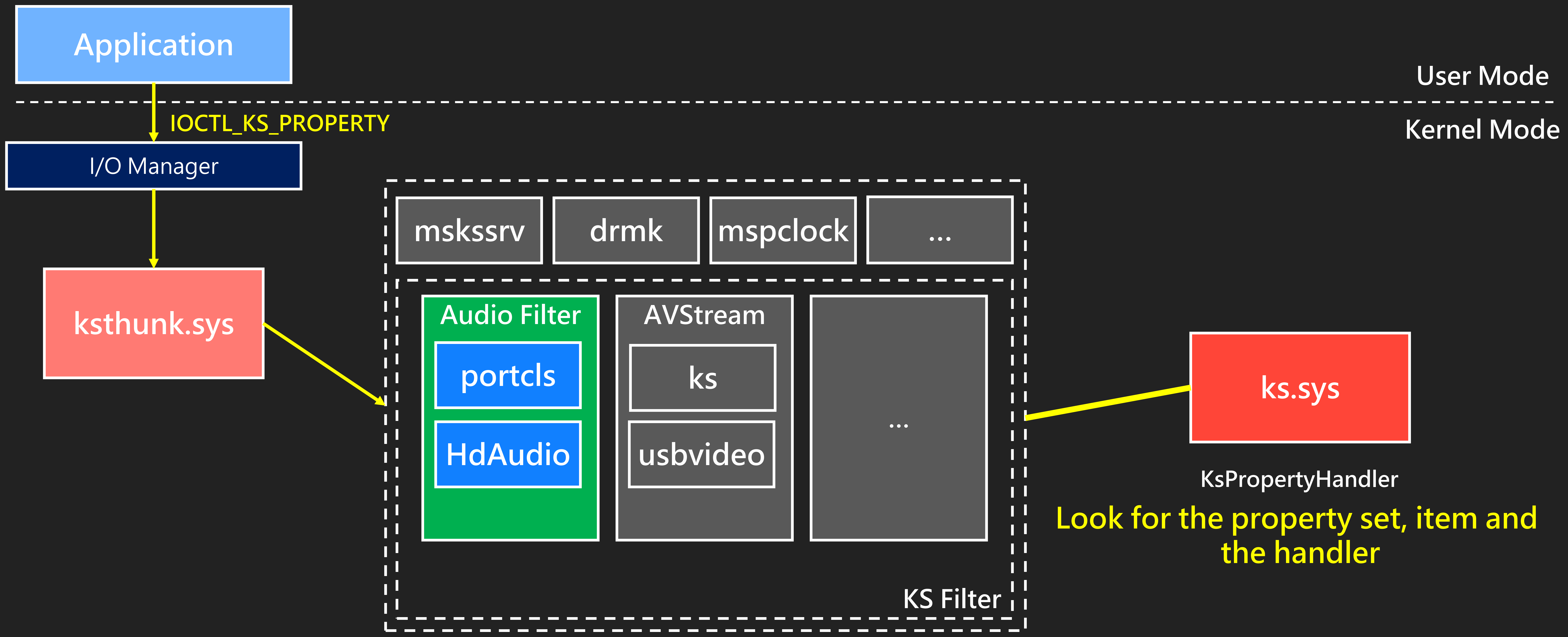
DEVCORE

The work flow of set pin state



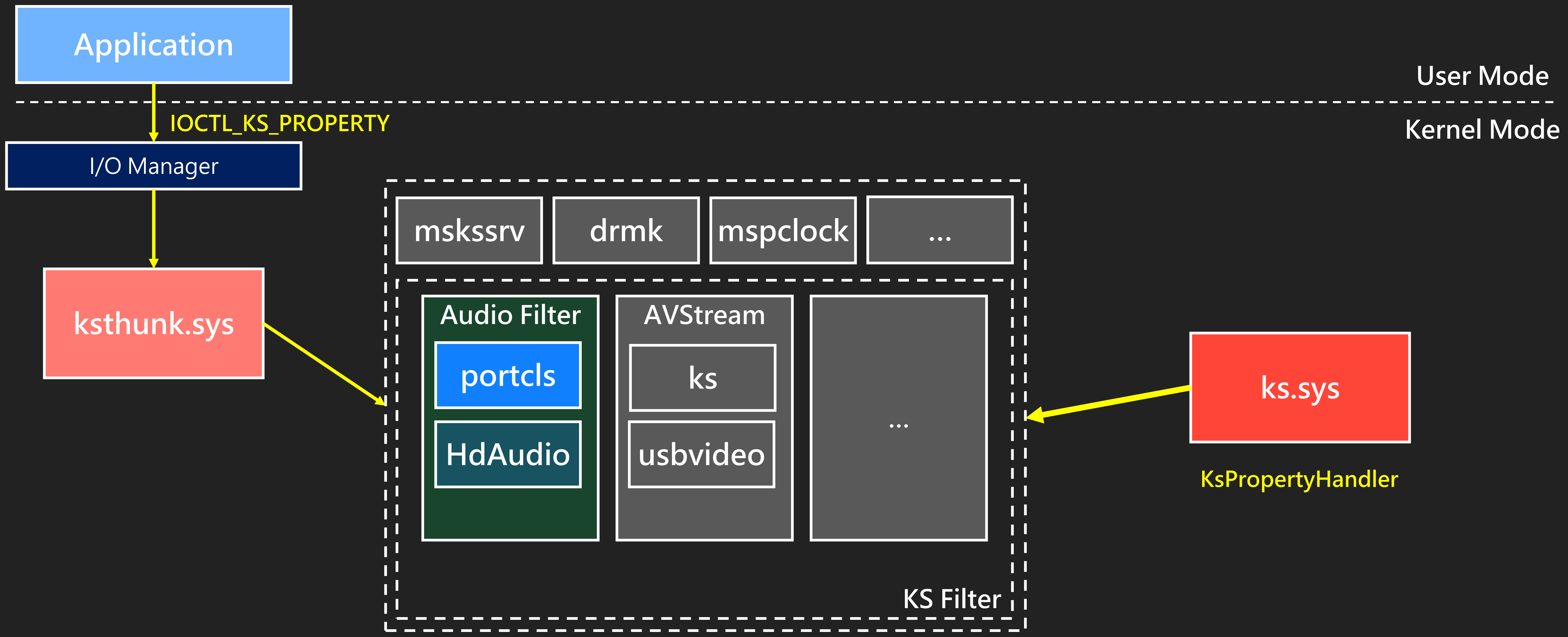
DEVCORE

The work flow of set pin state



DEVCORE

The work flow of set pin state



DEVCORE

portcls! PinPropertyDeviceState

From attacker's view

DEVCORE

From attacker's view

- There are many properties for each device
    - individual implementation

DEVCORE

From attacker's view

- There are many properties for each device
  - individual implementation
- No vulnerabilities in ks and ksthunk for a long time
  - **CVE-2020-16889 (found by @nghiadt1098)**
  - **CVE-2020-17045 (found by @nghiadt1098)**

DEVCORE

From attacker's view

- There are many properties for each device
  - individual implementation
- No vulnerabilities in ks and ksthunk for a long time
  - CVE-2020-16889 (found by @nghiadt1098)
  - CVE-2020-17045 (found by @nghiadt1098)
- **Each driver handles part of the content individually, which may lead to inconsistencies.**

DEVCORE

We found some trivial vulnerabilities in few days ...

DEVCORE

Vulnerabilities

- Portcls.sys
  - CVE-2024-38055 (OOB)
  - CVE-2024-38056
- Ksthunk
  - CVE-2024-38054 (OOB)
  - CVE-2024-38057

We found some interesting things

DEVCORE

Is really safe ?

```
if ( irp->RequestorMode )
{
  v14 = 0xC0000010;
}
else
{
  UserBuffer = (unsigned int *)irp->UserBuffer;
  v19[0] = 0LL;
  v19[1] = v9;
  FileObject = CurrentStackLocation->FileObject;
  v21 = FileObject;
  v14 = (*(__int64 (__fastcall **)(_QWORD, _QWORD, __int64 *))(Type3InputBuffer + 0x38))(
          *UserBuffer,
          0LL,
          v19);
}
```

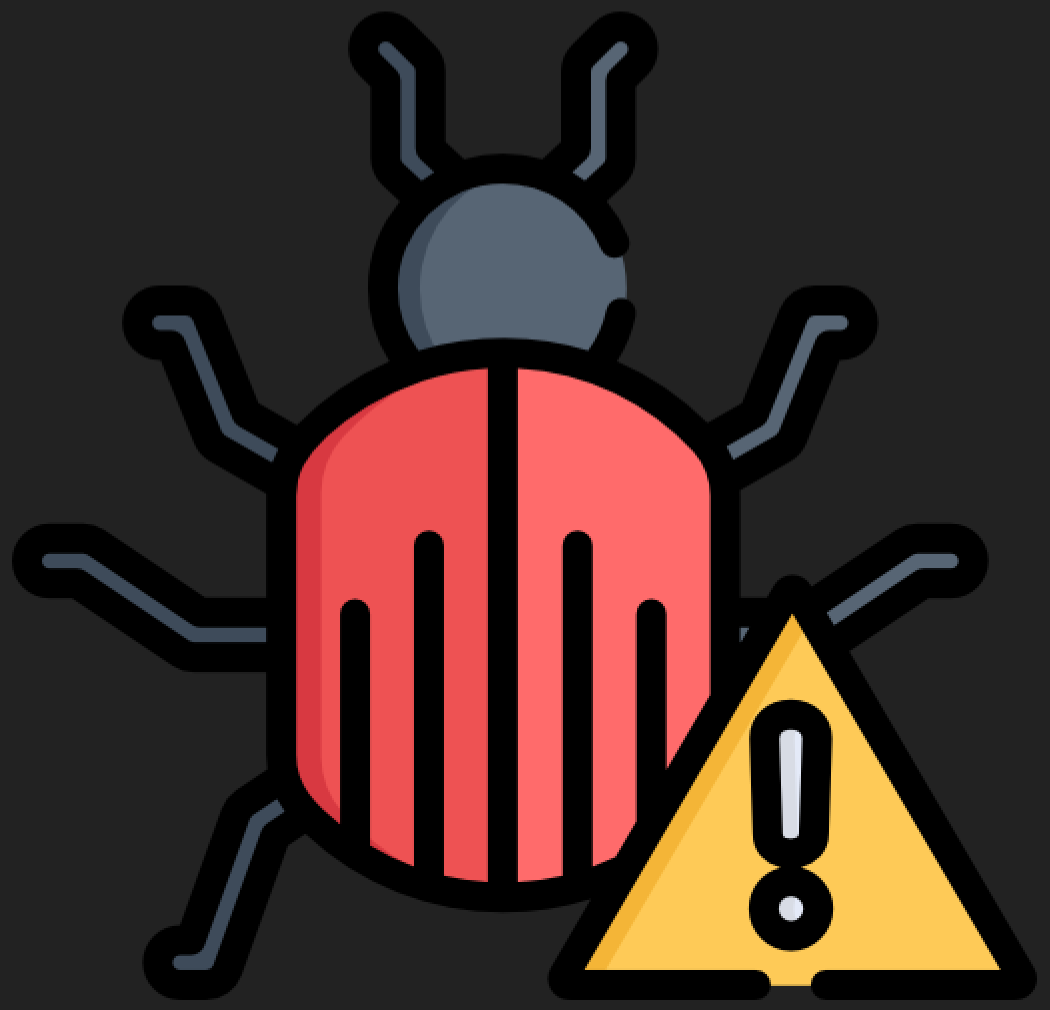
DEVCORE

Is really safe ?

UserMode(1)

```
if ( irp->RequestorMode )
{
  v14 = 0xC0000010;
}
else
{
  UserBuffer = (unsigned int *)irp->UserBuffer;
  v19[0] = 0LL;
  v19[1] = v9;
  FileObject = CurrentStackLocation->FileObject;
  v21 = FileObject;
  v14 = (*(__int64 (__fastcall **)(_QWORD, _QWORD, __int64 *))(Type3InputBuffer + 0x38))(
          *UserBuffer,
          0LL,
          v19);
}
```

DEVCORE

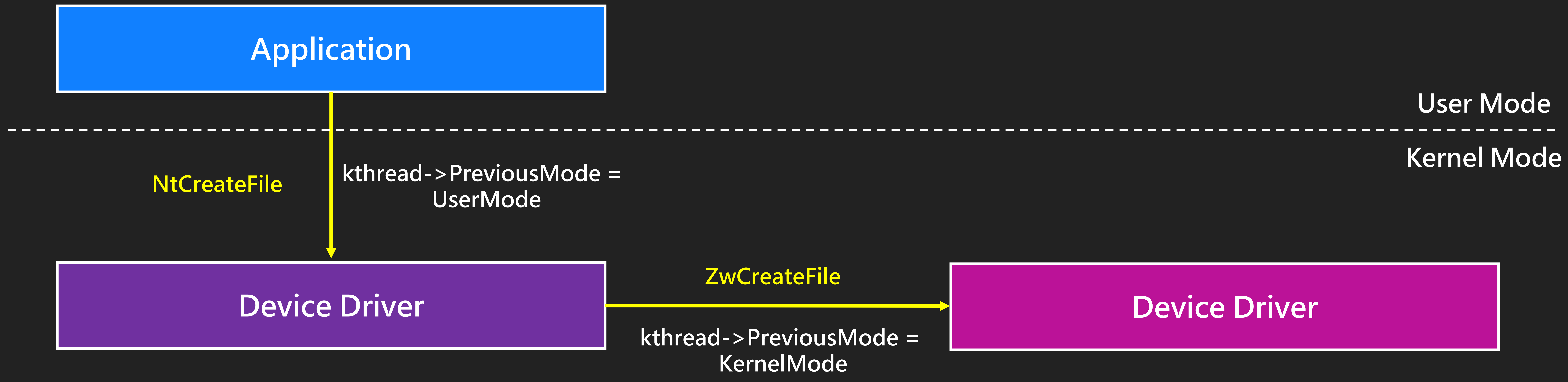


The Overlooked Bug Class

DEVCORE

PreviousMode

- A field in the thread object that indicates whether the parameters for a System Service Call originated in user mode or kernel mode.



DEVCORE

IRP RequestorMode

- IRP->RequestorMode
  - the execution mode of the original requester of the operation
  - A copy of the PreviousMode value from the thread object

DEVCORE

IRP RequestorMode

```
if ( Irp->RequestorMode )
{
  ProbeForRead(CurrentStackLocation->Parameters.DeviceIoControl.Type3InputBuffer, InputBufferLength, 1u)
  a4 = callback;
  outputLength = outlen;
}
```

```
MmProbeAndLockPages(Irp->MdlAddress, Irp->RequestorMode, IoWriteAccess);
RequestorMode = Irp->RequestorMode;
v16 = (unsigned __int8)HIBYTE(*(_WORD *)(a2 + 24)) >> 6;
Object = 0LL;
v14 = ObReferenceObjectByHandle(v8, v16, (POBJECT_TYPE)IoFileObjectType, RequestorMode, &Object, 0LL);
```

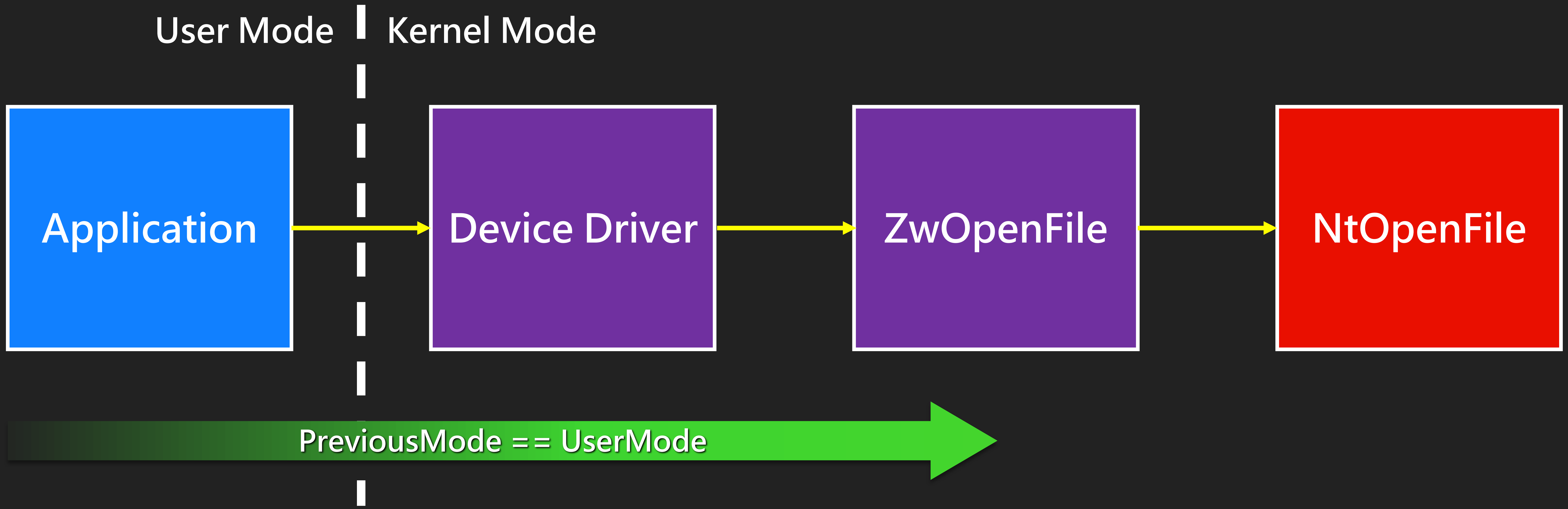
DEVCORE

But there are some issues in some cases ...

DEVCORE

A logical bug class

- Windows Kernel Logic Bug Class: Access Mode Mismatch in IO Manager by James Forshaw

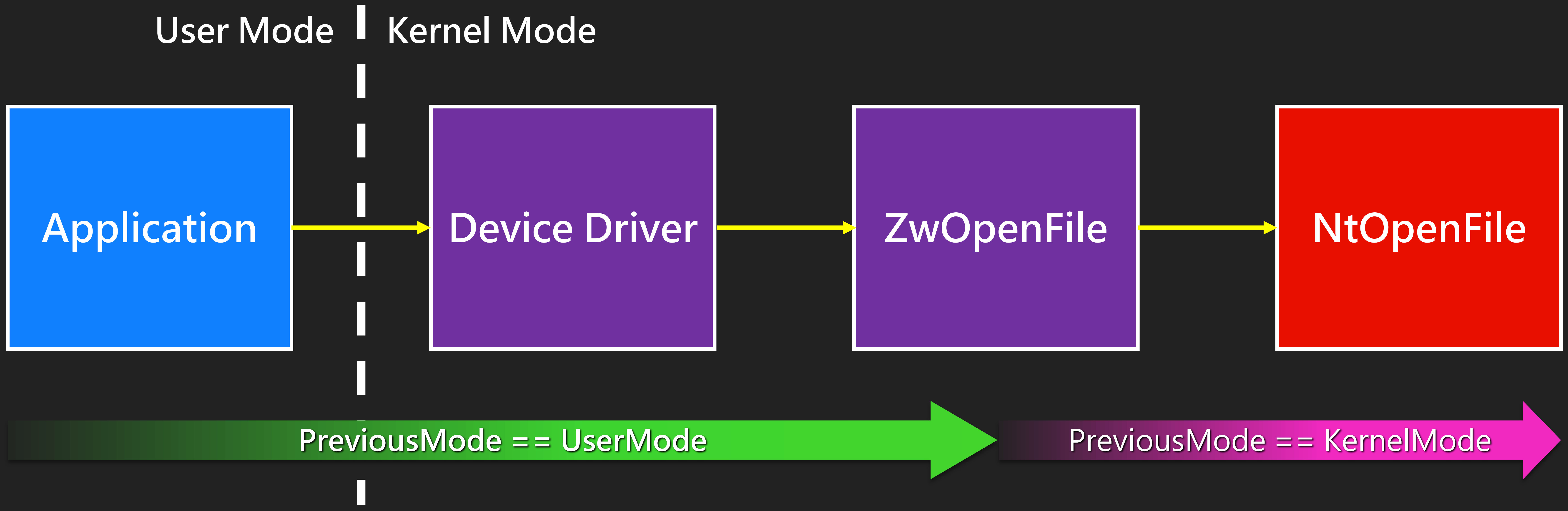


DEVCORE

https://googleprojectzero.blogspot.com/2019/03/windows-kernel-logic-bug-class-access.html

A logical bug class

- Windows Kernel Logic Bug Class: Access Mode Mismatch in IO Manager by James Forshaw

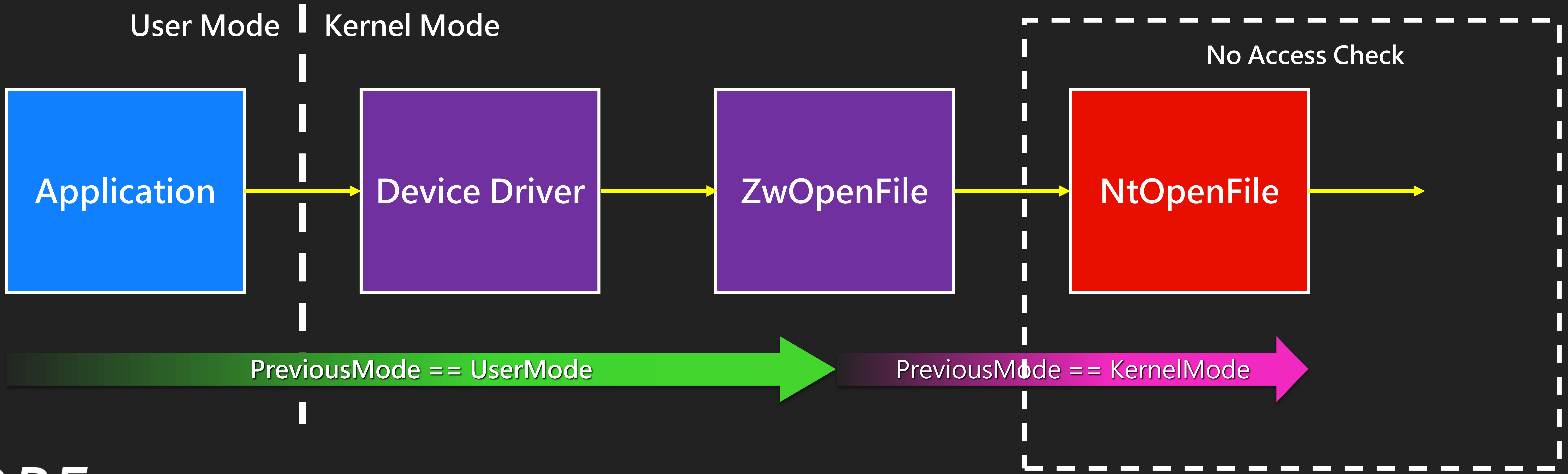


DEVCORE

https://googleprojectzero.blogspot.com/2019/03/windows-kernel-logic-bug-class-access.html

A logical bug class

- What happens if kernel call OpenFile and solely relies on RequestorMode for validation ?



DEVCORE

https://googleprojectzero.blogspot.com/2019/03/windows-kernel-logic-bug-class-access.html

A logical bug class

- What happens if kernel call OpenFile and solely relies on RequestorMode for validation ?
  - Bypass
    - Security Access Check
    - Memory Access Check

DEVCORE

https://googleprojectzero.blogspot.com/2019/03/windows-kernel-logic-bug-class-access.html

It focuses on Zw* system service call

DEVCORE

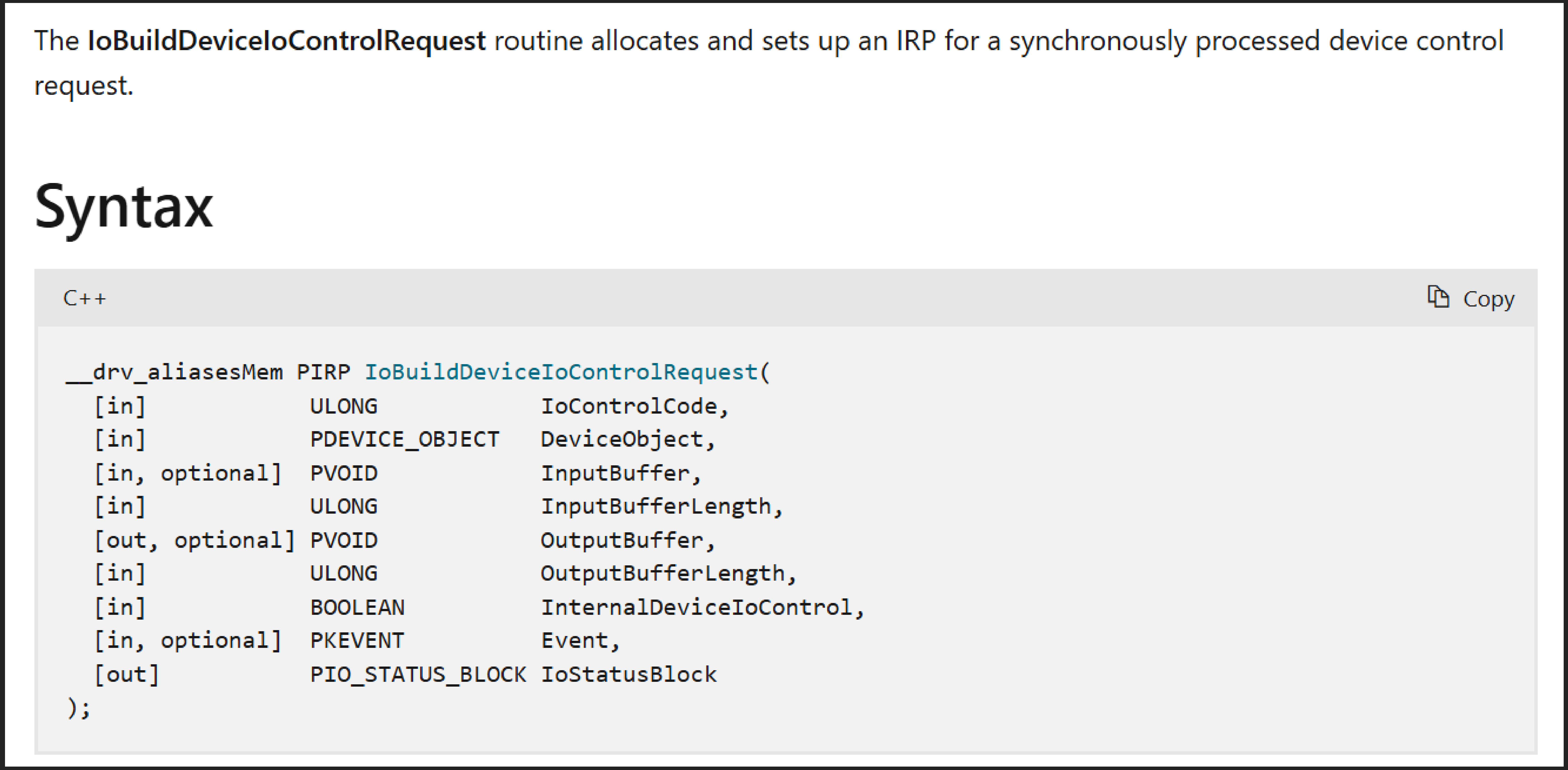
Are there other potential causes for this bug class?

