DEVCORE
SECURITY
CONSULTING

My journey on SMBGhost

Angelboy

angelboy@chroot.org

@scwuaptx

Whoami

- Angelboy
  - Researcher at DEVCORE
  - CTF Player
    - HITCON / 217
  - Chroot
  - Co-founder of pwnable.tw
  - Speaker
    - HITB GSEC 2018/AVTokyo 2018/VXCON

PWNABLE.TW

Outline

- Introduction
- Vulnerability - CVE-2020-0796
- Exploitation of SMBGhost
  - From crash to arbitrary memory writing
  - How can we get code execution from arbitrary memory writing in the past
  - Method 1 - System root hijack (need some condition)
  - Method 2 - Abusing MDL

Outline

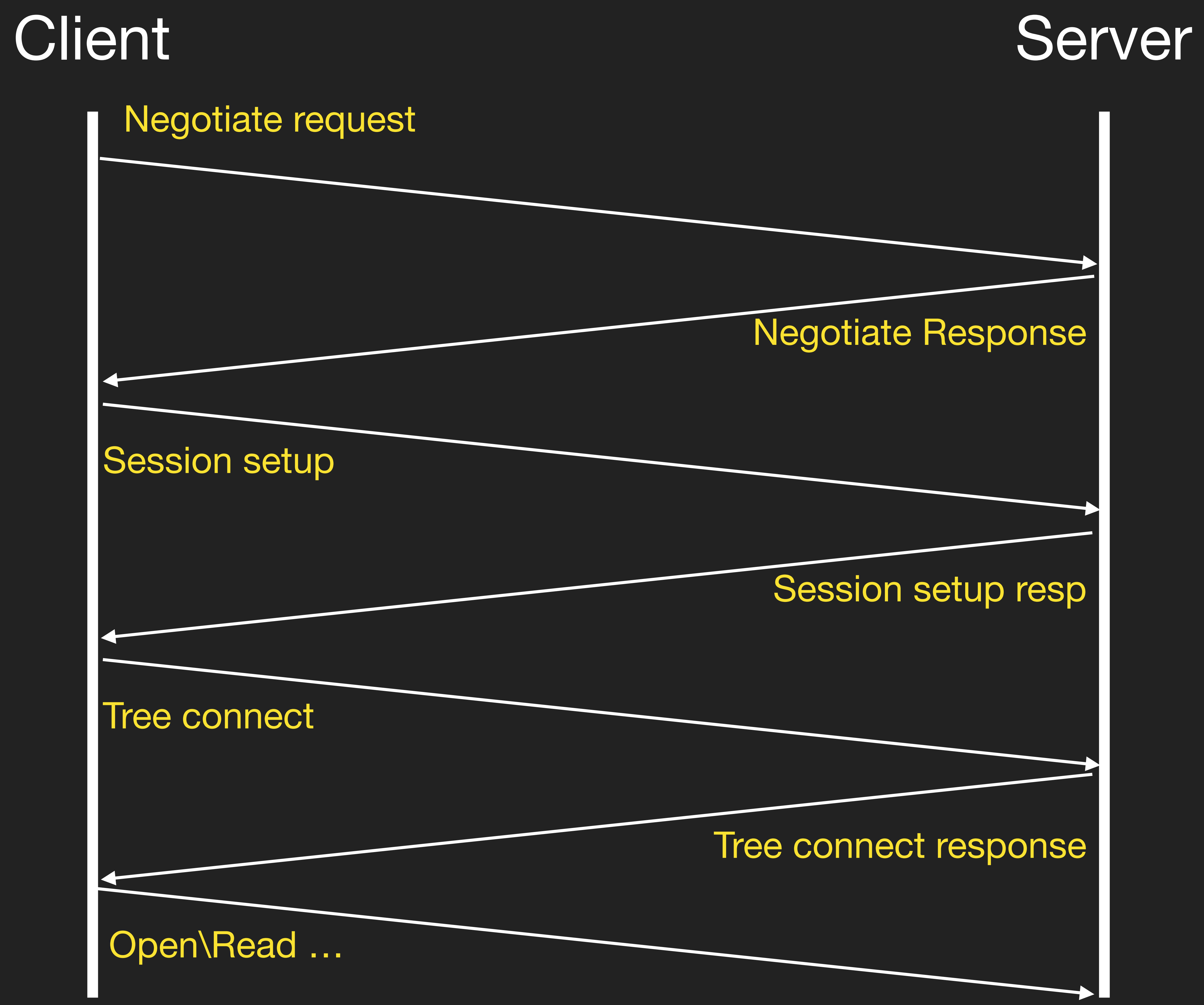
- Introduction
- Vulnerability - CVE-2020-0796
- Exploitation of SMBGhost
  - From crash to arbitrary memory writing
  - How can we get code execution from arbitrary memory writing in the past
  - Method 1 - System root hijack (need some condition)
  - Method 2 - Abusing MDL

Introduction

- Server Message Block (SMB) 是 Windows 中常見共享檔案的協定，基本上只要安裝完 Windows 就會在 445 port 開啟這樣的協定，在企業中更是常見
- MS17-010
  - EternalBlue
  - WannaCry
- CVE-2020-0796
  - SMBGhost

Introduction

- Server Message Block (SMB)



DEVCORE SECURITY CONSULTING

Outline

- Introduction
- Vulnerability - CVE-2020-0796
- Exploitation of SMBGhost
  - From crash to arbitrary memory writing
  - How can we get code execution from arbitrary memory writing in the past
  - Method 1 - System root hijack (need some condition)
  - Method 2 - Abusing MDL

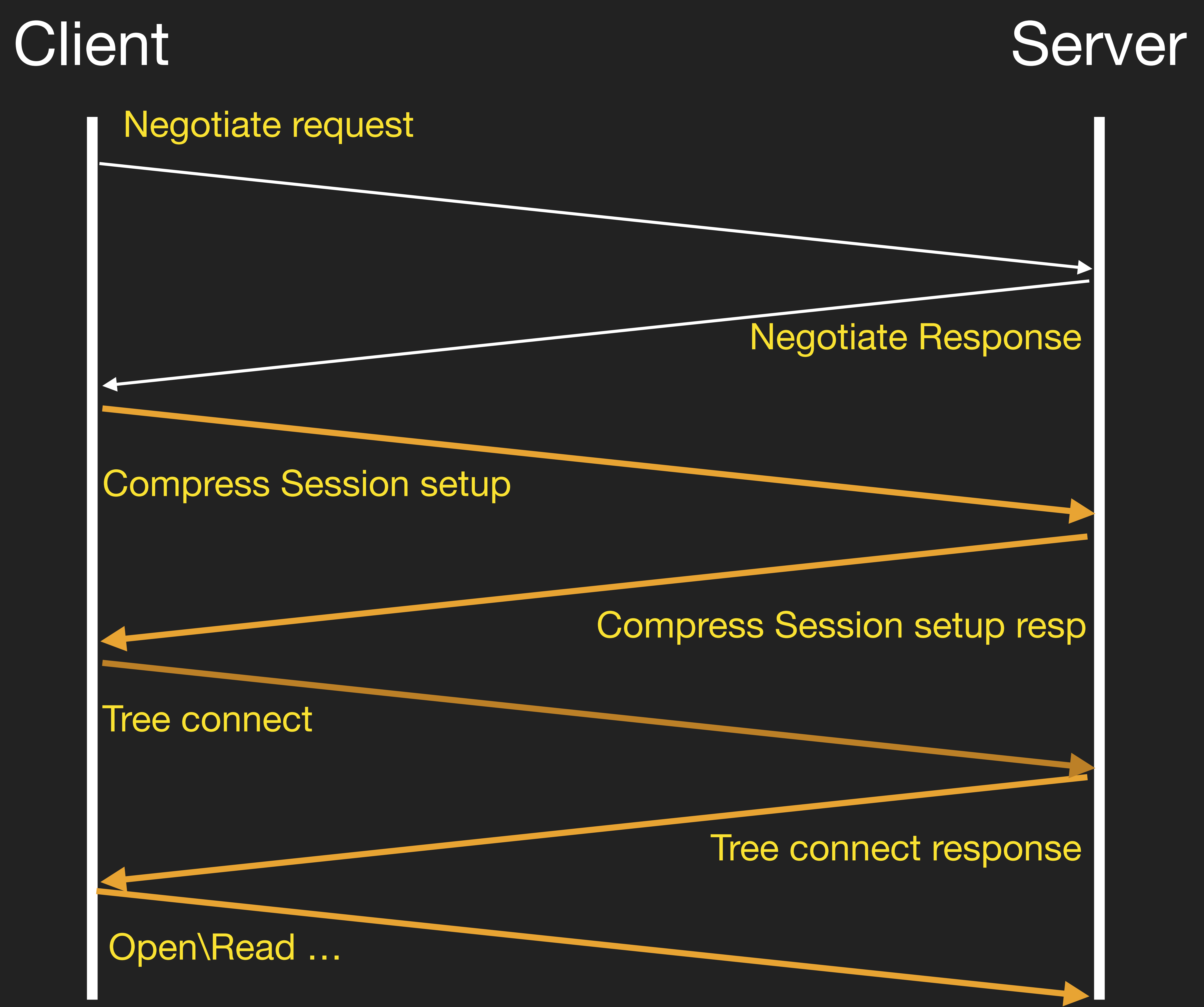
SMBGhost

Vulnerability

- Environment
  - Windows 10 19H1, 19H2
    - SMBCompression
      - SMB 3.1.1 後開始支援對 data 及 command 的壓縮
      - Protocol ID : 0x424D53FC (\xfcSMB)
        - 只要 smb 封包開頭是 (\xfcSMB) 就會用 decompress 來解讀後需內容

Introduction

- Server Message Block (SMB)



DEVCORE SECURITY CONSULTING

SMBGhost

Vulnerability

- SMBCompression Header

The SMB2 COMPRESSION_TRANSFORM_HEADER is used by the client or server when sending compressed messages. This optional header is only valid for the SMB 3.1.1 dialect[69].

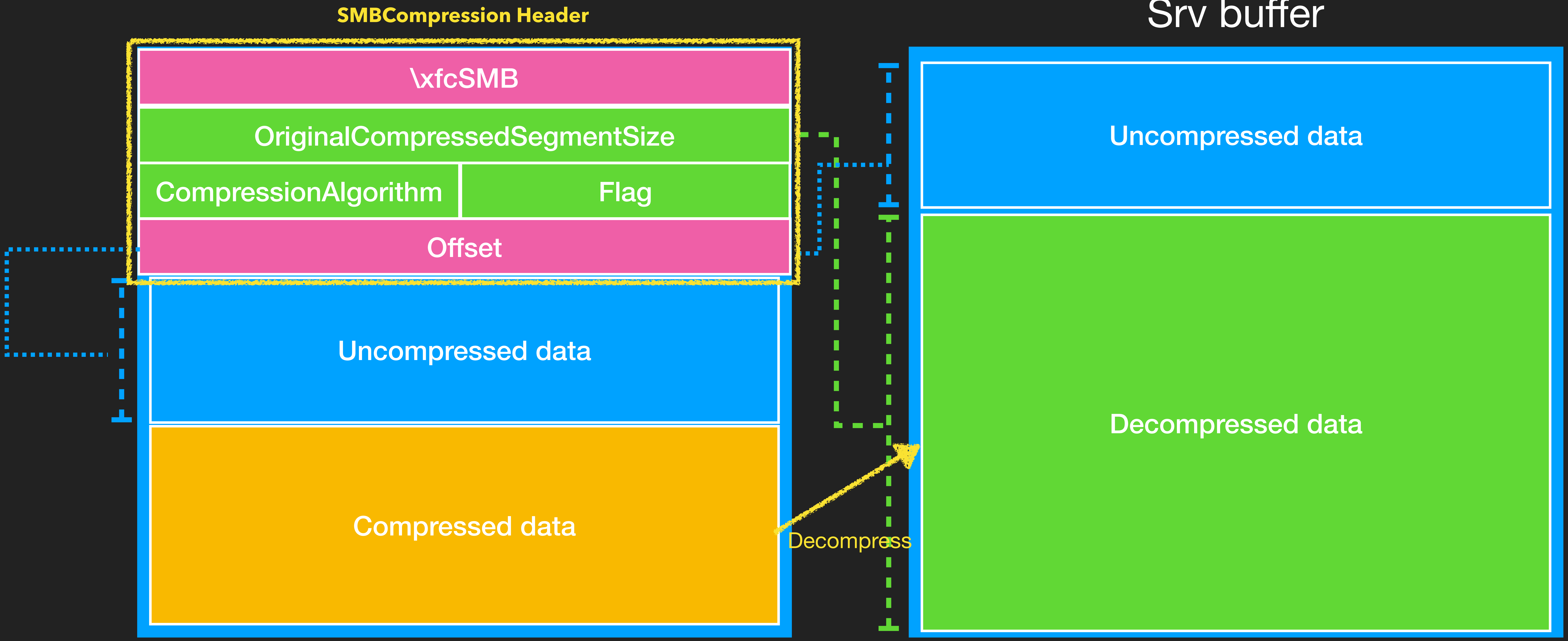
0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6
ProtocolId																										
OriginalCompressedSegmentSize																										
CompressionAlgorithm																Flags										
Offset/Length																										

SMBGhost

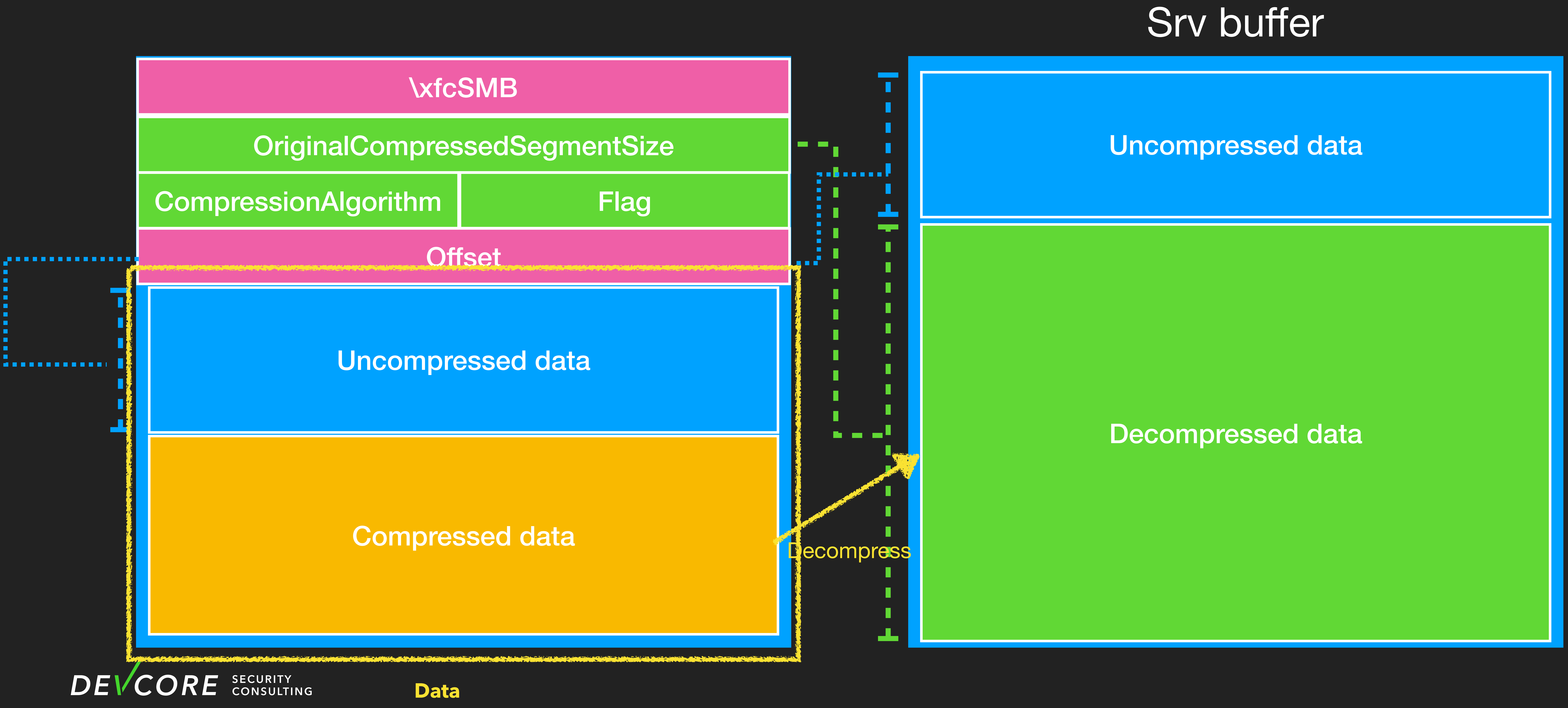
Vulnerability

- SMBCompression Header
  - OriginalCompressedSegmentSize
    - 原始壓縮數據的大小
  - CompressionAlgorithm
    - 壓縮演算法 LZNT1/LZ77/LZ77+Huffman
    - 可用的壓縮法會先在 Negotiate 封包先定義
  - Offset/Length
    - Data 中開始壓縮的 offset

SMBGhost


SMBCompression Header
\xfcSMB
OriginalCompressedSegmentSize
CompressionAlgorithm
Flag
Offset
Uncompressed data
Compressed data
Decompress
Srv buffer
Uncompressed data
Decompressed data

SMBGhost



DEVCORE SECURITY CONSULTING

SMBGhost

Vulnerability

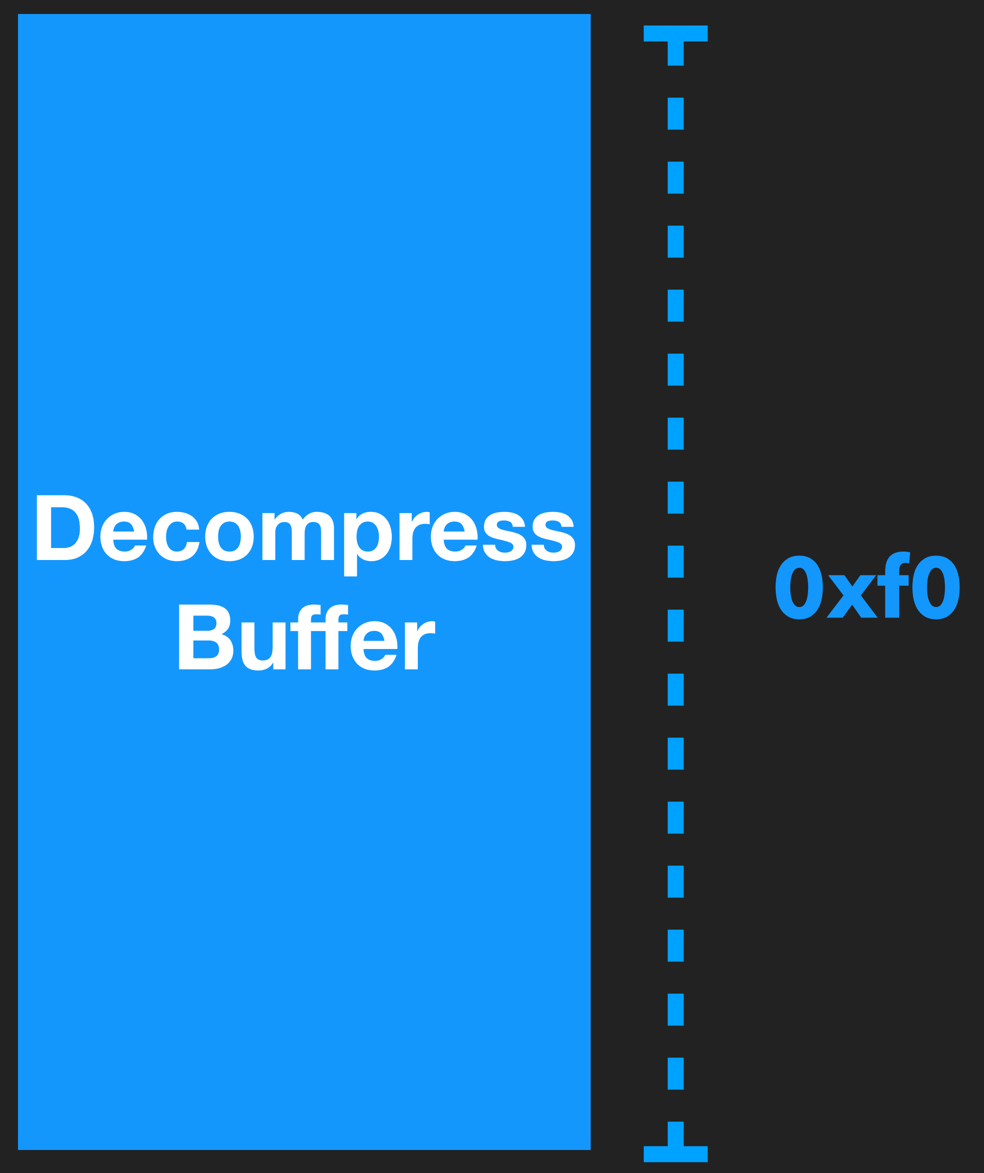
- srv2!srv2DecompressData
  - Integer overflow when it allocates decompress buffer
    - Lead to out of bounds

