Do My Best!

Windows Kernel Driver & Named Pipe LPE 버그헌팅

Contents

Intro

Team Members

Windows Kernel Driver & Named Pipe LPE 버그헌팅

김태영(PM)

방세연

김명규

이경재

장민기

임한글

멘토 최광준　PL 최호수

OUR TOPIC

Windows Kernel Driver & Named Pipe LPE 버그헌팅

주제 선정 이유?

- Windows는 전 세계 데스크톱 OS 시장의 대부분을 차지하며, 여전히 기업·개인 환경에서 가장 폭넓게 사용
- 따라서 Windows 환경에서 SYSTEM 권한으로 직접 도달할 수 있는 취약점은 공격 효과와 보안 파급력이 극대화

최종 목표

- 전 세계적으로 사용 빈도가 높은 Windows 환경에서 SYSTEM 권한까지 침해 가능한 Kernel Driver를 분석하여, 잠재적 취약점을 식별하고 Proof-of-Concept(POC)을 통해 검증

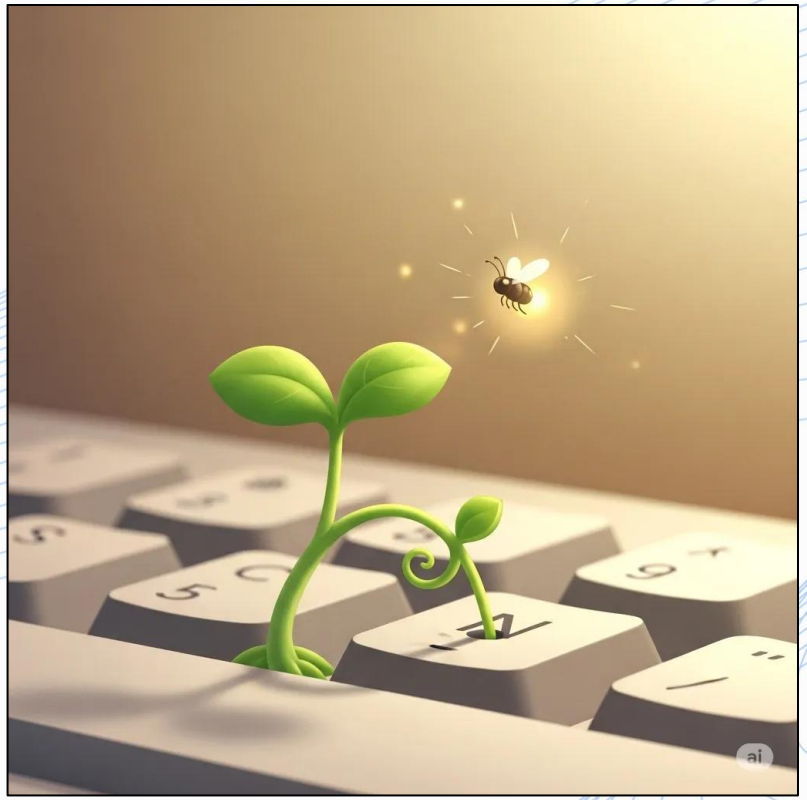
- SYSTEM 권한으로 실행되는 서비스의 Named Pipe를 분석하여, 권한 상승 가능성이 있는 취약한 IPC(Inter-Process Communication)를 식별하고 Proof-of-Concept(POC)을 통해 검증

우리는 어떤 사람이었는가?

1. 커널을 접해보지 못했고, 보안을 시작한지 평균 1,2년된 사람들

2. 기본적인 pwnable 지식만 가지고 무작정 프로젝트에 뛰어든 사람들 (근데 취약점은 찾고싶은..)