DEVCORE

Are there other potential causes for this bug class?

The Bug Pattern

- IoBuildDeviceIoControlRequest



The **IoBuildDeviceIoControlRequest** routine allocates and sets up an IRP for a synchronously processed device control request.

Syntax

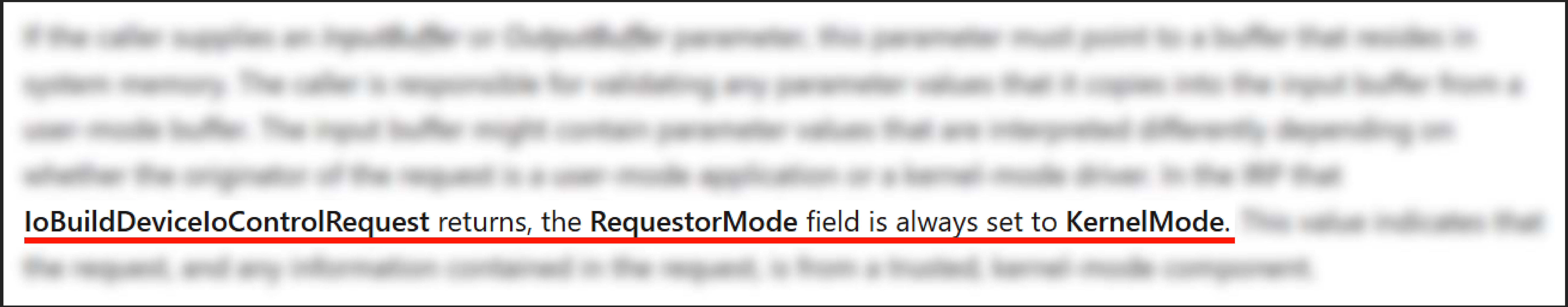
C++ Copy

```
__drv_aliasesMem PIRP IoBuildDeviceIoControlRequest(
  [in]            ULONG            IoControlCode,
  [in]            PDEVICE_OBJECT   DeviceObject,
  [in, optional]  PVOID            InputBuffer,
  [in]            ULONG            InputBufferLength,
  [out, optional] PVOID            OutputBuffer,
  [in]            ULONG            OutputBufferLength,
  [in]            BOOLEAN          InternalDeviceIoControl,
  [in, optional]  PKEVENT          Event,
  [out]           PIO_STATUS_BLOCK IoStatusBlock
);
```

DEVCORE

The Bug Pattern

- IoBuildDeviceIoControlRequest

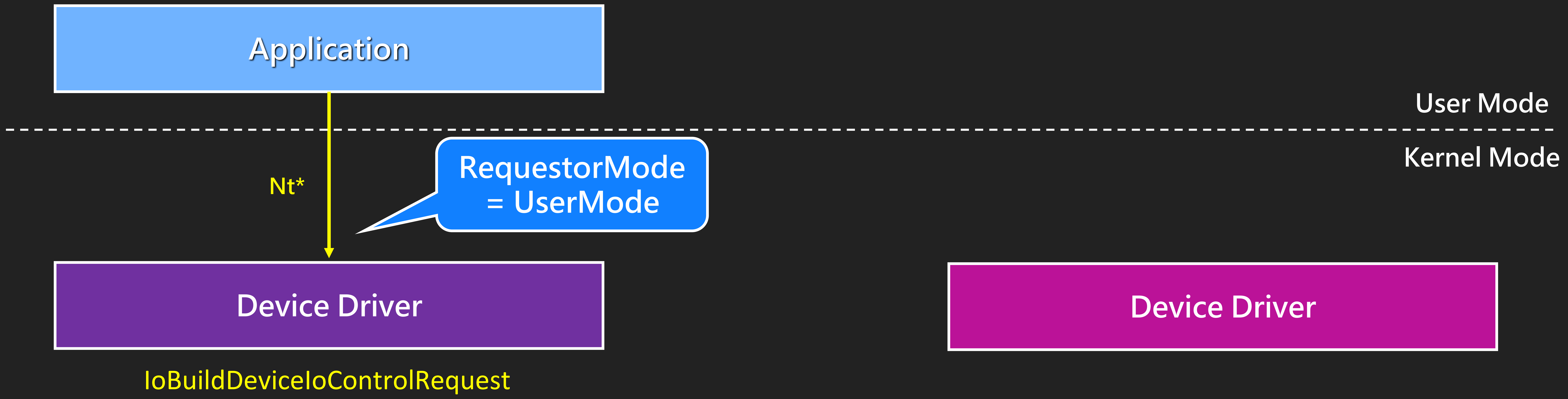


If the caller supplies an InputBuffer or OutputBuffer parameter, this parameter must point to a buffer that resides in system memory. The caller is responsible for validating any parameter values that it copies into the input buffer from a user-mode buffer. The input buffer might contain parameter values that are interpreted differently depending on whether the originator of the request is a user-mode application or a kernel-mode driver. In the IRP that **IoBuildDeviceIoControlRequest** returns, the **RequestorMode** field is always set to **KernelMode**. This value indicates that the request, and any information contained in the request, is from a trusted, kernel-mode component.

DEVCORE

The Bug Pattern

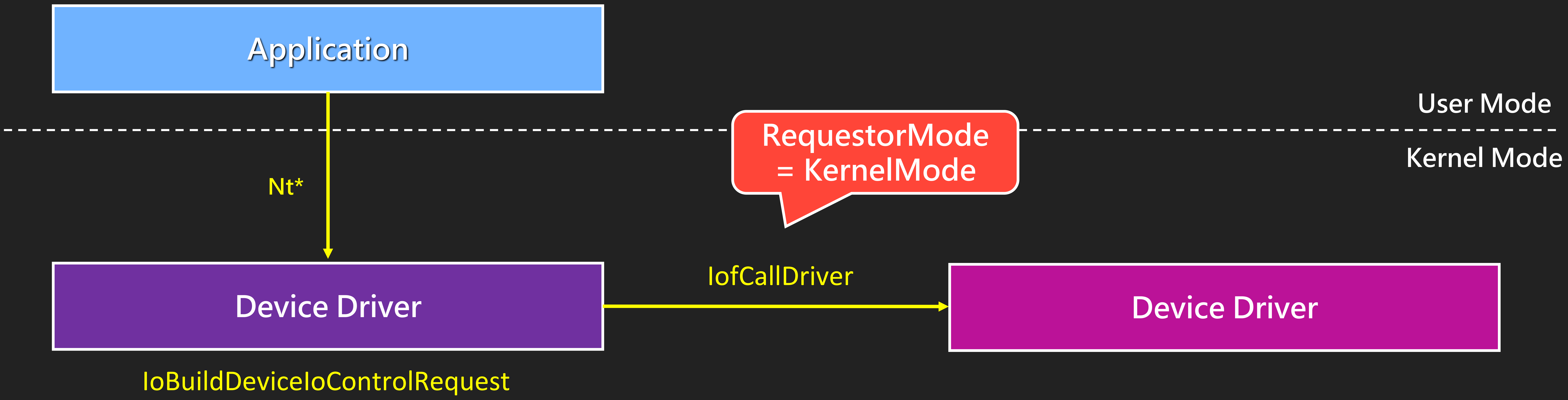
- IoBuildDeviceIoControlRequest



DEVCORE

The Bug Pattern

- IoBuildDeviceIoControlRequest



DEVCORE

After quick review of this bug pattern in KS

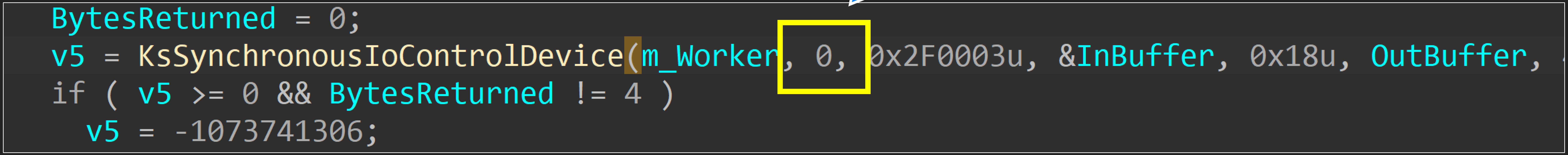
DEVCORE

```c
NTSTATUS __stdcall KsSynchronousIoControlDevice(
    PFILE_OBJECT FileObject,
    KPROCESSOR_MODE RequestorMode,
    ULONG IoControl,
    PVOID InBuffer,
    ULONG InSize,
    PVOID OutBuffer,
    ULONG OutSize,
    PULONG BytesReturned)
{

    KeInitializeEvent(&Event, NotificationEvent, 0);
    NewIrp = IoBuildDeviceIoControlRequest(
        IoControl,
        RelatedDeviceObject,
        InBuffer,
        InSize,
        OutBuffer,
        OutSize,
        0,
        &Event,
        &IoStatusBlock);
    ...
    NewIrp->RequestorMode = RequestorMode;
    ...
    Status = IofCallDriver(RelatedDeviceObject, NewIrp);
}
```

But ...

KernelMode

CKsPin::GetState



```
BytesReturned = 0;
v5 = KsSynchronousIoControlDevice(m_Worker, 0, 0x2F0003u, &InBuffer, 0x18u, OutBuffer,
if ( v5 >= 0 && BytesReturned != 4 )
  v5 = -1073741306;
```

DEVCORE

CKsPin::GetState

```
BytesReturned = 0;
v5 = KsSynchronousIoControlDevice(m_Worker, 0, 0x2F0003u, &InBuffer, 0x18u, OutBuffer,
if ( v5 >= 0 && BytesReturned != 4 )
  v5 = -1073741306;
```

SerializePropertySet

```
if ( SerialSize )
{
  v19 = KsSynchronousIoControlDevice(
          CurrentStackLocation->FileObject,
          0,
          CurrentS          rameters.DeviceIoControl.IoControlCode,
          PoolWithTag,
          InSize,
          (v16 + 0x20),
          SerialSize,
          &BytesReturned);
```

KernelMode

CKsPin::GetState

```
    BytesReturned = 0;
    v5 = ...ntrolDevice(m_Worker, 0, 0x2F0003u, &InBuffer, 0x18u, OutBuffer
    if
                  InSize,
                  (v16 + 0x20),
                  SerialSize,
                  &BytesReturned);
```

UnserializePropertySet

```
      if ( OutSize > v13 )
        goto error2;
                = KsSynchronousIoControlDevice(
                    CurrentStackLocation->FileObject,
                    0,
                    CurrentStackLocation->Parameters.DeviceIoControl.IoControlCode,
                    New_KsProperty_req,
                    InSize,
                    OutBuffer,
                    OutSize,
                    &BytesReturned);
```

KernelMode

DEVCORE

Look for the bug pattern in KS

1. KsSynchronousIoControlDevice
2. Controllable
   - InputBuffer
   - OutputBuffer
3. IOCTL relies on RequestorMode for security checks

Look for the bug pattern in KS

1. KsSynchronousIoControlDevice
2. Controllable
   - InputBuffer
   - OutputBuffer

```
KsSynchronousIoControlDevice(
  CurrentStackLocation->FileObject,
  0,
  CurrentStackLocation->Parameters.DeviceIoControl.IoControlCode,
  New_KsProperty_req,
  InSize,
  OutBuffer,
  OutSize,
  &BytesReturned);
```

DEVCORE

Look for the bug pattern in KS

1. KsSynchronousIoControlDevice
2. Controllable
   - InputBuffer
   - OutputBuffer
3. IOCTL relies on RequestorMode for security checks

```
MmProbeAndLockPages(mdl, irp->RequestorMode, IoWriteAccess);
```

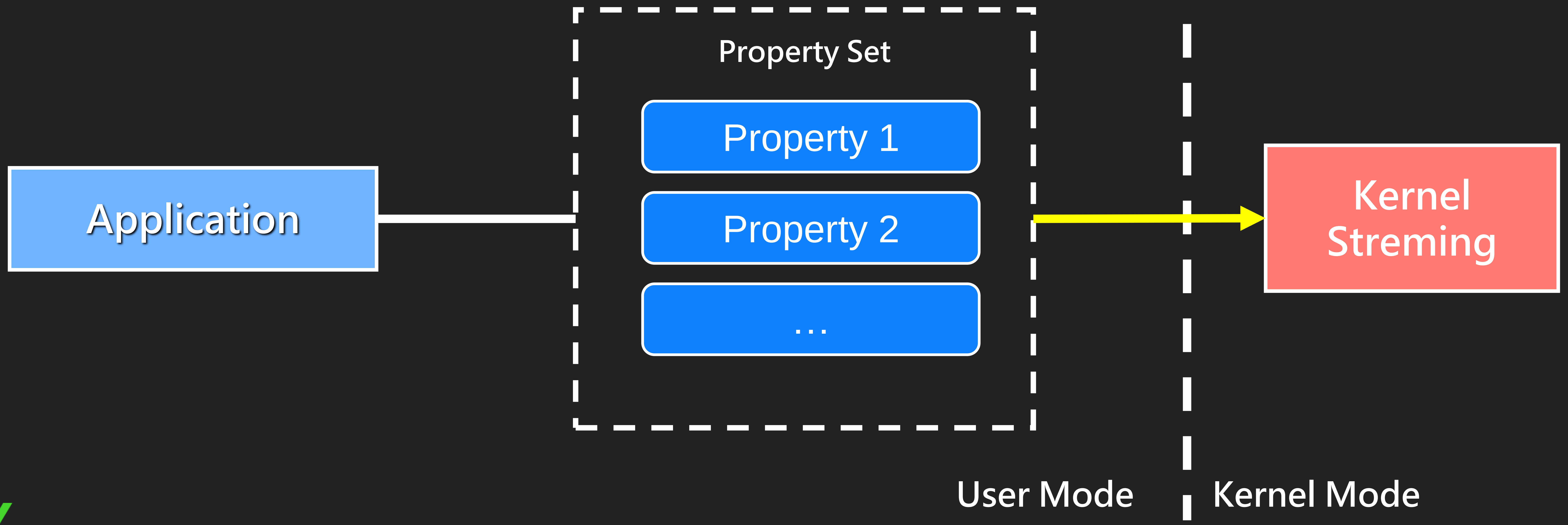
```
if ( irp->RequestorMode )
  ProbeForRead(CurrentStackLocation->Parameters.DeviceIoControl.Type3InputBuffer, inputbuf, 1u);
```

The Vulnerability & Exploitation

CVE-2024-35250

Unserialize the property set

- KSPROPERTY_TYPE_UNSERIALIZESET
  - Interaction with multiple properties with a single call



UnserializePropertySet

```
NTSTATUS __fastcall KspPropertyHandler(
        PIRP Irp,
        unsigned int propertysetscnt,
        KSPROPERTY_SET *propertyset,
        __int64 (__fastcall *a4)(_QWORD, _QWORD, _QWORD),
        int a5,
        __int64 NodeAutomationTable,
        unsigned int NodeCnt){

        // check if the UserProvideProperty->Set is in the propertyset

        ...
        if ( KsProperty_flag == KSPROPERTY_TYPE_UNSERIALIZESET )
          return UnserializePropertySet(Irp, sysbuf_, propertyset_);
        ...

}
```

UnserializePropertySet

```c
unsigned __int64 __fastcall UnserializePropertySet(
    PIRP irp,
    KSIDENTIFIER* UserProvideProperty,
    KSPROPERTY_SET* propertyset_)
{
    ...
    New_KsProperty_req = ExAllocatePoolWithTag(NonPagedPoolNx, InSize, 0x7070534Bu);
    ...
    memmove(New_KsProperty_req, CurrentStackLocation->Parameters.DeviceIoControl.Type3InputBuffer, InSize);
    ...
    status = KsSynchronousIoControlDevice(
            CurrentStackLocation->FileObject,
            0,
            CurrentStackLocation->Parameters.DeviceIoControl.IoControlCode,
            New_KsProperty_req,
            InSize,
            OutBuffer,
            OutSize,
            &BytesReturned);
    ...
}
```

UnserializePropertySet

```
unsigned __int64 __fastcall UnserializePropertySet(
    PIRP irp,
    KSIDENTIFIER* UserProvideProperty,
    KSPROPERTY_SET* propertyset_)
{
    ...
    New_KsProperty_req = ExAllocatePoolWithTag(NonPagedPoolNx, InSize, 0x7070534Bu);
    ...
    memmove(New_KsProperty_req, CurrentStackLocation->Parameters.DeviceIoControl.Type3InputBuffer, InSize);
    ...
    status = KsSynchronousIoControlDevice(
            CurrentStackLocation->FileObject,
            0,
            CurrentStackLocation->Parameters.DeviceIoControl.IoControlCode,
            New_KsProperty_req,
            InSize,
            OutBuffer,
            OutSize,
            &BytesReturned);
    ...
}
```

UnserializePropertySet

```
unsigned __int64 __fastcall UnserializePropertySet(
    PIRP irp,
    KSIDENTIFIER* UserProvideProperty,
    KSPROPERTY_SET* propertyset_)
{
    ...
    New_KsProperty_req = ExAllocatePoolWithTag(NonPagedPoolNx, InSize, 0x7070534Bu);
    ...
    memmove(New_KsProperty_req, CurrentStackLocation->Parameters.DeviceIoControl.Type3InputBuffer, InSize);
    ...
    status = KsSynchronousIoControlDevice(
            CurrentStackLocation->FileObject,
            0,
            CurrentStackLocation->Parameters.DeviceIoControl.IoControlCode,
            New_KsProperty_req,
            InSize,
            OutBuffer,
            OutSize,
            &BytesReturned);
    ...
}
```

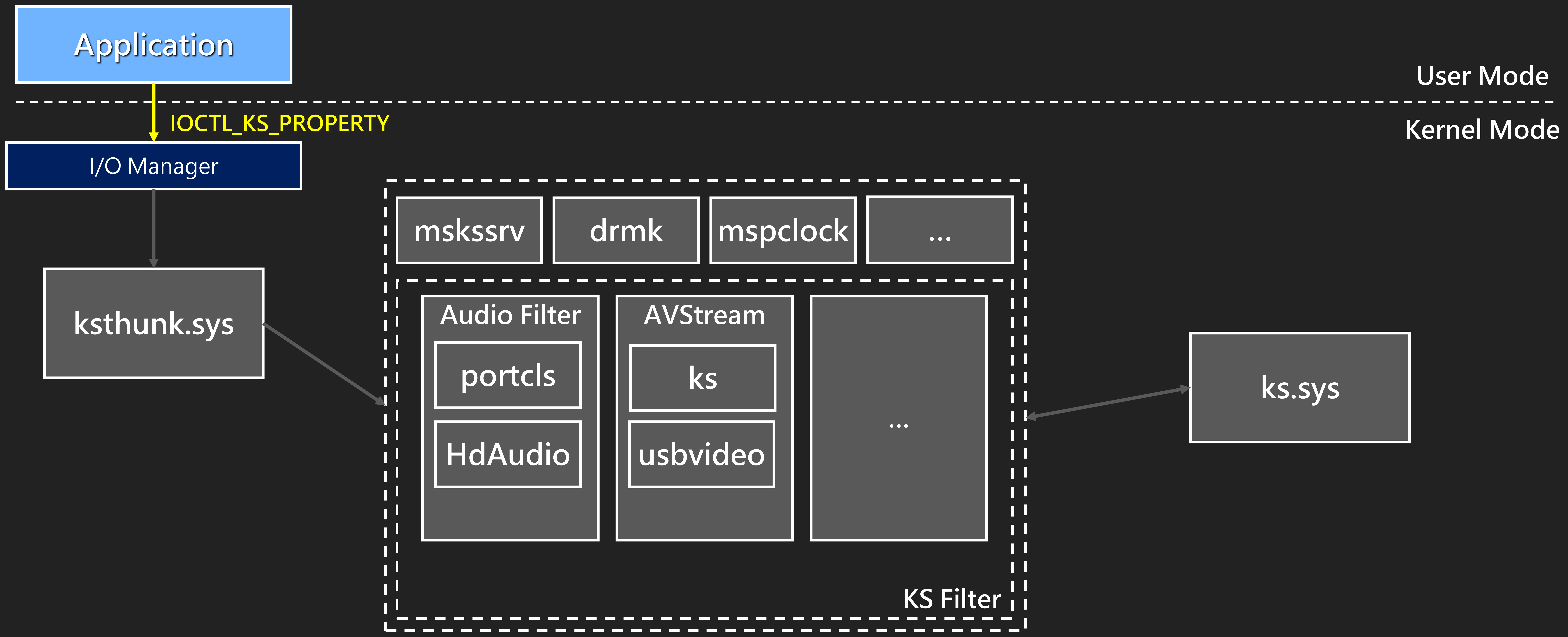
KernelMode

UnserializePropertySet

```
unsigned __int64 __fastcall UnserializePropertySet(
    PIRP irp,
    KSIDENTIFIER* UserProvideProperty,
    KSPROPERTY_SET* propertyset_)
{
    ...
    New_KsProperty_req = ExAllocatePoolWithTag(NonPagedPoolNx, InSize, 0x7070534Bu);
    ...
    memmove(New_KsProperty_req, CurrentStackLocation->Parameters.DeviceIoControl.Type3InputBuffer, InSize);
    ...
    status = KsSynchronousIoControlDevice(
            CurrentStackLocation->FileObject,
            0,
            CurrentStackLocation->Parameters.DeviceIoControl.IoControlCode,
            New_KsProperty_req,
            InSize,
            OutBuffer,
            OutSize,
            &BytesReturned);
    ...
}
```