Vulnerability - SMBGhost

```c
typedef struct _ALLOCATION_HEADER
{
    // ...
    PVOID buffer;
    // ...
} ALLOCATION_HEADER, *PALLOCATION_HEADER;

Srv2DecompressData(){
    ...
    (ALLOCATION_HEADER)decompress = SrvNetAllocateBuffer(OriginalCompressedSegmentSize + offset);
    ...
    NTSTATIS status = SmbCompressionDecompress(
        CompressionAlgorithm,
        compressdata_ptr,
        decompress->buffer+offset,
        OriginalCompressedSegmentSize,
        &FinalCompressdSize)
    if(status < 0 || FinalCompressedSize != OriginalCompressedSegmentSize){
        SrvNetFreeBuffer(decompress);
        return;
    }
    if(offset > 0){
        memcpy(decompress->buffer,header + sizeof(compress_trans_header),offset);
    }
```

Buffer size = 0xfffffff0 + 0x100= 0xf0



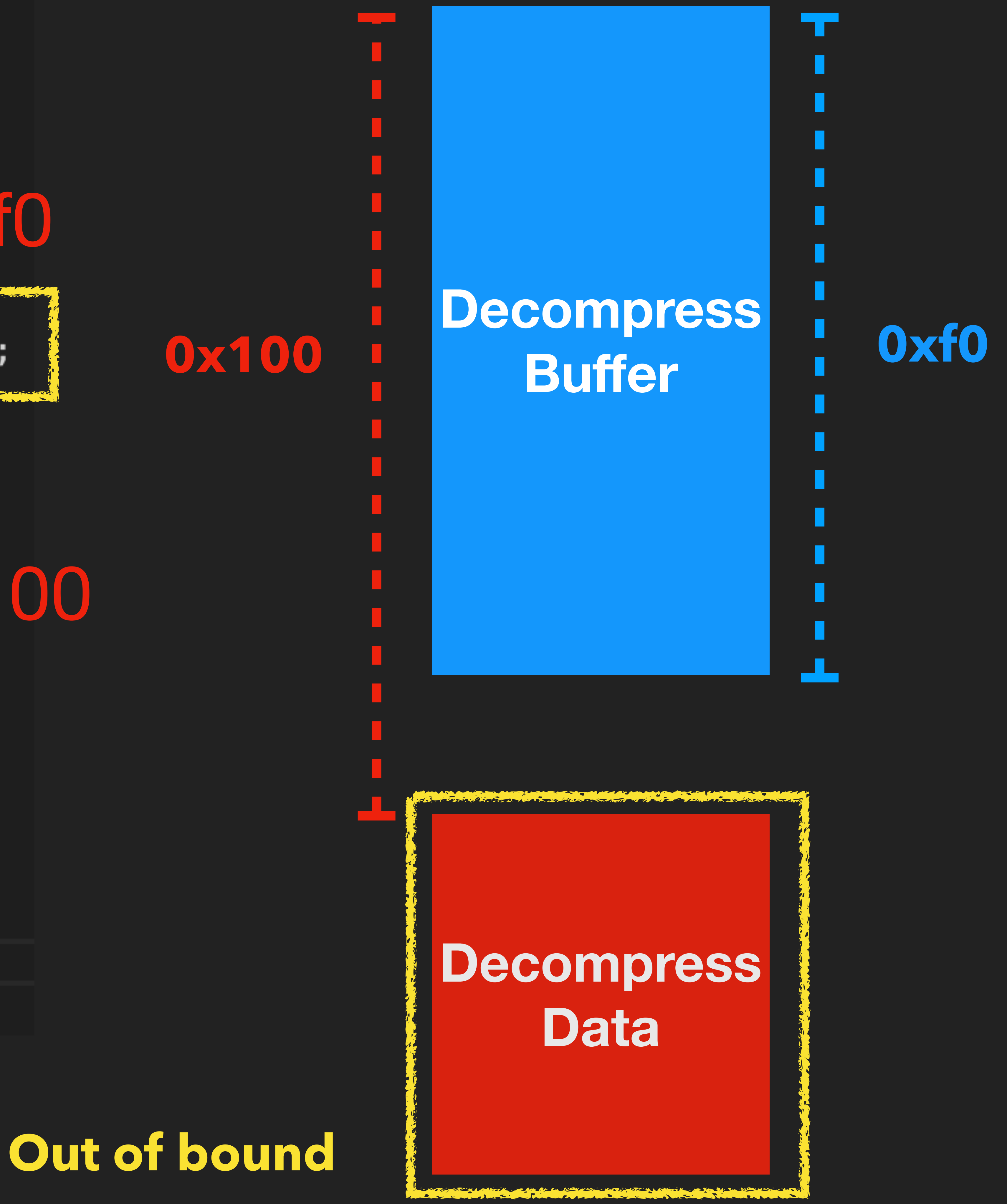
Vulnerability - SMBGhost

```c
typedef struct _ALLOCATION_HEADER
{
    // ...
    PVOID buffer;
    // ...
} ALLOCATION_HEADER, *PALLOCATION_HEADER;

Srv2DecompressData(){
    (ALLOCATION_HEADER)decompress = SrvNetAllocateBuffer(OriginalCompressedSegmentSize + offset);
    ...
    NTSTATIS status = SmbCompressionDecompress(
        CompressionAlgorithm,
        compressdata_ptr,
        decompress->buffer+offset,
        OriginalCompressedSegmentSize,
        &FinalCompressdSize)
    if(status < 0 || FinalCompressedSize != OriginalCompressedSegmentSize){
        SrvNetFreeBuffer(decompress);
        return;
    }
    if(offset > 0){
        memcpy(decompress->buffer,header + sizeof(compress_trans_header),offset);
    }
```

Buffer size = 0xfffffff0 + 0x100= 0xf0

Buffer+ offset = buffer + 0x100



DEVCORE SECURITY CONSULTING

Outline

- Introduction
- Vulnerability - CVE-2020-0796
- Exploitation of SMBGhost
    - From crash to arbitrary memory writing
    - How can we get code execution from arbitrary memory writing in the past
    - Method 1 - System root hijack (need some condition)
    - Method 2 - Abusing MDL

Exploitation of SMBGhost

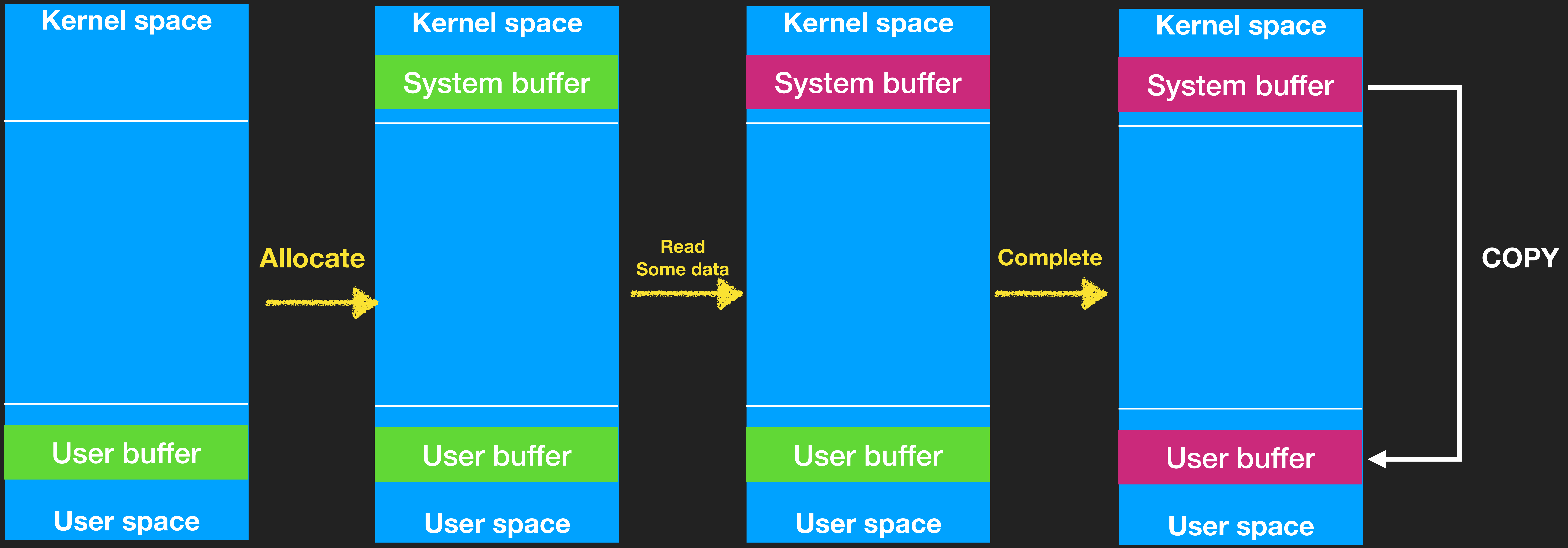
Methods for Accessing Data Buffers in windows driver

- 當 Driver 需要 access user 傳進來的 buffer 時，有可能執行 driver 的 thread 與 user request thread 不同或者 user buffer 在 page out 下，因為 driver IRQL >= 2 不會有 paging，而有可能導致無法正確 access 到，因此 Windows 由提供下列幾種方式來 access user data
    - Buffered I/O
    - Direct I/O
    - Neither Buffered Nor Direct I/O

Exploitation of SMBGhost

Methods for Accessing Data Buffers in windows driver

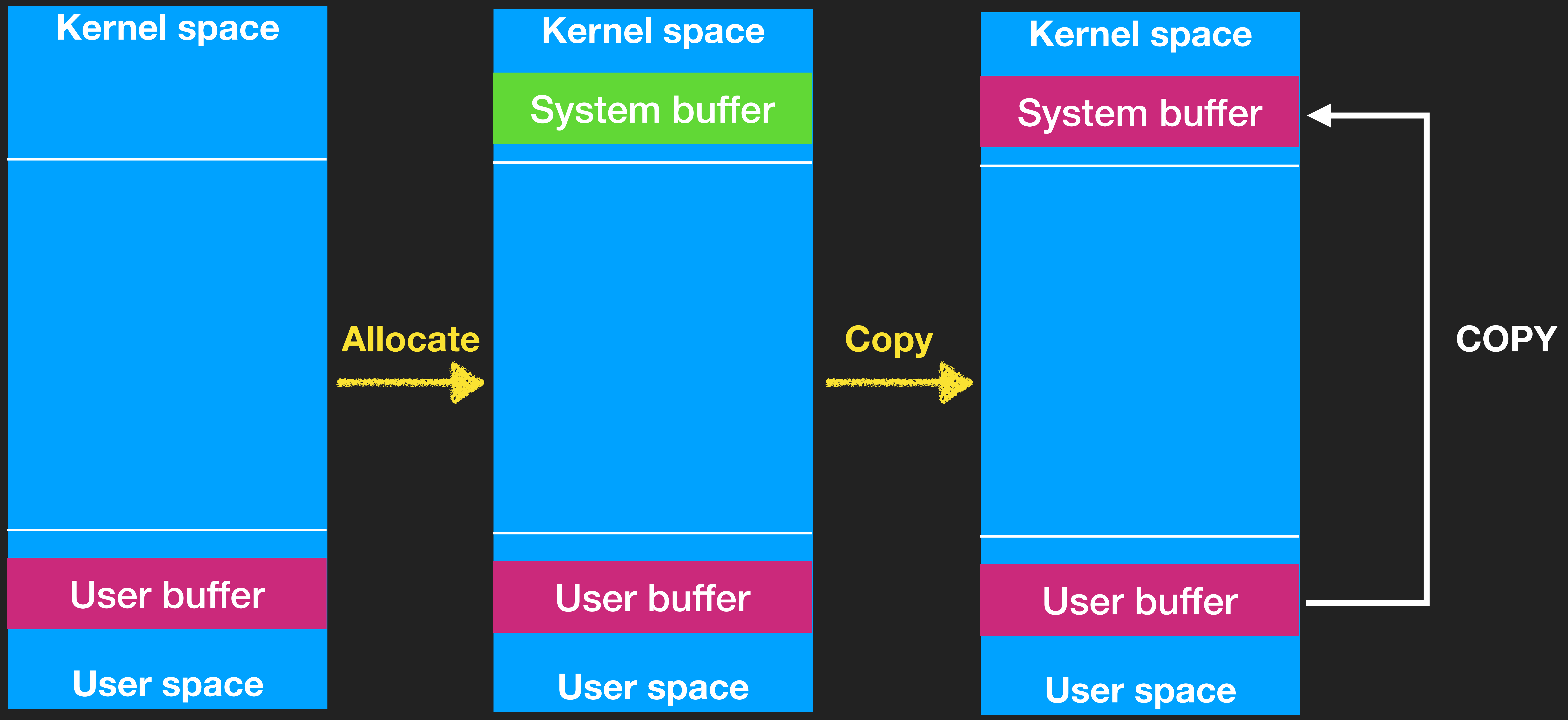
- Buffered I/O (read)



Exploitation of SMBGhost

Methods for Accessing Data Buffers in windows driver

- Buffered I/O (write)