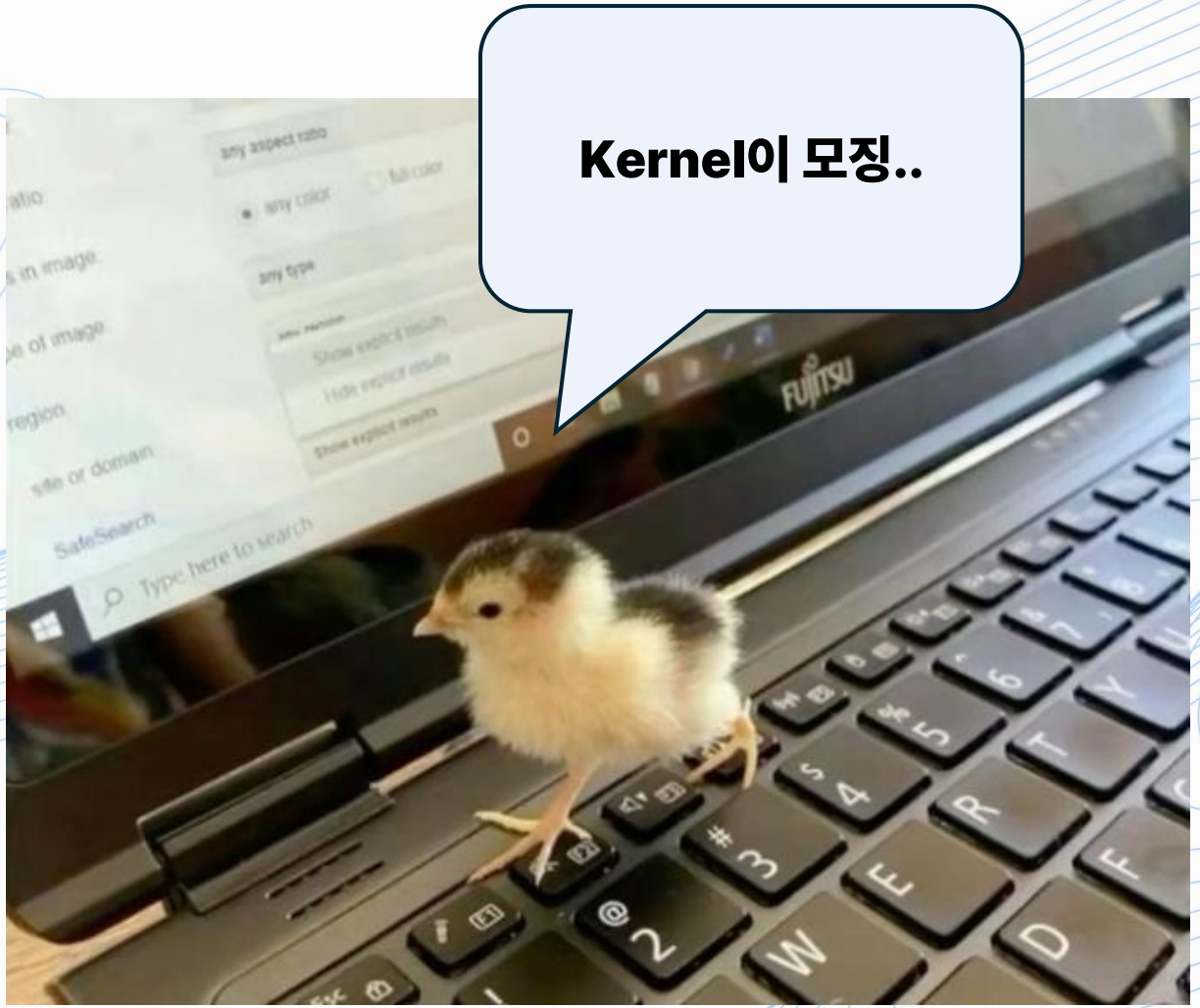
3. 시간 넘치는 사람들 (ㅎㅎ..)