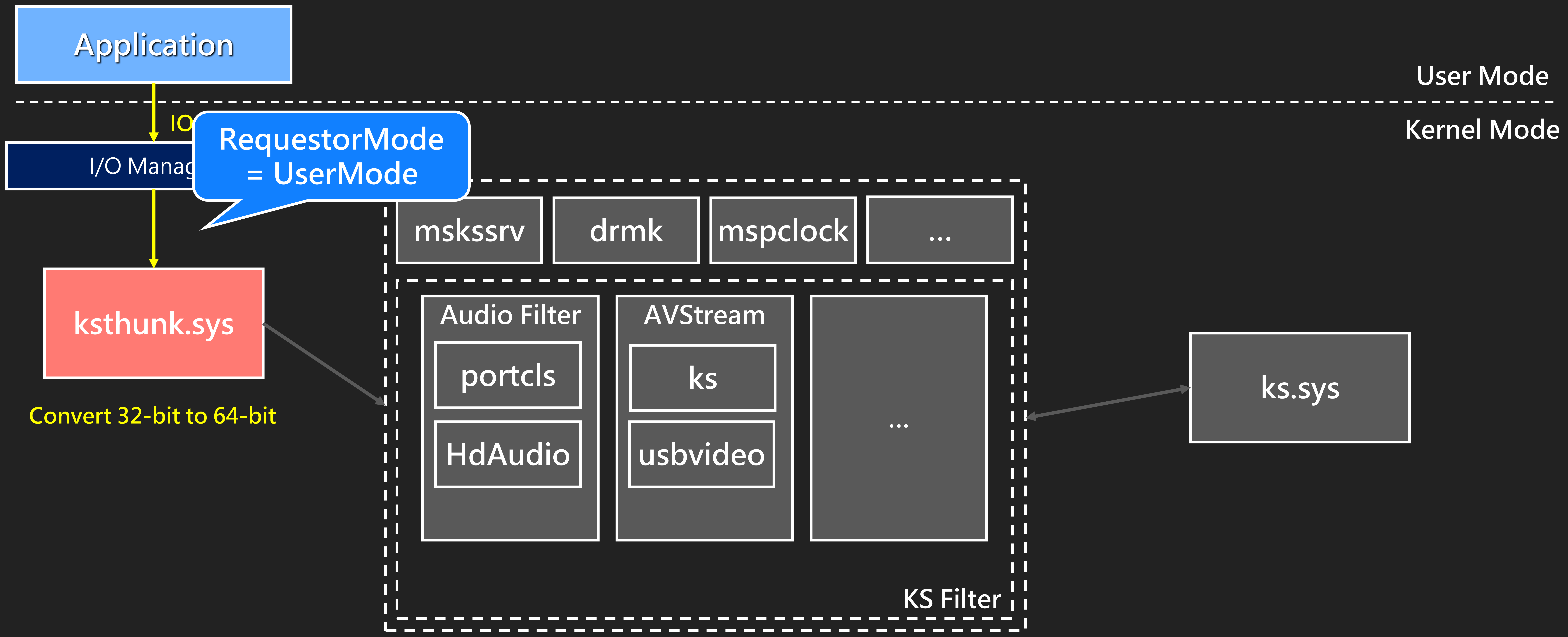
User Control

UnserializePropertySet



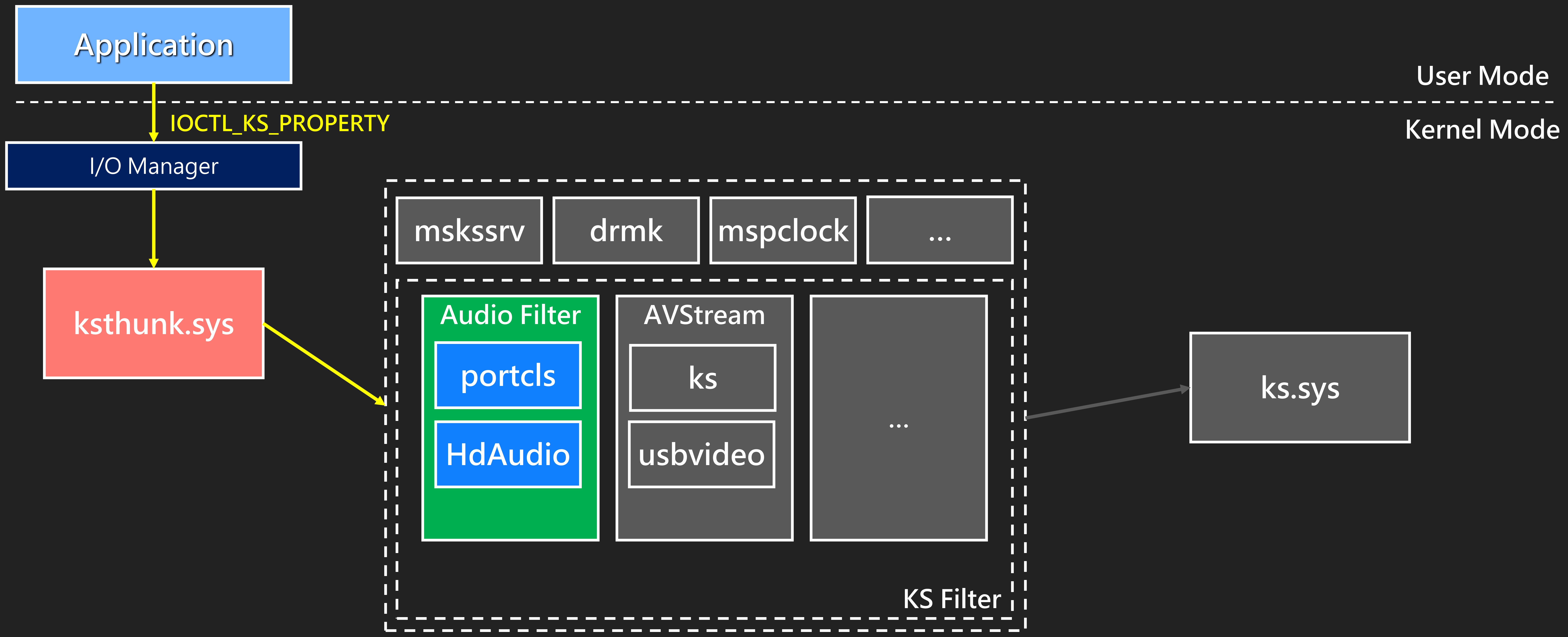
DEVCORE

UnserializePropertySet



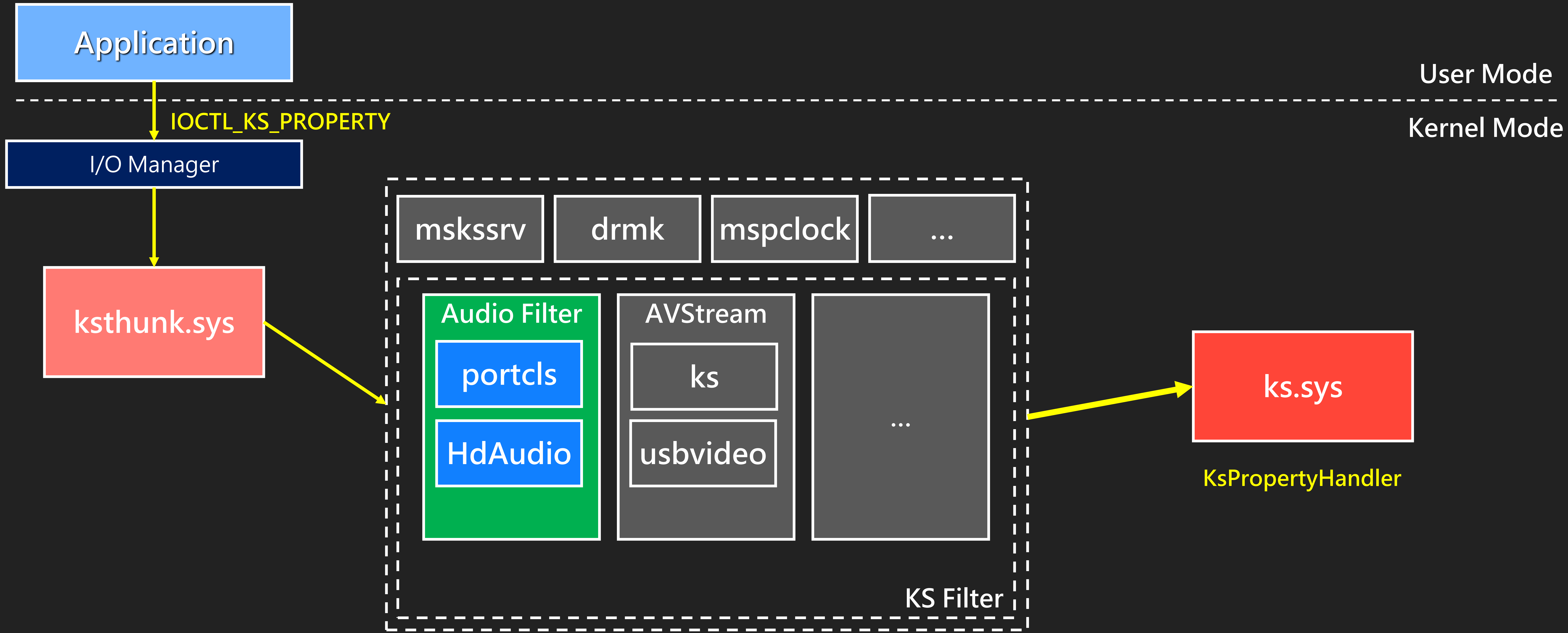
DEVCORE

UnserializePropertySet



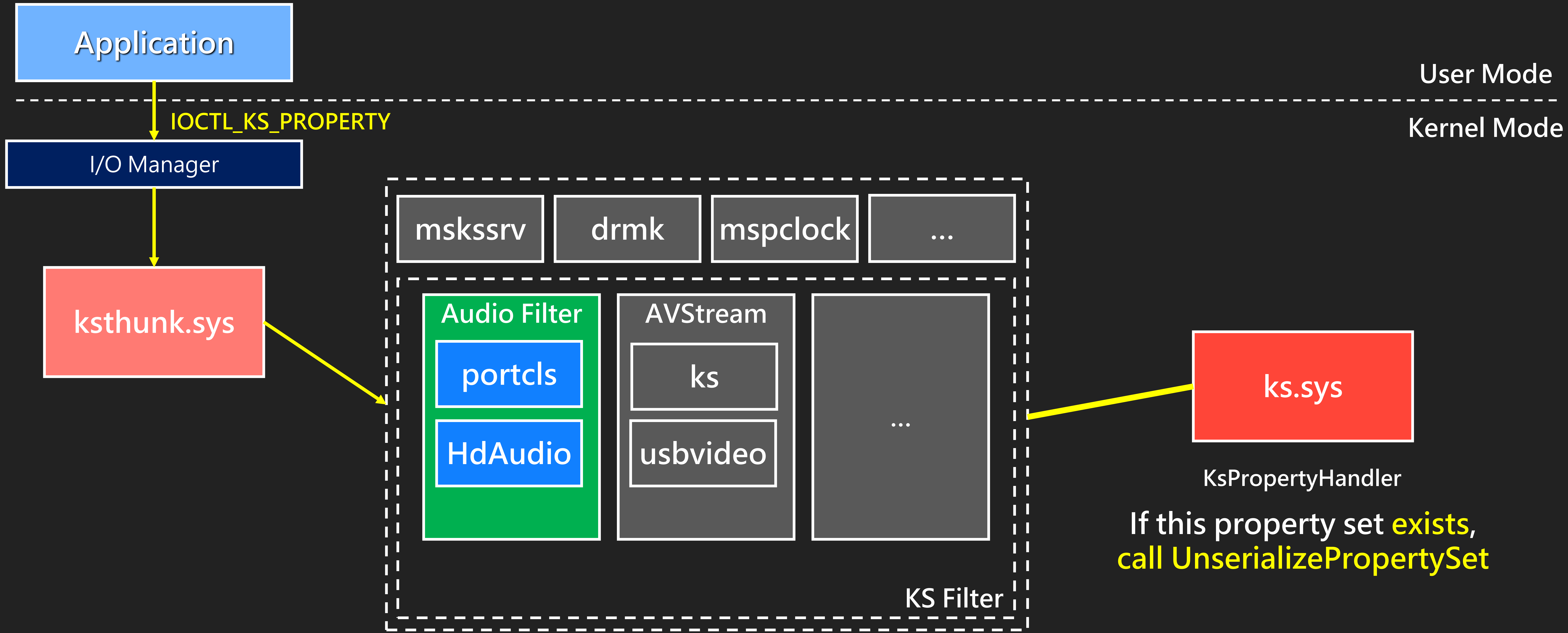
DEVCORE

UnserializePropertySet



DEVCORE

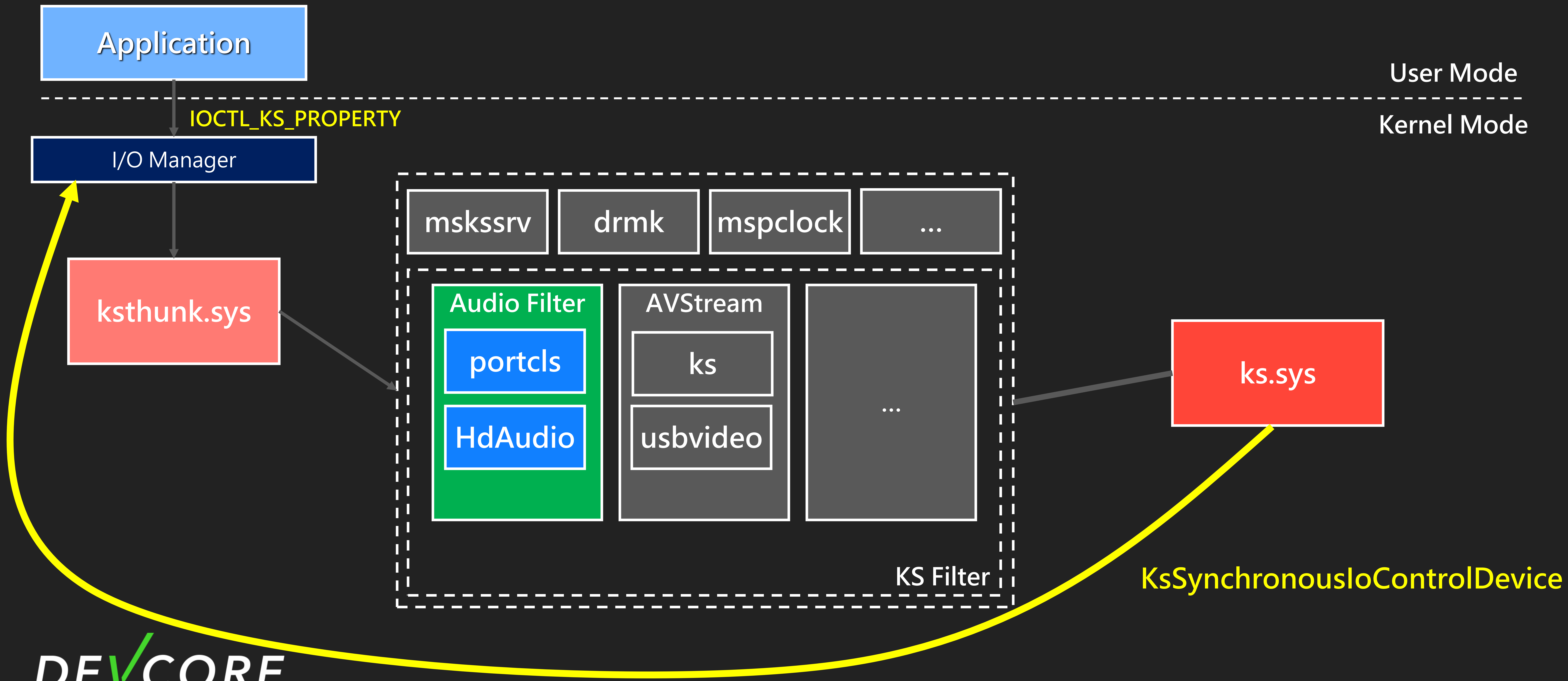
UnserializePropertySet



If this property set exists,
call UnserializePropertySet

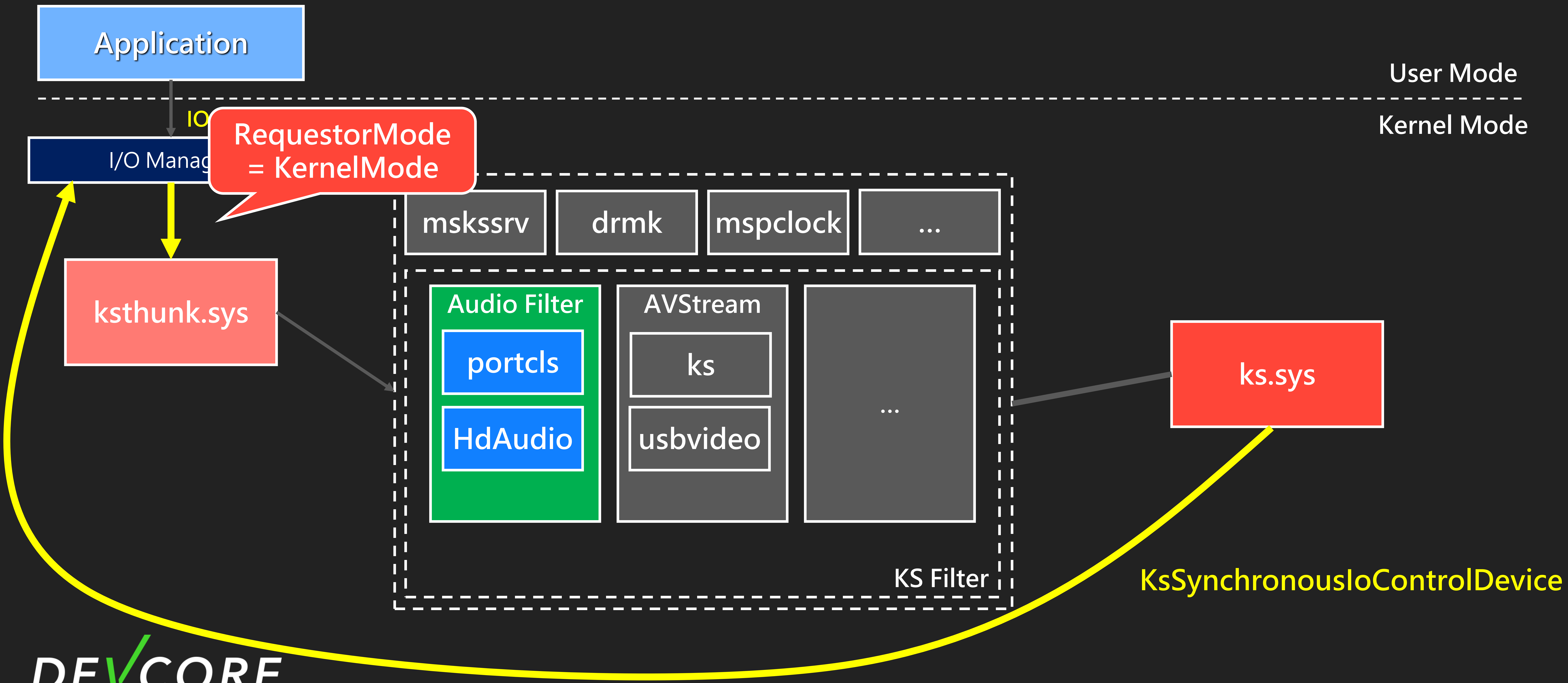
DEVCORE

UnserializePropertySet



DEVCORE

UnserializePropertySet



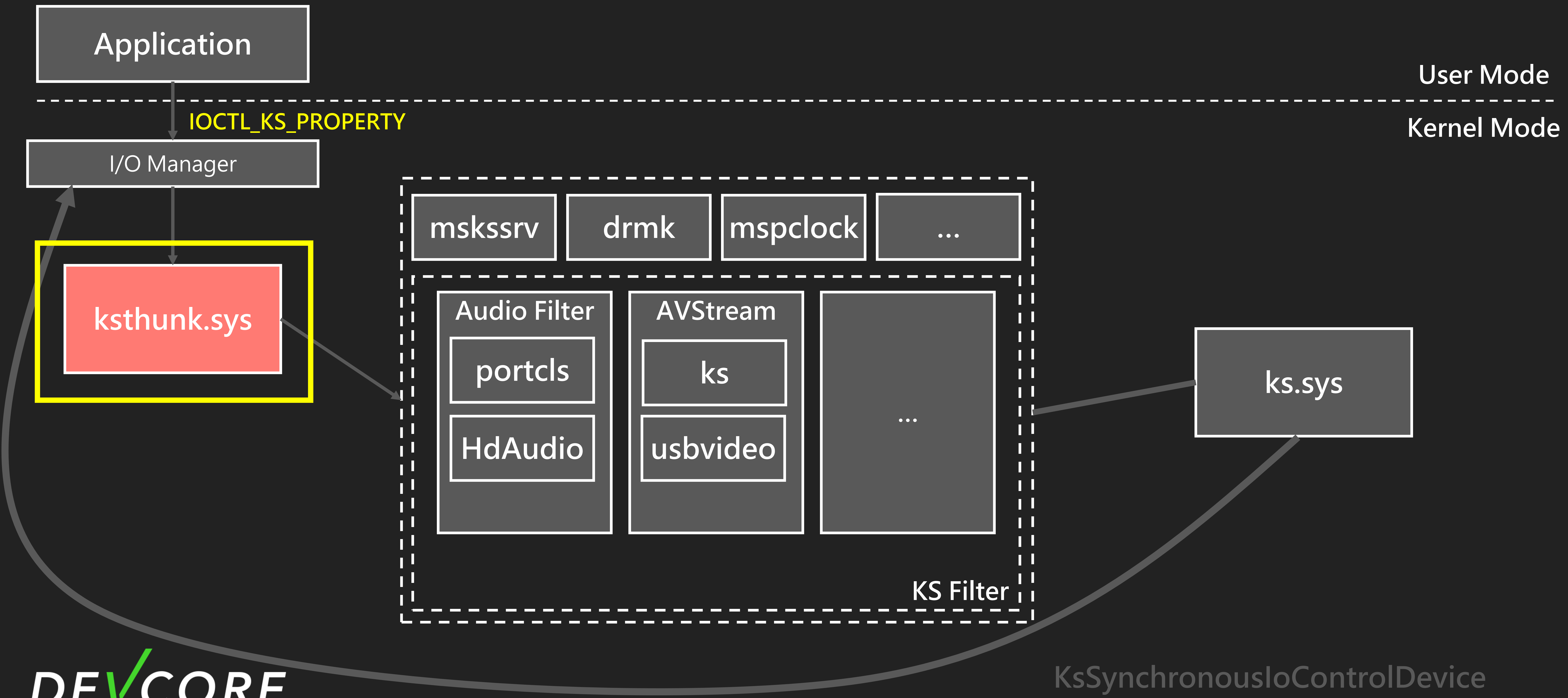
We can do arbitrary IOCTL_KS_PROPERTY with KernelMode now

DEVCORE

We need to find a target to EoP

DEVCORE

UnserializePropertySet



DEVCORE

ksthunk! DispatchIoctl

```
__int64 __fastcall CKSThunkDevice::CheckIrpForStackAdjustmentNative(__int64 a1, struct _IRP *irp, __int64 a3, int *a4)
{
  ...
    if ( *(_OWORD *)&Type3InputBuffer->Set == *(_OWORD *)&KSPROPSETID_DrmAudioStream
        && !type3inputbuf.Id
        && (type3inputbuf.Flags & 2) != 0 )        // KSPROPERTY_TYPE_SET
    {
        if ( irp->RequestorMode )
        {
        v14 = 0xC0000010;
        }
        else
        {
        UserBuffer = (unsigned int *)irp->UserBuffer;
        ...
        v14 = (*(__int64 (__fastcall **)(_QWORD, _QWORD, __int64 *))(Type3InputBuffer + 0x38))(// call Type3InputBuffer+0x38
                *UserBuffer,
                0LL,
                v19);
        }
    }
}
```

ksthunk! DispatchIoctl

```
__int64 __fastcall CKSThunkDevice::CheckIrpForStackAdjustmentNative(__int64 a1, struct _IRP *irp, __int64 a3, int *a4)
{
  ...
    if ( *(_OWORD *)&Type3InputBuffer->Set == *(_OWORD *)&KSPROPSETID_DrmAudioStream
        && !type3inputbuf.Id
        && (type3inputbuf.Flags & 2) != 0 )        // KSPROPERTY_TYPE_SET
    {
        if ( irp->RequestorMode )
        {
        v14 = 0xC0000010;
        }
        else
        {
        UserBuffer = (unsigned int *)irp->UserBuffer;
        ...
        v14 = (*(__int64 (__fastcall **)(_QWORD, _QWORD, __int64 *))(Type3InputBuffer + 0x38))(// call Type3InputBuffer+0x38
                *UserBuffer,
                0LL,
                v19);
        }
    }
}
```

ksthunk! DispatchIoctl

```
__int64 __fastcall CKSThunkDevice::CheckIrpForStackAdjustmentNative(__int64 a1, struct _IRP *irp, __int64 a3, int *a4)
{
  ...
    if ( *(_OWORD *)&Type3InputBuffer->Set == *(_OWORD *)&KSPROPSETID_DrmAudioStream
        && !type3inputbuf.Id
        && (type3inputbuf.Flags & 2) != 0 )        // KSPROPERTY_TYPE_SET
    {
        if ( irp->RequestorMode )
        {
        v14 = 0xC0000010;
        }
        else
        {
        UserBuffer = (unsigned int *)irp->UserBuffer;
        ...
        v14 = (*(__int64 (__fastcall **)(_QWORD, _QWORD, __int64 *))(Type3InputBuffer + 0x38))(// call Type3InputBuffer+0x38
                *UserBuffer,
                0LL,
                v19);
        }
    }
}
```

RequestorMode == KernelMode (0)

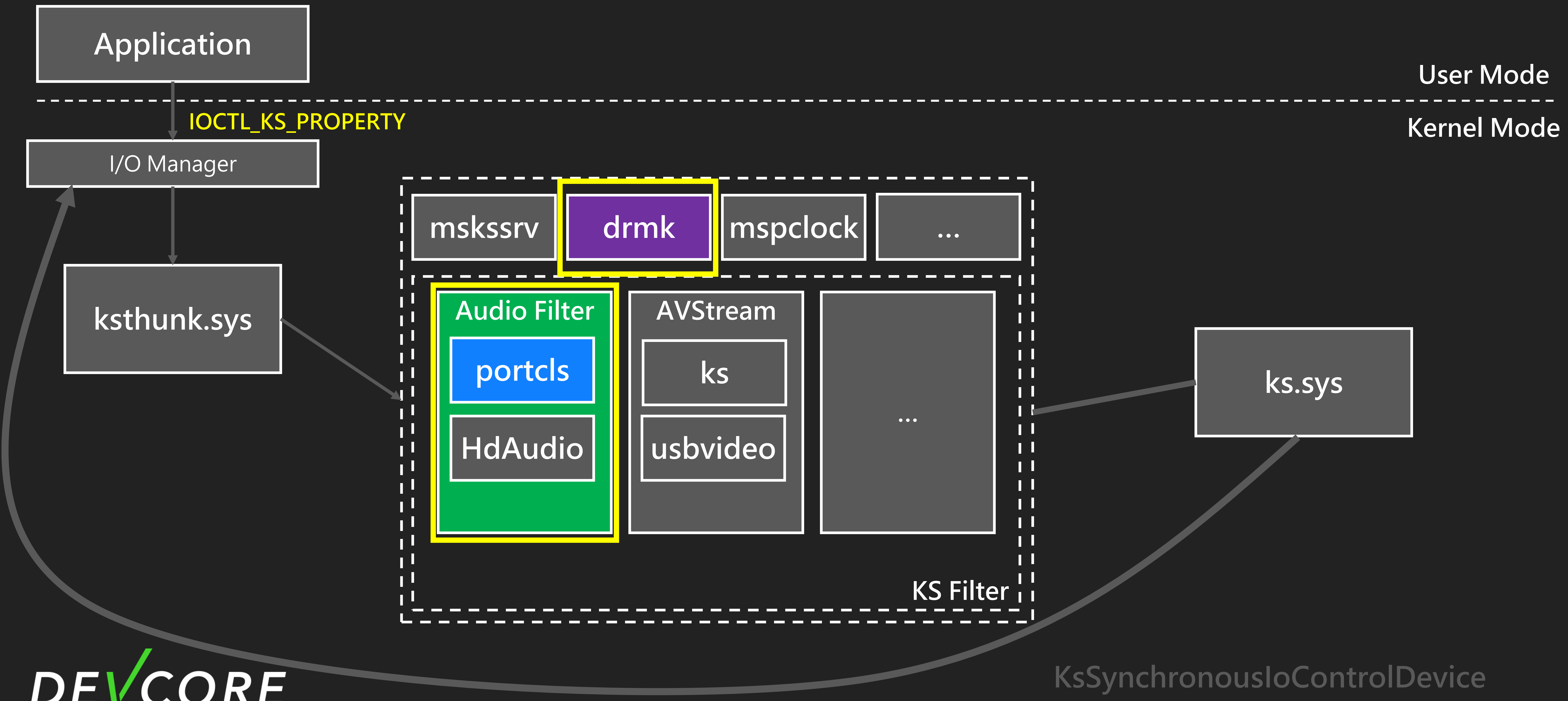
ksthunk! DispatchIoctl

```
__int64 __fastcall CKSThunkDevice::CheckIrpForStackAdjustmentNative(__int64 a1, struct _IRP *irp, __int64 a3, int *a4)
{
  ...
    if ( *(_OWORD *)&Type3InputBuffer->Set == *(_OWORD *)&KSPROPSETID_DrmAudioStream
        && !type3inputbuf.Id
        && (type3inputbuf.Flags & 2) != 0 )        // KSPROPERTY_TYPE_SET
    {
        if ( irp->RequestorMode )
        {
        v14 = 0xC0000010;
        }
        else
        {
        UserBuffer = (unsigned int *)irp->UserBuffer;
        ...
        v14 = (*(__int64 (__fastcall **)(_QWORD, _QWORD, __int64 *))(Type3InputBuffer + 0x38))(// call Type3InputBuffer+0x38
                *UserBuffer,
                0LL,
                v19);
        }
    }
}
```

ksthunk! DispatchIoctl

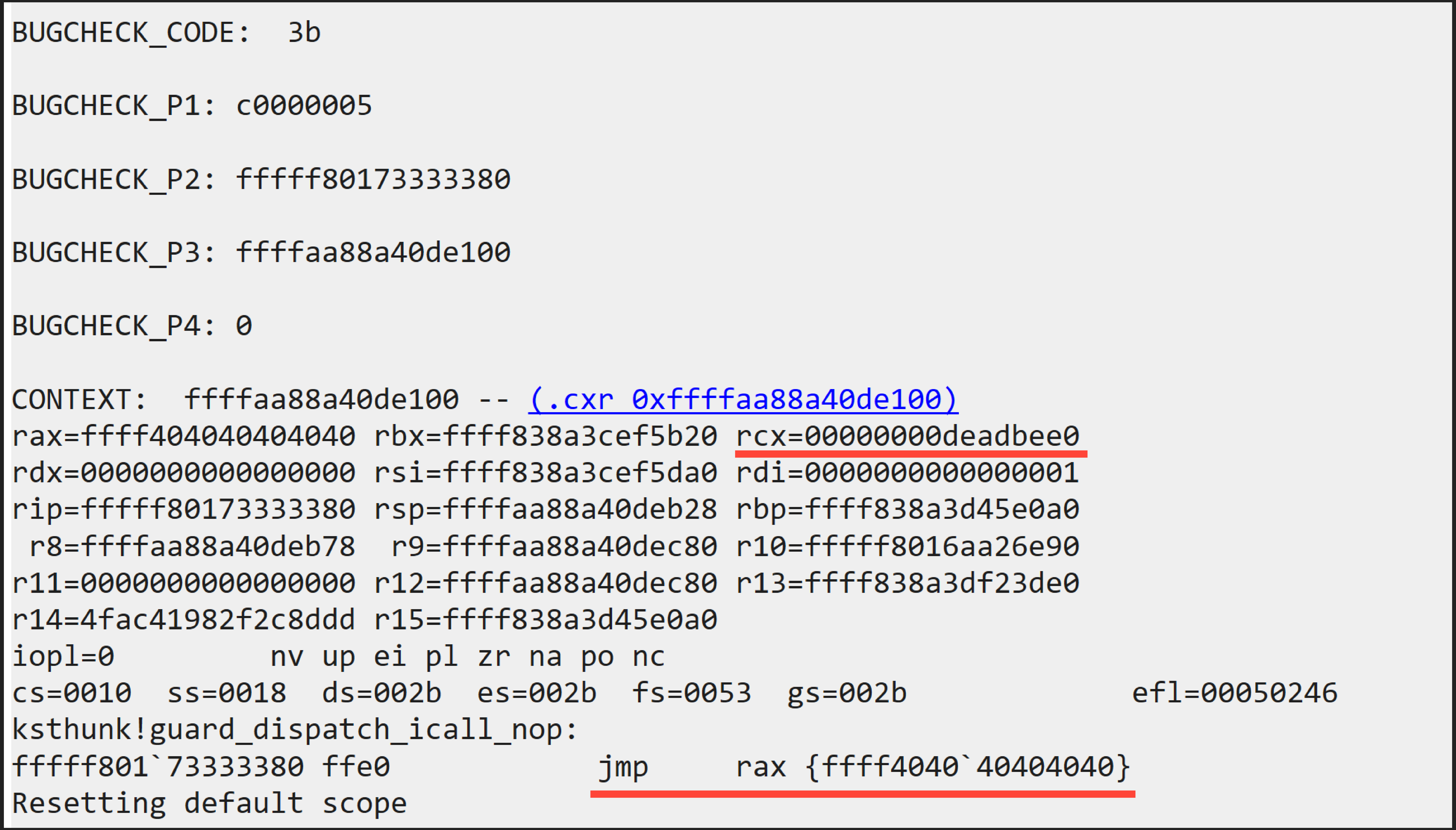
```
__int64 __fastcall CKSThunkDevice::CheckIrpForStackAdjustmentNative(__int64 a1, struct _IRP *irp, __int64 a3, int *a4)
{
  ...
    if ( *(_OWORD *)&Type3InputBuffer->Set == *(_OWORD *)&KSPROPSETID_DrmAudioStream
        && !type3inputbuf.Id
        && (type3inputbuf.Flags & 2) != 0 )       // KSPROPERTY_TYPE_SET
    {
        if ( irp->RequestorMode )
        {
        v14 = 0xC0000010;
        }
        else
        {
        UserBuffer = (unsigned int *)irp->UserBuffer;
        ...
        v14 = (*(__int64 (__fastcall **)(_QWORD, _QWORD, __int64 *))(Type3InputBuffer + 0x38))(// call Type3InputBuffer+0x38
                *UserBuffer,
                0LL,
                v19);
        }
    }
}
```