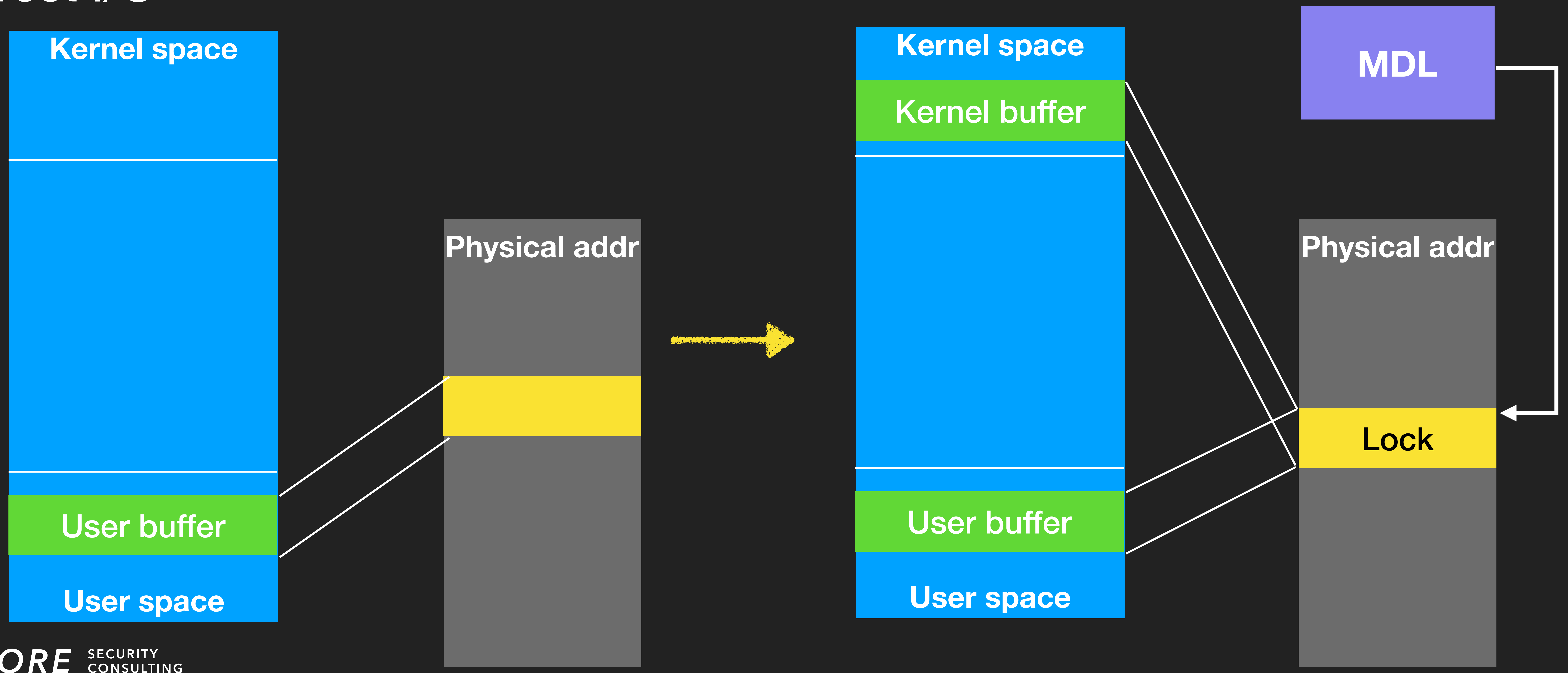


DEVCORE SECURITY CONSULTING

Exploitation of SMBGhost

Methods for Accessing Data Buffers in windows driver

- Direct I/O

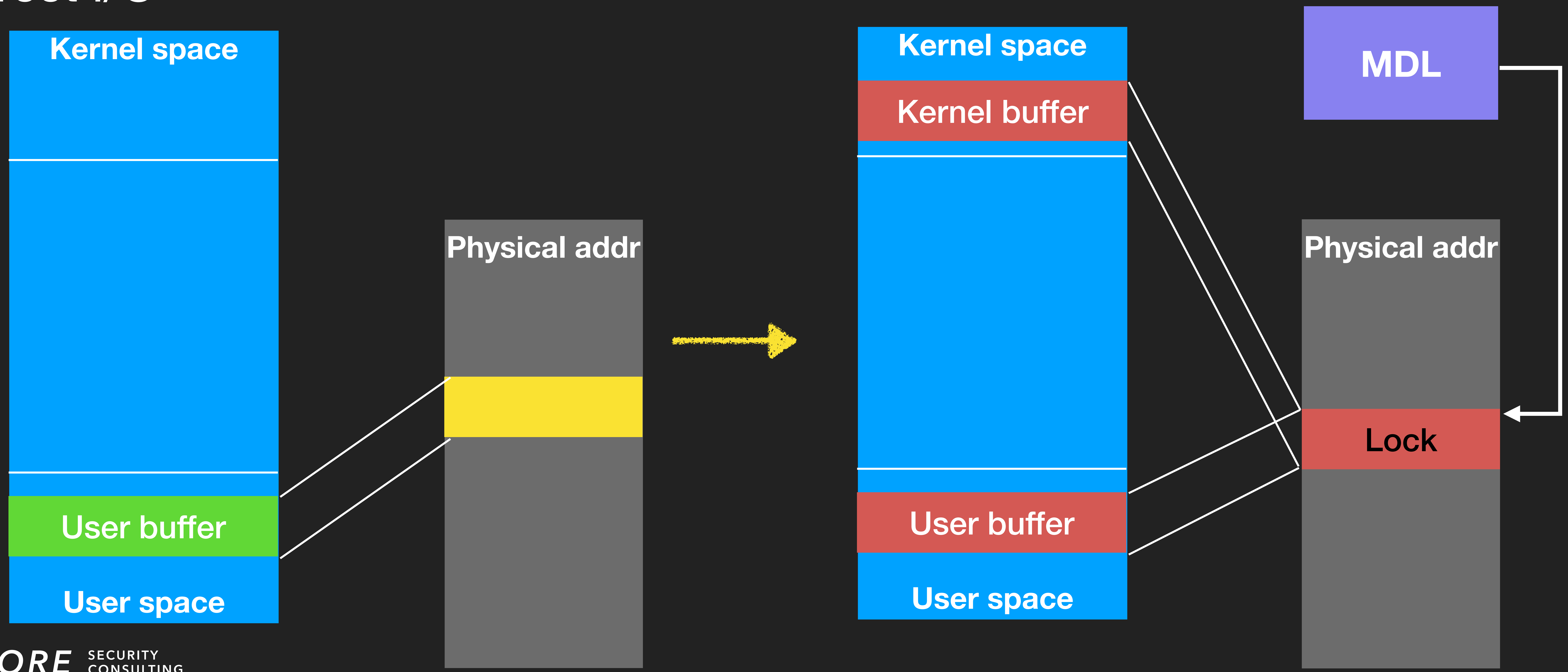


It will use kernel buffer to accesss data

Exploitation of SMBGhost

Methods for Accessing Data Buffers in windows driver

- Direct I/O



It will use kernel buffer to accesss data

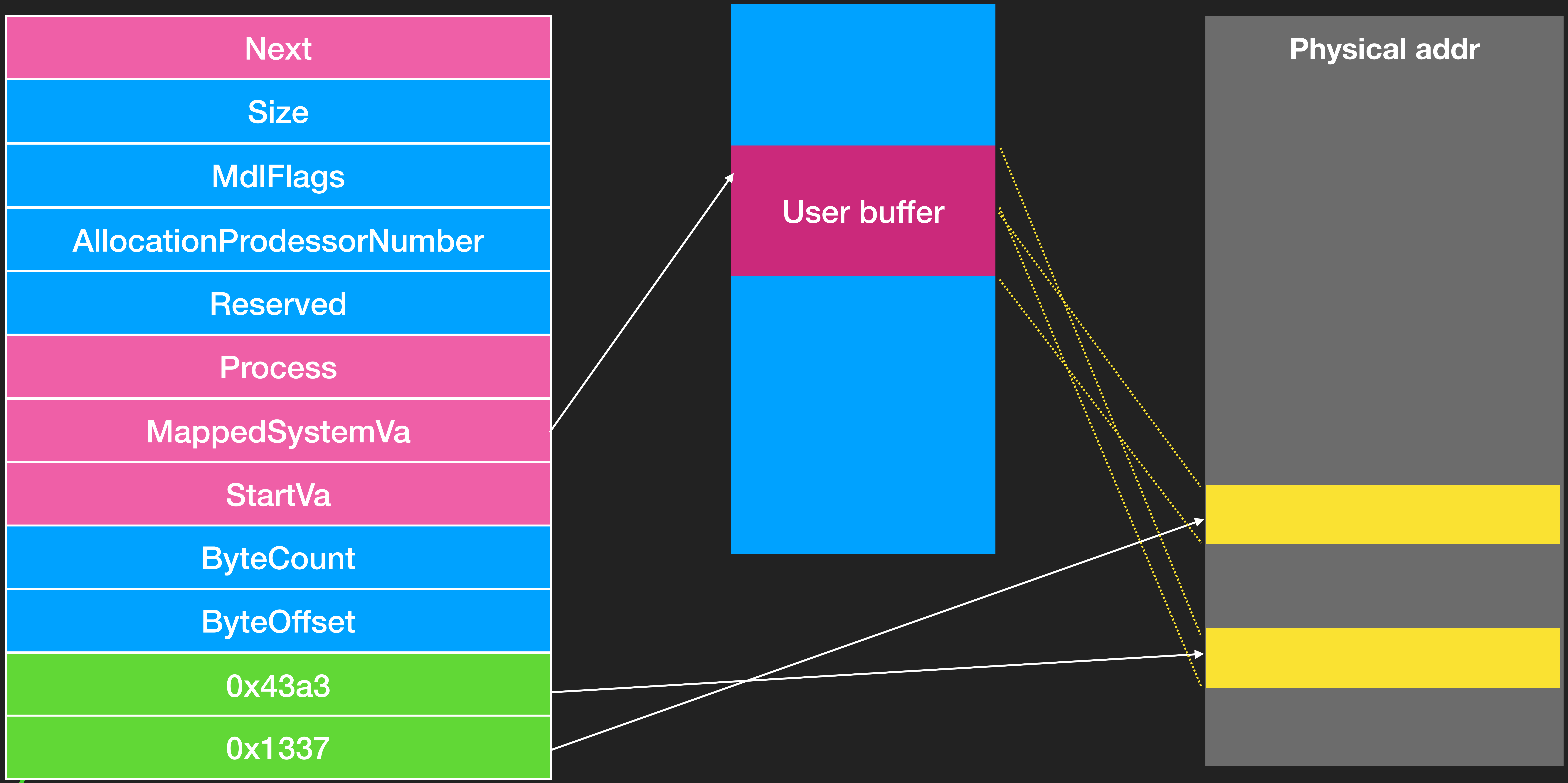
DEVCORE SECURITY CONSULTING

Exploitation of SMBGhost

Methods for Accessing Data Buffers in windows driver

- Memory descriptor list (MDL)
  - 描述一段連續虛擬記憶體區塊與 physical address (通常不連續) 對應的結構
  - 主要用於 I/O 操作，使用時會將提供的 Virtual Address lock 使得該記憶體區段變為 non-paged ，操作結束後會 unlock ，讓該記憶體區段變回 paged ，Virtual Address 可為 User Mode 或 Kernel Mode
  - 也可用於 DMA (Directly Memory Access)

Exploitation of SMBGhost



DEVCORE SECURITY CONSULTING _MDL

Exploitation of SMBGhost

_MDL

0x0	Next
0x8	Size
0xa	MdlFlags
0xc	AllocationProdessorNumber
0xe	Reserved
0x10	Process
0x18	MappedSystemVa
0x20	StartVa
0x28	ByteCount
0x2c	ByteOffset
0x30	Physical Address[x]

- _MDL
  - Next (_MDL)
    - 指向下一個 MDL 結構，用於 UserBuffer 為不連續的虛擬記憶體區段
  - Size
    - 該 MDL structure 的大小，取決於尾段 physical address 數量

Exploitation of SMBGhost

_MDL

0x0	Next
0x8	Size
0xa	MdlFlags
0xc	AllocationProdessorNumber
0xe	Reserved
0x10	Process
0x18	MappedSystemVa
0x20	StartVa
0x28	ByteCount
0x2c	ByteOffset
0x30	Physical Address[x]

- _MDL
  - MdlFlags
    - 描述該 MDL 狀態，如用於何處
  - Process (_EPROCESS)
    - 該 Virtual address 所屬的 Process 結構

Exploitation of SMBGhost

- Mdlflags

```
/* MDL definitions */

#define MDL_MAPPED_TO_SYSTEM_VA      0x0001
#define MDL_PAGES_LOCKED             0x0002
#define MDL_SOURCE_IS_NONPAGED_POOL  0x0004
#define MDL_ALLOCATED_FIXED_SIZE     0x0008
#define MDL_PARTIAL                  0x0010
#define MDL_PARTIAL_HAS_BEEN_MAPPED  0x0020
#define MDL_IO_PAGE_READ             0x0040
#define MDL_WRITE_OPERATION          0x0080
#define MDL_PARENT_MAPPED_SYSTEM_VA  0x0100
#define MDL_FREE_EXTRA_PTES          0x0200
#define MDL_DESCRIBES_AWE            0x0400
#define MDL_IO_SPACE                 0x0800
#define MDL_NETWORK_HEADER           0x1000
#define MDL_MAPPING_CAN_FAIL         0x2000
#define MDL_ALLOCATED_MUST_SUCCEED   0x4000
#define MDL_INTERNAL                 0x8000
```

Exploitation of SMBGhost

_MDL

0x0	Next
0x8	Size
0xa	MdlFlags
0xc	AllocationProdessorNumber
0xe	Reserved
0x10	Process
0x18	MappedSystemVa
0x20	StartVa
0x28	ByteCount
0x2c	ByteOffset
0x30	Physical Address[x]

- _MDL
  - MappedSystemVa
    - 該 buffer 的起始位置
  - StartVa
    - 該 buffer 所屬的 virtual address 的開頭 (page alignment)
    - 可以在 user space 或 kernel space

Exploitation of SMBGhost

_MDL

0x0	Next
0x8	Size
0xa	MdlFlags
0xc	AllocationProdessorNumber
0xe	Reserved
0x10	Process
0x18	MappedSystemVa
0x20	StartVa
0x28	ByteCount
0x2c	ByteOffset
0x30	Physical Address[x]

- _MDL
  - Physical Address
    - 該 buffer 所對應到的 physical address
    - 會 >> 12 後在存入
    - 在 Map MappedSystemVa 時，會將該 PA map 到虛擬記憶體上

Exploitation of SMBGhost

Methods for Accessing Data Buffers in windows driver

- Neither Buffered Nor Direct I/O
  - The operating system passes the application buffer's virtual starting address and size to the driver stack. The buffer is only accessible from drivers that execute in the application's thread context.
  - only highest-level drivers, such as FSDs, can use this method for accessing buffers.
  - 使用時必須注意傳入位置是否屬於 user space ，寫入時也需檢查要寫入的 buffer 是否屬於 user space ，漏洞常發生於此
    - ProbeForRead
    - ProbeForWrite

Exploitation of SMBGhost

Methods for Accessing Data Buffers in windows driver

- SMB uses Direct I/O
  - Receive and response buffer 會使用 MDL 來描述
  - 最後會透過 tcpip.sys 來傳遞
    - 可採用 DMA 形式來傳輸

Exploitation of SMBGhost

From crash to arbitrary memory writing

- SrvNetAllocateBuffer()
  - SMB 所使用的記憶體基本上由 srvnet.sys 所管理
    - Receive & Response buffer
  - 該記憶體管理使用 lookasidelist 來管理釋放的記憶體區塊，分配時如果 lookasidelist 有適合的 free block 就會使用該記憶體區塊
    - Size 區間為 [0x1100,0x2100,0x4100,0x8100,...,0x100100] ，相同大小 Freed block 會被放入同一個 linked list 中，另外每個 CPU 都會有自己獨立的 lookasidelist
    - 超過則使用系統的 ExPoolAllocate

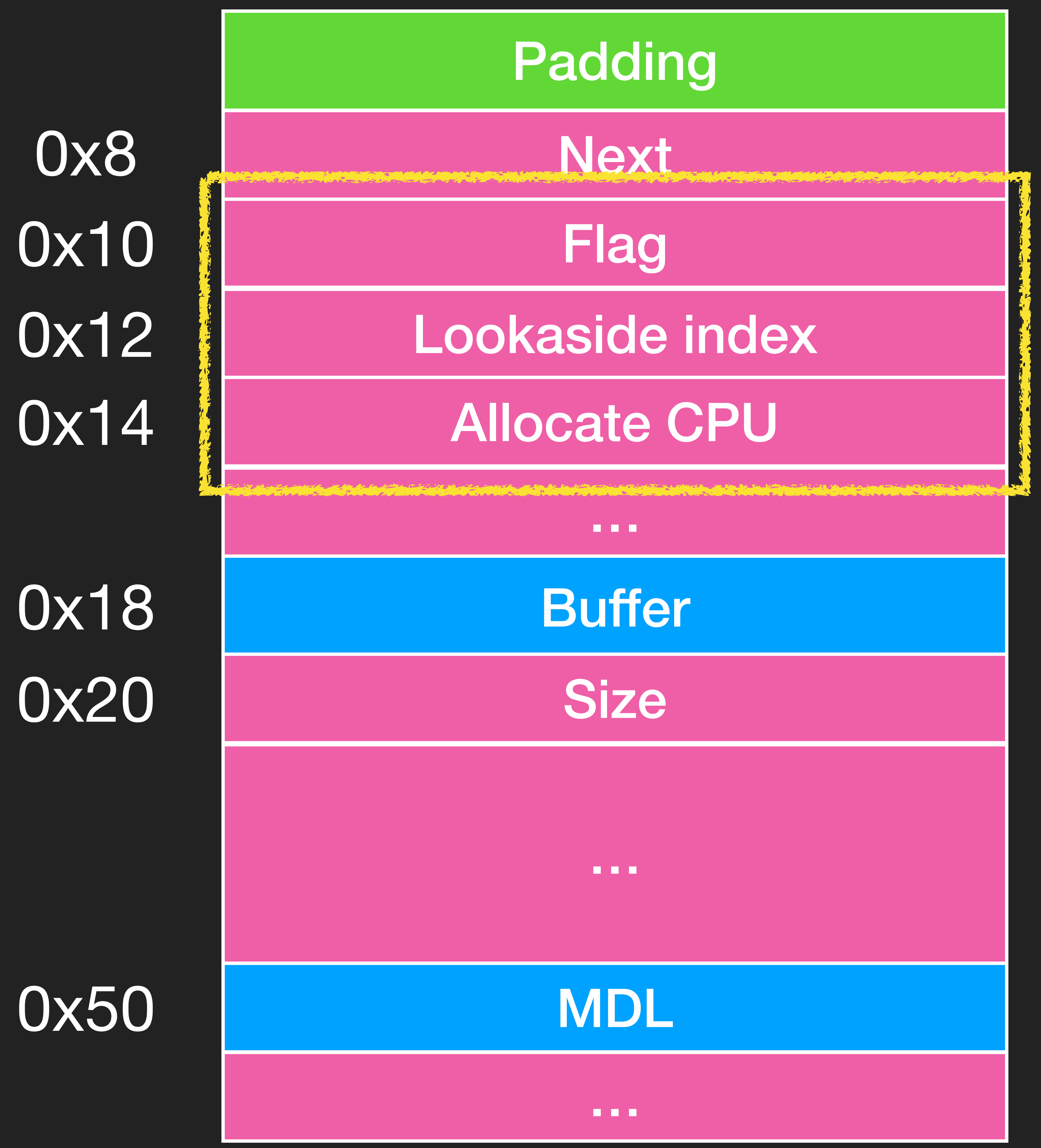
Exploitation of SMBGhost

From crash to arbitrary memory writing

- SrvNetAllocateBuffer Allocation Structure
  - 用來管理 srvnet 所分配的記憶體區塊的結構，每個分配出去的 buffer 都會有一個
  - SrvNetAllocateBuffer(…) 所回傳的是該結構
    - 與其他記憶體分配結構所回傳的不同，該 function 所回傳的是記憶體管理結構，而不是單純的記憶體區塊

Exploitation of SMBGhost

From crash to arbitrary memory writing



- SrvAllocStruct
  - Flag
    - Inused flag 表示該 buffer 是 freed 還是正在使用
  - Lookaside index
    - 在 lookaside list 中的 index
  - Allocate CPU
    - 分配時的 cpu number

Exploitation of SMBGhost

From crash to arbitrary memory writing

	Padding
0x8	Next
0x10	Flag
0x12	Lookaside index
0x14	Allocate CPU
	...
0x18	Buffer
0x20	Size
	...
0x50	MDL
	...

- SrvAllocStruct
  - Buffer
    - 該結構所管理的記憶體區塊，也就是使用時會用的 buffer
  - Size
    - 該 Buffer 的大小

Exploitation of SMBGhost

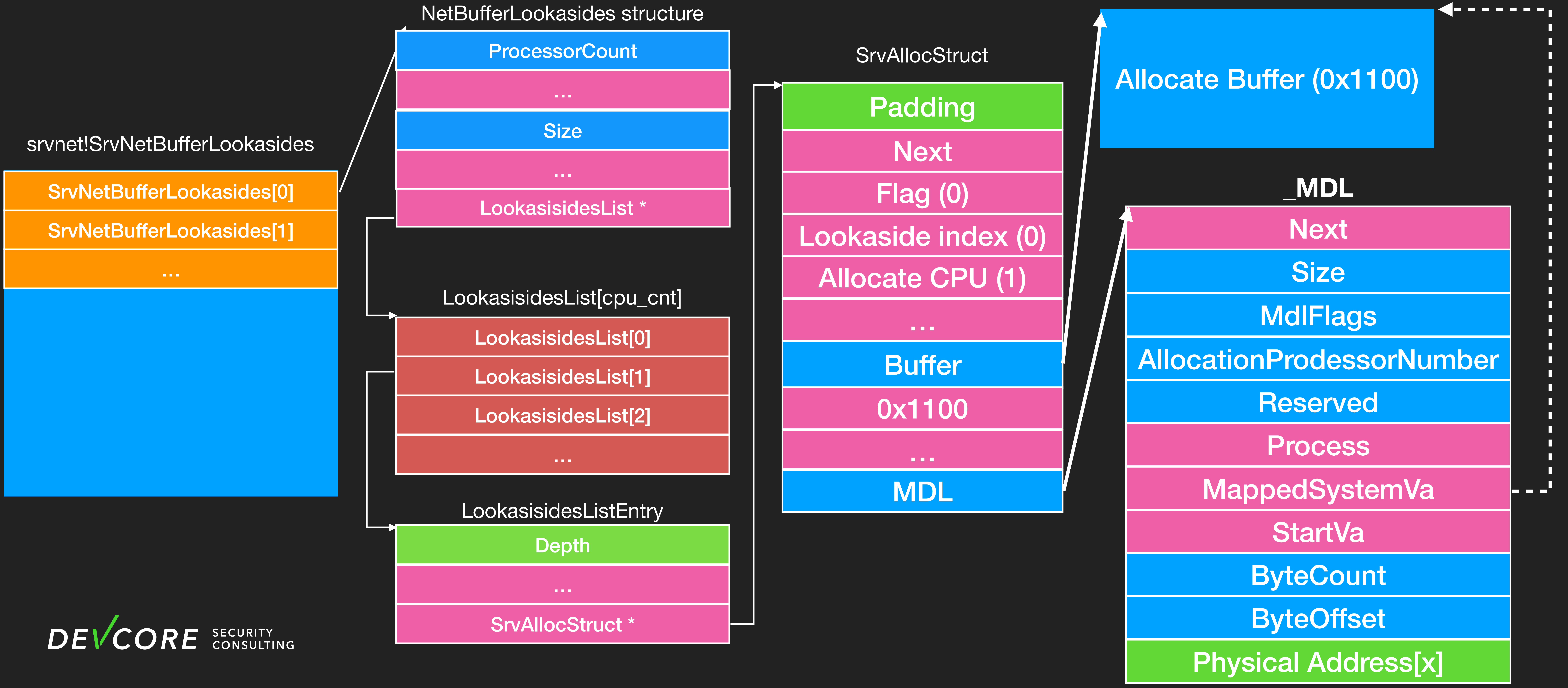
From crash to arbitrary memory writing

	Padding
0x8	Next
0x10	Flag
0x12	Lookaside index
0x14	Allocate CPU
	…
0x18	Buffer
0x20	Size
	…
0x50	MDL
	…

- SrvAllocStruct
  - MDL
    - 指向描述該 buffer 的 MDL ，在做記憶體讀寫、傳送封包時會使用該 MDL ，來做記憶體相關操作
  - Next
    - 指向下一塊 Freed SrvAllocStruct