Kernel에 대한 두려움

일단 Windows에 대해서도 제대로 알지 못하는데,
커널은 더더욱 raw한 영역이라고 생각함

→ **그런데 kernel driver를 분석한다? 3개월 동안 공부만 하다가 끝나는건 아닐까.. 하는 걱정**

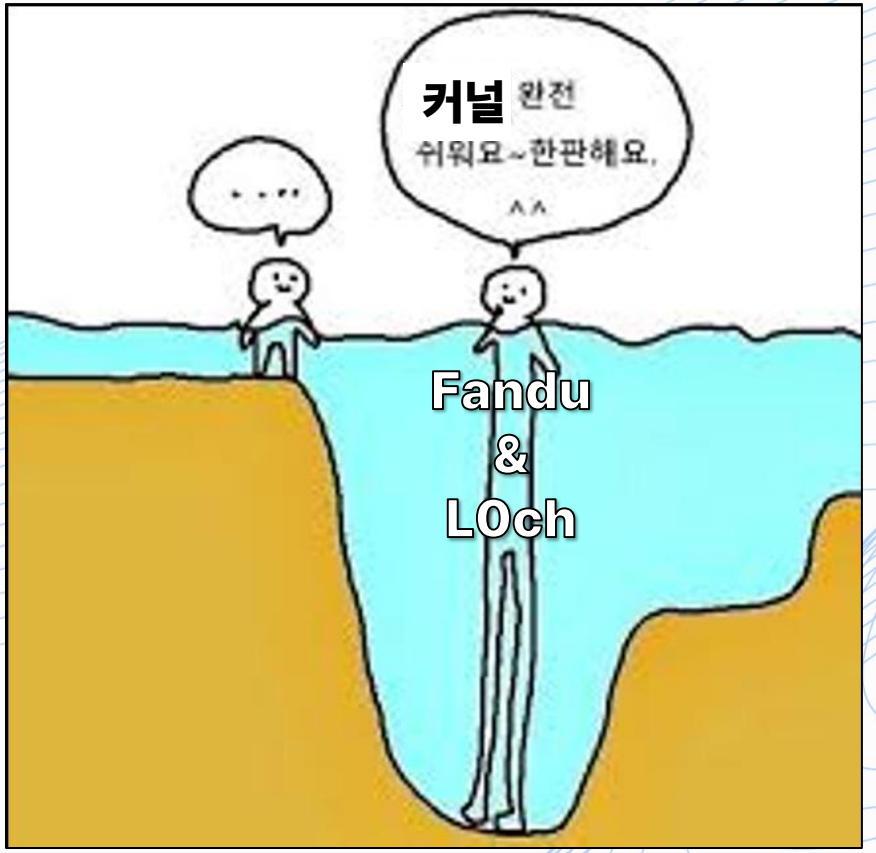


시작하게 된 계기

어차피 찾아야 할 제로데이, 기왕 할거 간지나게 커널 해보자!!

열정 있는 사람들끼리 모였을 때 이런 주제로 또 언제 해보겠어!!

커널 어렵지.. 근데 매일 12시간 씩 공부하면 할 수 있어요 (멘토님 왈)



시작하게 된 계기

하면 된다는 자신감을 얻어보자.
-> 우리도 제로데이 찾을 수 있다!

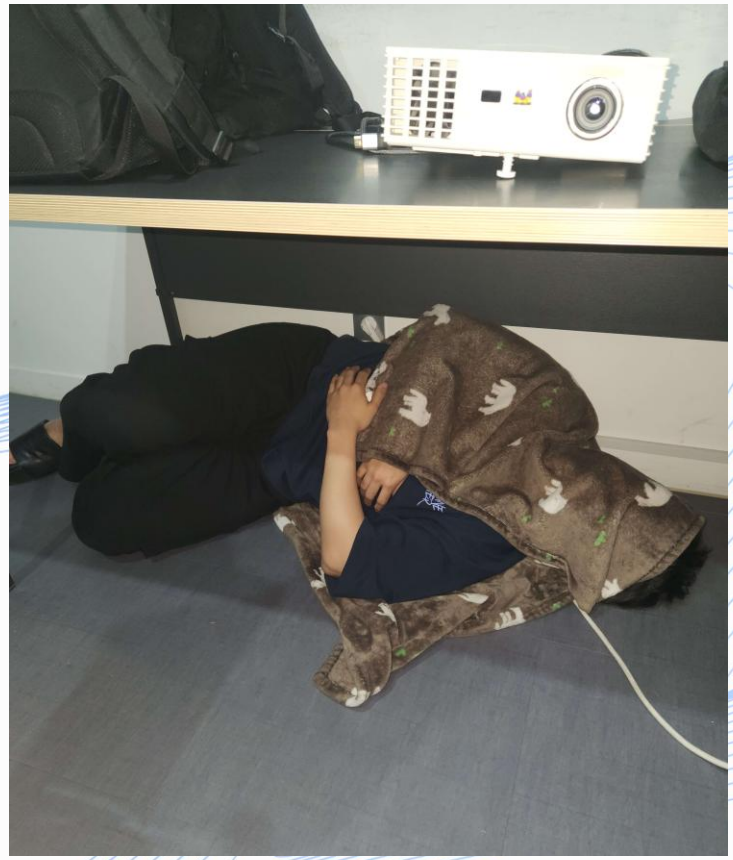
시작하게 된 계기 & 어떤 노력을 했는가?

하면 된다는 자신감을 얻어보자.
-> 우리도 제로데이 찾을 수 있다!

3개월이라는 기간의 Limit...

주 6일 오프라인으로 만나자
매일 **9am to 9pm (12시간)** 함께하자
프로젝트에 모든걸 쏟아보자

Windows LPE를 찾아 떠나는 여정



갈 곳이 없어진 우리들..

Road to LPE

우리의 결의

	김태영(충남 천안)	김명규(서울 은평구)	이경재(경기 일산)	장민기(경기 성남)	방세연(서울 강남구)	임한글(서울 서대문구)
월	12시 이후	Full	Full	Full	14시 이후	16시 이후
화	18시 이후	14시 이전, 16시 이후	16시 이후	X(온라인 가능)	18시 이후	16시 이후
수	18시 이후	18시 이후	18시 이후	Full	Full	Full
목	Full	Full	Full	X(온라인 가능)	X	18시 이후
금	Full	Full	Full	Full	Full	15시 이후
토	Full	Full	Full	Full	Full	Full
일	Full	Full	Full	Full	Full	Full