UnserializePropertySet



DEVCORE

ksthunk! DispatchIoctl



```
BUGCHECK_CODE:  3b

BUGCHECK_P1: c0000005

BUGCHECK_P2: fffff80173333380

BUGCHECK_P3: ffffaa88a40de100

BUGCHECK_P4: 0

CONTEXT:  ffffaa88a40de100 -- (.cxr 0xffffaa88a40de100)
rax=ffff404040404040 rbx=ffff838a3cef5b20 rcx=00000000deadbee0
rdx=0000000000000000 rsi=ffff838a3cef5da0 rdi=0000000000000001
rip=fffff80173333380 rsp=ffffaa88a40deb28 rbp=ffff838a3d45e0a0
 r8=ffffaa88a40deb78  r9=ffffaa88a40dec80 r10=fffff8016aa26e90
r11=0000000000000000 r12=ffffaa88a40dec80 r13=ffff838a3df23de0
r14=4fac41982f2c8ddd r15=ffff838a3d45e0a0
iopl=0         nv up ei pl zr na po nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00050246
ksthunk!guard_dispatch_icall_nop:
fffff801`73333380 ffe0            jmp     rax {ffff4040`40404040}
Resetting default scope
```

We have an arbitrary call with one argument now

DEVCORE

Exploitation

Mitigation on Win11

- kCFG
- kASLR
- SMEP
- …

Mitigation on Win11

- kCFG
- kASLR ✓
    - NtQuerySystemInformation
- SMEP ✓
    - Reuse Kernel Code
- …

DEVCORE

Bypass kCFG

- Find a valid function in Windows Kernel
  - Our goal is turn arbitrary call to arbitrary memory write

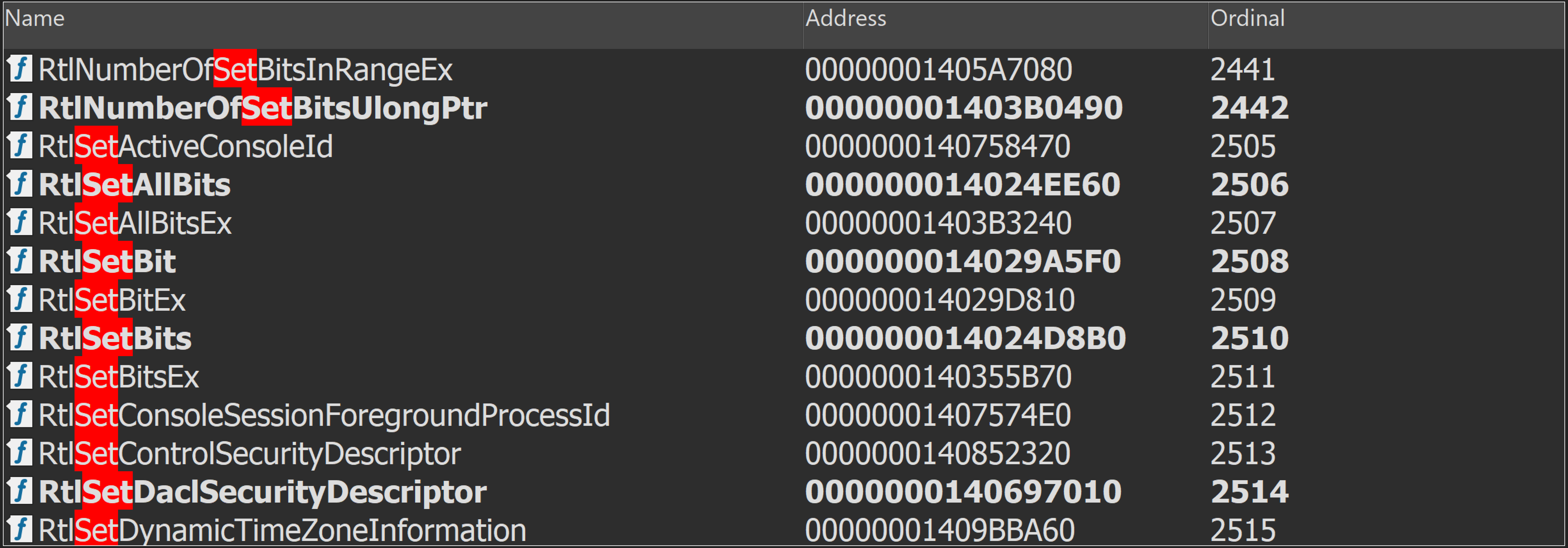
DEVCORE

Bypass kCFG

- Find a valid function in Windows Kernel
  - Our goal is turn arbitrary call to arbitrary memory write
    - Search *Set* function export from ntoskrnl.exe

DEVCORE

Bypass kCFG



Name	Address	Ordinal
RtlNumberOfSetBitsInRangeEx	00000001405A7080	2441
RtlNumberOfSetBitsUlongPtr	**00000001403B0490**	**2442**
RtlSetActiveConsoleId	0000000140758470	2505
RtlSetAllBits	**000000014024EE60**	**2506**
RtlSetAllBitsEx	00000001403B3240	2507
RtlSetBit	**000000014029A5F0**	**2508**
RtlSetBitEx	000000014029D810	2509
RtlSetBits	**000000014024D8B0**	**2510**
RtlSetBitsEx	0000000140355B70	2511
RtlSetConsoleSessionForegroundProcessId	00000001407574E0	2512
RtlSetControlSecurityDescriptor	0000000140852320	2513
RtlSetDaclSecurityDescriptor	**0000000140697010**	**2514**
RtlSetDynamicTimeZoneInformation	00000001409BBA60	2515

DEVCORE

Two hours later …

DEVCORE

Bypass kCFG

```
void __stdcall RtlSetAllBits(PRTL_BITMAP BitMapHeader)
{
  unsigned int *Buffer; // r8
  unsigned __int64 v2; // rdx

  Buffer = BitMapHeader->Buffer;
  v2 = (unsigned __int64)(4 * (((BitMapHeader->SizeOfBitMap & 0x1F) != 0) + (BitMapHeader->SizeOfBitMap >> 5))) >> 2;
  if ( v2 )
  {
    ...
    memset(Buffer, 0xFFu, 8 * (v2 >> 1));
    if ( (v2 & 1) != 0 )
      Buffer[v2 - 1] = -1;
  }
}
```

DEVCORE

Bypass kCFG

- RtlSetAllBits
  - The RtlSetAllBits routine sets all bits in a given bitmap variable.

```
NTSYSAPI VOID RtlSetAllBits(
  [in] PRTL_BITMAP BitMapHeader
);
```

```
struct _RTL_BITMAP
{
    ULONG SizeOfBitMap;
    ULONG* Buffer;
};
```

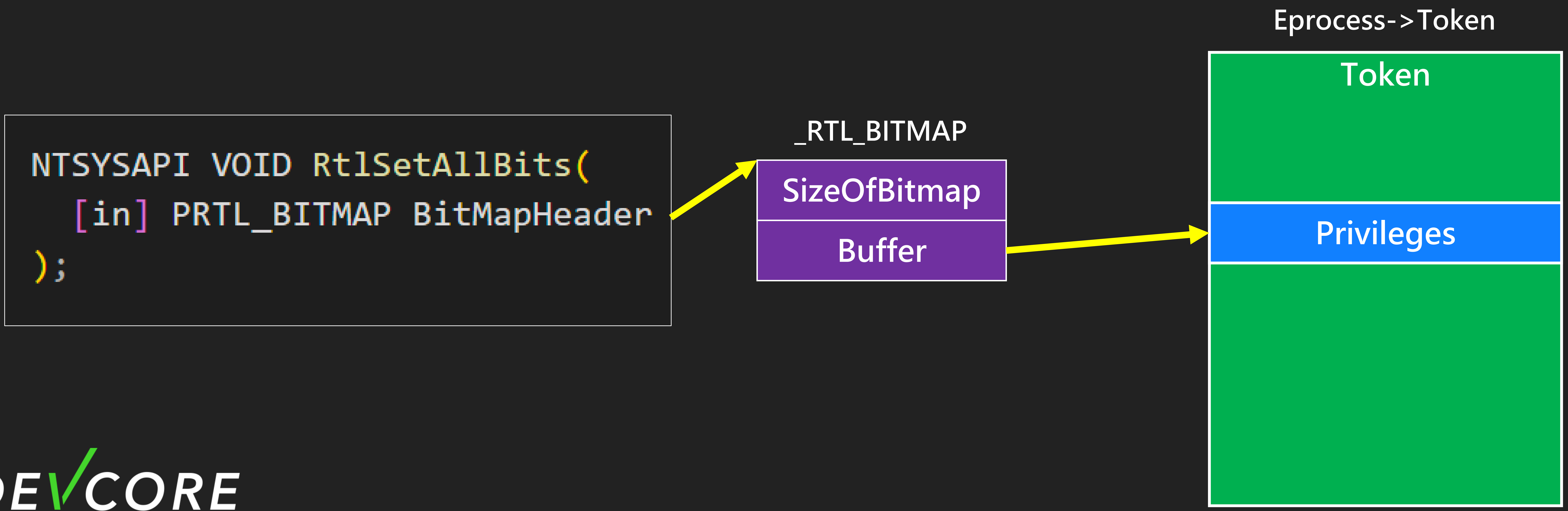
DEVCORE

We can set all bits in arbitrary memory

DEVCORE

Abuse token privilege

- We can use the primitive to
    - Enable all privilege in current process token

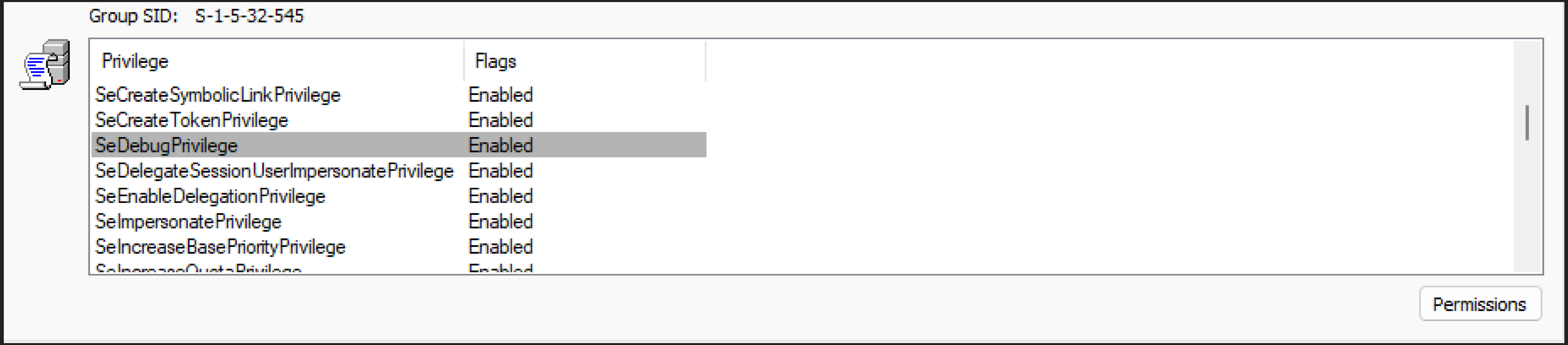


```
NTSYSAPI VOID RtlSetAllBits(
  [in] PRTL_BITMAP BitMapHeader
);
```

DEVCORE

Abuse token privilege

- We can use the primitive to
  - Enable all privilege in current process token



Group SID: S-1-5-32-545

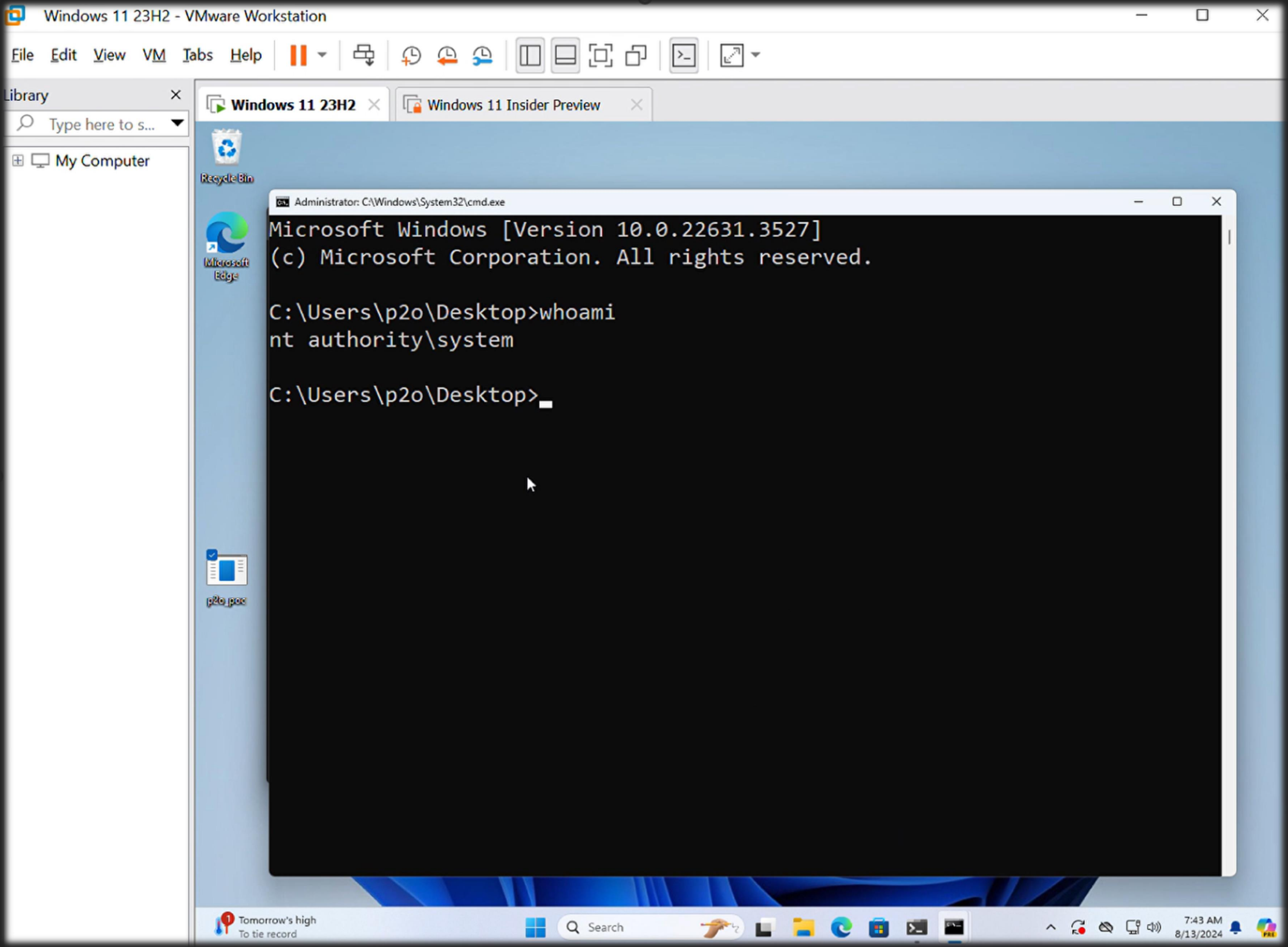
Privilege	Flags
SeCreateSymbolicLinkPrivilege	Enabled
SeCreateTokenPrivilege	Enabled
SeDebugPrivilege	Enabled
SeDelegateSessionUserImpersonatePrivilege	Enabled
SeEnableDelegationPrivilege	Enabled
SeImpersonatePrivilege	Enabled
SeIncreaseBasePriorityPrivilege	Enabled
SeIncreaseQuotaPrivilege	Enabled

DEVCORE

The Last Step

- Well-known EoP method with SeDebugPrivilege
  - Open process of winlogon.exe
  - Set thread attribute to PROC_THREAD_ATTRIBUTE_PARENT_PROCESS
  - Spawn cmd.exe

DEVCORE

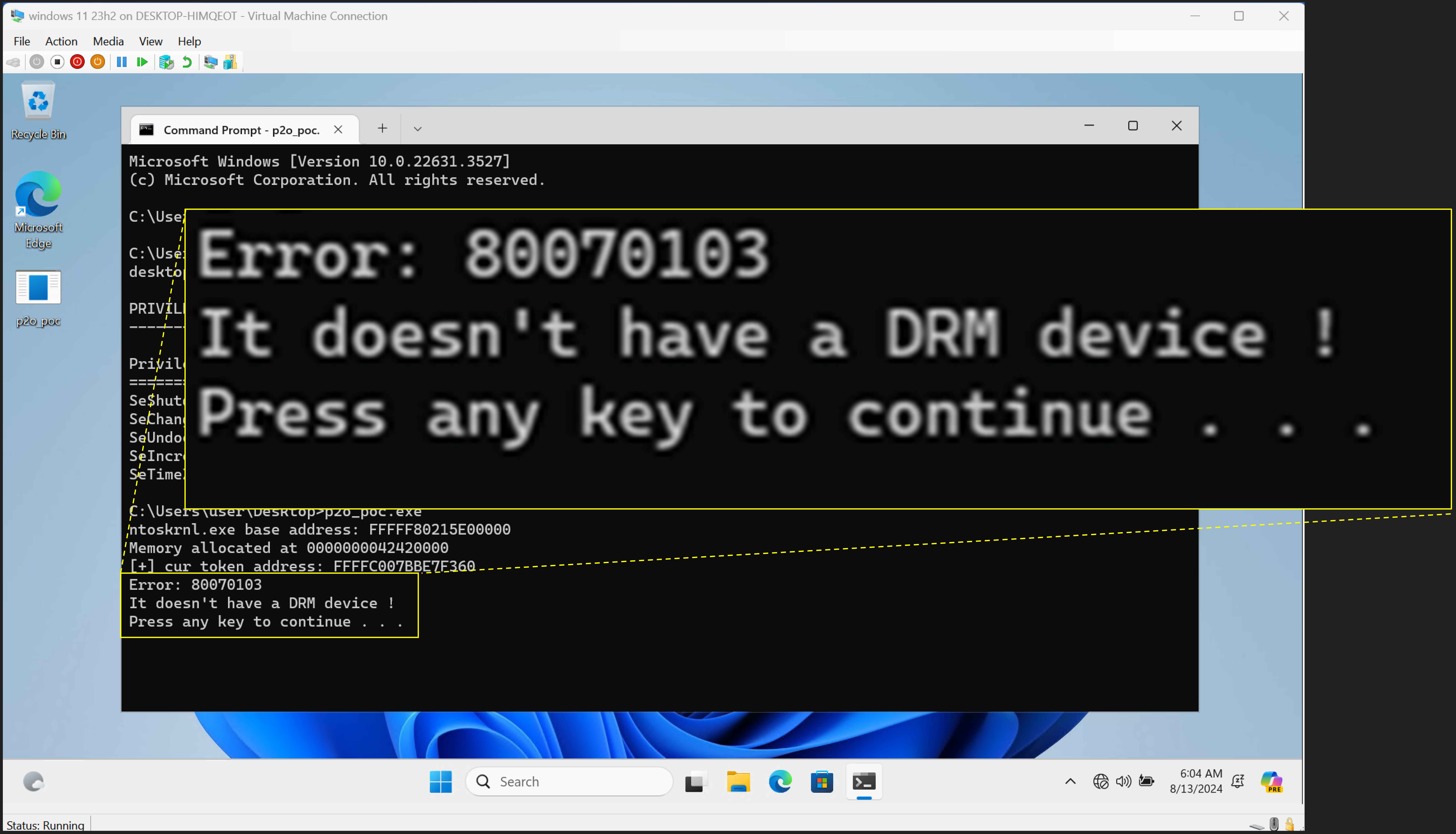


```
Microsoft Windows [Version 10.0.22631.3527]
(c) Microsoft Corporation. All rights reserved.

C:\Users\p2o\Desktop>whoami
nt authority\system

C:\Users\p2o\Desktop>_
```

It's like a Proxy to Kernel !

DEVCORE

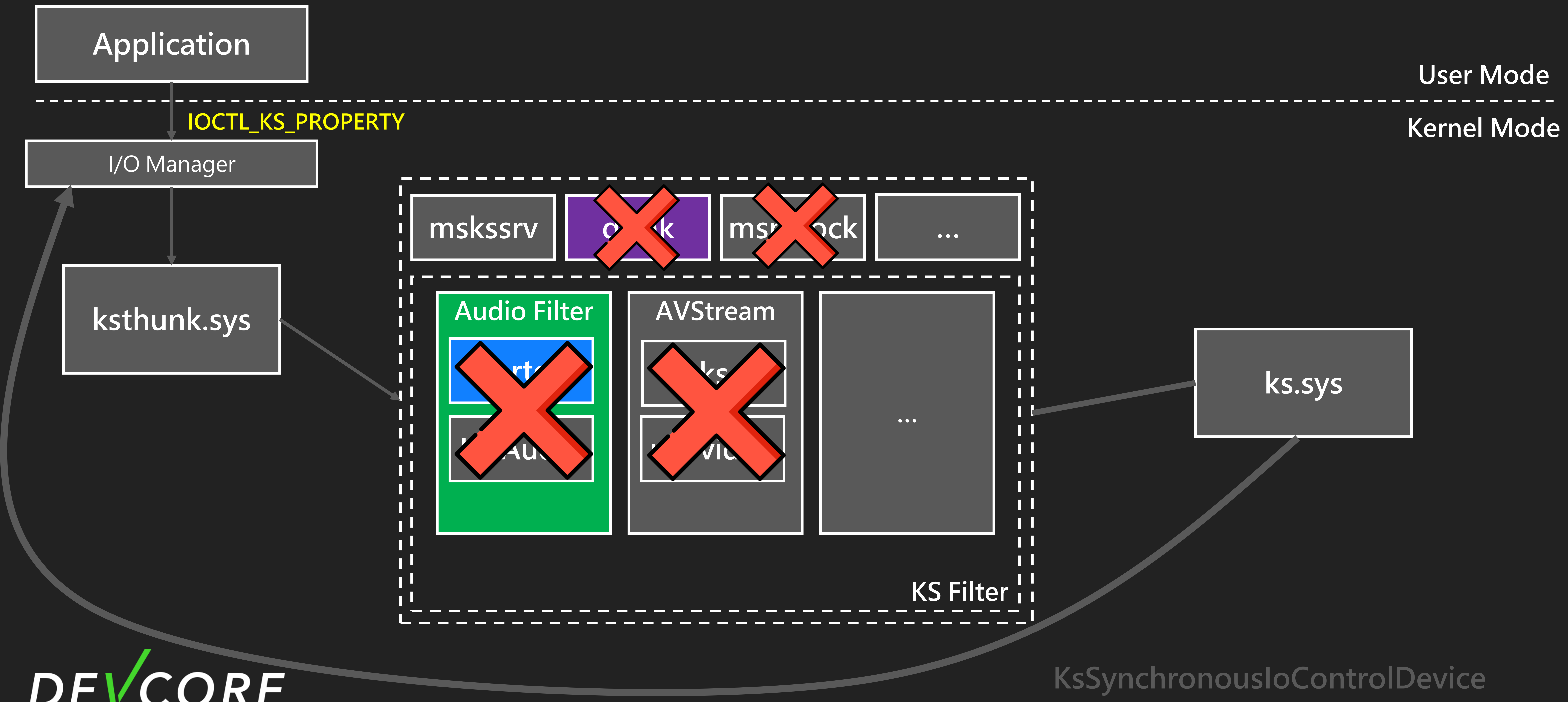
However ...



```
Microsoft Windows [Version 10.0.22631.3527]
(c) Microsoft Corporation. All rights reserved.

C:\Use

C:\Use
deskto

PRIVIL
------

Privil
======
SeShut
SeChan
SeUndo
SeIncr
SeTime

C:\Users\user\Desktop>p2o_poc.exe
ntoskrnl.exe base address: FFFFF80215E00000
Memory allocated at 0000000042420000
[+] cur token address: FFFFC007BBE7F360
Error: 80070103
It doesn't have a DRM device !
Press any key to continue . . .
```

```
Error: 80070103
It doesn't have a DRM device !
Press any key to continue . . .
```