Exploitation of SMBGhost

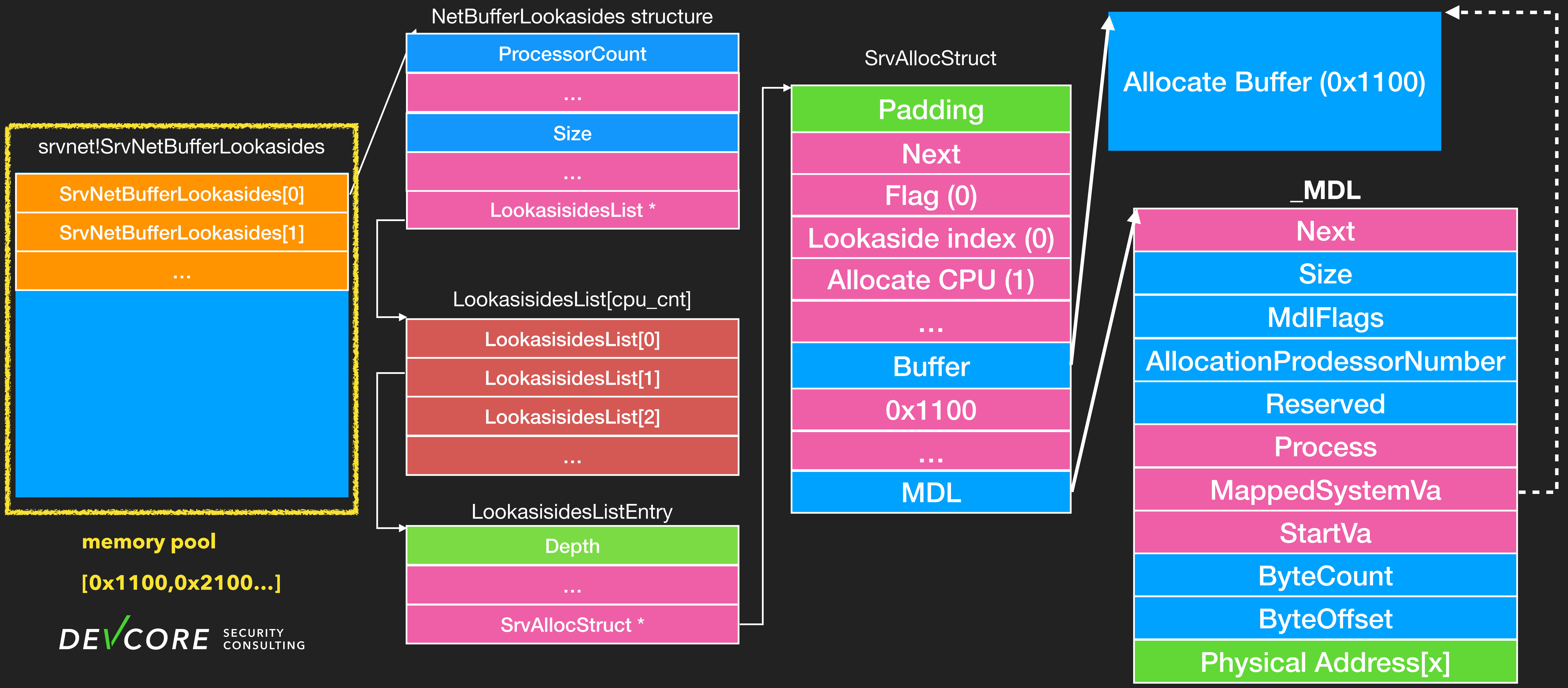
From crash to arbitrary memory writing



DEVCORE SECURITY CONSULTING

Exploitation of SMBGhost

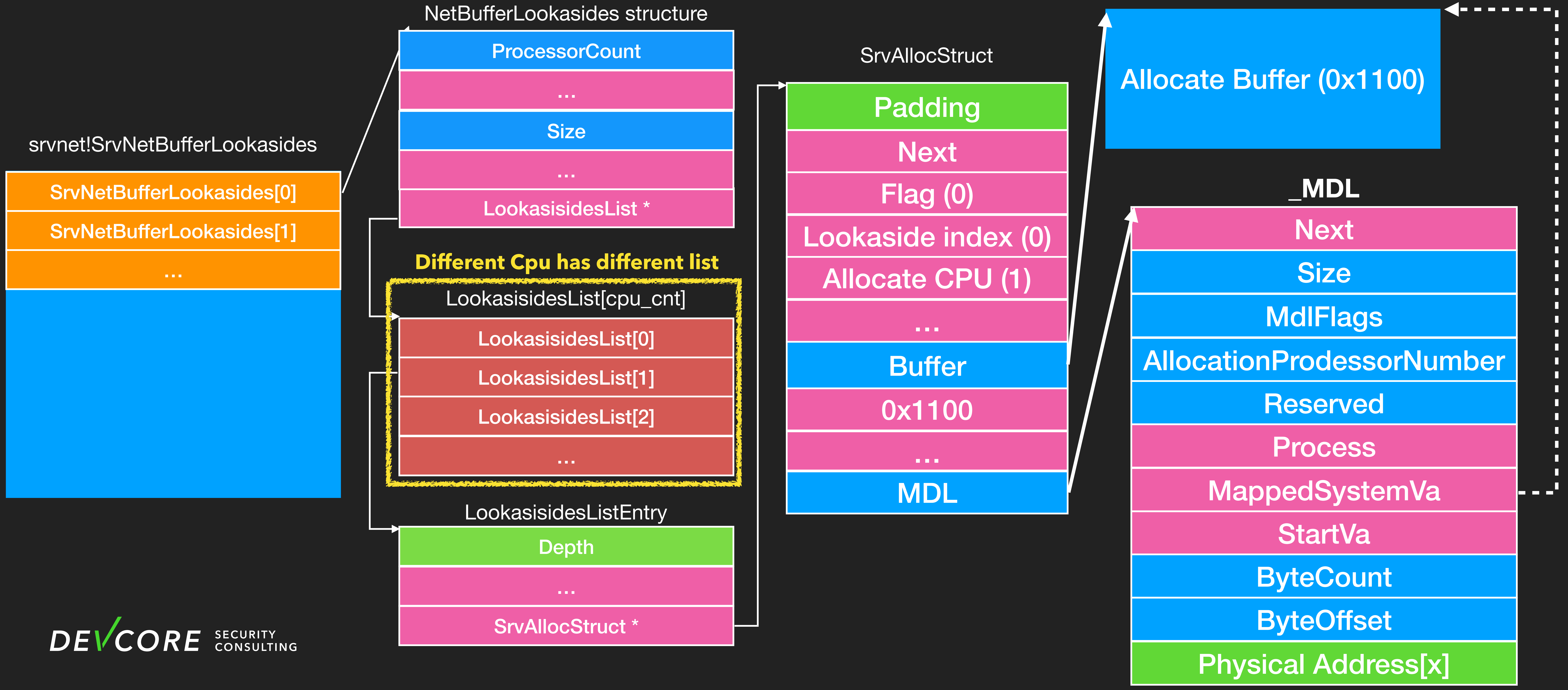
From crash to arbitrary memory writing



DEVCORE SECURITY CONSULTING

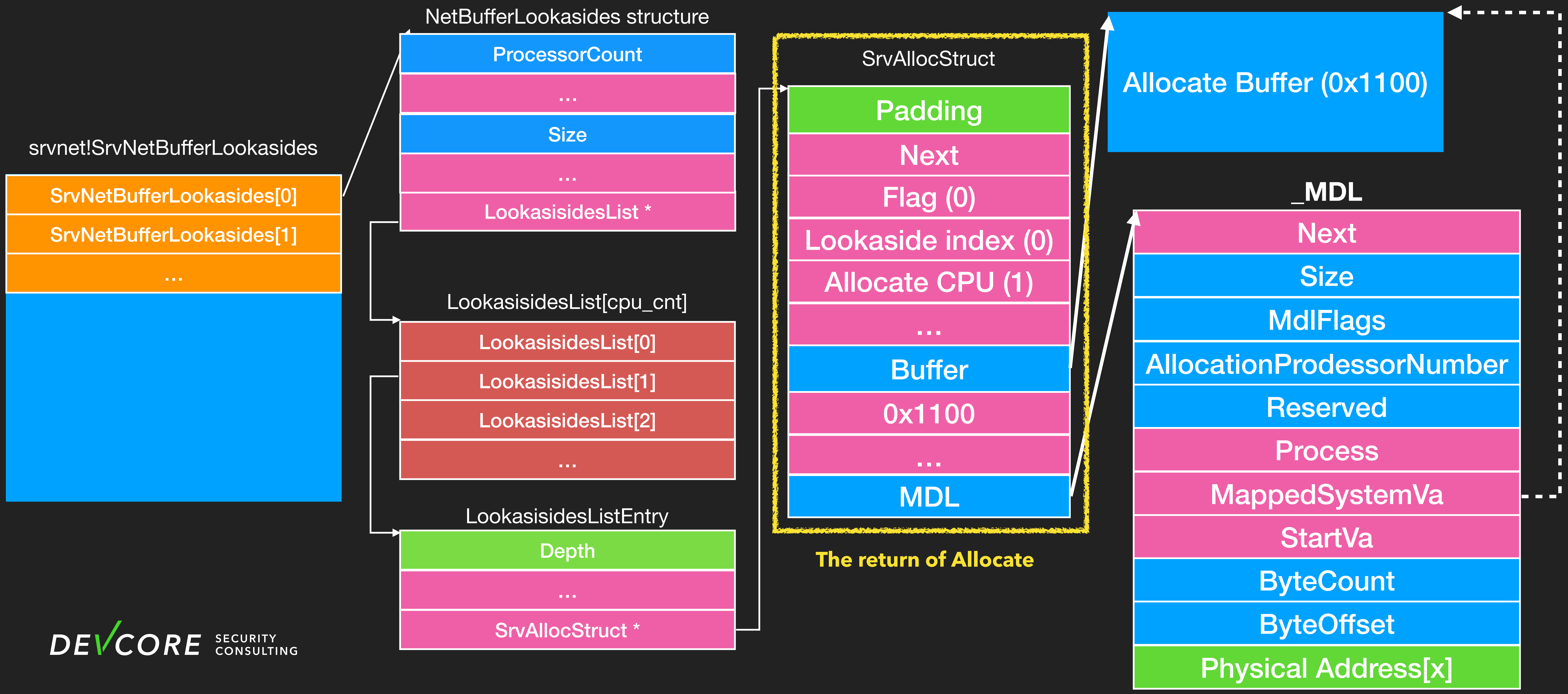
Exploitation of SMBGhost

From crash to arbitrary memory writing



Exploitation of SMBGhost

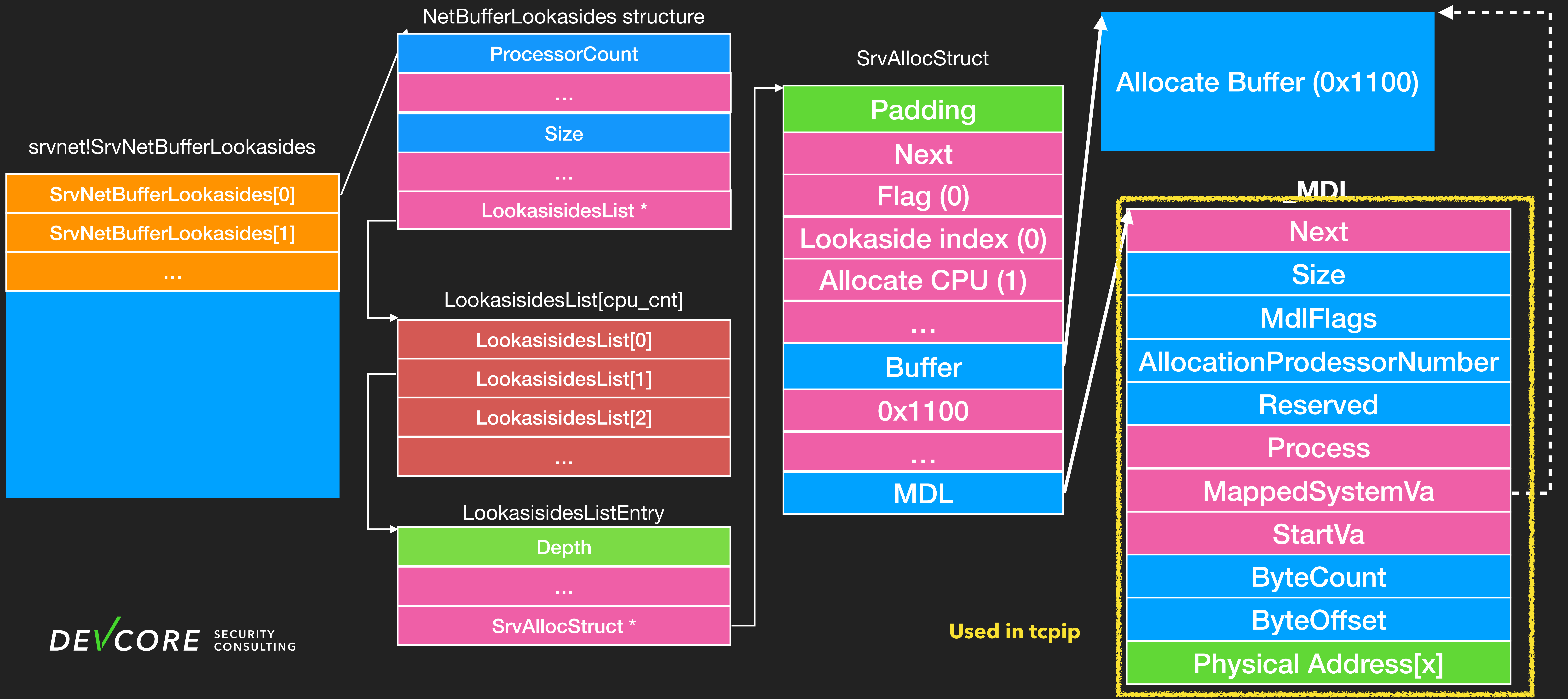
From crash to arbitrary memory writing



DEVCORE SECURITY CONSULTING

Exploitation of SMBGhost

From crash to arbitrary memory writing



DEVCORE SECURITY CONSULTING

Exploitation of SMBGhost

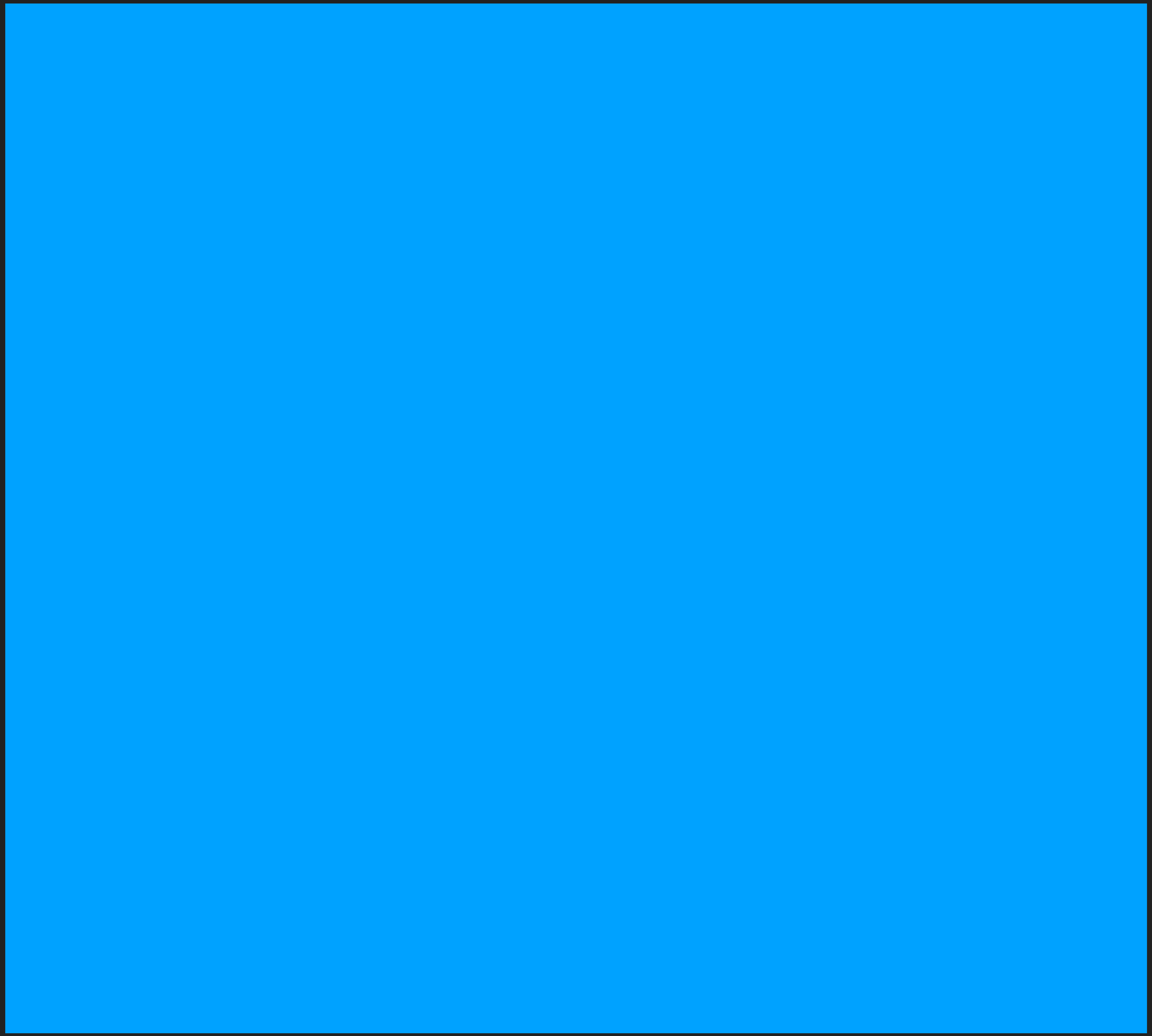
From crash to arbitrary memory writing

- SrvNetAllocateBuffer Allocate
  - 如果沒有在 SrvNetBufferLookasides 找到適合的 buffer，則會使用系統的動態記憶體分配 ExPoolAlloacate 分配 buffer + SrvAllocaStruct + MDL 大小的空間
    - 並將 SrvAllocaStruct 及 MDL 初始化
    - SrvAllocaStruct 及 MDL 會放在該 buffer 尾端