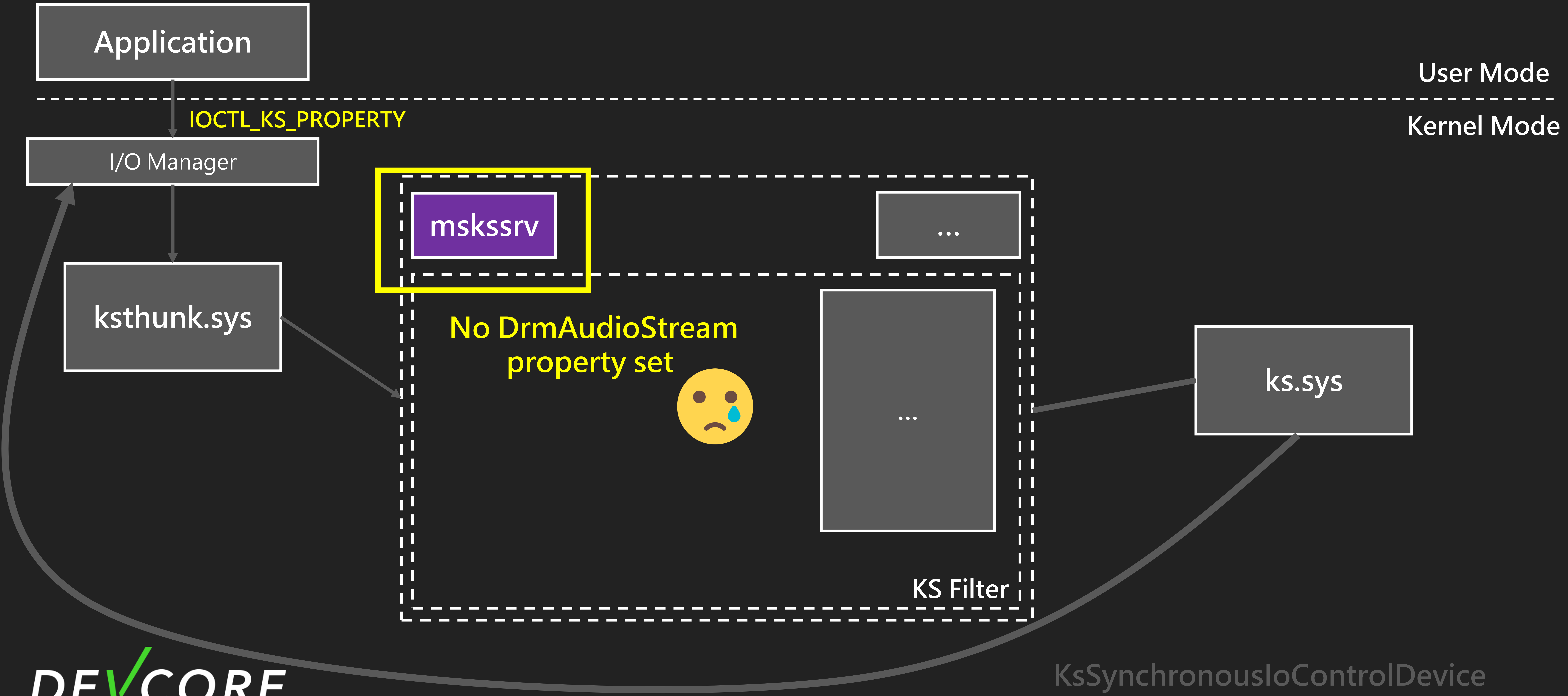
KS Device in Hyper-V



KsSynchronousIoControlDevice

DEVCORE

KS Device in Hyper-V



DEVCORE

CVE-2024-30084

IOCTL_KS_PROPERTY

- Neither I/O
    - Using user input buffer directly
- Inputbuffer = Parameters.DeviceIoControl.Type3InputBuffer
- Outputbuffer = Irp->UserBuffer

DEVCORE

KspPropertyHandler

User input buffer

```
NTSTATUS __fastcall KspPropertyHandler(
        PIRP Irp,
        unsigned int propertysetscnt,
        KSPROPERTY_SET *propertyset,
        ...){

        memmove(SystemBuffer[outlen_padding],
        CurrentStackLocation->Parameters.DeviceIoControl.Type3InputBuffer,
        InputBufferLength);
        ...
        Guid = *&SystemBuffer[outlen_padding];
        // Check if the Guid is in the property set


        ...
        if ( KsProperty_flag == KSPROPERTY_TYPE_UNSERIALIZESET )
          return UnserializePropertySet(Irp, sysbuf_, propertyset_);
        ...

}
```

KspPropertyHandler

```
NTSTATUS __fastcall KspPropertyHandler(
        PIRP Irp,
        unsigned int propertysetscnt,
        KSPROPERTY_SET *propertyset,
        ...){

        memmove(SystemBuffer[outlen_padding],
        CurrentStackLocation->Parameters.DeviceIoControl.Type3InputBuffer,
        InputBufferLength);
        ...
        Guid = *&SystemBuffer[outlen_padding];
        // Check if the Guid is in the property set

        ...
        if ( KsProperty_flag == KSPROPERTY_TYPE_UNSERIALIZESET )
          return UnserializePropertySet(Irp, sysbuf_, propertyset_);
        ...

}
```

Let's take a look at UnserializePropertySet again

DEVCORE

UnserializePropertySet

```
unsigned __int64 __fastcall UnserializePropertySet(
    PIRP irp,
    KSIDENTIFIER* UserProvideProperty,
    KSPROPERTY_SET* propertyset_)
{
    ...
    New_KsProperty_req = ExAllocatePoolWithTag(NonPagedPoolNx, InSize, 0x7070534Bu);
    ...
    memmove(New_KsProperty_req, CurrentStackLocation->Parameters.DeviceIoControl.Type3InputBuffer, InSize);
    ...
    status = KsSynchronousIoControlDevice(
            CurrentStackLocation->FileObject,
            0,
            CurrentStackLocation->Parameters.DeviceIoControl.IoControlCode,
            New_KsProperty_req,
            InSize,
            OutBuffer,
            OutSize,
            &BytesReturned);
    ...
}
```

Copy User input again ! ?

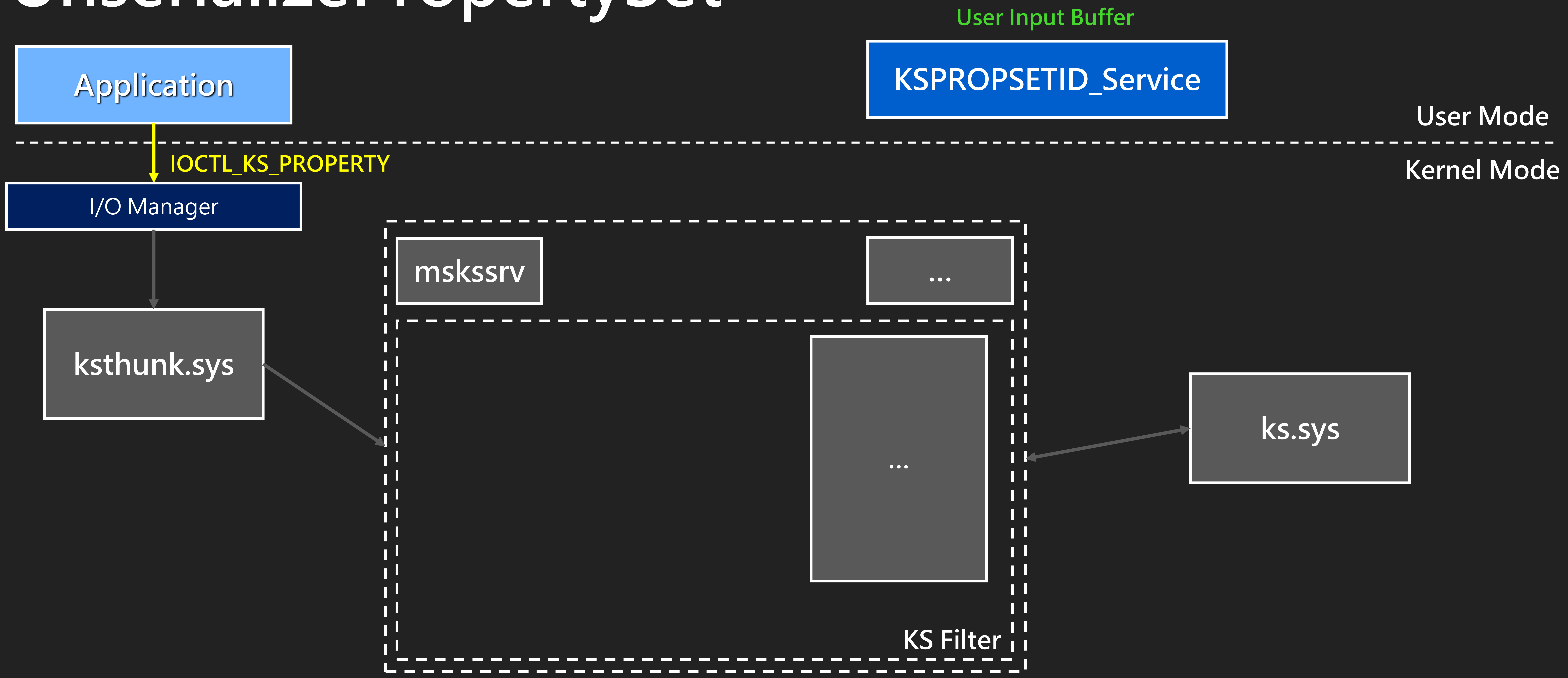
UnserializePropertySet

```
unsigned __int64 __fastcall UnserializePropertySet(
    PIRP irp,
    KSIDENTIFIER* UserProvideProperty,
    KSPROPERTY_SET* propertyset_)
{
    ...
    New_KsProperty_req = ExAllocatePoolWithTag(NonPagedPoolNx, InSize, 0x7070534Bu);
    ...
    memmove(New_KsProperty_req, CurrentStackLocation->Parameters.DeviceIoControl.Type3InputBuffer, InSize);
    ...
    status = KsSynchronousIoControlDevice(
            CurrentStackLocation->FileObject,
            0,
            CurrentStackLocation->Parameters.DeviceIoControl.IoControlCode,
            New_KsProperty_req,
            InSize,
            OutBuffer,
            OutSize,
            &BytesReturned);
    ...
}
```

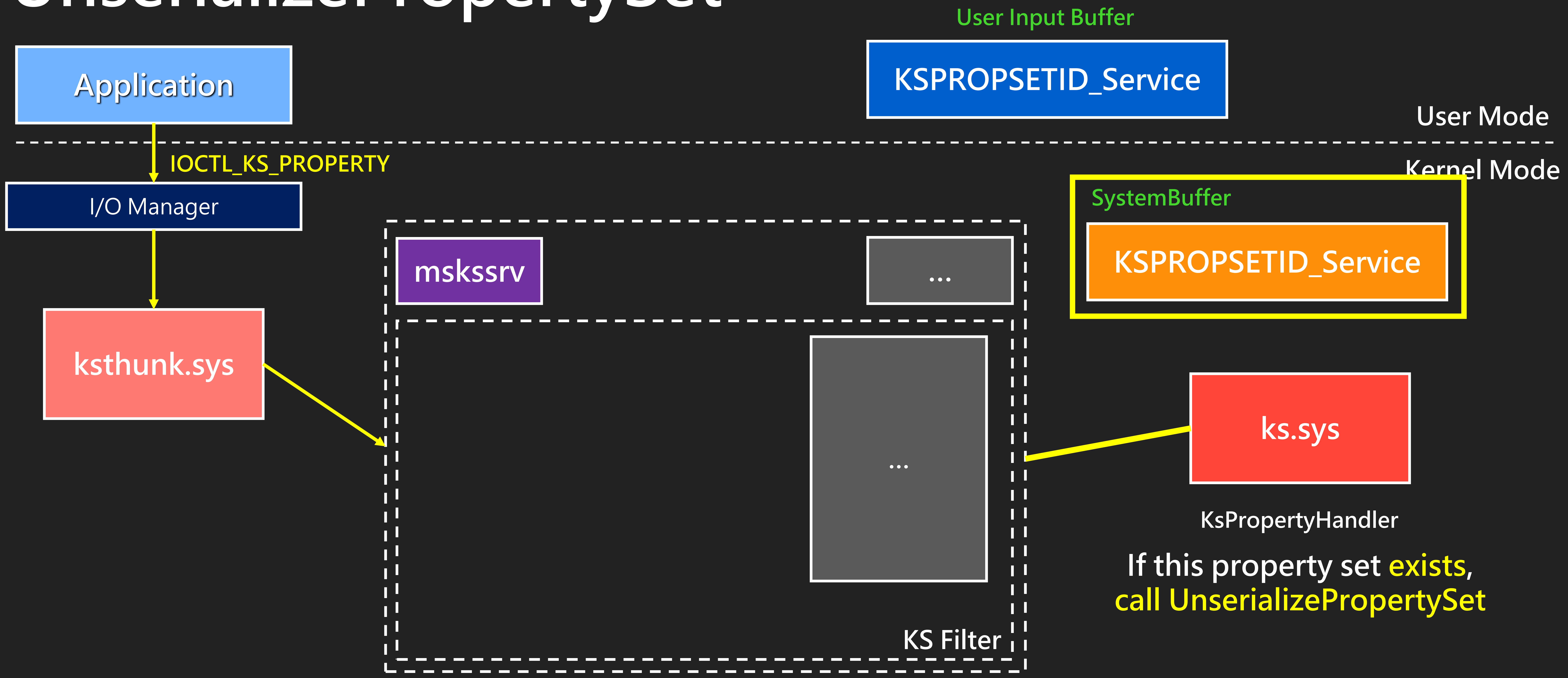
Copy User input again !?

Double Fetch

UnserializePropertySet



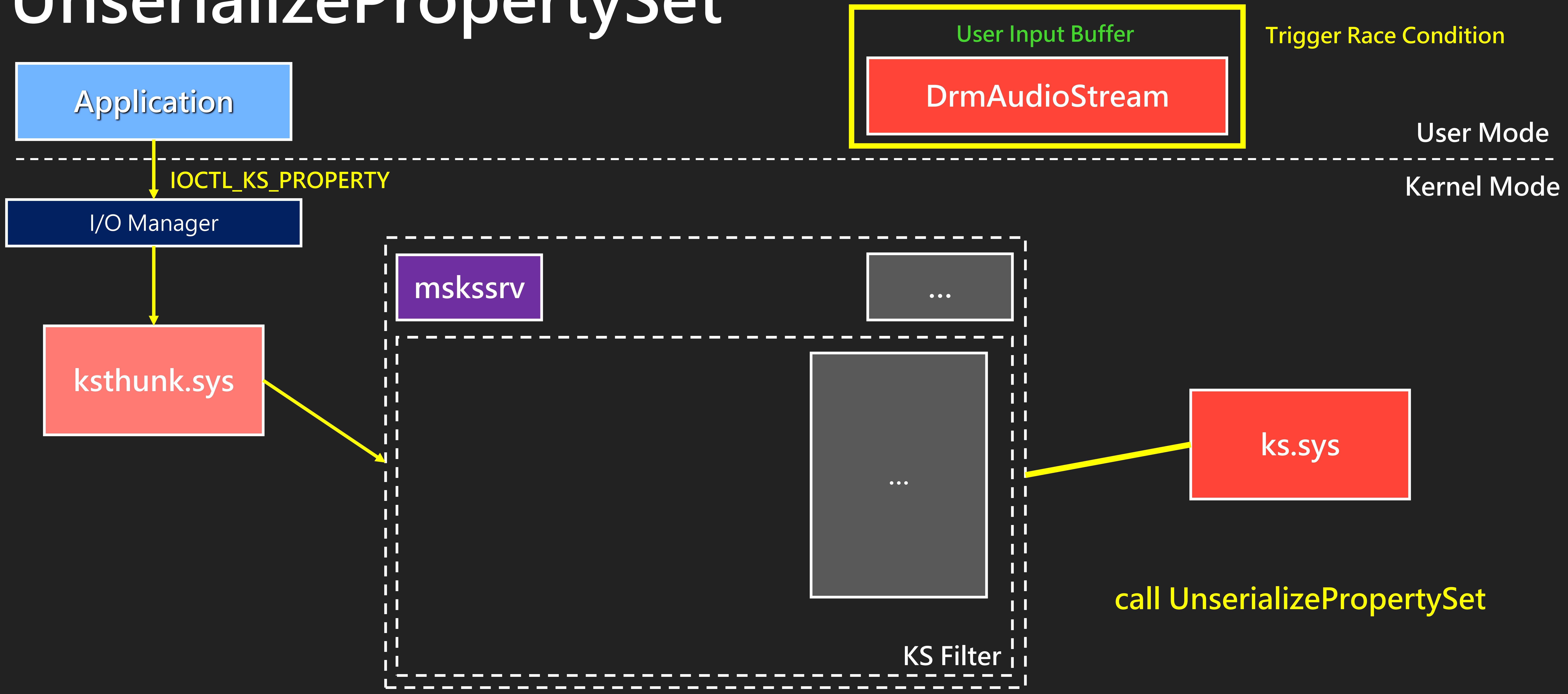
UnserializePropertySet



If this property set exists,
call UnserializePropertySet

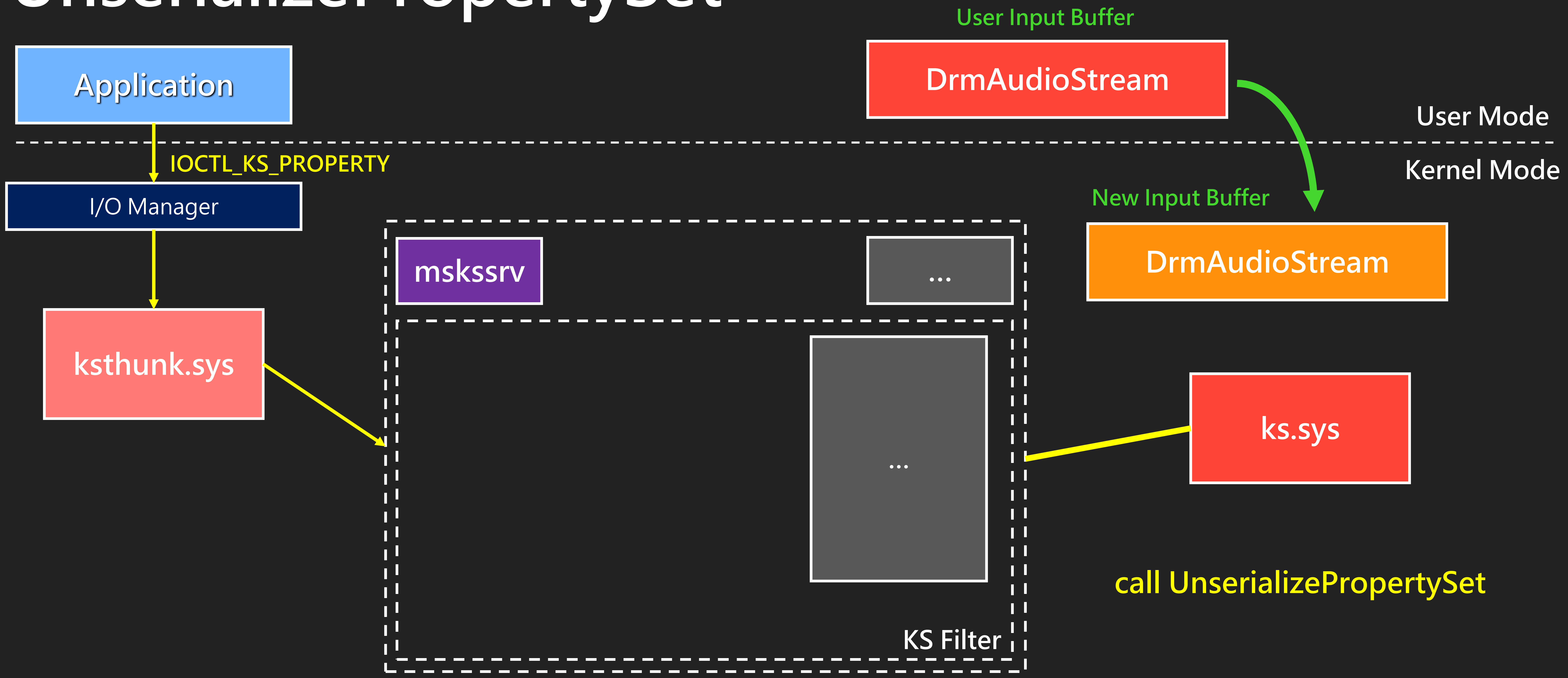
DEVCORE

UnserializePropertySet



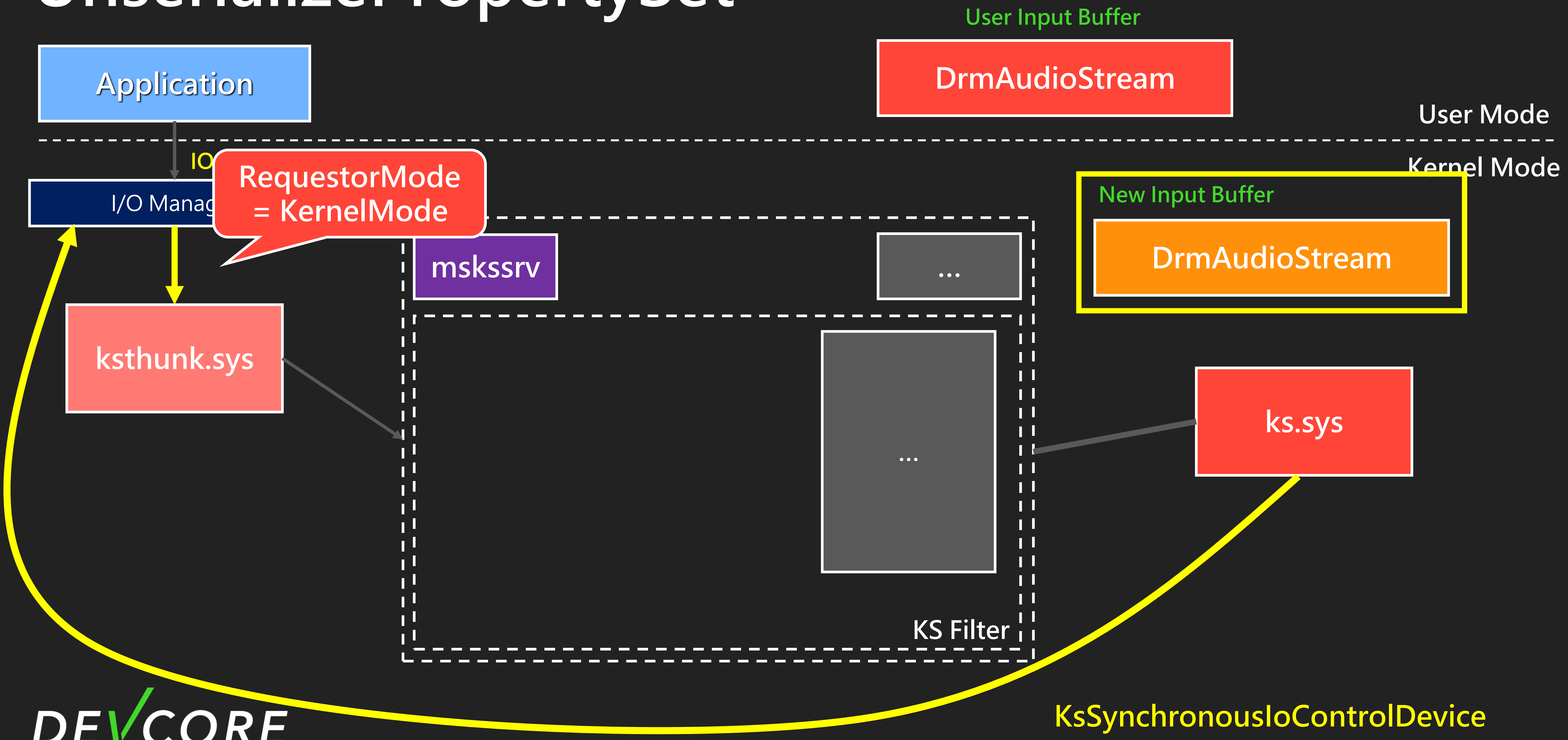
DEVCORE

UnserializePropertySet

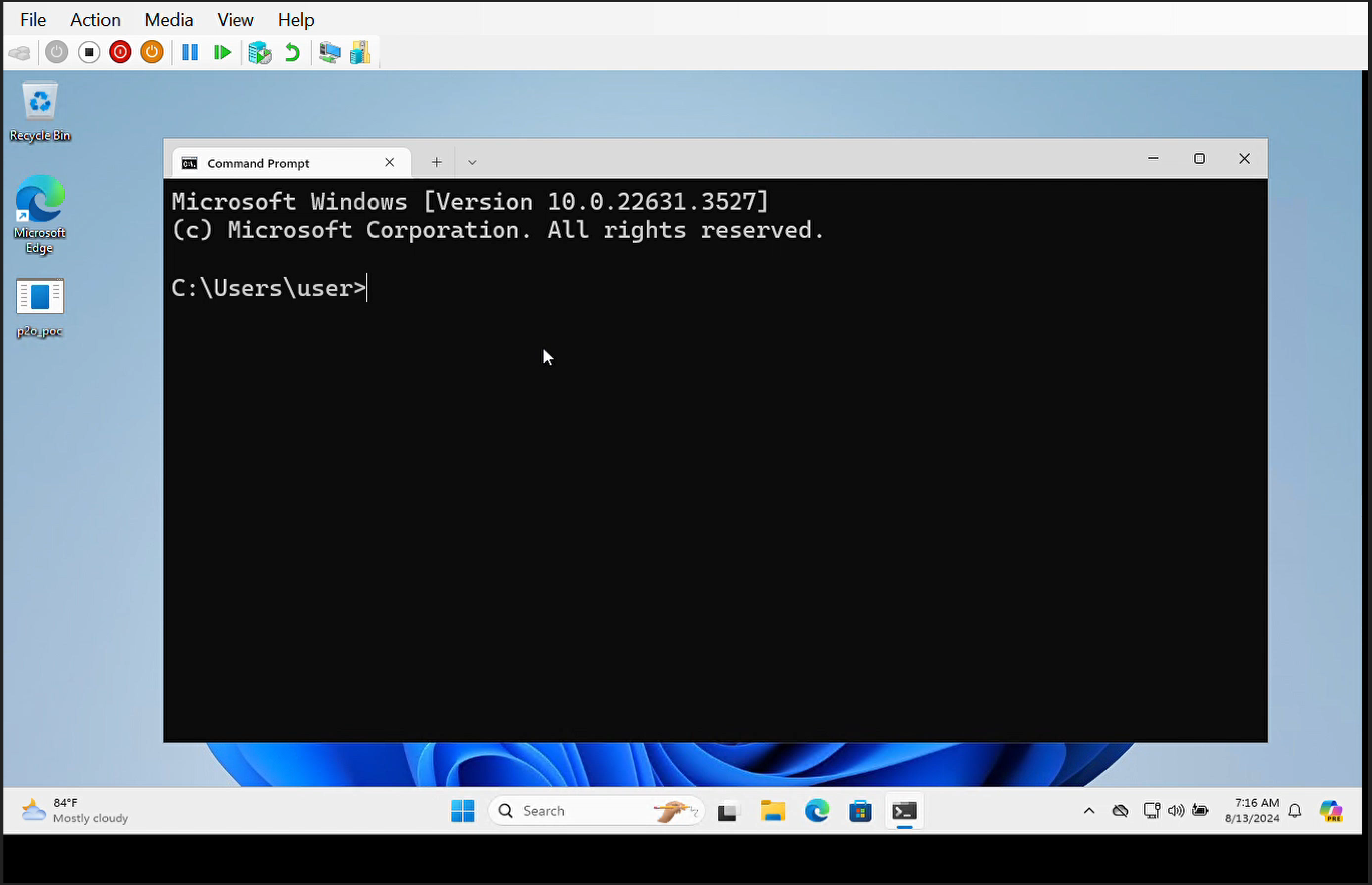


DEVCORE

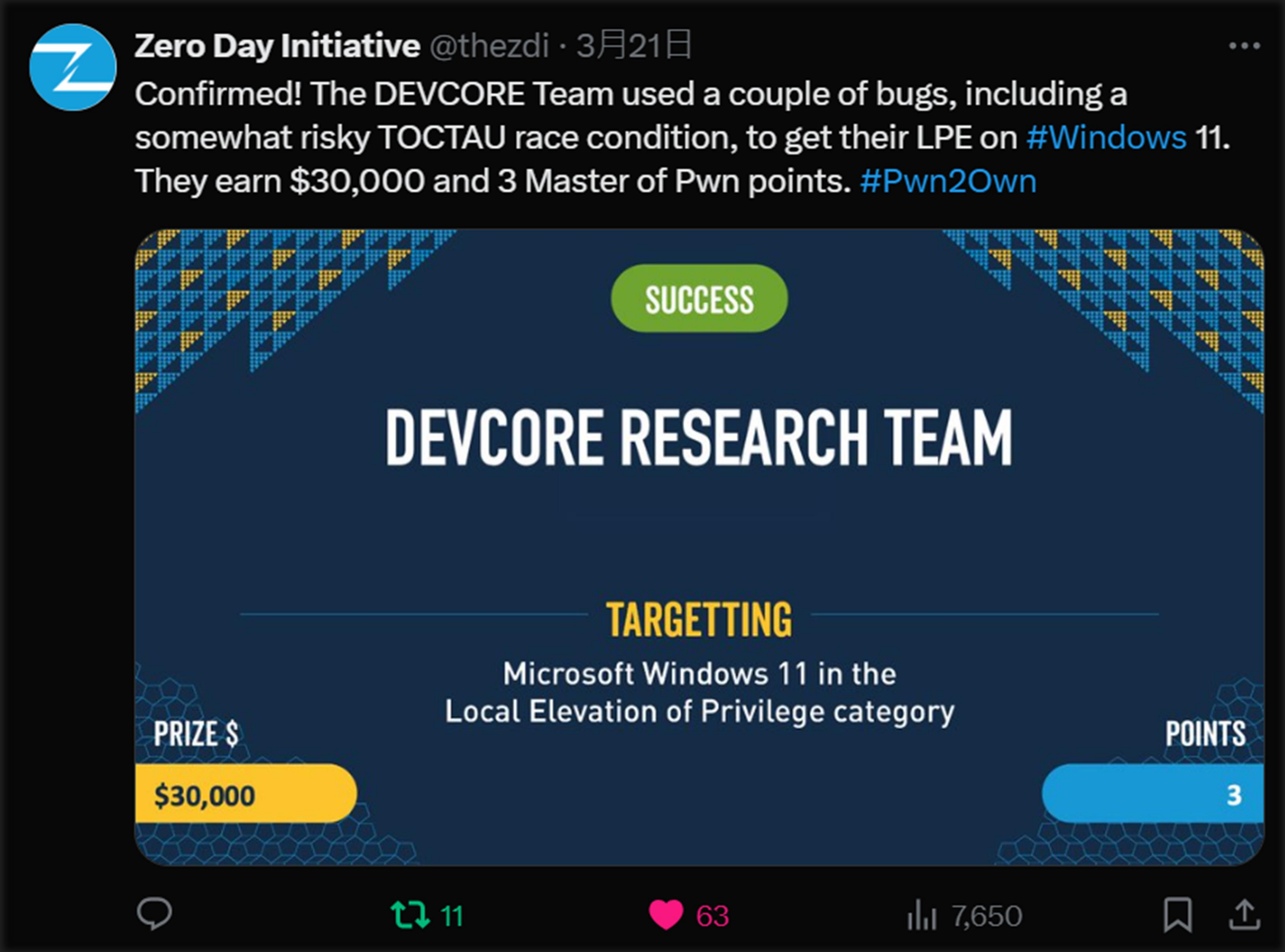
UnserializePropertySet



DEVCORE



```
Microsoft Windows [Version 10.0.22631.3527]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>
```



Zero Day Initiative @thezdi · 3月21日

Confirmed! The DEVCORE Team used a couple of bugs, including a somewhat risky TOCTAU race condition, to get their LPE on #Windows 11. They earn $30,000 and 3 Master of Pwn points. #Pwn2Own

SUCCESS

DEVCORE RESEARCH TEAM

TARGETTING

Microsoft Windows 11 in the Local Elevation of Privilege category

PRIZE $

$30,000

POINTS

3

DEVCORE

Is that the end of it ?

DEVCORE

CVE-2024-30090

KS Event

KS Event

- Event sets are groups of related events for which a listener can request notification.
- Client can register event for
  - Device State Change
  - Time interval
  - …

DEVCORE

KS Event

- Use IOCTL_KS_ENABLE_EVENT to register
    - EVENT_HANDLE
    - SEMAPHORE_HANDLE

```
typedef struct {
  ULONG NotificationType;
  union {
    struct {
      HANDLE     Event;
      ...
    } EventHandle;
    struct {
      HANDLE Semaphore;
      ...
    } SemaphoreHandle;
  }
  ...
} KSEVENTDATA, *PKSEVENTDATA;
```

DEVCORE

kstunk! ThunkEnableEventIrp

- Transfer 32-bit IOCTL_KS_ENABLE_EVENT requests to 64-bit requests

```
__int64 __fastcall CKSThunkDevice::DispatchIoctl(CKernelFilterDevice *a1, IRP *irp, unsigned int a3, NTSTATUS *a4)
{
  ...
  if ( IoIs32bitProcess(irp) && irp->RequestorMode )
  {
    ...
    if ( CurrentStackLocation->Parameters.DeviceIoControl.IoControlCode == IOCTL_KS_ENABLE_EVENT )
      return CKSAutomationThunk::ThunkEnableEventIrp(v12, a2, v11, a4);
    ...
  }
  else if ( CurrentStackLocation->Parameters.DeviceIoControl.IoControlCode == IOCTL_KS_PROPERTY )
  {
    //Pass down
    return CKSThunkDevice::CheckIrpForStackAdjustmentNative((__int64)a1, irp, v11, a4);
  }
}
```

ThunkEnableEventIrp

```
__int64 __fastcall CKSAutomationThunk::ThunkEnableEventIrp(__int64 ioctlcode_d, PIRP irp, __int64 a3, int *a4)
{
  ...
  if ( (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_ENABLE
    || (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_ONESHOT
    || (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_ENABLEBUFFERED )
  {
    // Convert 32-bit requests and pass down directly
  }
  else if ( (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_QUERYBUFFER )
  {
    ...
    newinputbuf = (KSEVENT *)ExAllocatePoolWithTag((POOL_TYPE)0x600, (unsigned int)(inputbuflen + 8), 'bqSK');
    ...
    memcpy(newinputbuf,Type3InputBuffer,0x28);
    ...
    v18 = KsSynchronousIoControlDevice(
            v25->FileObject,
            0,
            IOCTL_KS_ENABLE_EVENT,
            newinputbuf,
            inputbuflen + 8,
            OutBuffer,
            outbuflen,
            &BytesReturned);
    ...
  }
  ...
}
```

User input

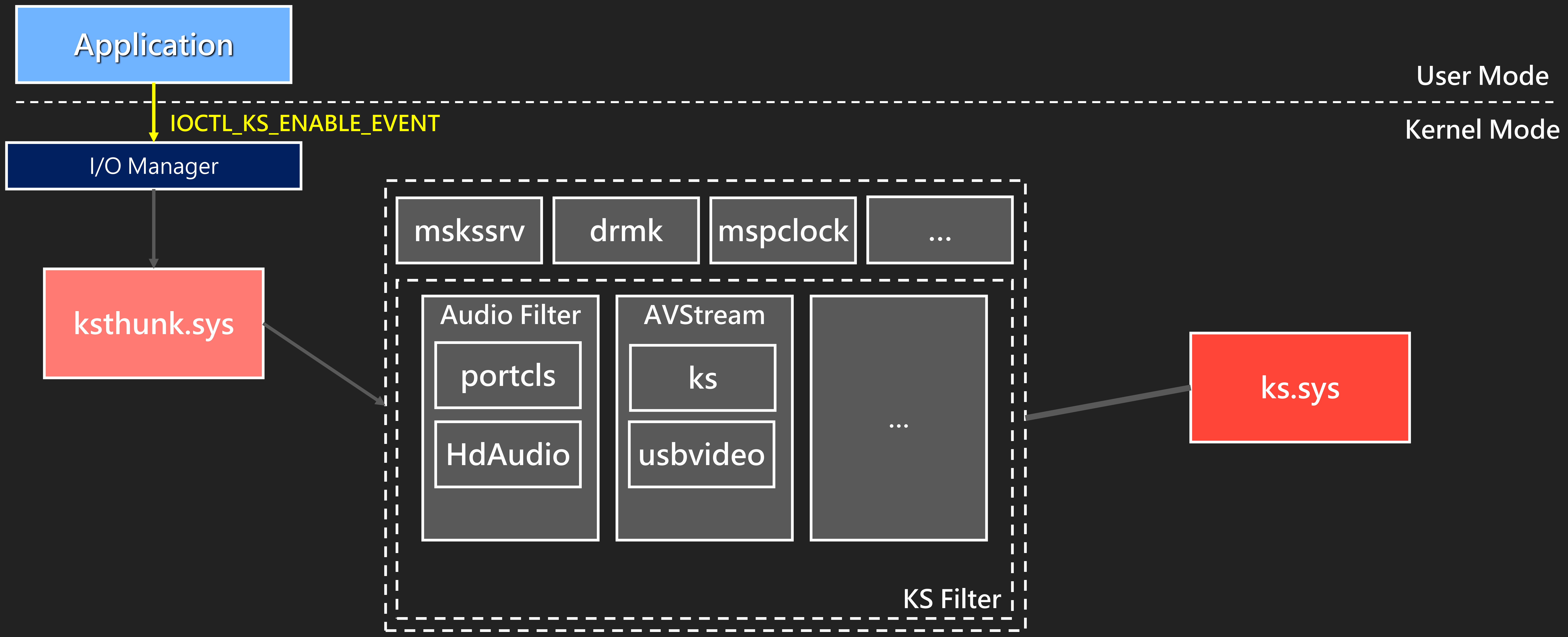
ThunkEnableEventIrp

```
__int64 __fastcall CKSAutomationThunk::ThunkEnableEventIrp(__int64 ioctlcode_d, PIRP irp, __int64 a3, int *a4)
{
  ...
  if ( (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_ENABLE
    || (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_ONESHOT
    || (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_ENABLEBUFFERED )
  {
    // Convert 32-bit requests and pass down directly
  }
  else if ( (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_QUERYBUFFER )
  {
    ...
    newinputbuf = (KSEVENT *)ExAllocatePoolWithTag((POOL_TYPE)0x600, (unsigned int)(inputbuflen + 8), 'bqSK');
    ...
    memcpy(newinputbuf,Type3InputBuffer,0x28);
    ...
    v18 = KsSynchronousIoControlDevice(
            v25->FileObject,
            0,
            IOCTL_KS_ENABLE_EVENT,
            newinputbuf,
            inputbuflen + 8,
            OutBuffer,
            outbuflen,
            &BytesReturned);
    ...
  }
  ...
}
```

ThunkEnableEventIrp

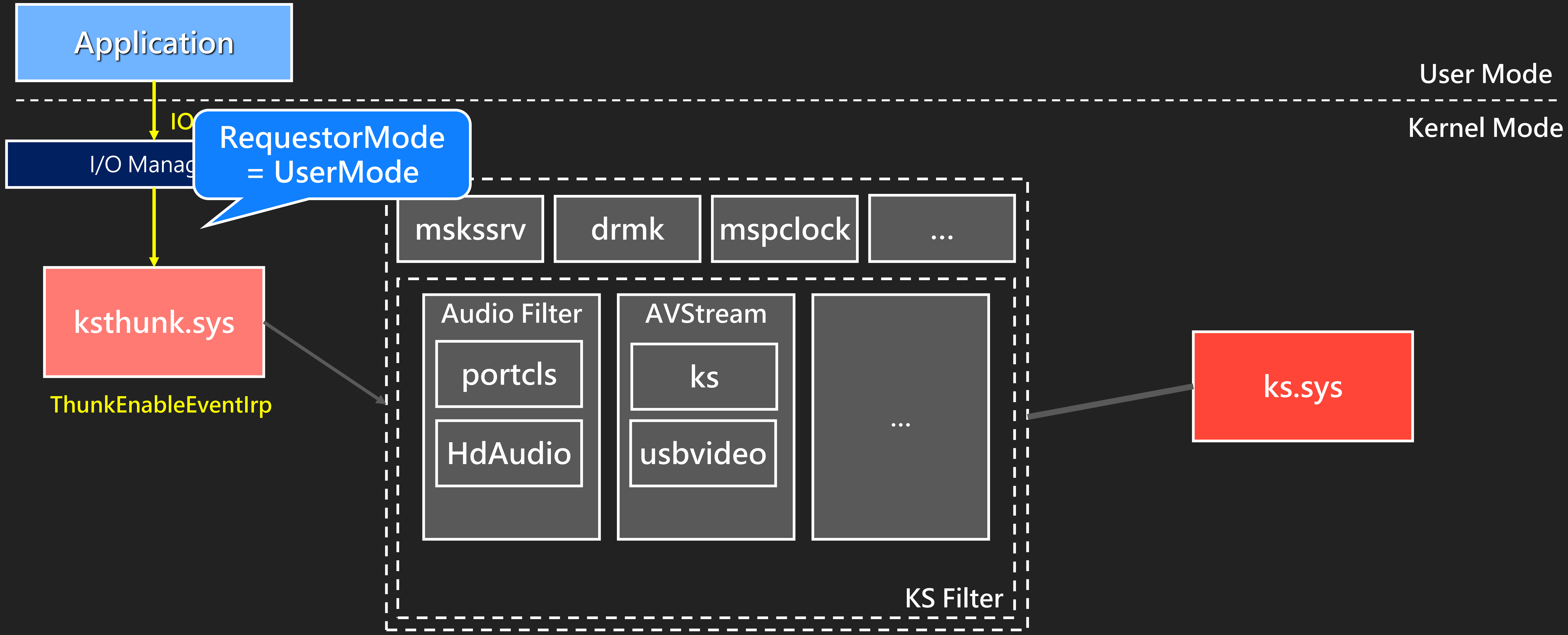
```
__int64 __fastcall CKSAutomationThunk::ThunkEnableEventIrp(__int64 ioctlcode_d, PIRP irp, __int64 a3, int *a4)
{
  ...
  if ( (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_ENABLE
    || (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_ONESHOT
    || (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_ENABLEBUFFERED )
  {
    // Convert 32-bit requests and pass down directly
  }
  else if ( (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_QUERYBUFFER )
  {
    ...
    newinputbuf = (KSEVENT *)ExAllocatePoolWithTag((POOL_TYPE)0x600, (unsigned int)(inputbuflen + 8), 'bqSK');
    ...
    memcpy(newinputbuf,Type3InputBuffer,0x28);
    ...
    v18 = KsSynchronousIoControlDevice(
            v25->FileObject,
            0,       KernelMode
            IOCTL_KS_ENABLE_EVENT,
            newinputbuf,
            inputbuflen + 8,
            OutBuffer,
            outbuflen,
            &BytesReturned);
    ...
  }
  ...
}
```

ThunkEnableEventIrp



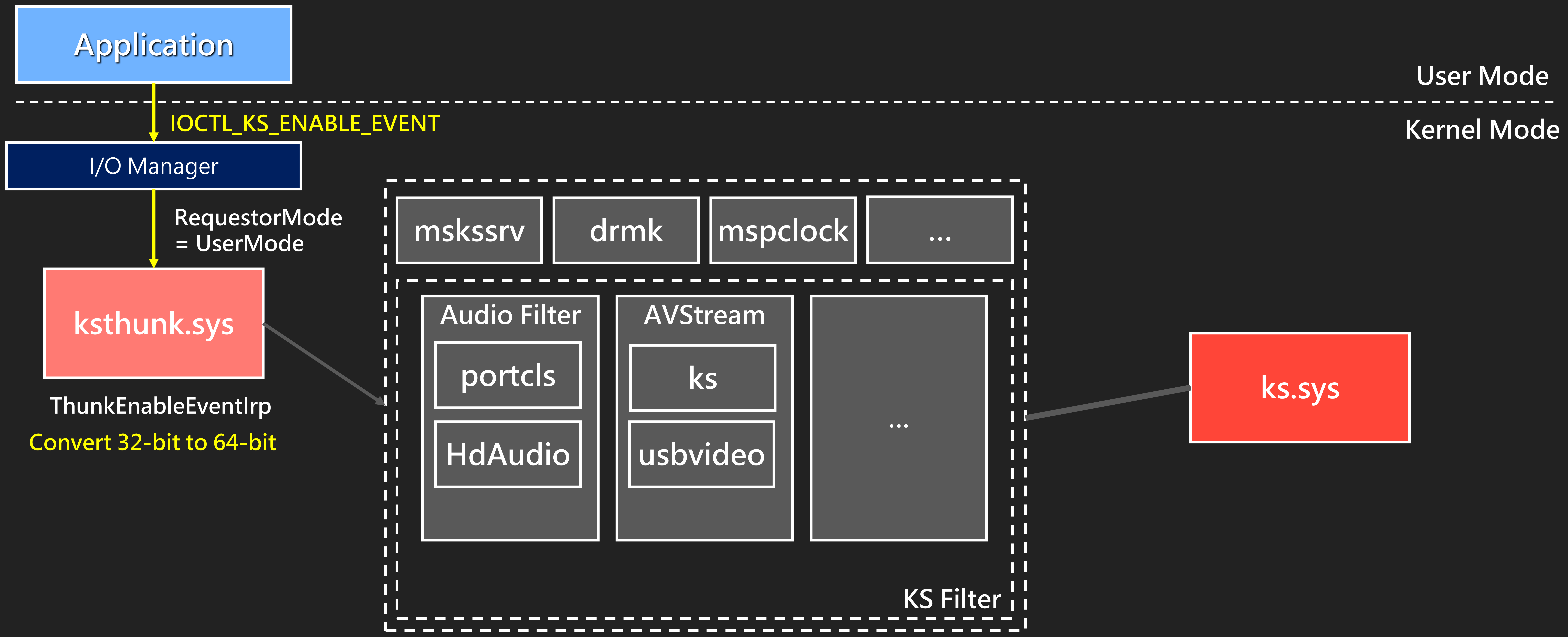
DEVCORE

ThunkEnableEventIrp



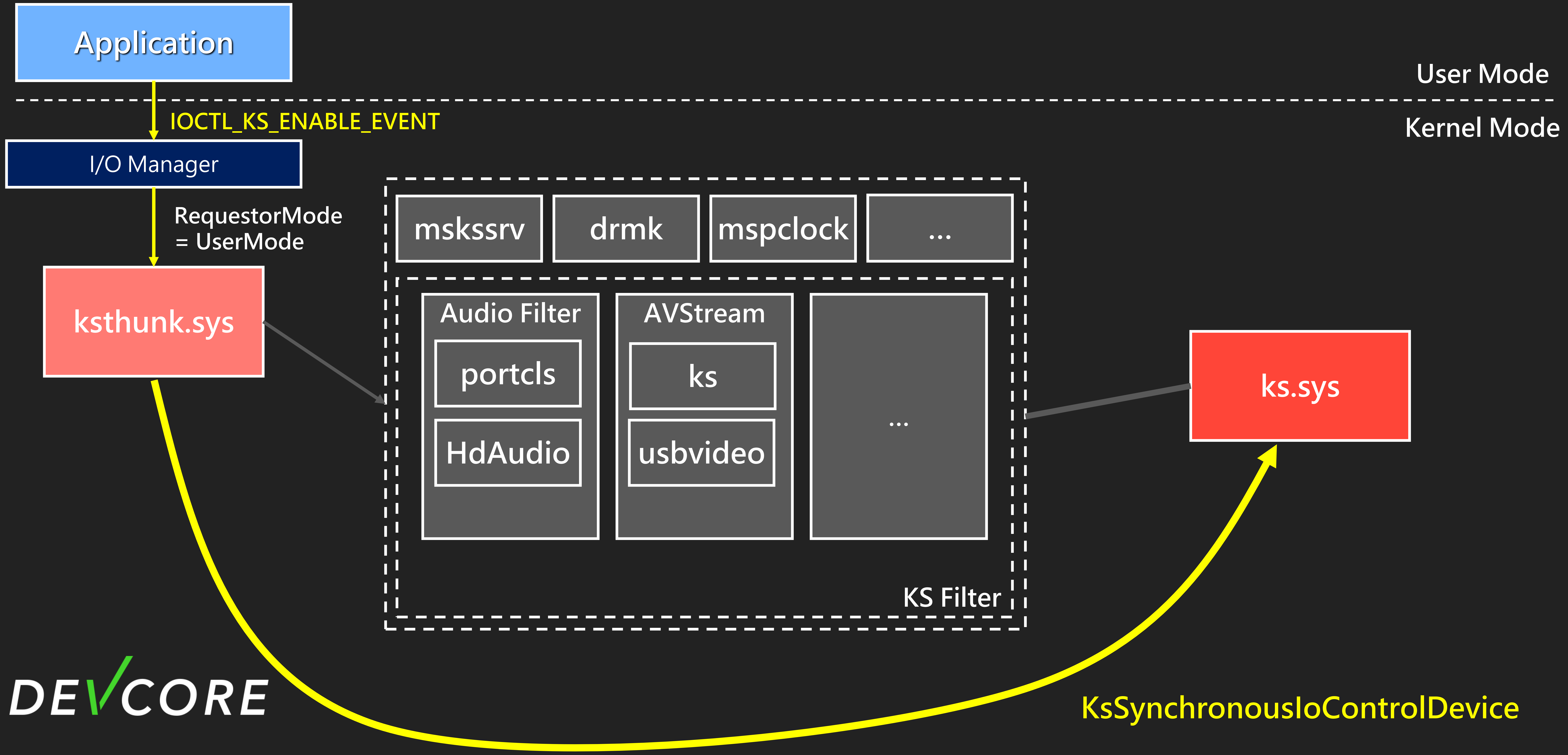
DEVCORE

ThunkEnableEventIrp



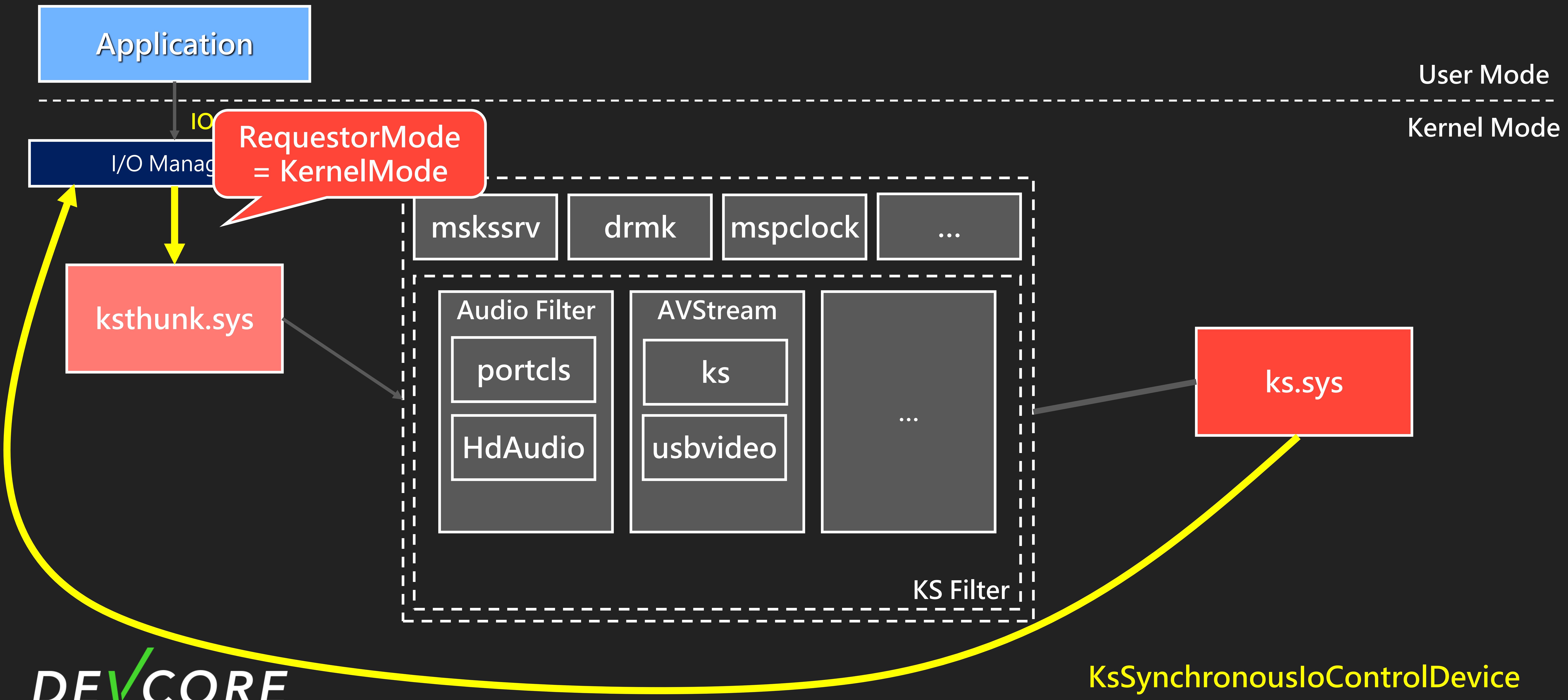
DEVCORE

ThunkEnableEventIrp



DEVCORE

ThunkEnableEventIrp



DEVCORE

We can do arbitrary IOCTL_KS_ENABLE_EVENT with KernelMode now

DEVCORE

We need to find a target to EoP

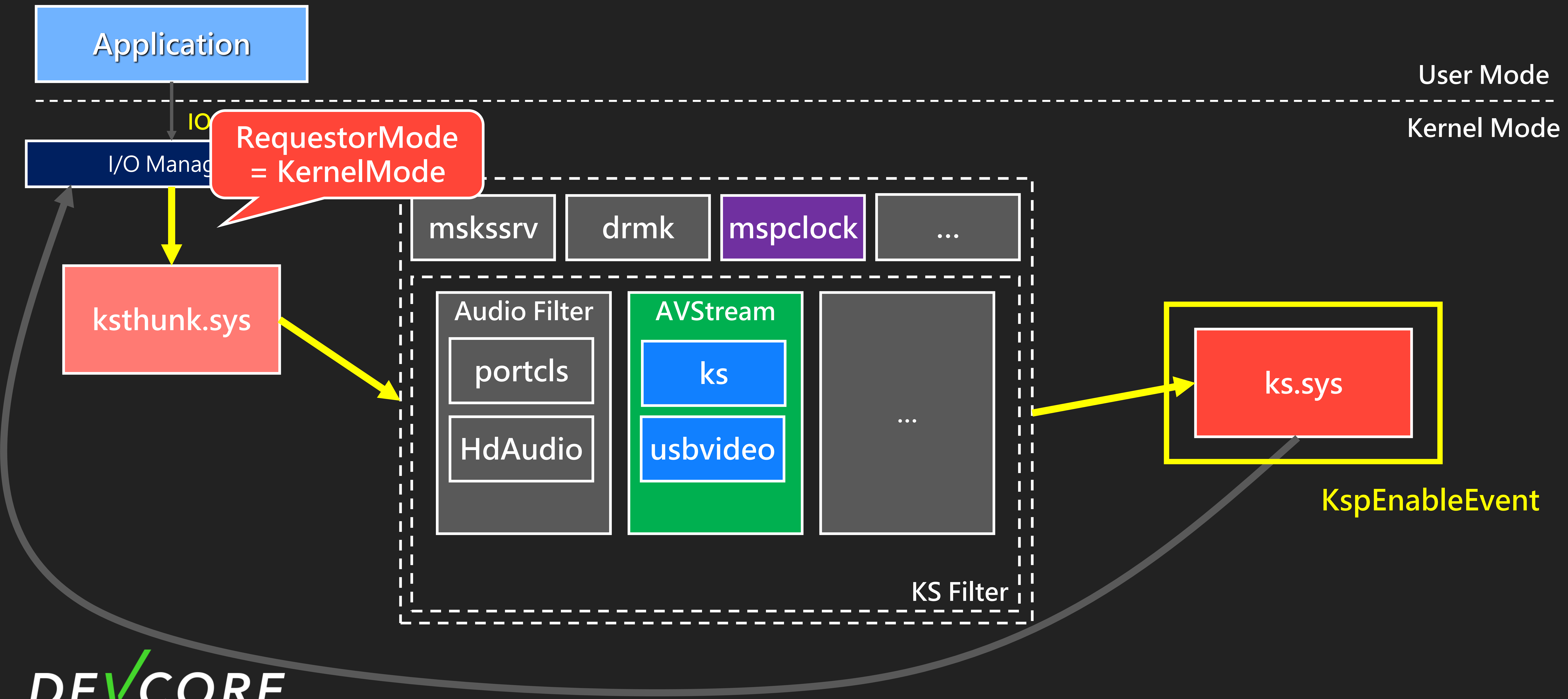
DEVCORE

But we didn't find a suitable target in ksthunk

We decide to pass it down to look for target

DEVCORE

ThunkEnableEventIrp



DEVCORE

We found some interesting ...