Exploitation of SMBGhost

From crash to arbitrary memory writing

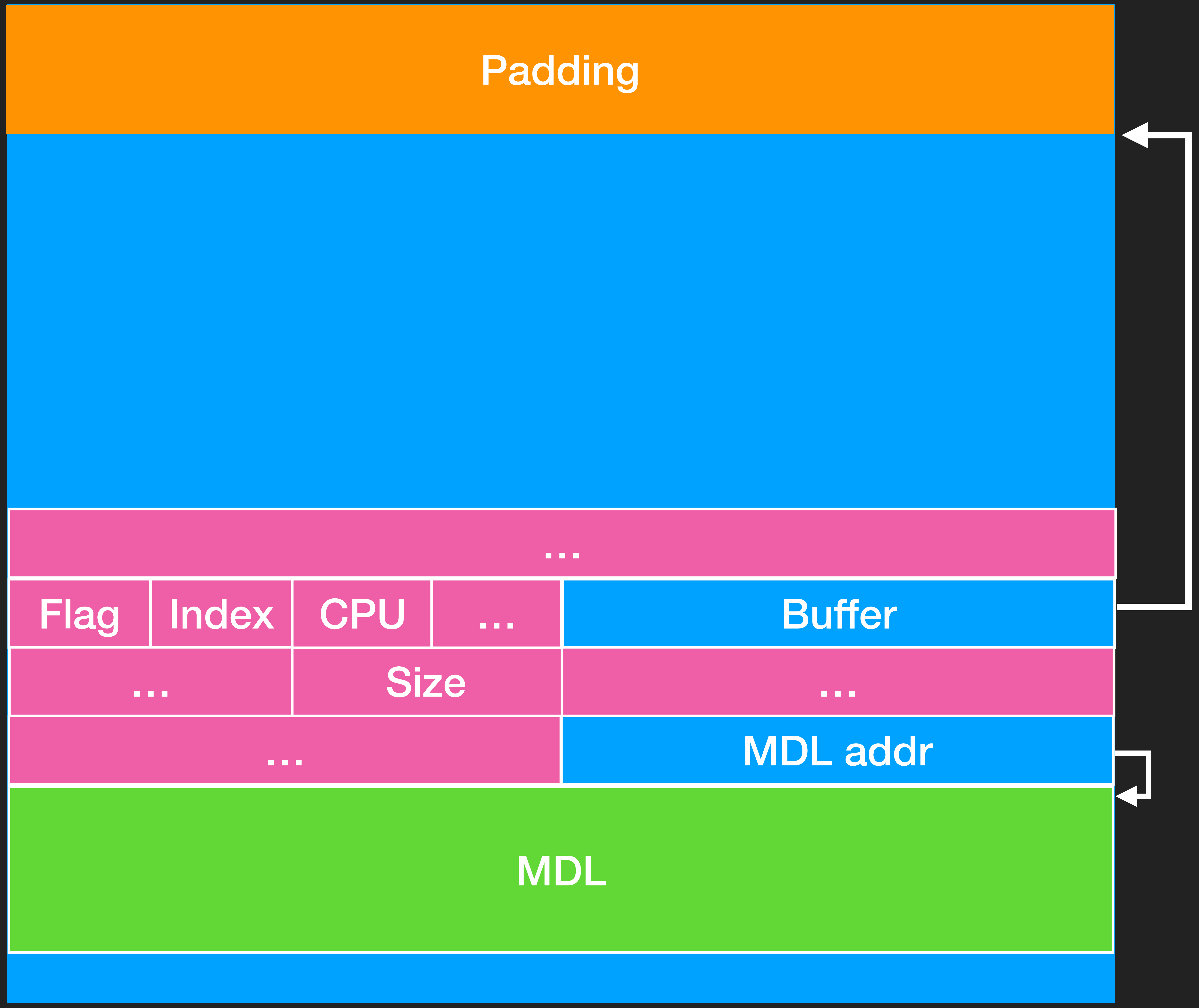
ExAllocatePoolWithTag



Exploitation of SMBGhost

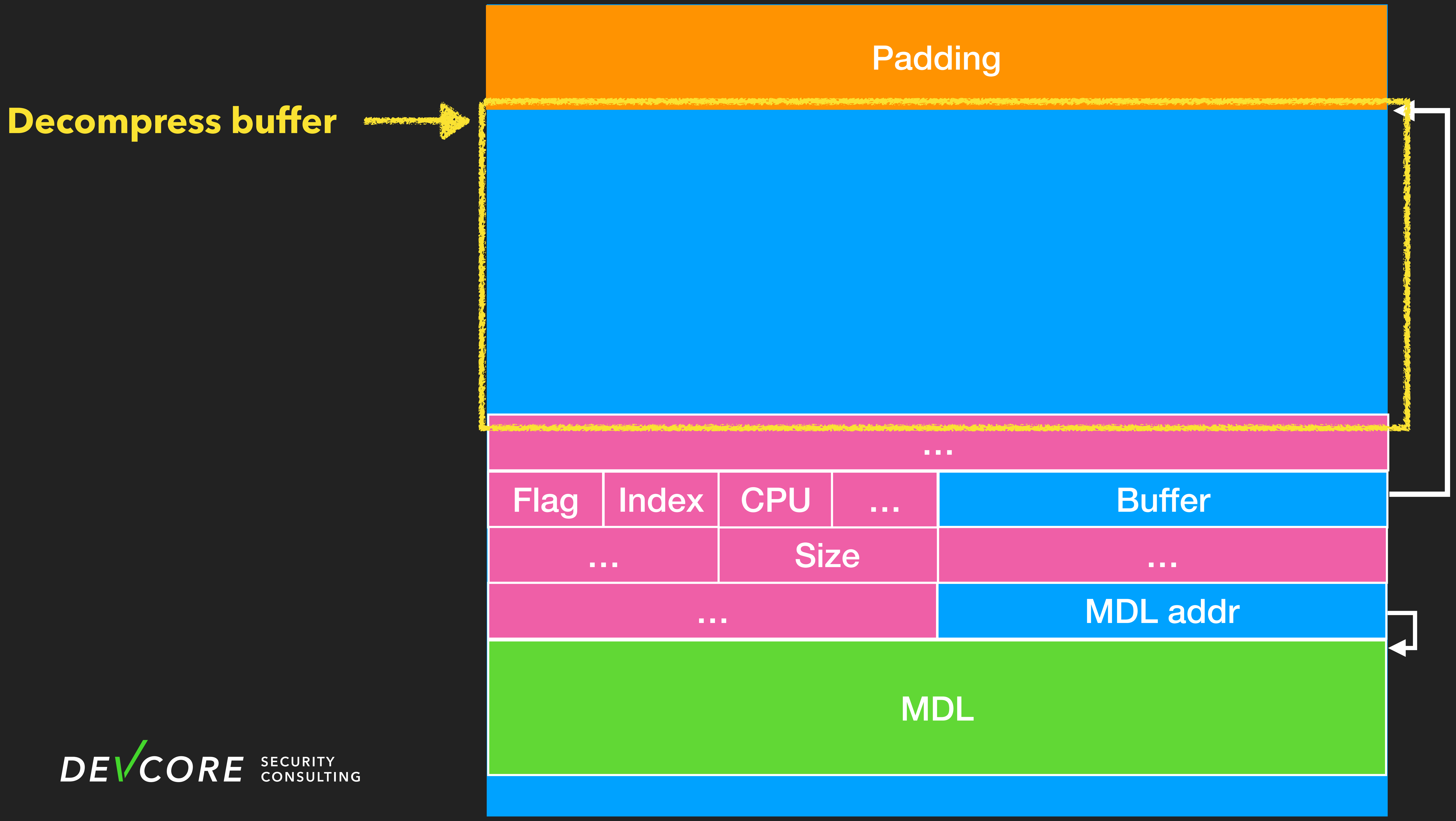
From crash to arbitrary memory writing

Initialize MDL
& SrvAllocaStruct



Exploitation of SMBGhost

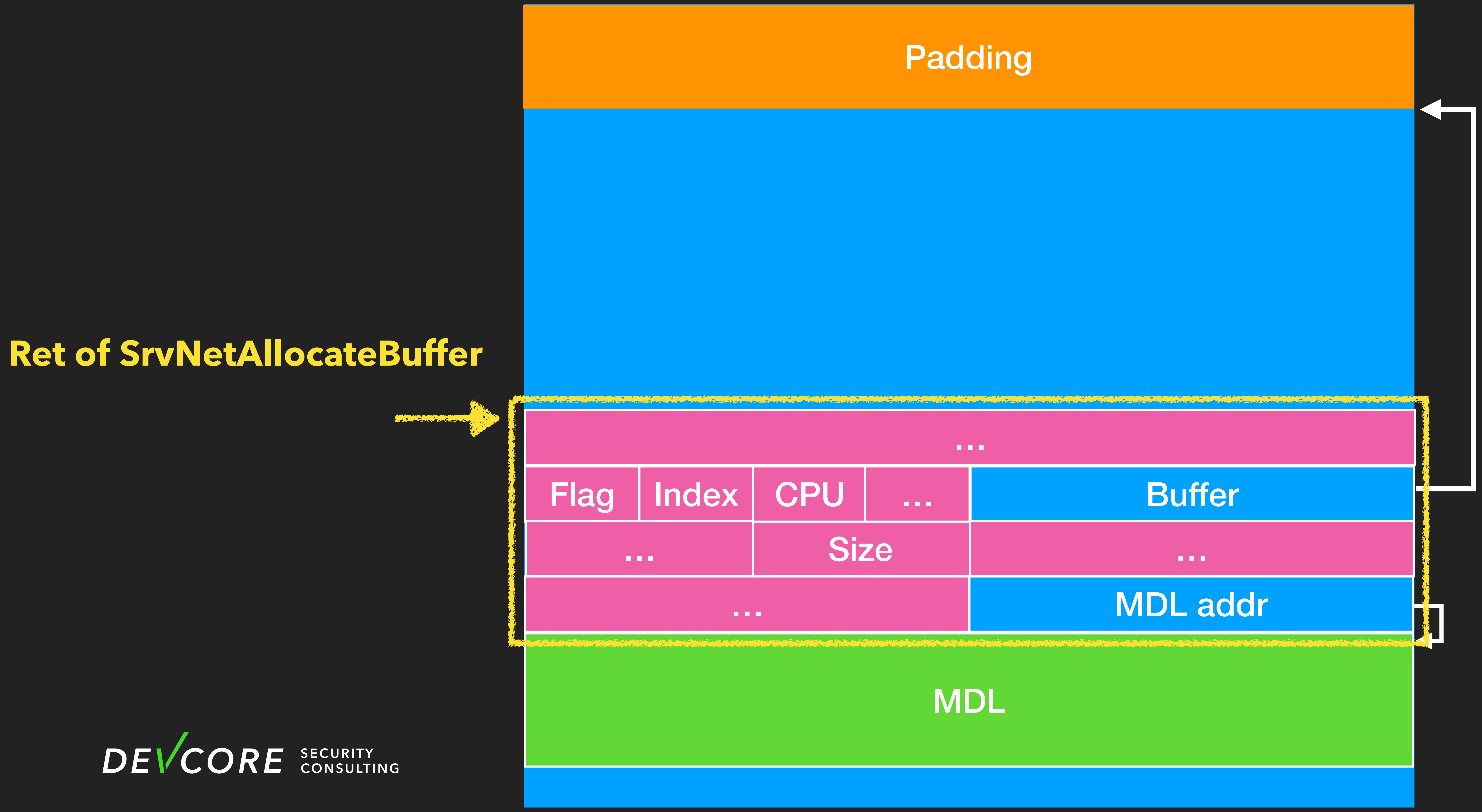
From crash to arbitrary memory writing



DEVCORE SECURITY CONSULTING

Exploitation of SMBGhost

From crash to arbitrary memory writing



DEVCORE SECURITY CONSULTING

Exploitation of SMBGhost

From crash to arbitrary memory writing

Backdoor ?

CTF Challenge ?

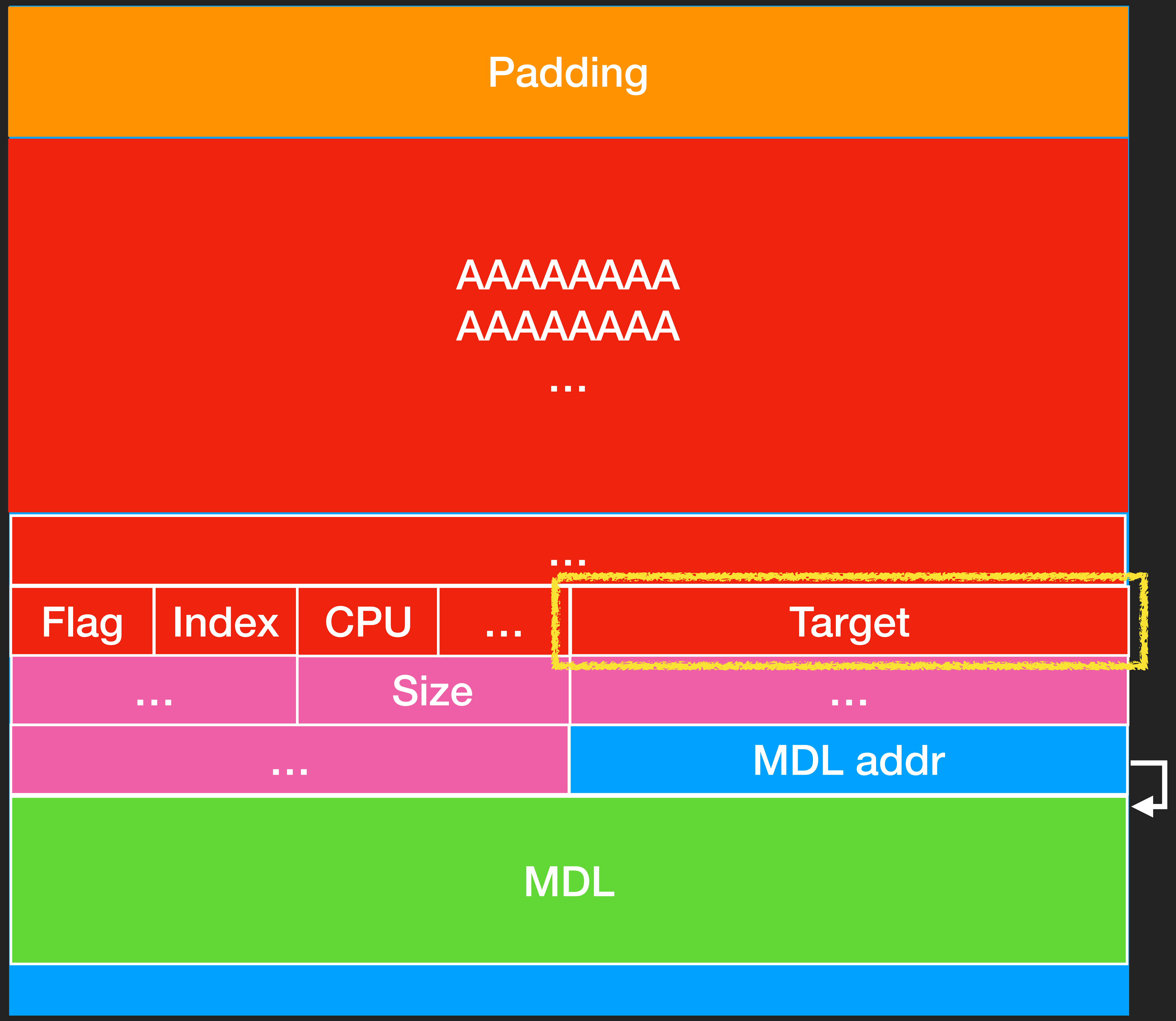
Exploitation of SMBGhost

From crash to arbitrary memory writing

- We can craft a special size to use the vulnerability to overwrite Buffer pointer first
  - After decompressing, it will overwrite the buffer pointer with target address
  - If it decompresses successfully, it will copy the data which uncompress from the original data to the buffer.
    - That is, we can do arbitrary memory writing !

Exploitation of SMBGhost

From crash to arbitrary memory writing



Exploitation of SMBGhost

```
typedef struct _ALLOCATION_HEADER
{
    // ...
    PVOID buffer;
    // ...
} ALLOCATION_HEADER, *PALLOCATION_HEADER;

Srv2DecompressData(){
    ...
    (ALLOCATION_HEADER)decompress = SrvNetAllocateBuffer(OriginalCompressedSegmentSize + offset);
    ...
    NTSTATIS status = SmbCompressionDecompress(
        CompressionAlgorithm,
        compressdata_ptr,
        decompress->buffer+offset,
        OriginalCompressedSegmentSize,
        &FinalCompressdSize)
    if(status < 0 || FinalCompressedSize != OriginalCompressedSegmentSize){
        SrvNetFreeBuffer(decompress);
        return;
    }
    if(offset > 0){
        memcpy(decompress->buffer,header + sizeof(compress_trans_header),offset);
    }
```

Exploitation of SMBGhost

```c
typedef struct _ALLOCATION_HEADER
{
    // ...
    PVOID buffer;
    // ...
} ALLOCATION_HEADER, *PALLOCATION_HEADER;

Srv2DecompressData(){
    ...
    (ALLOCATION_HEADER)decompress = SrvNetAllocateBuffer(OriginalCompressedSegmentSize + offset);
```

Arbitrary memory writing

memcpy(target,original data,offset)

```c
    if(status < 0 || FinalCompressedSize != OriginalCompressedSegmentSize){
        SrvNetFreeBuffer(decompress);
        return;
    }
    if(offset > 0){
        memcpy(decompress->buffer,header + sizeof(compress_trans_header),offset);
    }
```

Outline

- Introduction
- Vulnerability - CVE-2020-0796
- Exploitation of SMBGhost
  - From crash to arbitrary memory writing
  - How can we get code execution from arbitrary memory writing in the past
  - Method 1 - System root hijack (need some condition)
  - Method 2 - Abusing MDL

Exploitation of SMBGhost

How can we get code execution from arbitrary memory writing in the past

- Abusing the HalpInterruptController
- IAT Overwrite
  - We need bypass KASLR and read only protection
- …

Exploitation of SMBGhost

How can we get code execution from arbitrary memory writing in the past

- Abusing the HalpInterruptController
  - HAL (hardware abstraction layer)
    - A loadable kernel-mode module (hal.dll) that provides the low level interface to the hardware platform
    - hide the low-level hardware details from drivers and the operating system
      - I/O interface, interrupt controller …

Exploitation of SMBGhost

How can we get code execution from arbitrary memory writing in the past

- Abusing the HalpInterruptController
  - HalpInterruptController
    - We can overwrite HalpApicRequestInterrupt to control RIP
      - It will be called quite frequently by windows
      - 在 Windows 8 之前 HAL Heap 為固定且為可讀可寫可執行，因此可將 shell code 也寫上面跳過去就好
        - EternalBlue

Exploitation of SMBGhost

How can we get code execution from arbitrary memory writing in the past

- Abusing the HalpInterruptController
  - Win 8 之後，HAL 變成可讀可寫不可執行，控制 RIP 後，要到 shellcode 執行變得困難許多
  - 因此後來須先想辦法繞掉 DEP 再去跑 shellcode
    - 因為在 windows 中，純 ROP 不像 linux 中可以直接有方便的 API 置換掉 token，所以如果可以跑 shell code 會讓利用簡單很多

DEVCORE SECURITY CONSULTING

Exploitation of SMBGhost

How can we get code execution from arbitrary memory writing in the past

- Abusing the HalpInterruptController
  - 在 Windows 10 1703 之前，HAL heap 位置固定在 0xfffffffffd00000
  - SMBGhost 在 1903 及 1909 ，HAL 位置無法預測

Outline

- Introduction
- Vulnerability - CVE-2020-0796
- Exploitation of SMBGhost
    - From crash to arbitrary memory writing
    - How can we get code execution from arbitrary memory writing in the past
    - Method 1 - System root hijack (need some condition)
    - Method 2 - Abusing MDL

Exploitation of SMBGhost

System Root hijack

- 在有 SMBGhost 任意寫入後，因為是直接走 tcp ，無法在 target 機器上，有任何的互動，我們也沒有當前目標的任何位置，變成有任意寫入卻不知道可寫哪
- Win 10 後絕大多數的記憶體位置都會有 KASLR ，因此無法參考 EternalBlue 直接寫 HAL 位置 (Win10 前都固定位置)

Exploitation of SMBGhost

System Root hijack

- 在搜尋 memory 後，發現 _KUSER_SHARED_DATA 永遠都會在 0xfffff78000000000 ，從古至今都沒變過，且都是可寫
- 大多數都是用來計算時間，取得系統資訊，User space 會 mapping 在 0x7ffe0000，使得 user 取一些資訊時可以不用透過 system call
- 上面不少系統資訊
  - SystemTime
  - Syscall
    - 以前會把 syscall instruction 放這邊，類似 linux 中的 vdso
  - NtSystemRoot

Exploitation of SMBGhost

System Root hijack

```
struct _KUSER_SHARED_DATA
{
    ULONG TickCountLowDeprecated;                                    //0x0
    ULONG TickCountMultiplier;                                       //0x4
    volatile struct _KSYSTEM_TIME InterruptTime;                     //0x8
    volatile struct _KSYSTEM_TIME SystemTime;                        //0x14
    volatile struct _KSYSTEM_TIME TimeZoneBias;                      //0x20
    USHORT ImageNumberLow;                                           //0x2c
    USHORT ImageNumberHigh;                                          //0x2e
    WCHAR NtSystemRoot[260];                                         //0x30
    ULONG MaxStackTraceDepth;                                        //0x238
    ULONG CryptoExponent;                                            //0x23c
    ULONG TimeZoneId;                                                //0x240
    ULONG LargePageMinimum;                                          //0x244
    ULONG AitSamplingValue;                                          //0x248
    ULONG AppCompatFlag;                                             //0x24c
```

Exploitation of SMBGhost

System Root hijack

- NtSystemRoot
  - Used by ntdll!LdrpLoadDLL
    - 裡面會使用 ntdll!RtlGetNtSystemRoot 取得 dll 路徑，這個 api 就是直接從 _KUSER_SHARED_DATA 取值
    - 原本路徑為 (C:\Windows\)
    - 因此只要將該路徑改掉，就可以做 DLL Hijacking
      - 就算是 KnownDlls 也全都可以 Hijack

Exploitation of SMBGhost

System Root hijack

Direction	Typ	Address	Text
Up	p	LdrpBuildSystem32FileName+1A	call RtlGetNtSystemRoot
D...	p	RtlpAssemblyStorageMapResolutionDefaultCallback+5D	call RtlGetNtSystemRoot
D...	p	IsOverlaySupportedPath+2B	call RtlGetNtSystemRoot
D...	p	RtlpDiskSpeedInitialize+C	call RtlGetNtSystemRoot
D...	p	RtlQueryResourcePolicy+9F30F	call RtlGetNtSystemRoot
D...	p	RtlQueryResourcePolicy:loc_1800A8E43	call RtlGetNtSystemRoot
D...	p	LdrpGetModuleName:loc_1800BDD21	call RtlGetNtSystemRoot
D...	p	CsrpLocalSetupForSecureProcess+3A	call RtlGetNtSystemRoot
D...	p	CsrpLocalSetupForSecureProcess+FA	call RtlGetNtSystemRoot
D...	p	CsrpLocalSetupForSecureProcess+125	call RtlGetNtSystemRoot
D...	p	LdrpInitializeProcess:loc_1800D32FD	call RtlGetNtSystemRoot
D...	p	LdrpInitializeProcess+ED2	call RtlGetNtSystemRoot
D...	p	AVrfpLoadAndInitializeProvider+B7	call RtlGetNtSystemRoot
D...	o	.rdata:0000000180119B86	dd rva RtlGetNtSystemRoot
D...	o	.rdata:000000018013E35C	RUNTIME_FUNCTION <rva RtlGetNtSystemRoot, rva byte_1
D...	o	.rdata:off_18014C528	dd rva RtlDispatchAPC, rva A_SHAFinal, rva A_SHAInit, rva
D...	o	.pdata:000000018016C398	RUNTIME_FUNCTION <rva RtlGetNtSystemRoot, rva byte_1

Exploitation of SMBGhost

System Root hijack

- NtSystemRoot
  - C:\Windows\ -> \??\UNC\{IP}\Windows\
    - Sechost.dll
    - Hijack svchost.exe (User : System)
      - Svchost 在 createthread 時會去 loaddll ，有時需等一下
- 如果有 kernel 任意寫入，亦可用於提權，Low integrity 也適用

Exploitation of SMBGhost

System Root hijack

11:45:...	MsMpEng.exe	2916	ReadFile	\\10.211.55.31\Windows\System32\sechost.dll	SUCCESS	Offset: 503,808, Length: 32,768, I/O Flags: Non-cache...
11:45:...	MsMpEng.exe	2916	ReadFile	\\10.211.55.31\Windows\System32\sechost.dll	SUCCESS	Offset: 393,216, Length: 32,768, I/O Flags: Non-cache...
11:45:...	csrss.exe	552	CreateFile	\\10.211.55.31\Windows\SYSTEM32\gdi32full.dll	ACCESS DENIED	Desired Access: Read Attributes, Disposition: Open, Op...
11:45:28.4951564 PM	exe	2916	ReadFile	\\10.211.55.31\Windows\System32\sechost.dll	SUCCESS	Offset: 569,344, Length: 4,096, I/O Flags: Non-cached...
11:45:...	MsMpEng.exe	2916	ReadFile	\\10.211.55.31\Windows\System32\sechost.dll	SUCCESS	Offset: 73,728, Length: 32,768, I/O Flags: Non-cached...
11:45:...	csrss.exe	552	CreateFile	\\10.211.55.31\Windows\SYSTEM32\gdi32full.dll	ACCESS DENIED	Desired Access: Read Attributes, Disposition: Open, Op...
11:45:...	csrss.exe	552	CreateFile	\\10.211.55.31\Windows\SYSTEM32\gdi32full.dll	ACCESS DENIED	Desired Access: Read Attributes, Disposition: Open, Op...
11:45:...	MsMpEng.exe	2916	CreateFileMapp...	\\10.211.55.31\Windows\System32\en-US\sechost.dll.mui	FILE LOCKED WITH ONLY READERS	SyncType: SyncTypeCreateSection, PageProtection: P...
11:45:...	MsMpEng.exe	2916	QueryStandardI...	\\10.211.55.31\Windows\System32\en-US\sechost.dll.mui	SUCCESS	AllocationSize: 4,096, EndOfFile: 2,560, NumberOfLink...
11:45:...	MsMpEng.exe	2916	ReadFile	\\10.211.55.31\Windows\System32\en-US\sechost.dll.mui	SUCCESS	Offset: 0, Length: 2,560, I/O Flags: Non-cached, Pagin...
11:45:...	csrss.exe	552	CreateFile	\\10.211.55.31\Windows\SYSTEM32\gdi32full.dll	ACCESS DENIED	Desired Access: Read Attributes, Disposition: Open, Op...
11:45:...	csrss.exe	552	CreateFile	\\10.211.55.31\Windows\SYSTEM32\gdi32full.dll	ACCESS DENIED	Desired Access: Read Attributes, Disposition: Open, Op...
11:45:...	csrss.exe	552	CreateFile	\\10.211.55.31\Windows\SYSTEM32\gdi32full.dll	ACCESS DENIED	Desired Access: Read Attributes, Disposition: Open, Op...
11:45:...	csrss.exe	552	CreateFile	\\10.211.55.31\Windows\SYSTEM32\gdi32full.dll	ACCESS DENIED	Desired Access: Read Attributes, Disposition: Open, Op...
11:45:...	csrss.exe	552	CreateFile	\\10.211.55.31\Windows\SYSTEM32\gdi32full.dll	ACCESS DENIED	Desired Access: Read Attributes, Disposition: Open, Op...