KspEnableEvent

```
__int64 __fastcall KspEnableEvent(
        ...)
{
    ...
    EventData = ExAllocatePoolWithTag(...);
    memcpy(EventData,Irp->UserBuffer,...);
    ...
    EventEntryEx->EventEntry.NotificationType = EventData->NotificationType;
    switch ( EventEntryEx_->EventEntry.NotificationType )
   {
      case KSEVENTF_EVENT_HANDLE:
          ...
          break;
      case KSEVENTF_EVENT_OBJECT:
      case DPC:
      case KSEVENTF_KSWORKITEM:
        if(Irp->RequestorMode)
          goto error;
      ...
   }
   Eventitem->AddEventHandler(Irp, EventData, PEventEntry);
}
```

KS Event

- The output buffer is a KSEVENTDATA structure used to specify a notification method.
  - Call from kernel driver
    - EVENT_OBJECT
    - DPC
    - KSWORKITEM
    - …

```
typedef struct {
  ULONG NotificationType;
    struct {
      PVOID     Event;
      ...
    } EventObject;
    struct {
      PKDPC     Dpc;
      ...
    } Dpc;
  ...
} KSEVENTDATA, *PKSEVENTDATA;
```

We can provide arbitrary kernel object to it !

DEVCORE

But ...

ThunkEnableEventIrp

```
__int64 __fastcall CKSAutomationThunk::ThunkEnableEventIrp(__int64 ioctlcode_d, PIRP irp, __int64 a3, int *a4)
{
  ...
  if ( (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_ENABLE
    || (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_ONESHOT
    || (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_ENABLEBUFFERED )
  {
    // Convert 32-bit requests and pass down directly
  }
  else if ( (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_QUERYBUFFER )
  {
    ...
    newinputbuf = (KSEVENT *)ExAllocatePoolWithTag((POOL_TYPE)0x600, (unsigned int)(inputbuflen + 8), 'bqSK');
    ...
    memcpy(newinputbuf,Type3InputBuffer,0x28);
    ...
    v18 = KsSynchronousIoControlDevice(
            v25->FileObject,
            0,
            IOCTL_KS_ENABLE_EVENT,
            newinputbuf,
            inputbuflen + 8,
            OutBuffer,
            outbuflen,
            &BytesReturned);
    ...
  }
  ...
}
```

ThunkEnableEventIrp

```
__int64 __fastcall CKSAutomationThunk::ThunkEnableEventIrp(__int64 ioctlcode_d, PIRP irp, __int64 a3, int *a4)
{
  ...
  if ( (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_ENABLE
    || (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_ONESHOT
    || (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_ENABLEBUFFERED )
  {
    // Convert 32-bit requests and pass down directly
  }
  else if ( (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_QUERYBUFFER )
  {
    ...
    newinputbuf = (KSEVENT *)ExAllocatePoolWithTag((POOL_TYPE)0x600, (unsigned int)(inputbuflen + 8), 'bqSK');
    ...
    memcpy(newinputbuf,Type3InputBuffer,0x28);
    ...
    v18 = KsSynchronousIoControlDevice(
            v25->FileObject,
            0,
            IOCTL_KS_ENABLE_EVENT,
            newinputbuf,
            inputbuflen + 8,
            OutBuffer,
            outbuflen,
            &BytesReturned);
    ...
  }
  ...
}
```

Fortunately,
there are double fetch everywhere.

DEVCORE

ThunkEnableEventIrp

```
  else if ( (v25->Parameters.DeviceIoControl.Type3InputBuffer->Flags & 0xEFFFFFFF) == KSEVENT_TYPE_QUERYBUFFER )
  {
    ...
    newinputbuf = (KSEVENT *)ExAllocatePoolWithTag((POOL_TYPE)0x600, (unsigned int)(inputbuflen + 8), 'bqSK');
    ...
    memcpy(newinputbuf,Type3InputBuffer,0x28);
    ...
    v18 = KsSynchronousIoControlDevice(
            v25->FileObject,
            0,
            IOCTL_KS_ENABLE_EVENT,
            newinputbuf,
            inputbuflen + 8,
            OutBuffer,
            outbuflen,
            &BytesReturned);
    ...
  }
  ...
}
```

Race window

If we trigger the event, it would call KsGenerateEvent

KsGenerateEvent

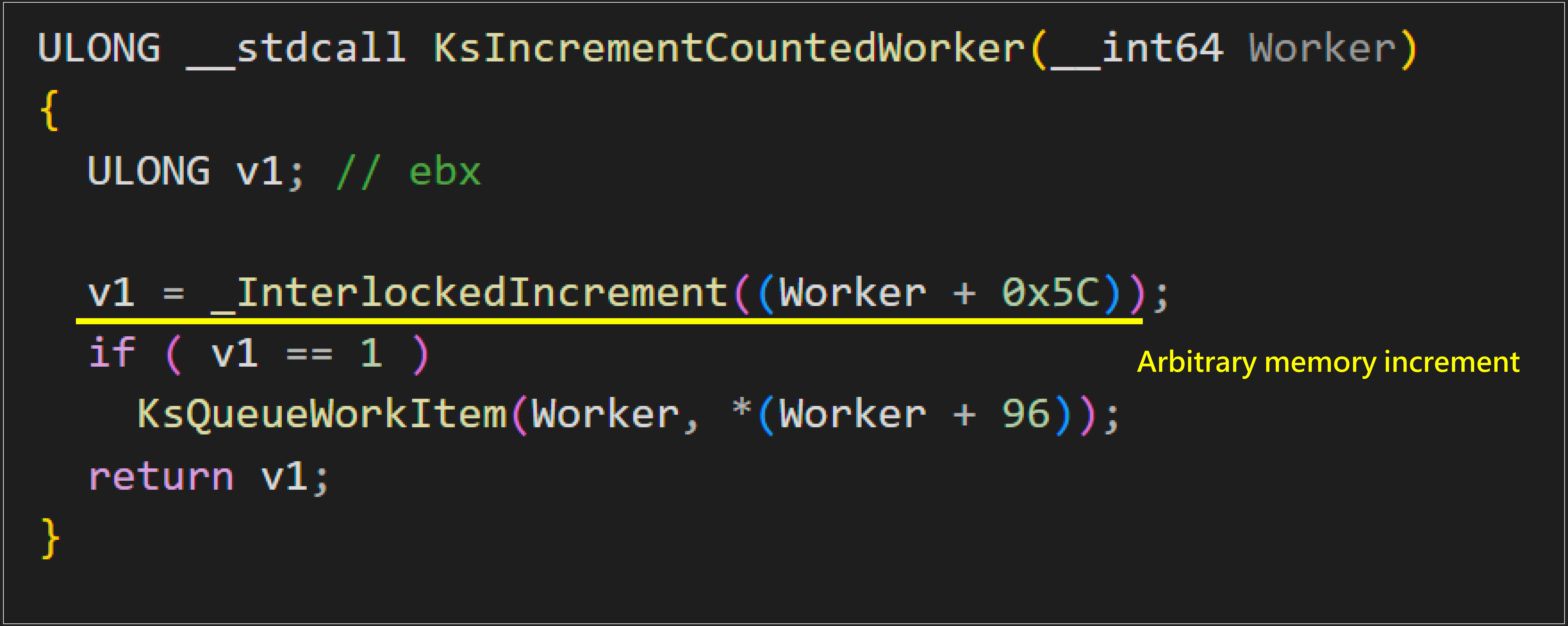
```
NTSTATUS __stdcall KsGenerateEvent(PKSEVENT_ENTRY EventEntry)
{

  switch ( EventEntry->NotificationType )
  {
    case KSEVENTF_DPC:
      ...
      if ( !KeInsertQueueDpc(EventEntry->EventData->Dpc.Dpc, EventEntry->EventData, 0LL) )
        _InterlockedAdd(&EventEntry->EventData->EventObject.Increment, 0xFFFFFFFF);
      ...
    case KSEVENTF_KSWORKITEM:
      ...
      KsIncrementCountedWorker(eventdata->KsWorkItem.KsWorkerObject);

  }
}
```

Arbitrary register DPC

KsGenerateEvent

```
NTSTATUS __stdcall KsGenerateEvent(PKSEVENT_ENTRY EventEntry)
{

  switch ( EventEntry->NotificationType )
  {
    case KSEVENTF_DPC:
      ...
      if ( !KeInsertQueueDpc(EventEntry->EventData->Dpc.Dpc, EventEntry->EventData, 0LL) )
        _InterlockedAdd(&EventEntry->EventData->EventObject.Increment, 0xFFFFFFFF);
      ...
    case KSEVENTF_KSWORKITEM:
      ...
      KsIncrementCountedWorker(eventdata->KsWorkItem.KsWorkerObject);

  }
}
```

KsIncrementCountedWorker



```
ULONG __stdcall KsIncrementCountedWorker(__int64 Worker)
{
  ULONG v1; // ebx

  v1 = _InterlockedIncrement((Worker + 0x5C));
  if ( v1 == 1 )
    KsQueueWorkItem(Worker, *(Worker + 96));
  return v1;
}
```

Arbitrary memory increment

We have arbitrary increment primitive now

Arbitrary increment primitive to EoP

- There are many well-known method
  - Abuse token privilege
  - IoRing
  - …

DEVCORE

It seems trivial, but ...

DEVCORE

Arbitrary increment primitive to EoP

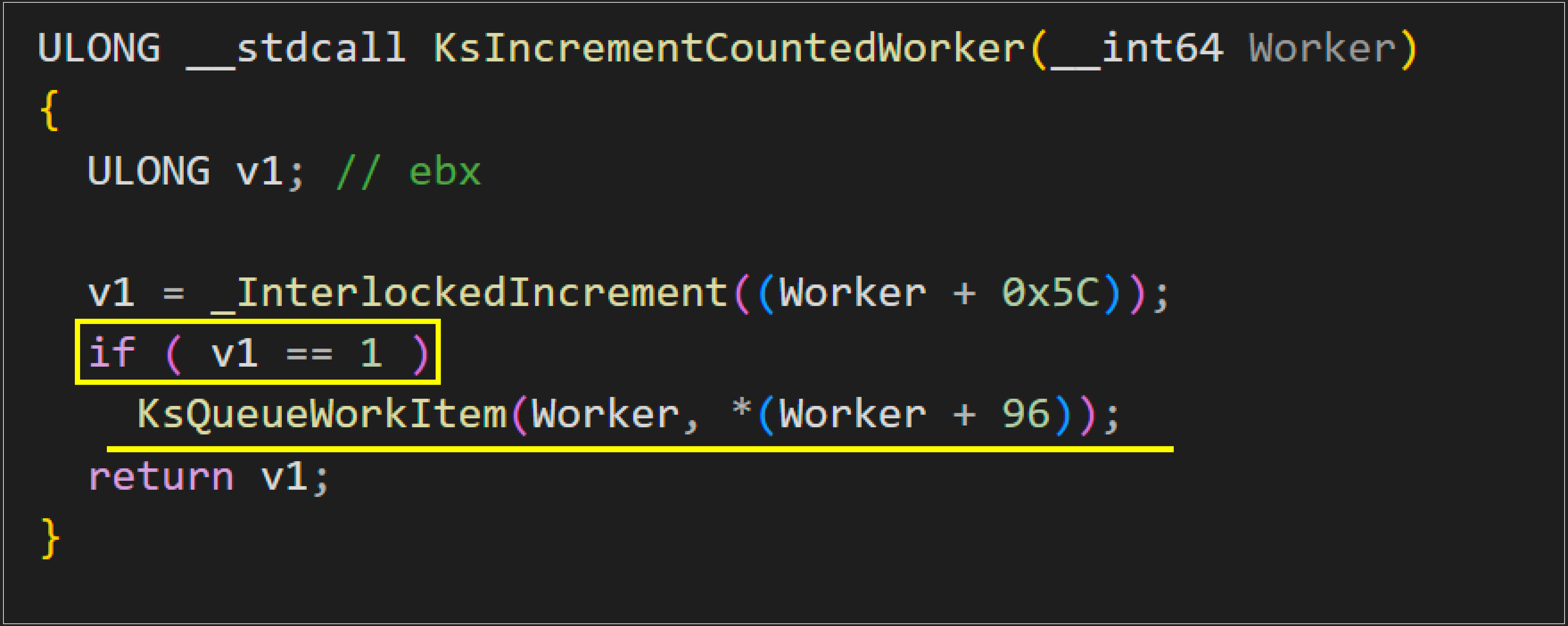
- Abuse token privilege
    - Need to overwrite Privileges.Enable and Privileges.Present
        - Need to trigger the bug multiple times
        - It may take a long time

DEVCORE

Arbitrary increment primitive to EoP

- IoRing
  - Need to overwrite IoRing->RegBuffersCount and IoRing->RegBuffers
    - Good Candidate
    - Only need to trigger the bug twice

KsIncrementCountedWorker



```
ULONG __stdcall KsIncrementCountedWorker(__int64 Worker)
{
  ULONG v1; // ebx

  v1 = _InterlockedIncrement((Worker + 0x5C));
  if ( v1 == 1 )
    KsQueueWorkItem(Worker, *(Worker + 96));
  return v1;
}
```

Let's find a new way !

DEVCORE

Arbitrary increment primitive to EoP

- Abuse token privilege
  - The goal is to obtain SeDebugPrivilege
    - Open process of winlogon.exe

DEVCORE

Why does having SeDebugPrivilege allow you to open high-privilege process?

DEVCORE

PsOpenProcess

```
if ( SeSinglePrivilegeCheck(SeDebugPrivilege, AccessMode_) )
{
  if ( (AccessState.RemainingDesiredAccess & MAXIMUM_ALLOWED) != 0 )
    AccessState.PreviouslyGrantedAccess |= PROCESS_ALL_ACCESS;
  else
    AccessState.PreviouslyGrantedAccess |= AccessState.RemainingDesiredAccess;
  AccessState.RemainingDesiredAccess = 0;
}
v20 = ObOpenObjectByPointer(
        Process,
        HandleAttributes,
        &AccessState,
        0,
        (POBJECT_TYPE)PsProcessType,
        AccessMode,
        &Handle);
```

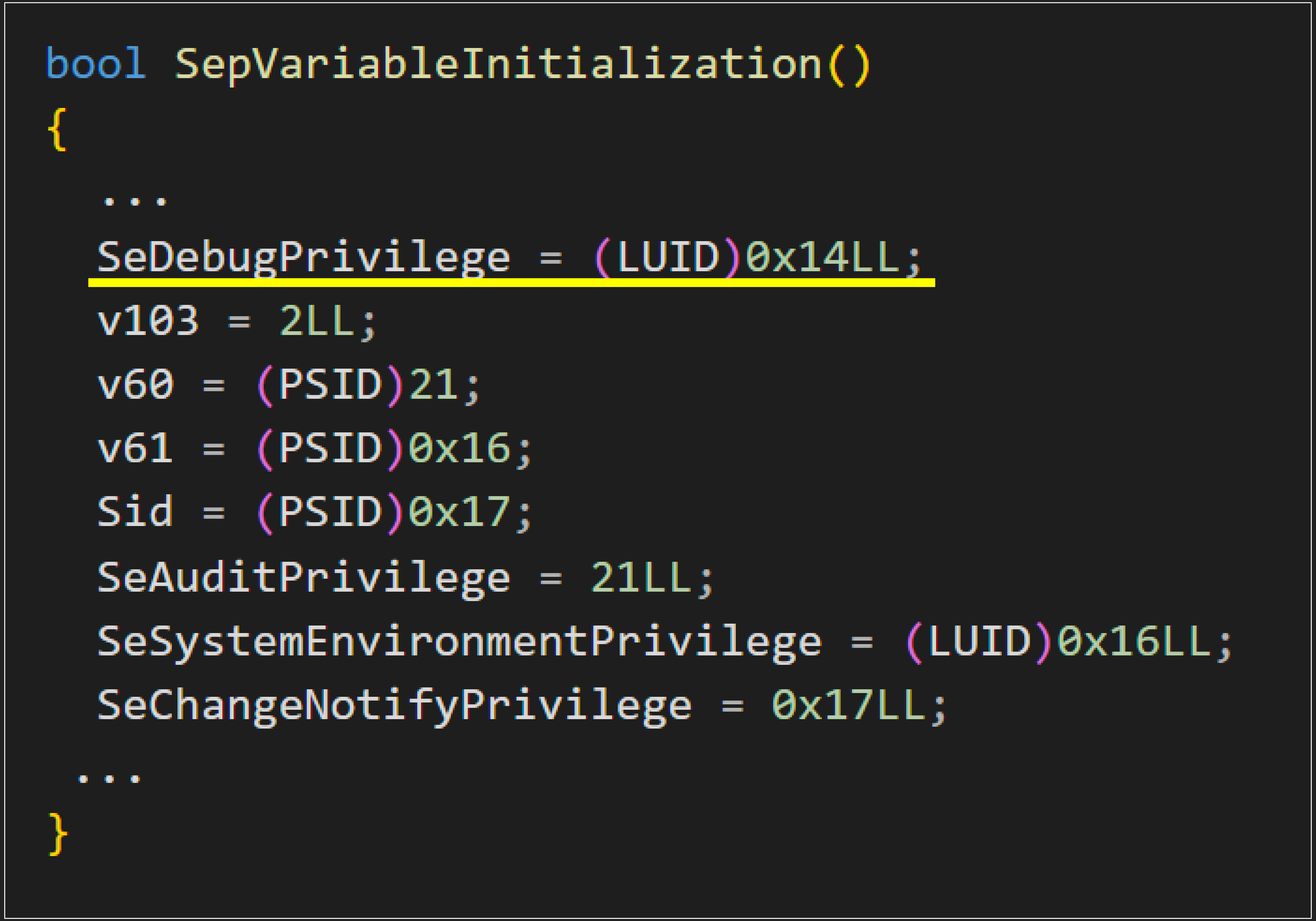
DEVCORE

PsOpenProcess

```
if ( SeSinglePrivilegeCheck(SeDebugPrivilege, AccessMode_) )
{
  if ( (AccessState.RemainingDesiredAccess & MAXIMUM_ALLOWED) != 0 )
    AccessState.PreviouslyGrantedAccess |= PROCESS_ALL_ACCESS;
  else
    AccessState.PreviouslyGrantedAccess |= AccessState.RemainingDesiredAccess;
  AccessState.RemainingDesiredAccess = 0;
}
v20 = ObOpenObjectByPointer(
        Process,
        HandleAttributes,
        &AccessState,
        0,
        (POBJECT_TYPE)PsProcessType,
        AccessMode,
        &Handle);
```

DEVCORE

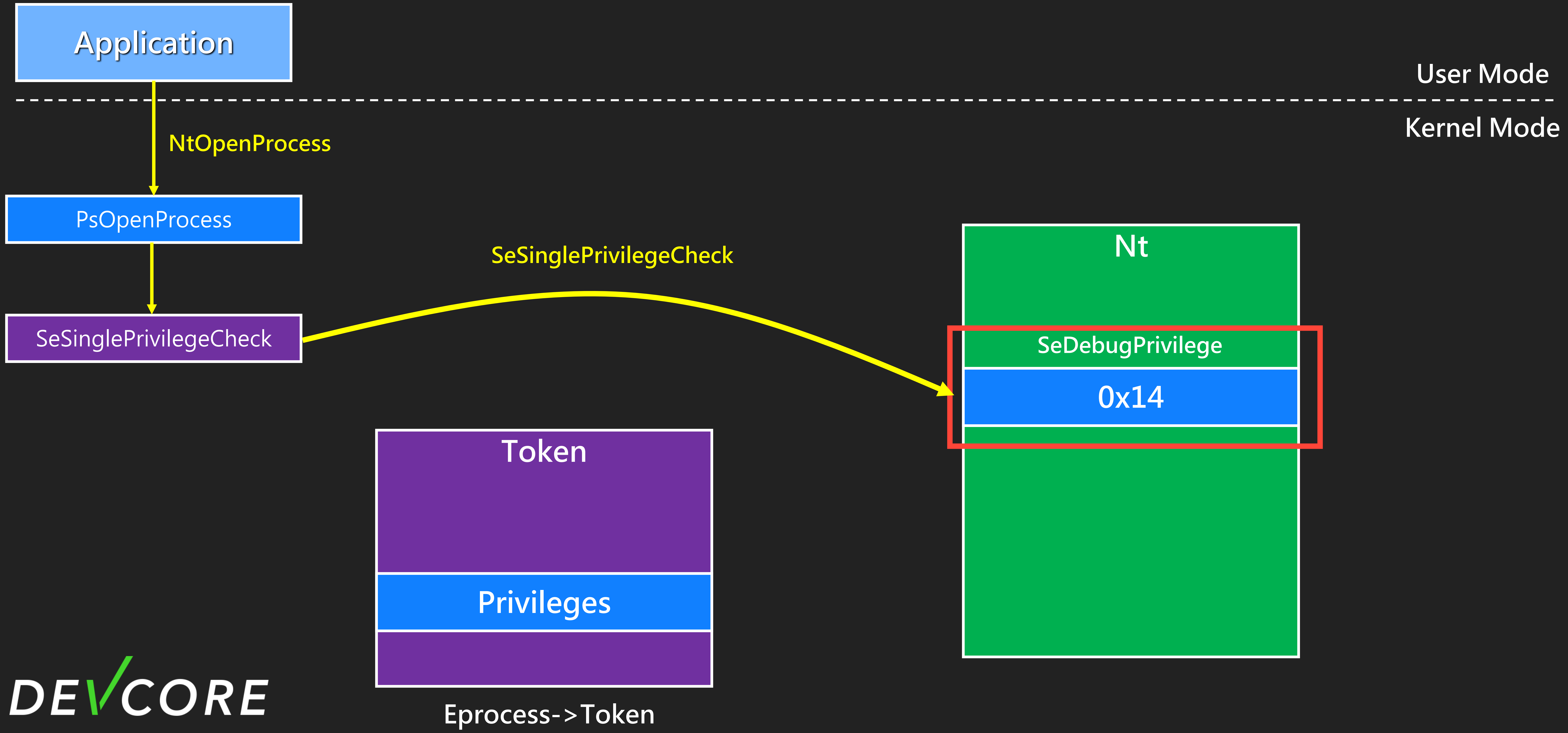
PsOpenProcess



```
bool SepVariableInitialization()
{
  ...
  SeDebugPrivilege = (LUID)0x14LL;
  v103 = 2LL;
  v60 = (PSID)21;
  v61 = (PSID)0x16;
  Sid = (PSID)0x17;
  SeAuditPrivilege = 21LL;
  SeSystemEnvironmentPrivilege = (LUID)0x16LL;
  SeChangeNotifyPrivilege = 0x17LL;
 ...
}
```

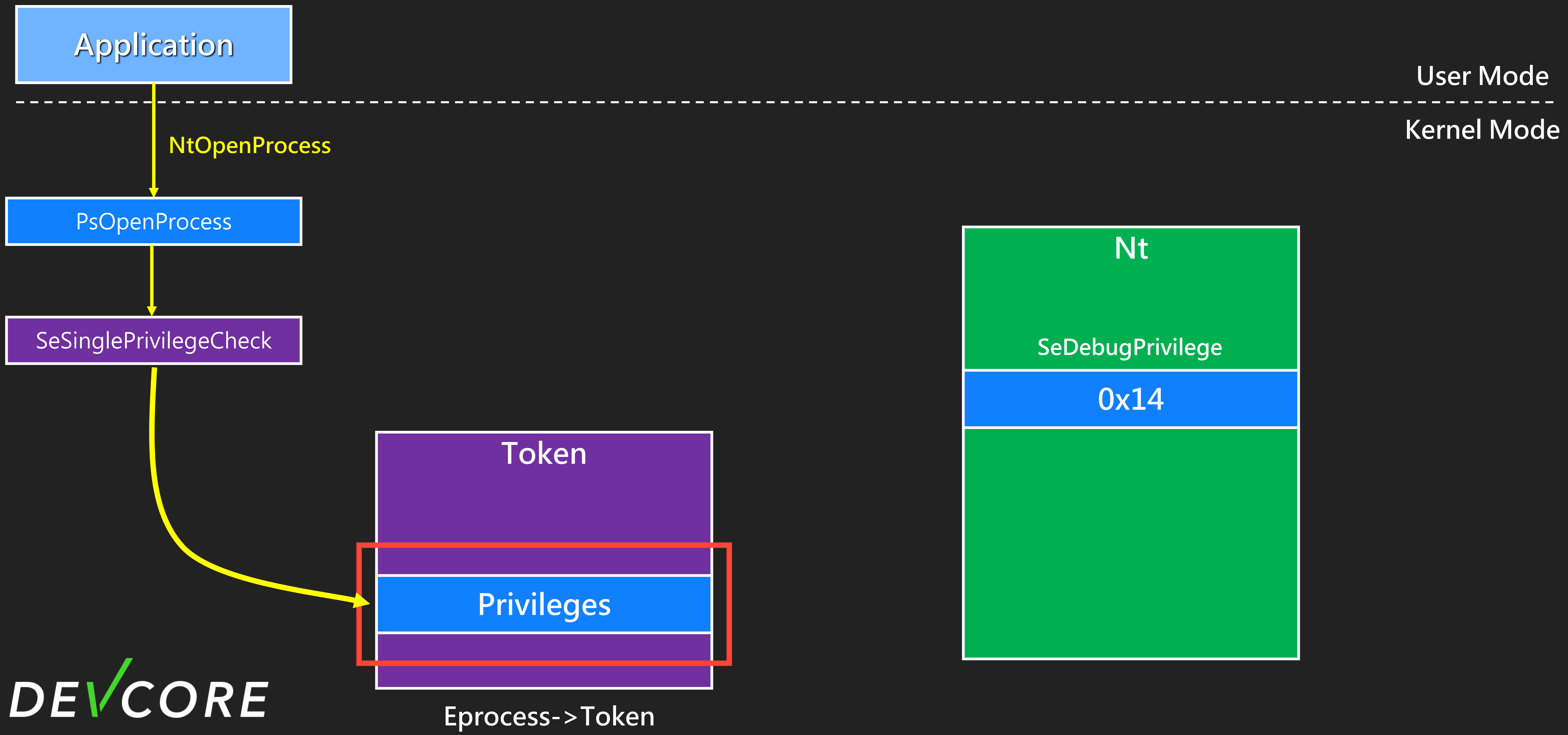
DEVCORE

Make abusing token privilege great again



DEVCORE

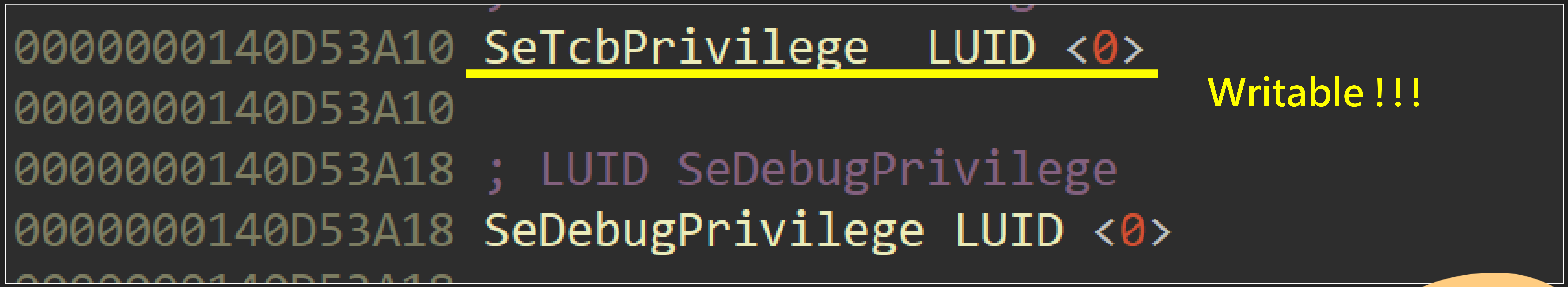
Make abusing token privilege great again



One more interesting …

DEVCORE

nt! SeDebugPrivilege



```
0000000140D53A10 SeTcbPrivilege  LUID <0>
0000000140D53A10
0000000140D53A18 ; LUID SeDebugPrivilege
0000000140D53A18 SeDebugPrivilege LUID <0>
```

Writable !!!

DEVCORE

Make abusing token privilege great again !

DEVCORE

Make abusing token privilege great again

```
C:\Users\angelboy>whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                          State
============================= ==================================== ========
SeShutdownPrivilege           Shut down the system                 Disabled
SeChangeNotifyPrivilege       Bypass traverse checking             Enabled
SeUndockPrivilege             Remove computer from docking station Disabled
SeIncreaseWorkingSetPrivilege Increase a process working set       Disabled
SeTimeZonePrivilege           Change the time zone                 Disabled
```

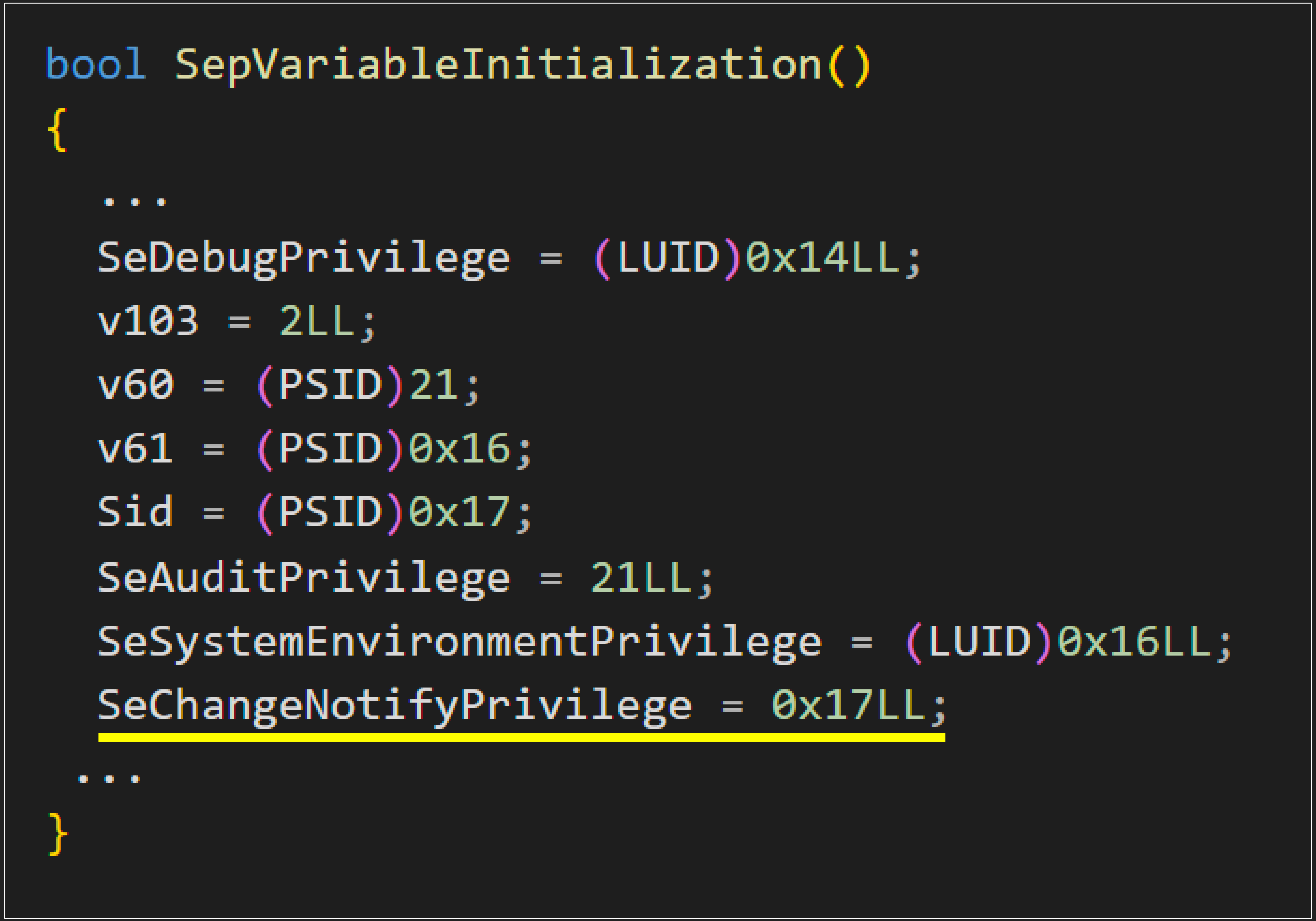
DEVCORE

Make abusing token privilege great again

```
C:\Users\angelboy>whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                          State
============================= ==================================== ========
SeShutdownPrivilege           Shut down the system                 Disabled
SeChangeNotifyPrivilege       Bypass traverse checking             Enabled
SeUndockPrivilege             Remove computer from docking station Disabled
SeIncreaseWorkingSetPrivilege Increase a process working set       Disabled
SeTimeZonePrivilege           Change the time zone                 Disabled
```

DEVCORE

nt! SeChangeNotifyPrivilege



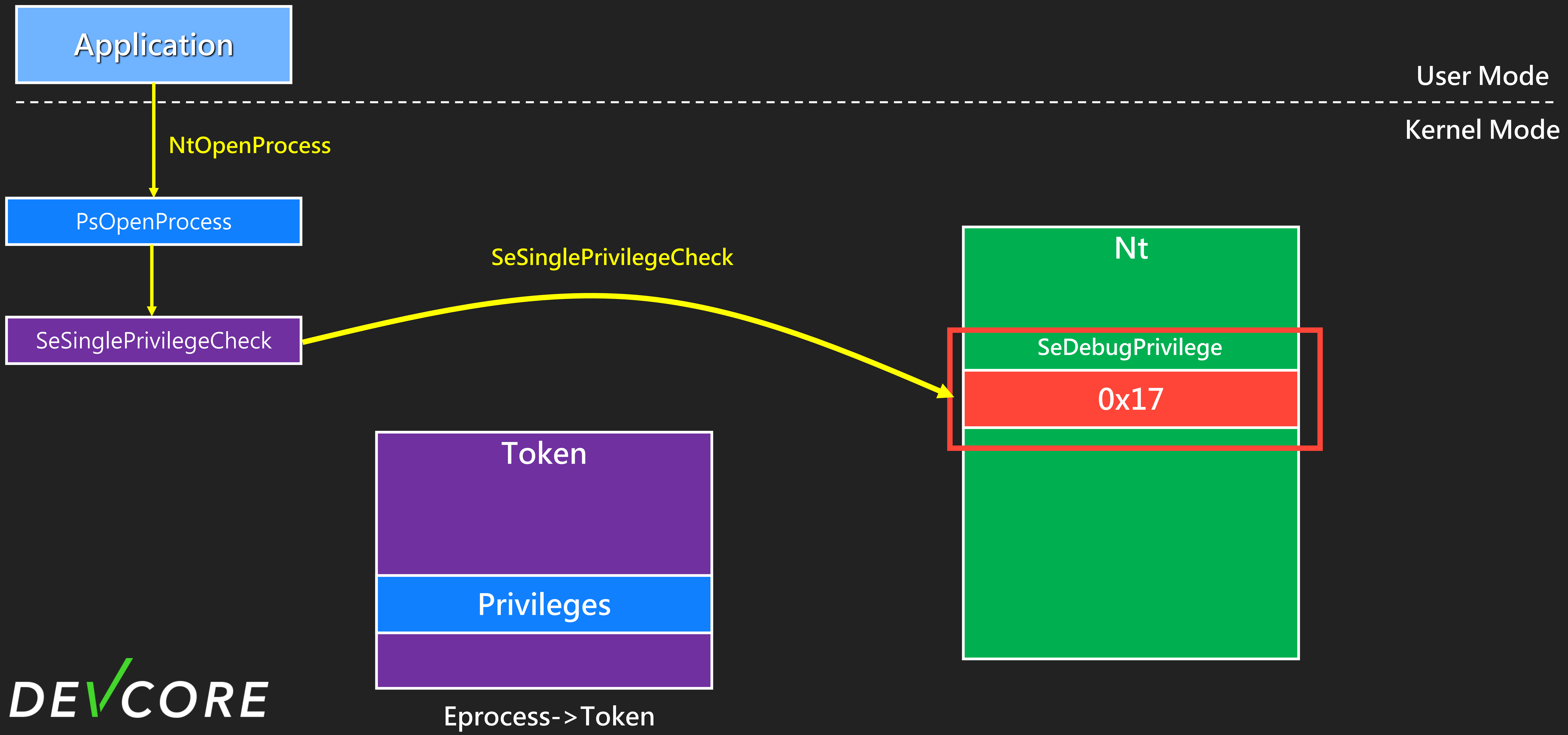
```
bool SepVariableInitialization()
{
  ...
  SeDebugPrivilege = (LUID)0x14LL;
  v103 = 2LL;
  v60 = (PSID)21;
  v61 = (PSID)0x16;
  Sid = (PSID)0x17;
  SeAuditPrivilege = 21LL;
  SeSystemEnvironmentPrivilege = (LUID)0x16LL;
  SeChangeNotifyPrivilege = 0x17LL;
 ...
}
```

DEVCORE

How about changing the value of
nt! SeDebugPrivilege from 0x14 to 0x17 ?

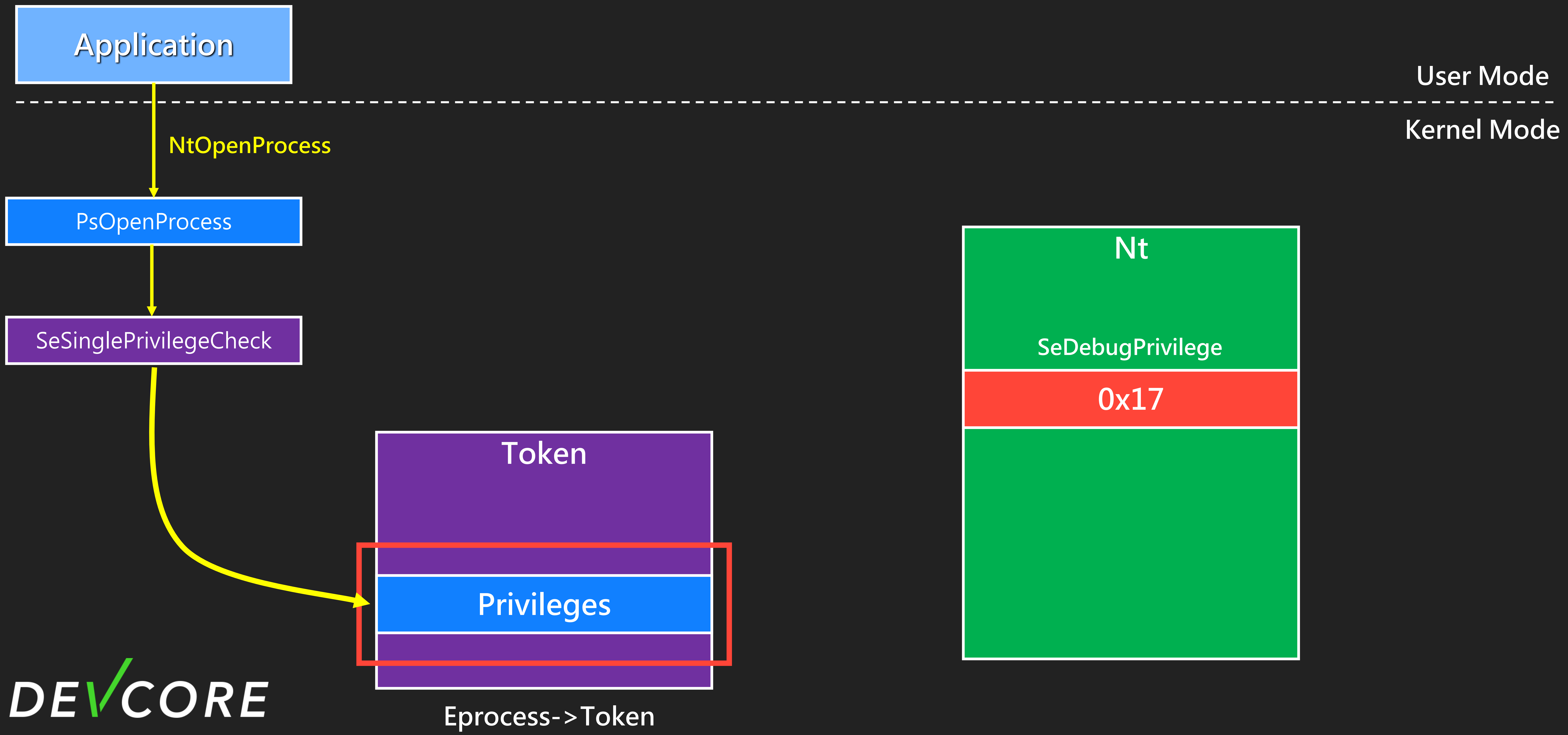
DEVCORE

Make abusing token privilege great again



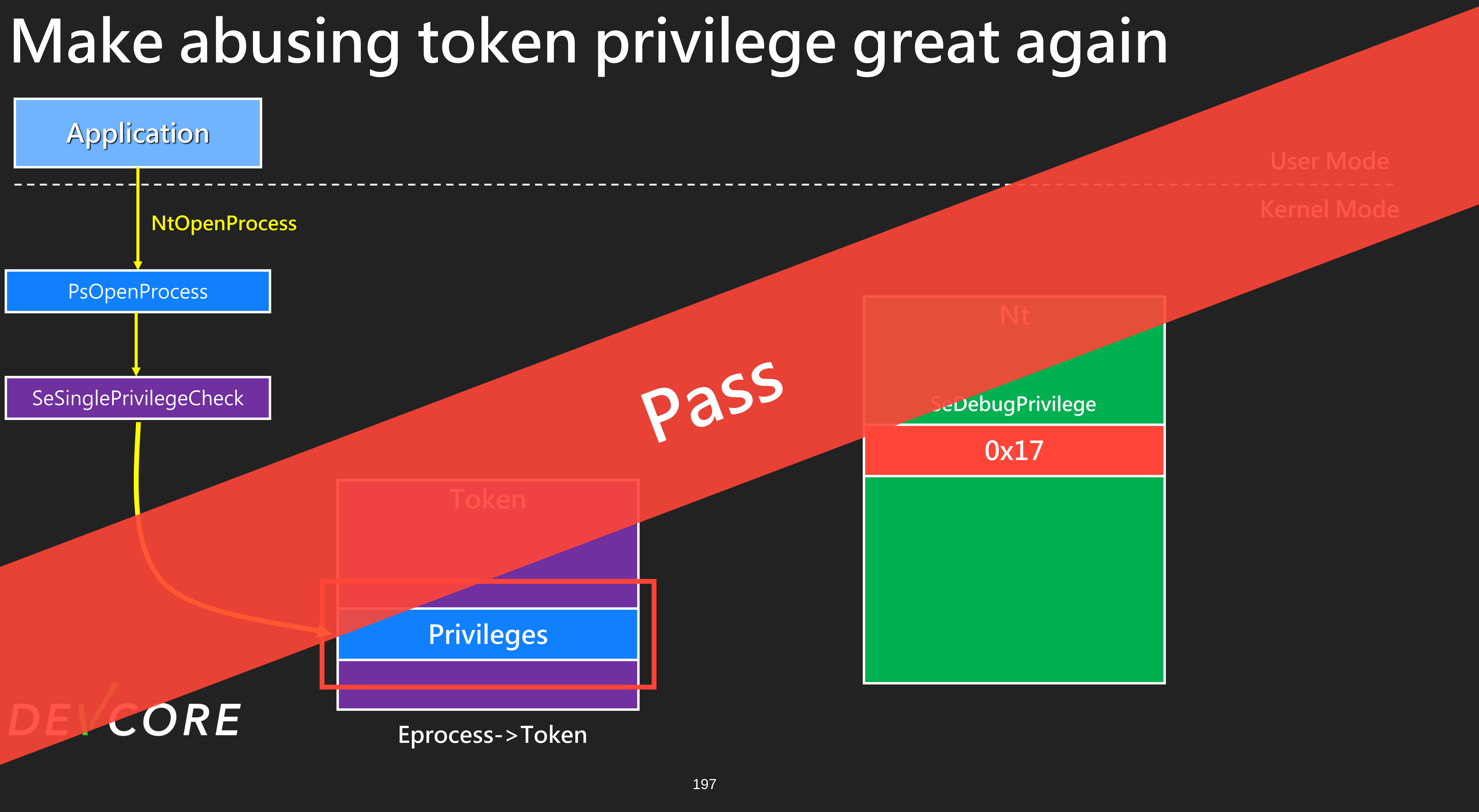
DEVCORE

Make abusing token privilege great again



DEVCORE

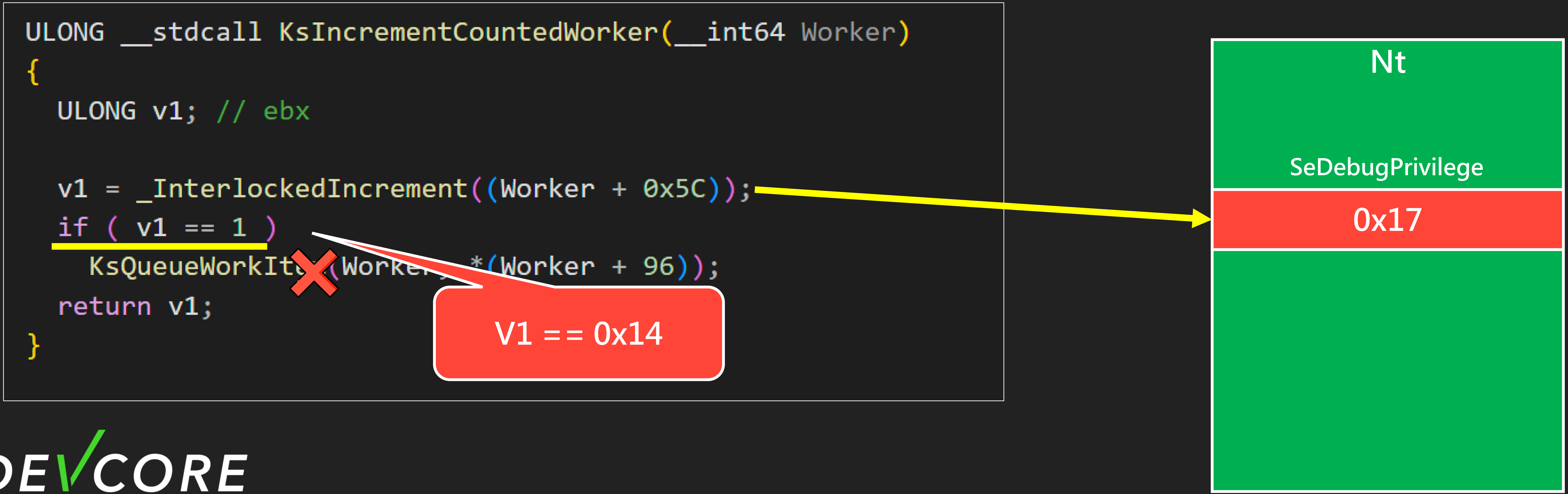
Make abusing token privilege great again



DEVCORE

Make abusing token privilege great again

- We can use arbitrary increment primitive to
  - Increase nt!SeDebugPrivilege to 0x17

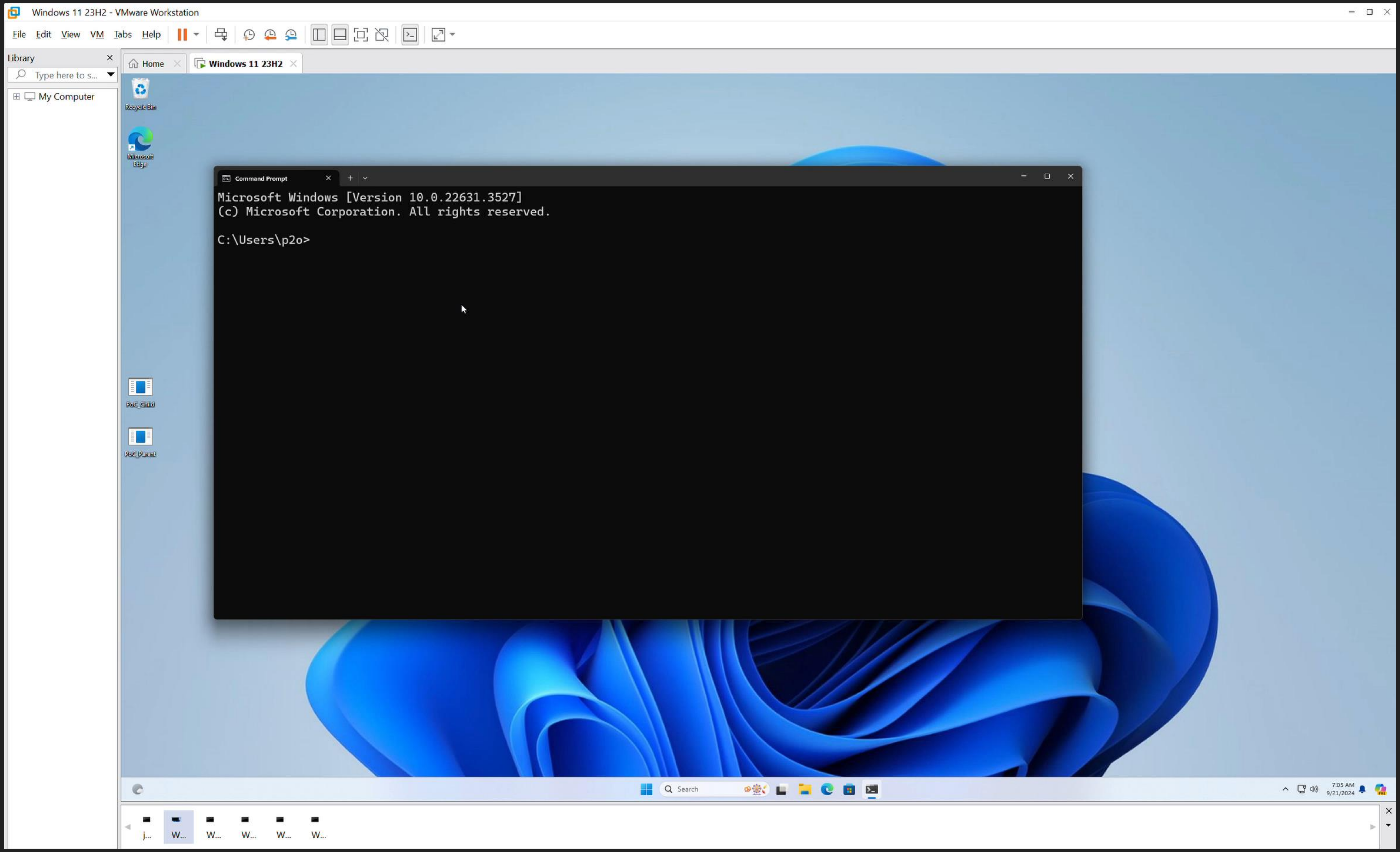


```
ULONG __stdcall KsIncrementCountedWorker(__int64 Worker)
{
  ULONG v1; // ebx

  v1 = _InterlockedIncrement((Worker + 0x5C));
  if ( v1 == 1 )
    KsQueueWorkIt (Worke   *(Worker + 96));
  return v1;
}
```

DEVCORE

Make abusing token privilege great again

- Not only nt!SeDebugPrivilege, but …
  - SeTcbPrivilege = 0x7
  - SeTakeOwnershipPrivilege = 0x9
  - SeLoadDriverPrivilege = 0xa
  - …



```
Microsoft Windows [Version 10.0.22631.3527]
(c) Microsoft Corporation. All rights reserved.

C:\Users\p2o>
```

DEVCORE

Proxying to Kernel again !

DEVCORE

Exploitability

The following table provides an exploitability assessment for this vulnerability at the time of original publication.

Publicly disclosed	No
Exploited	No
Exploitability assessment	Exploitation Less Likely

The Next

The Next

- The Overlook bug class
  - It may be possible to find more related proxy type bug
    - IoBuildDeviceIoControlRequest
    - IofCallDriver
    - ...
  - The timing of setting Irp->RequestorMode to KernelMode is very important.

DEVCORE

The Next

- The Attack Surface
  - kernel streaming has many components
    - Low-hanging fruit
      - Hdaudio.sys
      - Usbvideo.sys
      - …

DEVCORE

Takeaways

- Looking at historical vulnerabilities is indispensable
- When current exploitation methods no longer work, explore the core mechanics - you may discover new approaches.

DEVCORE

Is that the end of it ?

DEVCORE

CVE-2024-38125

CVE-2024-38055

CVE-2024-38056

CVE-2024-38054

CVE-2024-38144

CVE-2024-38191

CVE-2024-38052

CVE-2024-35250

CVE-2024-30084

CVE-2024-38057

CVE-2024-30090

To Be Continued ...

DEVCORE

DEVCORE

Thanks!

scwuaptx

angelboy@devco.re