Exploitation of SMBGhost

System Root hijack

- NtSystemRoot
  - Problem
    - 走 UNC 的話需要 target access 過任何 unc 一次，讓 unc driver 做初始化
    - 有機會造成 BSOD 原因是因為會讓 csrss.exe load dll 失敗，因為 csrss 只會 load 有微軟簽章過的 dll ，只要 load 失敗就會造成該 process 掛掉 ，就會造成 BSOD
    - 需要一 hijack 到 svchost 就馬上將 path 改回，但這之前不能先讓 csrss load dll
    - 在 SMBGhost 的例子中不夠通用😢，且易失敗

Outline

- Introduction
- Vulnerability - CVE-2020-0796
- Exploitation of SMBGhost
  - From crash to arbitrary memory writing
  - How can we get code execution from arbitrary memory writing in the past
  - Method 1 - System root hijack (need some condition)
  - Method 2 - Abusing MDL

Exploitation of SMBGhost
Abusing MDL

- Ricerca Security
  - @hugeh0ge @_N4NU_
  - The first stable exploit
    - Only write a blog
    - Did not release exploit

Exploitation of SMBGhost

Abusing MDL

- 在 _KUSER_SHARED_DATA 尾端空白處構造 MDL 結構
- SMB 在做 Response 時會用 SrvAllocStruct 結尾的 MDL ptr 所描述的記憶體區塊來作為 Response buffer
- tcpip.sys 回傳時，就會透過 DMA 讀取 MDL 中的 physical address 的內容來回傳
  - 如果可以構造 MDL 到就可以達成任意物理記憶體讀取
- 但正常情況下 Response buffer 會是新分配的 buffer

Exploitation of SMBGhost

Abusing MDL

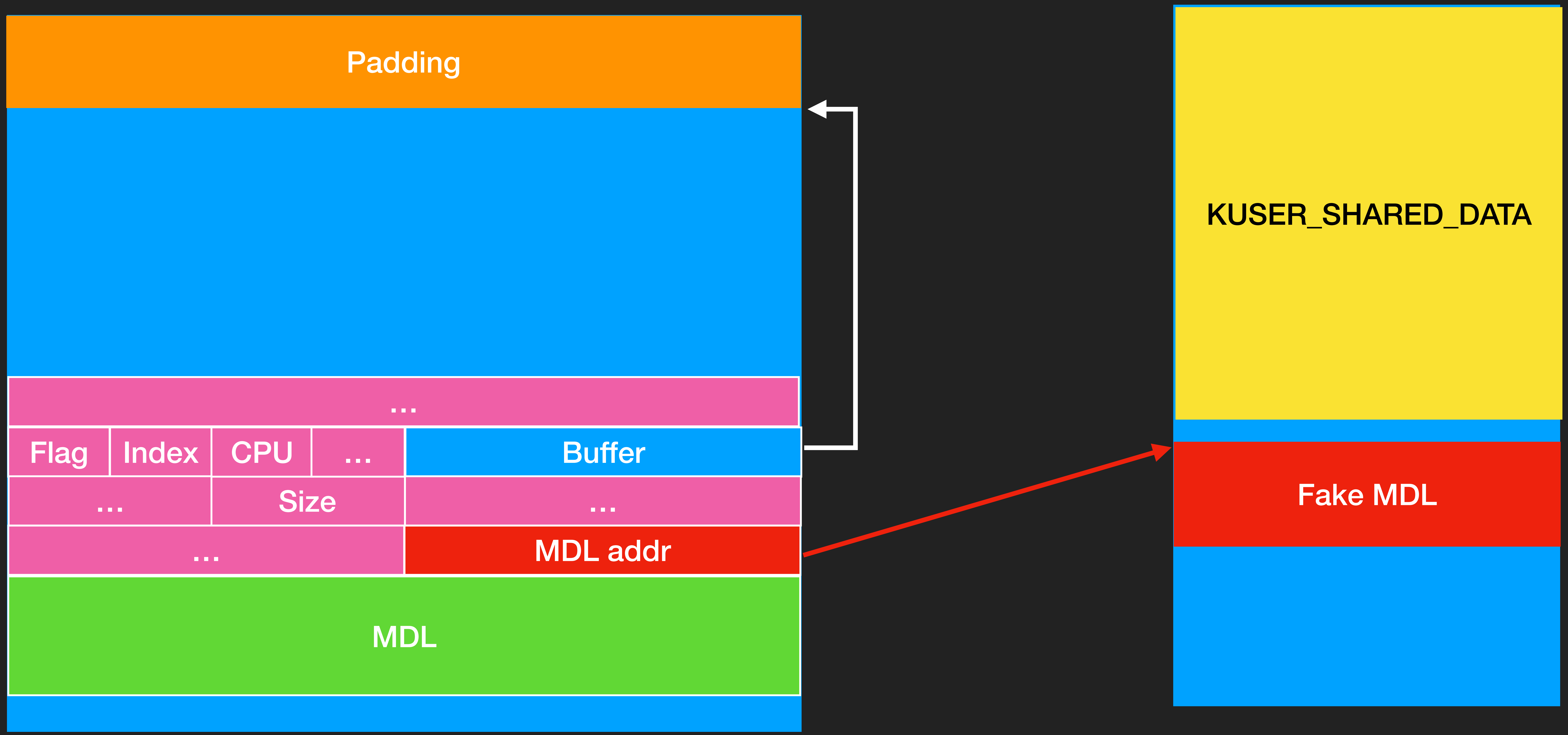
```
Srv2DecompressData(){
    ...
    (ALLOCATION_HEADER)decompress = SrvNetAllocateBuffer(OriginalCompressedSegmentSize + offset);
    ...
    NTSTATIS status = SmbCompressionDecompress(
        CompressionAlgorithm,
        compressdata_ptr,
        decompress->buffer+offset,
        OriginalCompressedSegmentSize,
        &FinalCompressdSize)
    if(status < 0 || FinalCompressedSize != OriginalCompressedSegmentSize){
        SrvNetFreeBuffer(decompress);
        return;
    }
    if(offset > 0){
        memcpy(decompress->buffer,header + sizeof(compress_trans_header),offset);
    }
    Srv2ReplaceReceiveBuffer(smbstruct,decompress);
}
```

Exploitation of SMBGhost

Abusing MDL

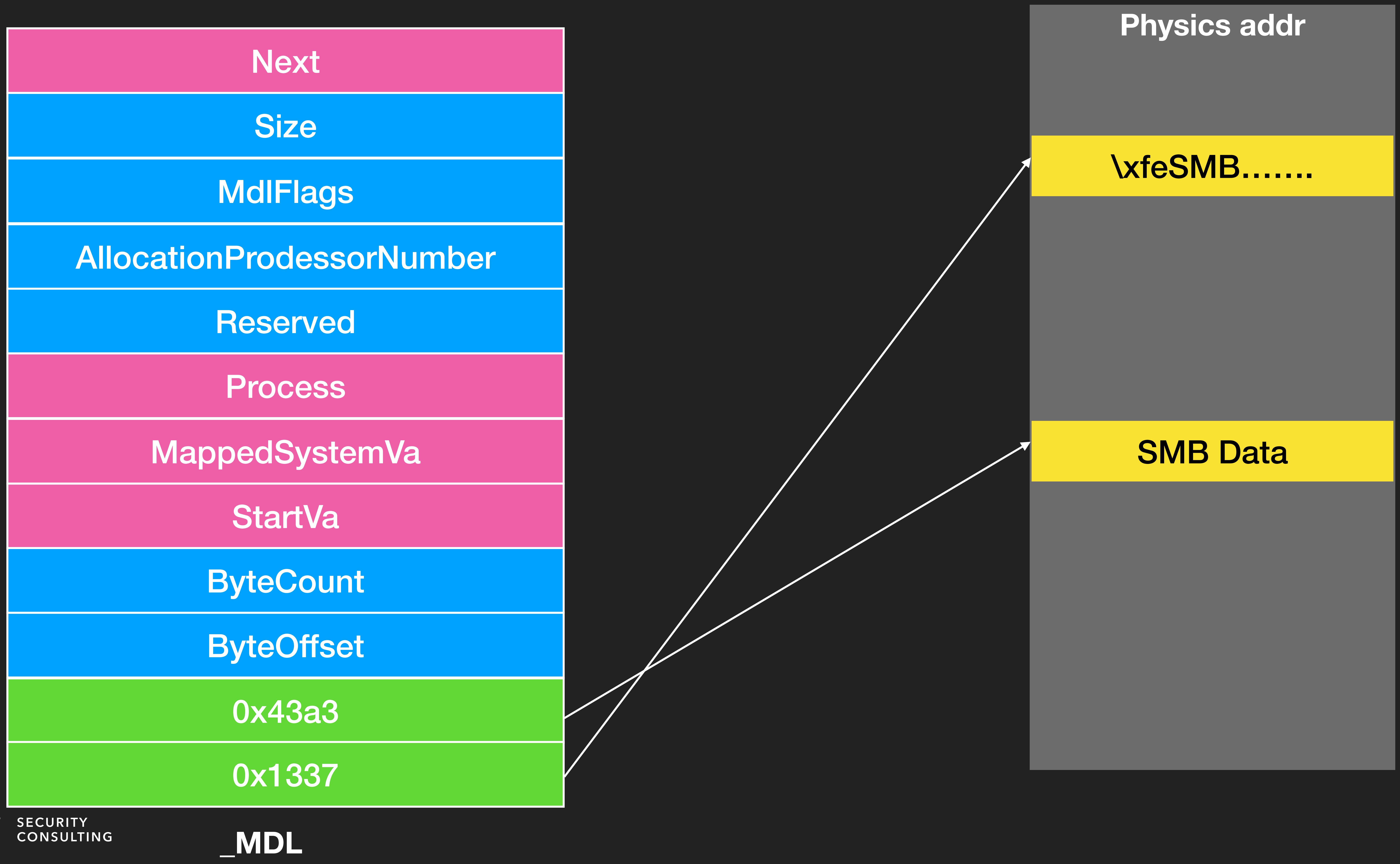
- 如果在 SMB cmd 執行 error 時，會執行到 Smb2SetError
    - 其中會呼叫 Srv2SetResponseBufferToReceiveBuffer
- 也就是說在 decompress 後，Smb 執行 Error 的情況下，下面三個 buffer 會相等
    - Receive buffer
    - Response buffer
    - Decompress buffer

Exploitation of SMBGhost

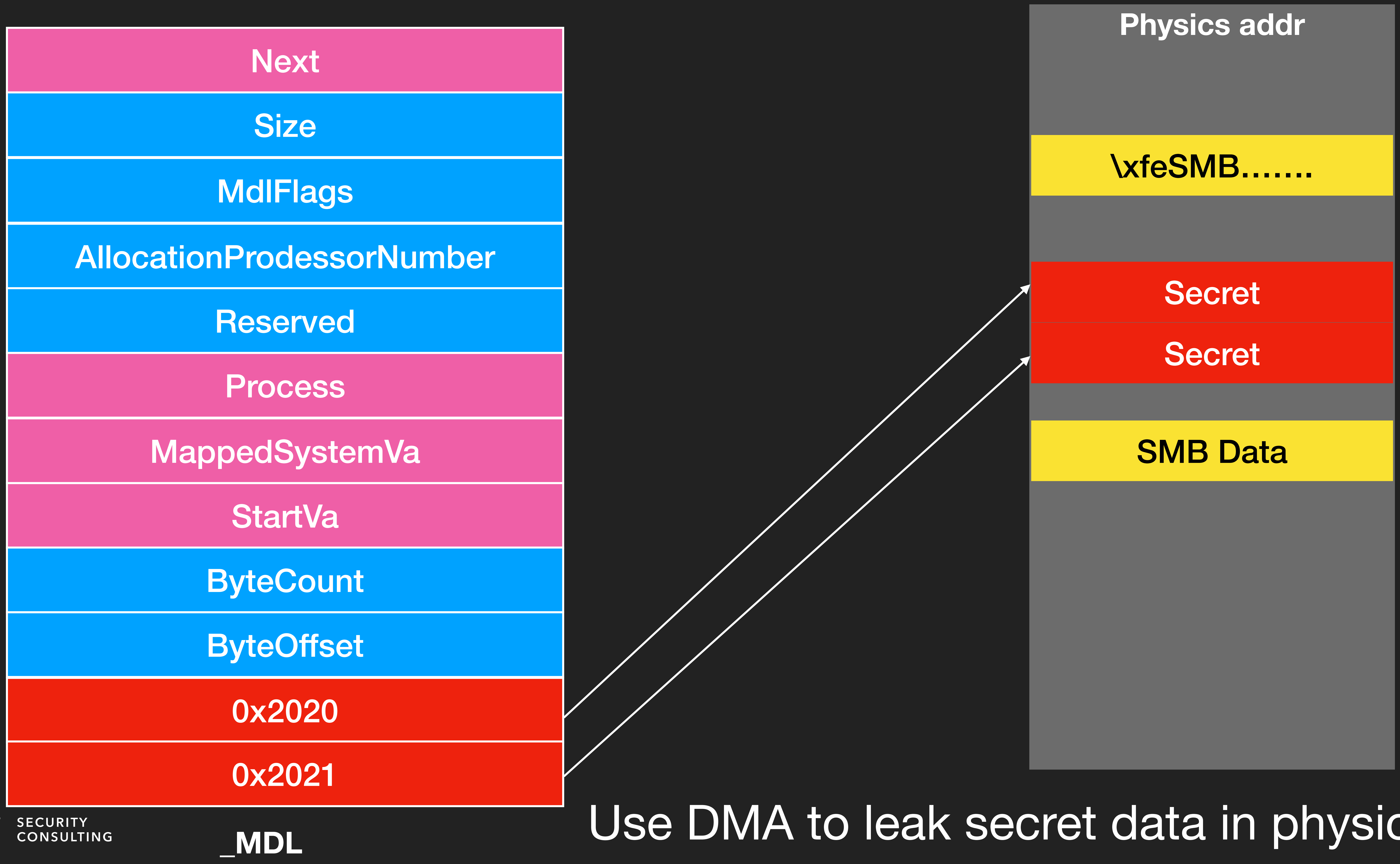


DEVCORE SECURITY CONSULTING

Exploitation of SMBGhost



Exploitation of SMBGhost

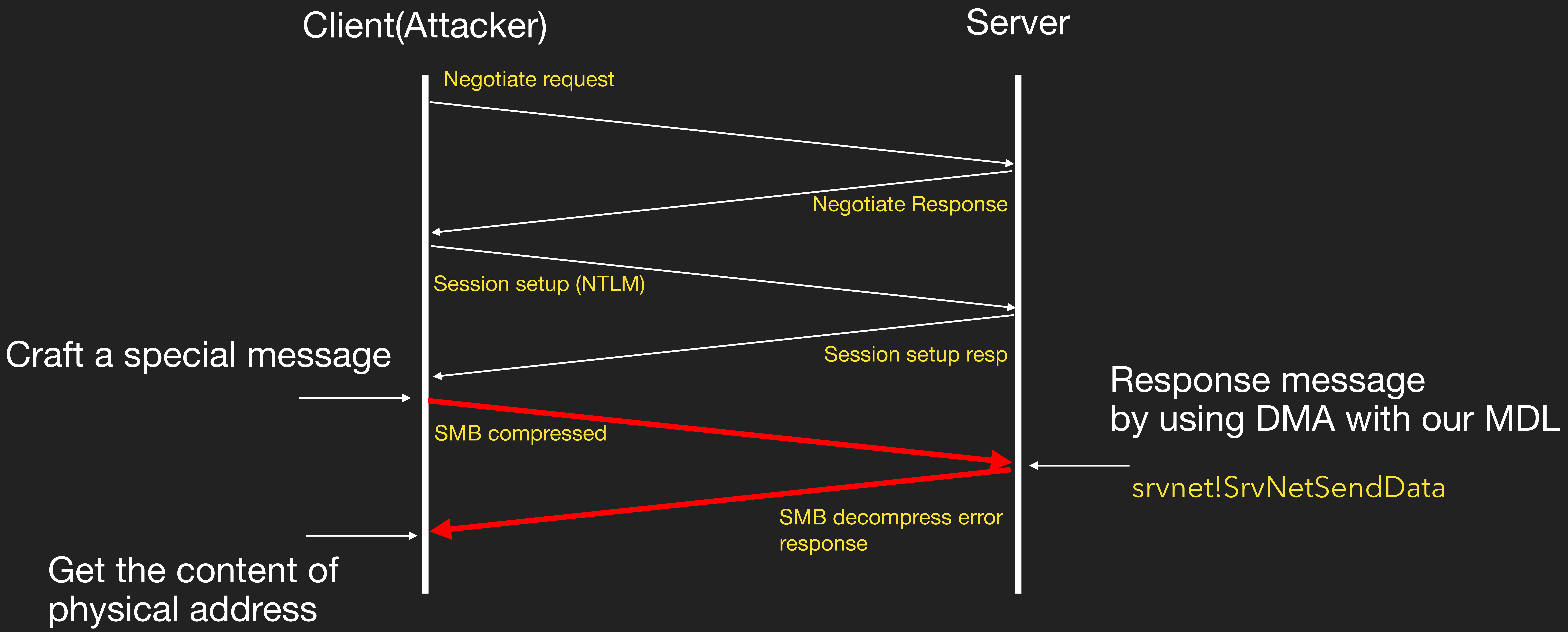


Use DMA to leak secret data in physical memory

DEVCORE SECURITY CONSULTING

Exploitation of SMBGhost

Abusing MDL



Exploitation of SMBGhost

Abusing MDL

- 但在某些環境下 DMA 會失效
- SrvNetSendData
  - Smb 最後用來傳遞封包的 function，會預先處理回傳的封包 buffer，最後會通過 tcpip 如果有使用 MDL 則會採用 MDL 所描述的記憶體來傳輸
  - 我們可以構造特別的 MDL ，事先將 physical 事先用 double-mapping 方式 mapping 到 Virtual Address 中，之後 Response 時就會用該 system buffer 的 data

DEVCORE SECURITY CONSULTING

Exploitation of SMBGhost

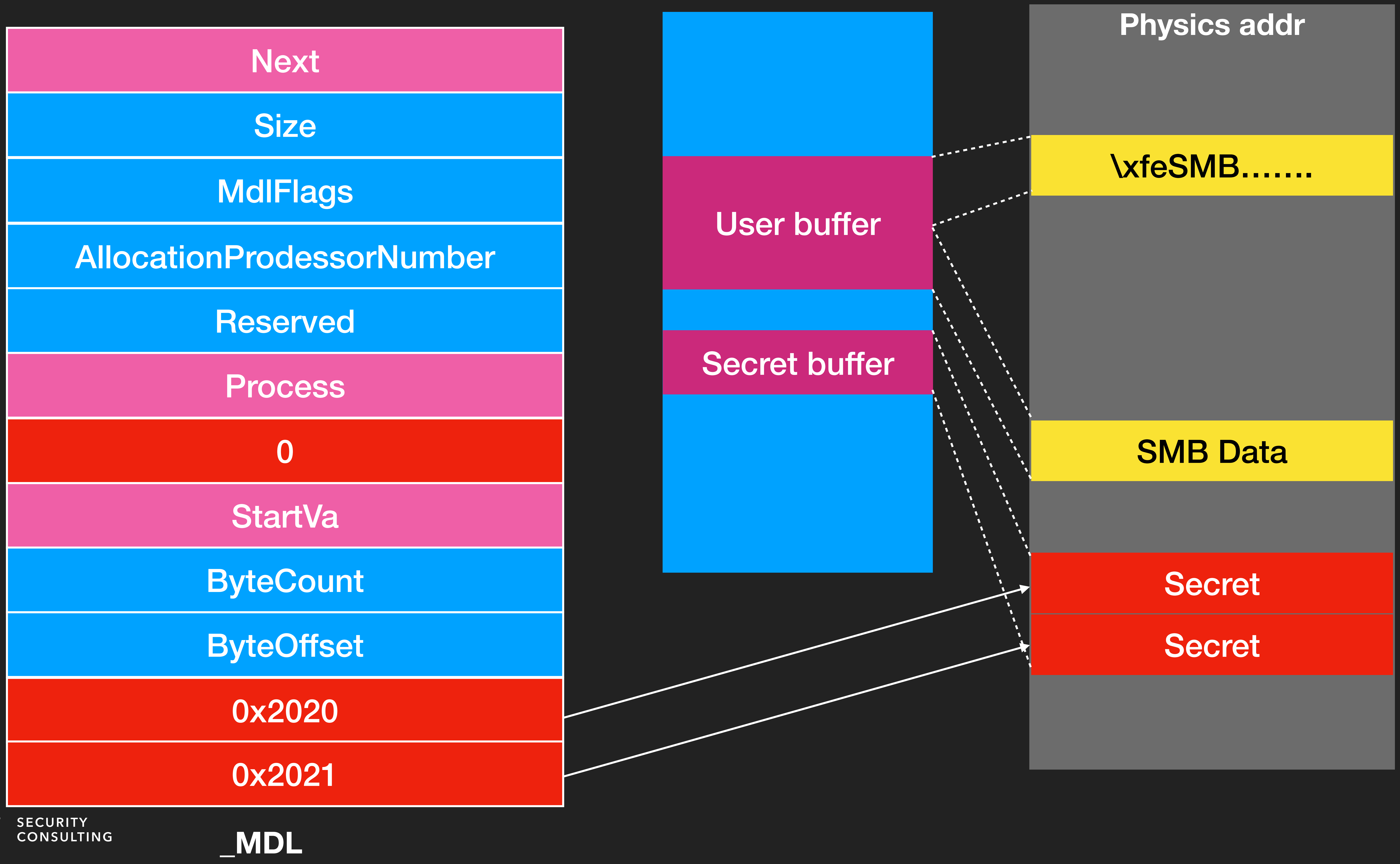
Abusing MDL

- SrvNetSendData

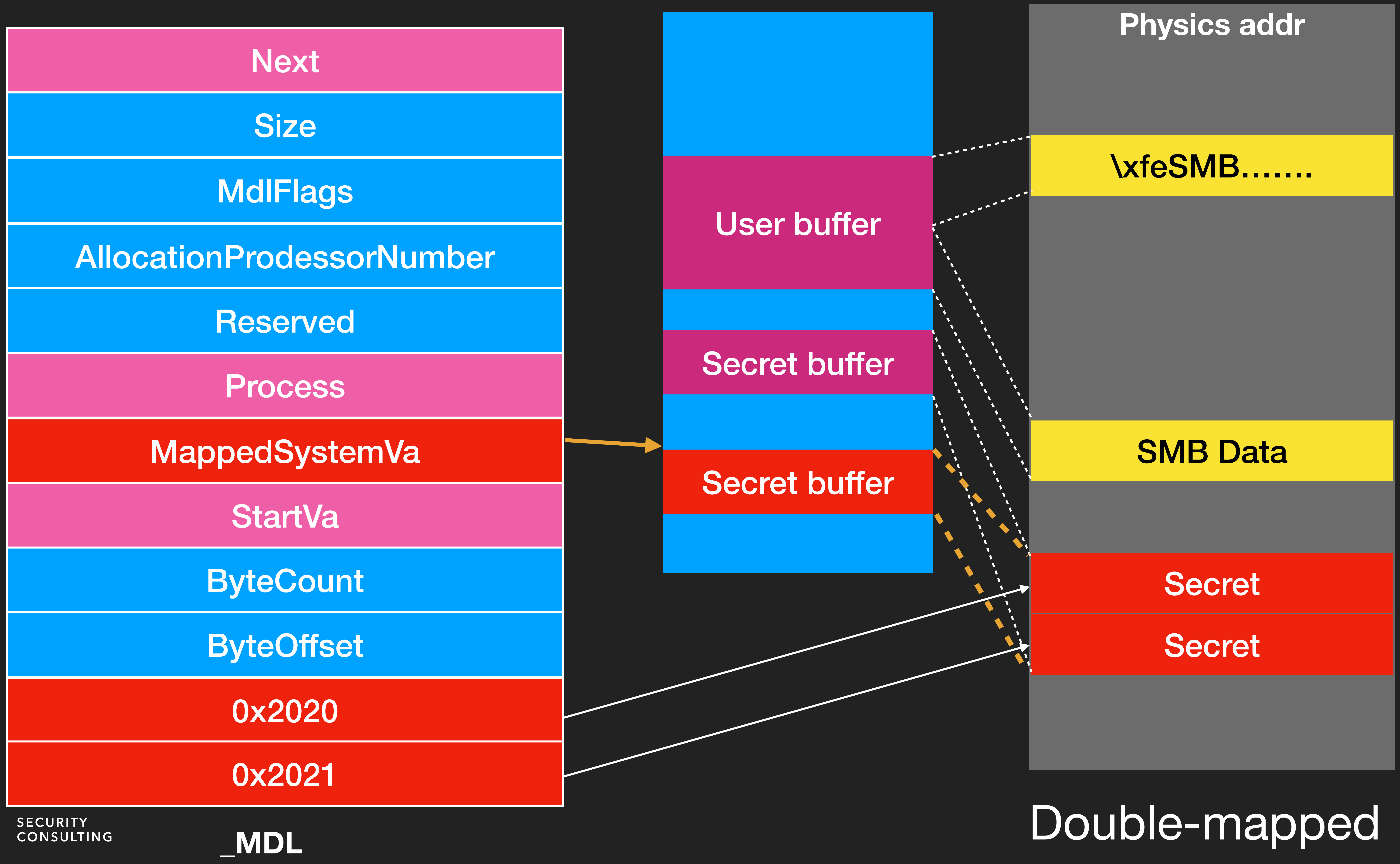
```
if ( v9 == 1 )
{
  if ( LODWORD(v6[5].mdl) )                         // compress
  {
    v18 = v6->mdl;
    v19 = (v18->MdlFlags & 5) != 0 ? v18->MappedSystemVa : MmMapLockedPagesSpecifyCache(
                                                          v18,
                                                          0,
                                                          MmCached,
                                                          0i64,
                                                          0,
                                                          NormalPagePriority);
    if ( v19[6]                                     // command
                                                    //
      && (int)SrvNetCompressData((unsigned int *)v6) < 0
      && WPP_GLOBAL_Control != (PDEVICE_OBJECT)&WPP_GLOBAL_Control
      && (HIDWORD(WPP_GLOBAL_Control->Timer) & 0x2000) != 0
      && BYTE1(WPP_GLOBAL_Control->Timer) >= 2u )
```

DEVCORE SECURITY CONSULTING

Exploitation of SMBGhost



Exploitation of SMBGhost



DEVCORE SECURITY CONSULTING

Exploitation of SMBGhost

Abusing MDL

- 我們可以從 physical address 0x1000 位置周圍讀取內容，在 windows 10 中是固定 mapping 到 Hal heap
    - 可獲得 HAL Heap 的 Virtual Address
    - 也可獲得 hal.dll 位置
- From Alex Ionescu’s talk Getting Physical with USB Type-C: Windows 10 RAM Forensics and UEFI Attacks

Exploitation of SMBGhost

Abusing MDL

- 接下來可以再次構造 MDL
  - 我們只要可以構造 MDL->MappedSystemVA 就可以讓該 buffer 作為 response buffer
    - Arbitrary virtual memory reading
  - 可以從 hal.dll 獲得 nt 及 HalpInterruptController 位置

Exploitation of SMBGhost

Abusing MDL

Exploitation of SMBGhost

Abusing MDL

- SrvNetSendData
  - 該 function 會初始化成 SMB 封包的格式，如填入 size of SMB header，如果 leak 的位置是無法寫入會造成 Access violation(BSOD)

```
v6->mdl->MappedSystemVa = (char *)v6->mdl->MappedSystemVa - 4;
v6->mdl->ByteOffset -= 4;
v6->mdl->ByteCount += 4;
*(_DWORD *)v6->mdl->MappedSystemVa = _byteswap_ulong(v6->size);
```

Exploitation of SMBGhost

Abusing MDL

- How to solve the access violation problem ?
  - 利用 compress 功能 (SrvNetCompressData)
    - 在 SrvNetSendData 中，如果啟用 compress ，則會分配新的 buffer 來放 mdl->MappedSystemVA ，因此會將要 leak 的 data 做 compress 後回傳，寫入 SMB header 資訊也會使用新的 buffer

```
new_netalloc = SrvNetAllocateBuffer(LODWORD(old->size), 0i64);
v3 = 1;
v7 = TdiCopyMdlToBuffer(old->mdl, 0, new_netalloc->buf, 0, old->size, &BytesCopied);
```

Exploitation of SMBGhost

Bypass DEP

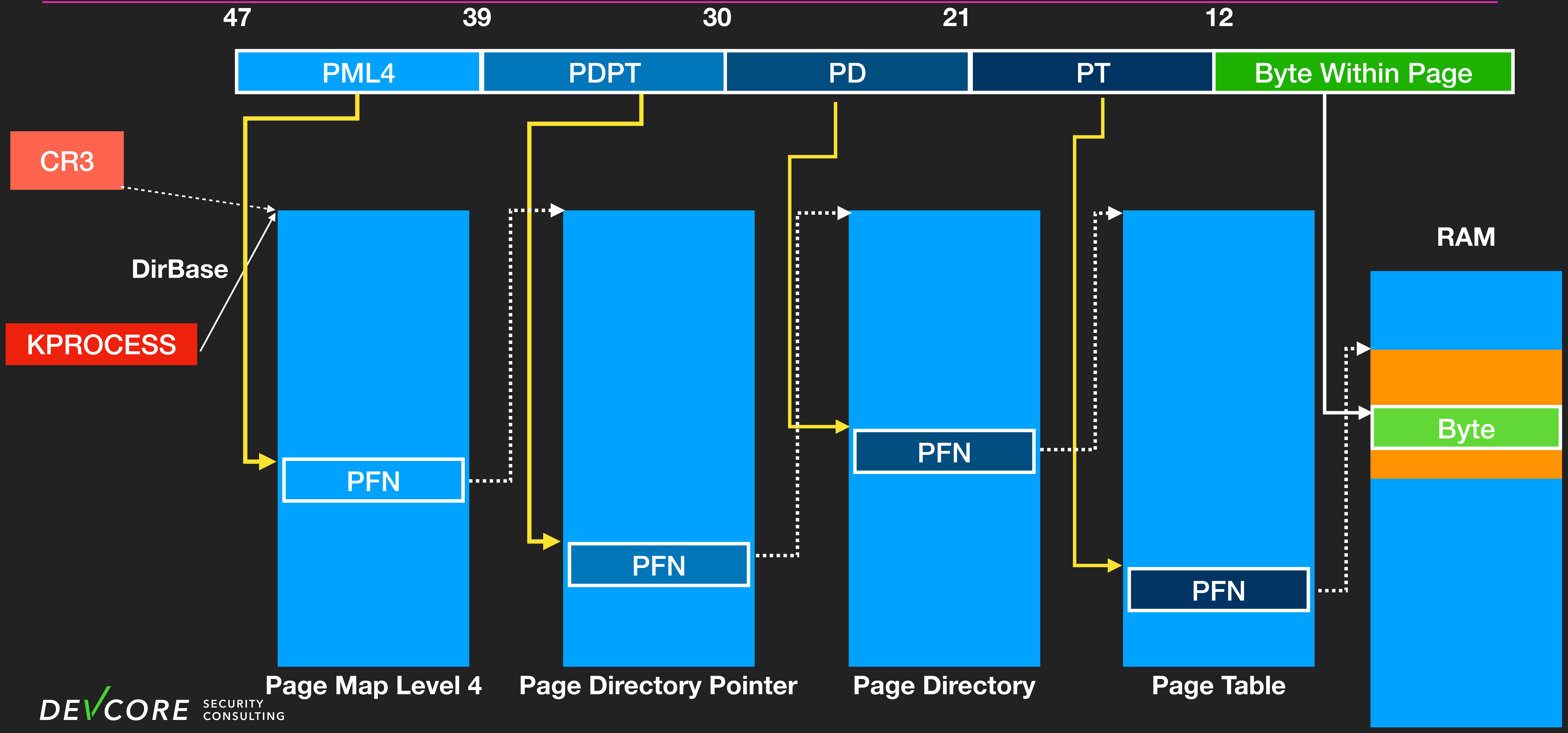
- Overwrite PTE to make some page to ERW
- ExAllocatePool
    - Allocate a NonPagedPool memory

Exploitation of SMBGhost

Bypass DEP

- PTE (Page Table Entry)
  - Each page of virtual address is associated with PTE, with contains the PA to which the virtual one is mapped.
  - Page 權限主要靠 PTE 來決定，如果我們可以修改到 PTE 也就可以任意修改 memory 權限 bypass DEP

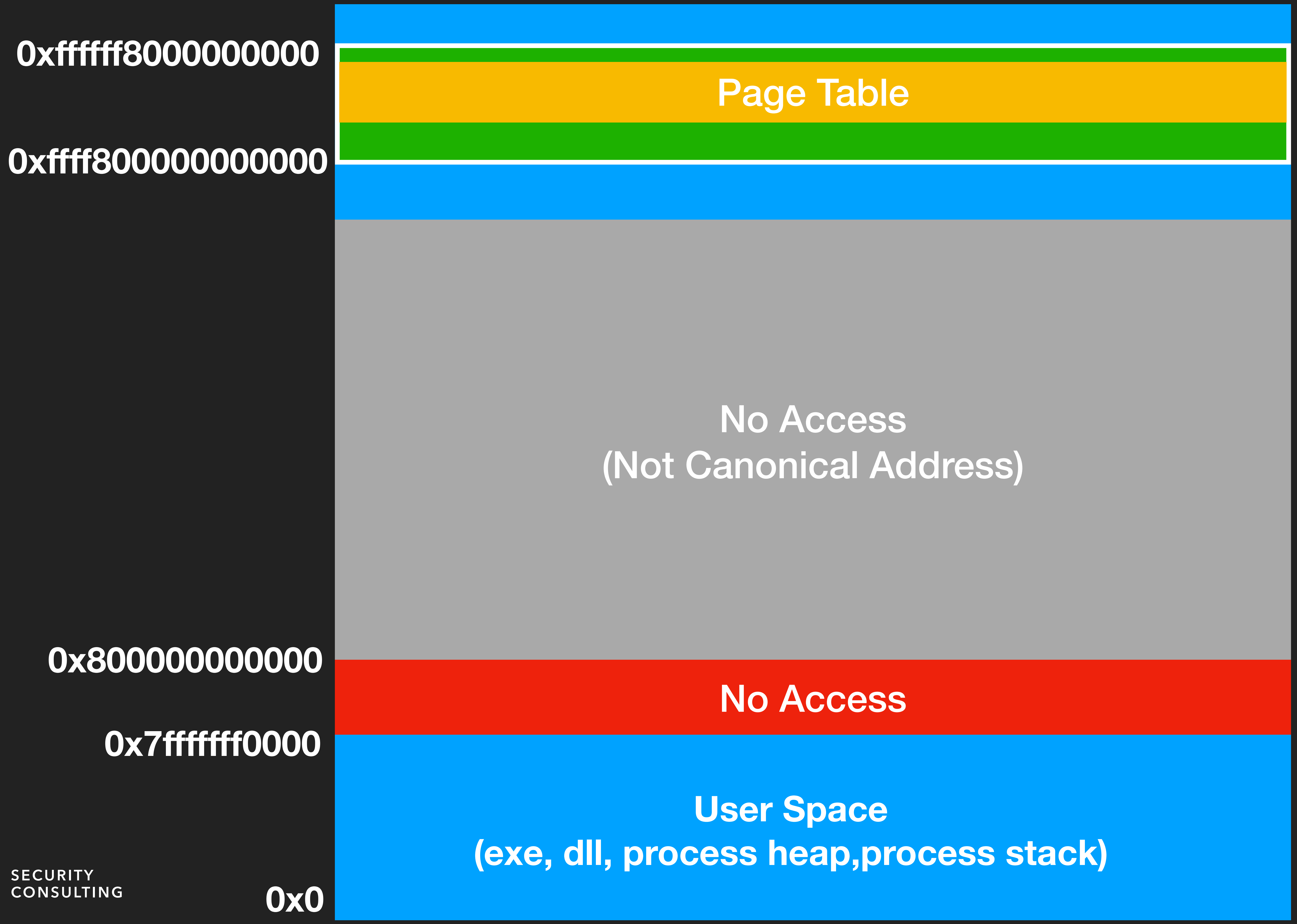
Exploitation of SMBGhost



DEVCORE SECURITY CONSULTING

Exploitation of SMBGhost

Bypass DEP



DEVCORE SECURITY CONSULTING

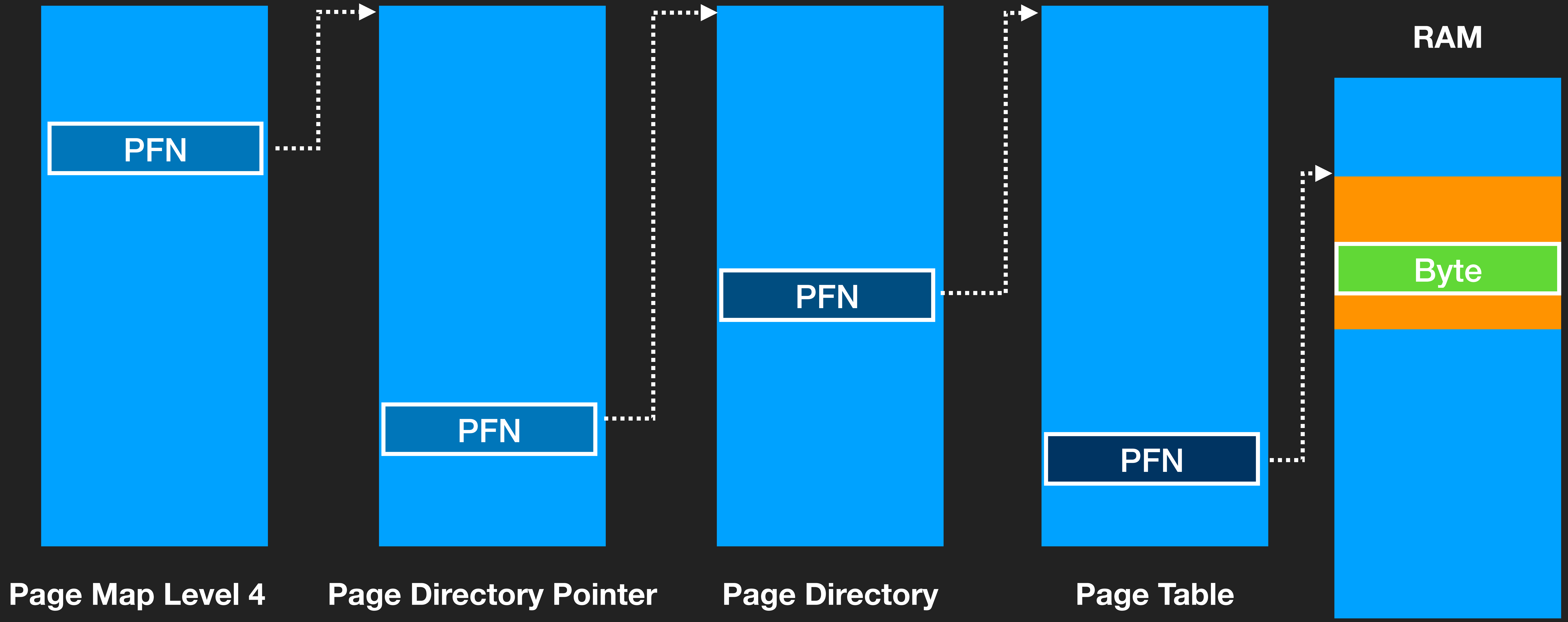
Exploitation of SMBGhost

Bypass DEP

- How windows to manage page table ?
  - Self-ref entry
    - This technique consists of using one entry at the highest paging level by pointing to itself.
    - In 64 bits, this entry is located in the PML4
    - That is, there is a PTE point to PML4 so that system can modify any PTE to manage page table.

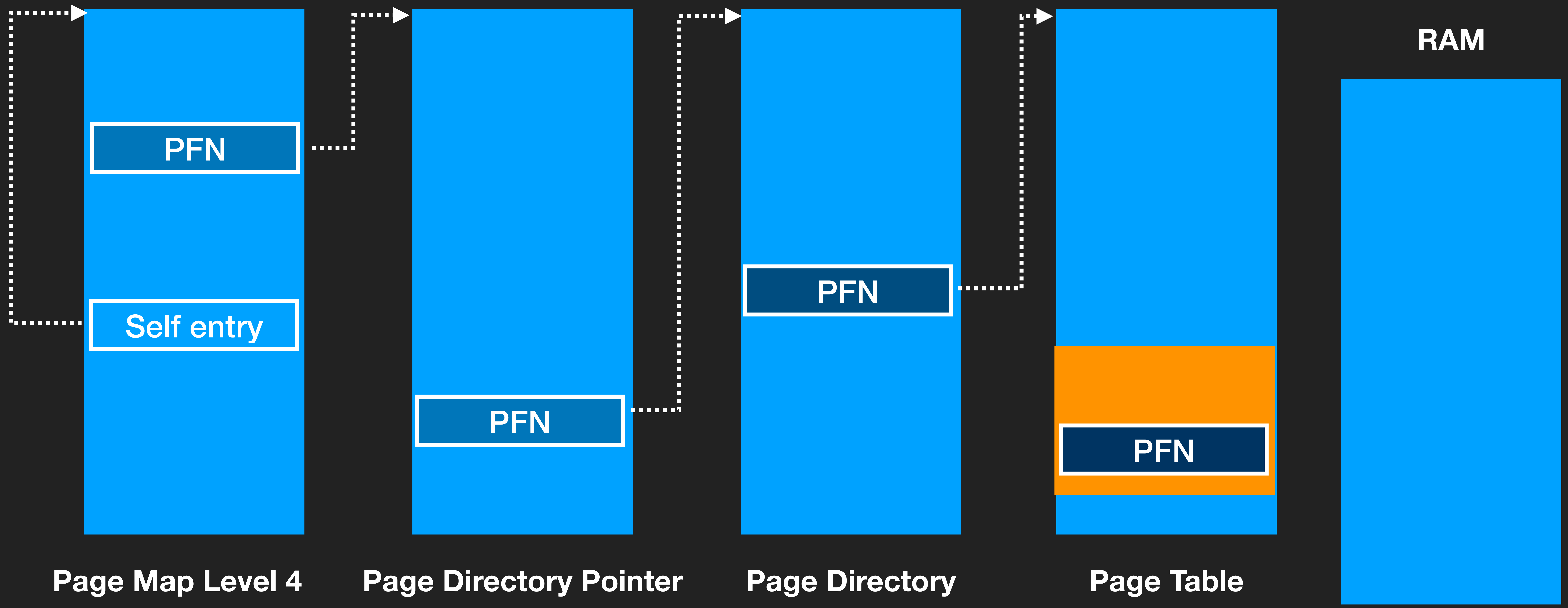
Exploitation of SMBGhost

Bypass DEP



Exploitation of SMBGhost

Bypass DEP



DEVCORE SECURITY CONSULTING

Exploitation of SMBGhost

Bypass DEP

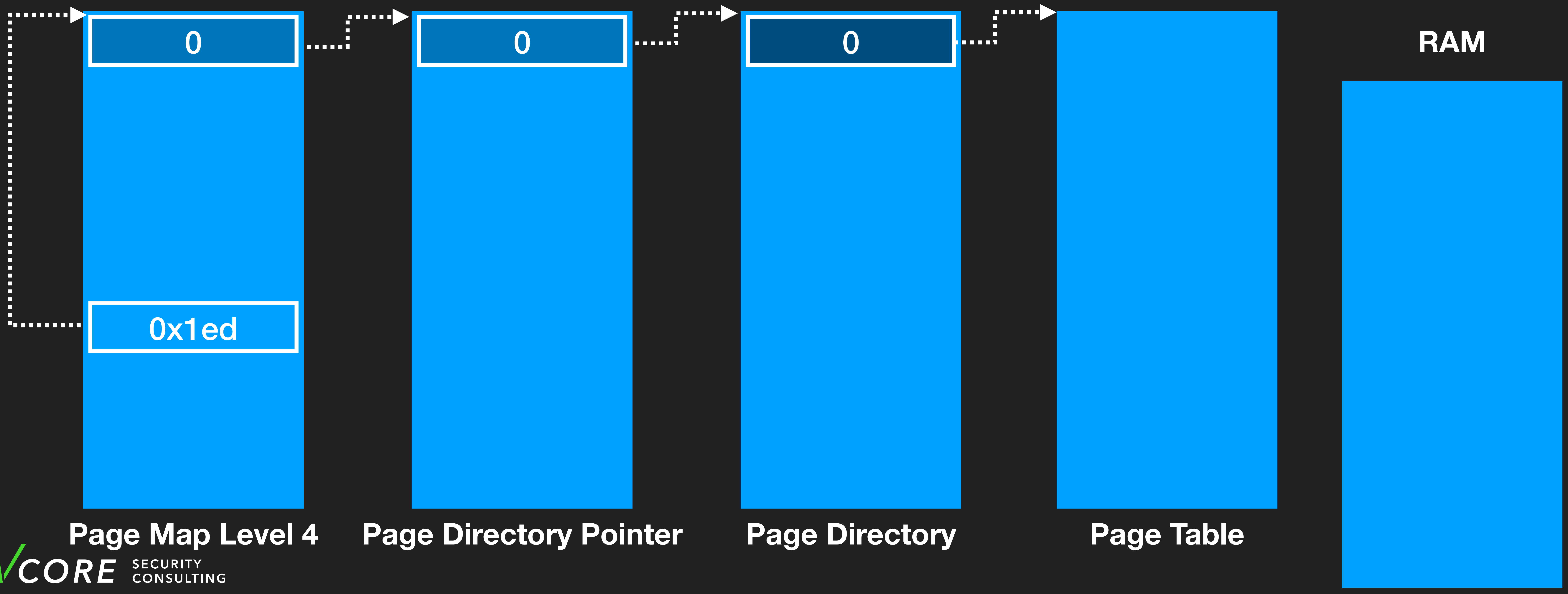
- How to locate PTE of 0
  - Find the Self-ref entry
    - 0xfffff000000000000 + 0x800000000*(self-ref entry)
  - It also the base of Page table

Exploitation of SMBGhost

Bypass DEP

Assume self-ref entry is 0x1ed

0xfffff00000000000 + 0x800000000*0x1ed = 0xffffff6800000000



DEVCORE SECURITY CONSULTING

Exploitation of SMBGhost

Bypass DEP

- 在過去 windows 是以 0x1ed 作為 self-modify entry 因此 Page table 的位置是固定的，導致 attack 可用這特性來輕易修改 page table
    - 在近期的版本中，這數值則是從 0x100-0x1ff 隨機一個數值，每次開機都會不同，如果有任意記憶體讀取，可從 nt!MmPteBase 獲得 Page table 位置
    - 如果有任意 physical address 也可以從 0x1ad000 (PML4 of system process)找出 self-entry 位置，從而推出 Page table 在 virtual address 的位置

Exploitation of SMBGhost

Bypass DEP

- 找到 Sef-ref entry 之後，我們可以算出想改權限 page 的位置的 PTE，將最高一 bit (NX bit) 清除，該 page 就會有執行權限

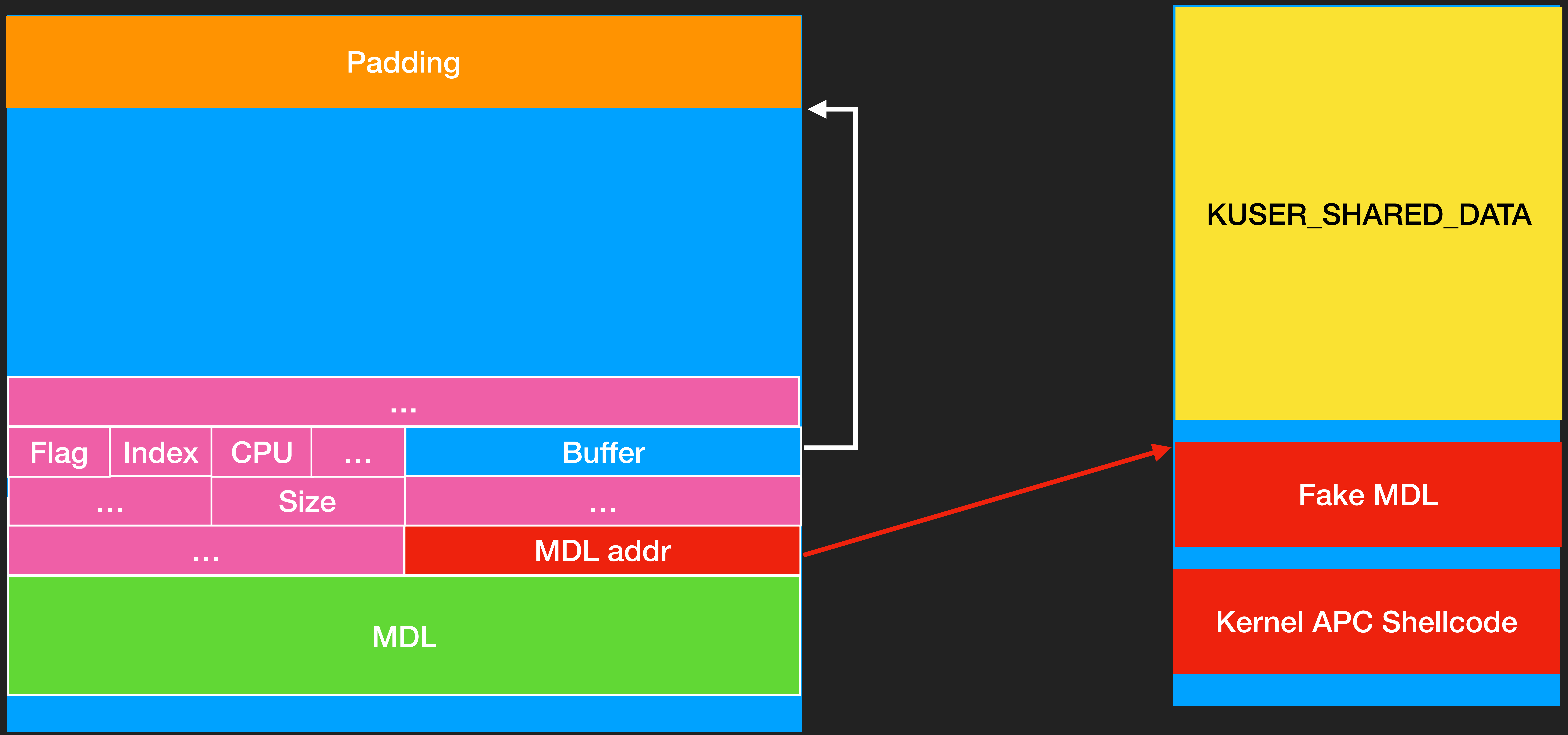
```
PML4_SELF_REF_INDEX = 0x1ed
def get_pxe_address(address):
    entry = PML4_SELF_REF_INDEX;
    result = address >> 9; # >> 12 <<  3 (*8)
    lower_boundary = (0xFFFF << 48) | (entry << 39);
    upper_boundary = ((0xFFFF << 48) | (entry << 39) + 0x8000000000 - 1) & 0xFFFFFFFFFFFFFFF8;
    result = result | lower_boundary;
    result = result & upper_boundary;
    return result
```

Exploitation of SMBGhost

Bypass DEP

- 我們可修改 _KUSER_SHARED_DATA page 權限，改為可讀可寫可執行
- 放 shellcode & recover shellcode 至 KUSER_SHARD_DATA 尾端
  - 覆蓋 HalpApicRequestInterrupt
    - Control RIP !
    - 因為 HalpApicRequestInterrupt 會不斷被呼叫到，需要馬上先還原該 pointer 位置

Exploitation of SMBGhost



Exploitation of SMBGhost

Shellcode

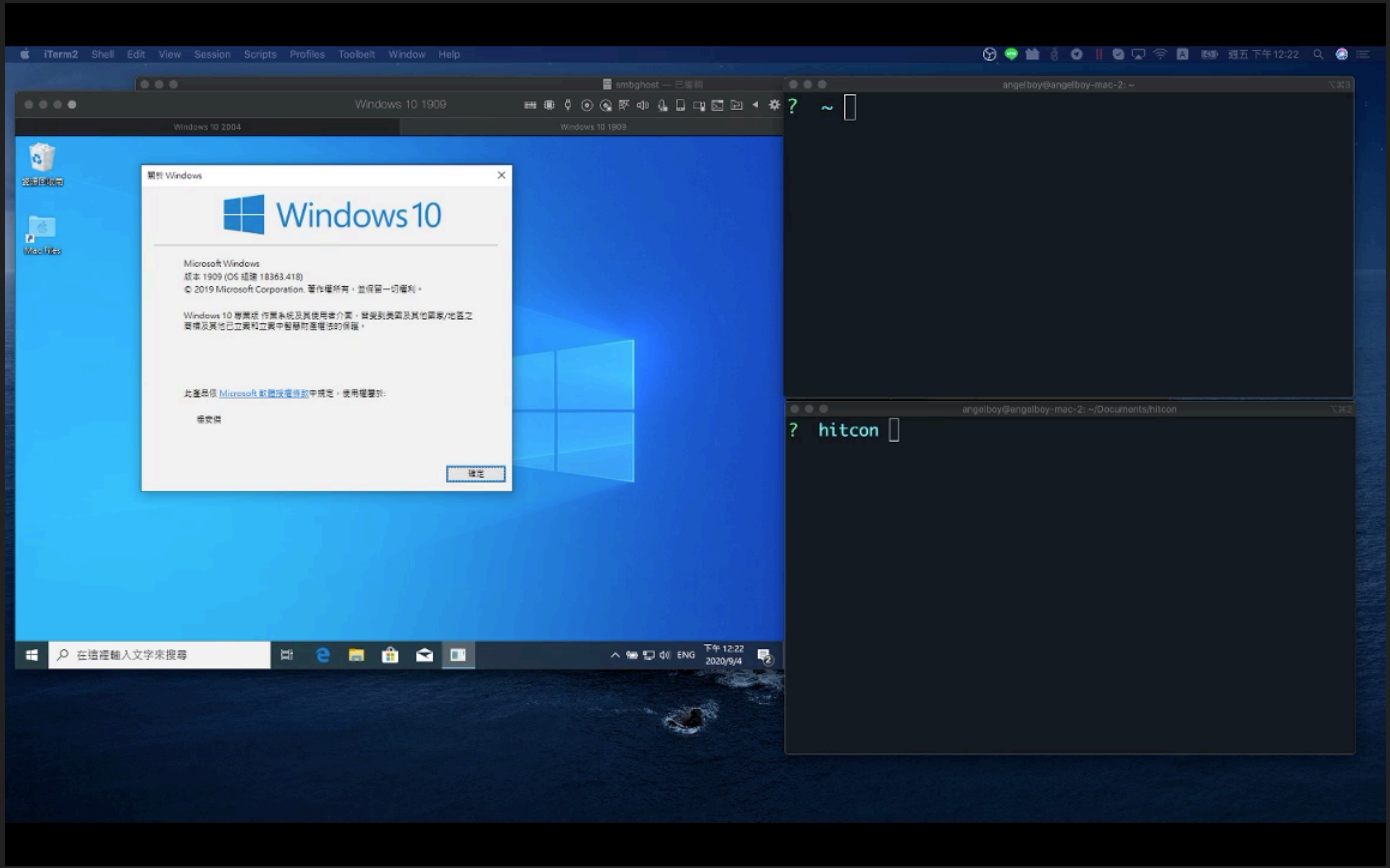
- APC Injection shellcode
  - 我們只有 kernel code execution 沒有 user mode 的 process 的互動
  - 我們此時就可以利用 APC injection 方式，將 user mode APC inject 到高權限的 Process 中
  - 這邊需要特別注意 IRQL，如果沒寫好，會踩到 paged memory

Exploitation of SMBGhost

Shellcode

- KAPC shellcode
  - KAPC
    - Find the target thread (svchost.exe)
    - Allocate execute memory in user-space
    - Copy shellcode to the memory
    - Queue the use-mode APC
  - UAPC
    - Reverse shell

Exploitation of SMBGhost
Demo



DEVCORE SECURITY CONSULTING

Exploitation of SMBGhost

Recover

- 在獲得 shell 後須立刻將 _KUSER_SHARED_DATA page 上的 data 還原
- 因為 PatchGuard 有保護該 page 尾端，如果沒有清空則有可能會 BSOD
  - CRITICAL_STRUCTURE_CORRUPTION

Conclusion

- Although Ntsystem root hijack is not useful in this case, it also a powerful method for EoP.
- It is a very funny ~~backdoor~~ bug, and can learn a lot of knowledge about Windows kernel
    - If you want learn windows kernel exploit, it an excellent case for you.


:(
您的電腦發生問題，因此必須重新啟動。
我們剛剛正在收集某些錯誤資訊，接著我們會為您重新啟動。
已完成 100%
如需此問題與可能修正的詳細資訊，請瀏覽 https://www.windows.com/stopcode
致電支援人員時，請提供此資訊:
停止代碼: CRITICAL PROCESS DIED
已完成 15%
停止代碼: PAGE FAULT

Thanks

- Lucas Leong
  - @_wmliang_

Q & A

Thank you for listening

angelboy@chroot.org

@scwuaptx

Reference

- https://ricercasecurity.blogspot.com/2020/04/ill-ask-your-body-smbghost-pre-auth-rce.html
- https://blog.zecops.com/vulnerabilities/exploiting-smbghost-cve-2020-0796-for-a-local-privilege-escalation-writeup-and-poc/
- https://www.coresecurity.com/blog/getting-physical-extreme-abuse-of-intel-based-paging-systems-part-1
- https://www.coresecurity.com/blog/getting-physical-extreme-abuse-of-intel-based-paging-systems-part-2-windows
- http://www.alex-ionescu.com/
- https://docs.microsoft.com/zh-tw/windows/
- https://www.matteomalvica.com/blog/2019/07/06/windows-kernel-shellcode/#token-stealing