1. 평균 하루 **12시간**을 목표로
2. 만나면 **9 to 9** 은 기본!
3. 온라인으로 참석 시 디스코드로 참여

우리의 결의

1. 공부한 내용은 **Notion에 기록**
2. **주 1회 회의**를 통해 진행 상황 공유

Background

- Windows WDM 리버스 엔지니어링 방법론 학습
- x64 ABI 규칙 개요
- PML4 Self-Reference Entry
- Paging
- 보호기법
- I/O Ring
- I/O Ring for Exploit
- I/O Ring API Usage
- IRP RequestorMode
- Kernel Streaming with CVE-2024-35250
- TOKEN_PRIVILEGES
- PreviousMode
- Named Pipe

Wargame

- HEVD 문제 풀이

1

WDM 리버싱 및 익스플로잇 실습

Voidsec 방법론 기반 WDM 구조와 취약 커널 드라이버 분석 실습

2

HackSys Extreme Vulnerable Driver

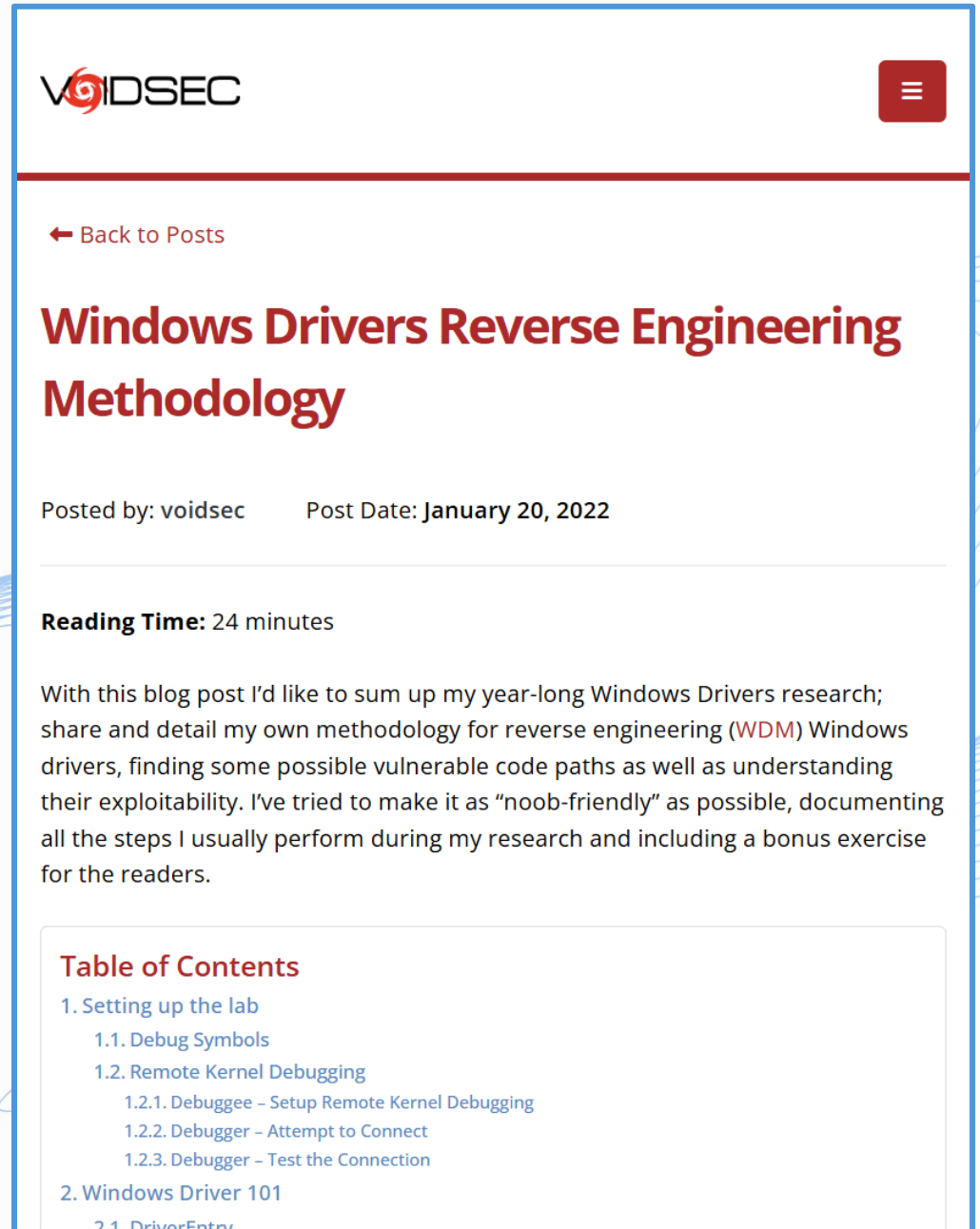
Stack Overflow / Arbitrary Overwrite / Arbitrary Increment

3

실전 1-day 취약점 분석

CVE-2023-20598 CVE-2024-35250

이 3가지를 한 달 안에 끝낸다.



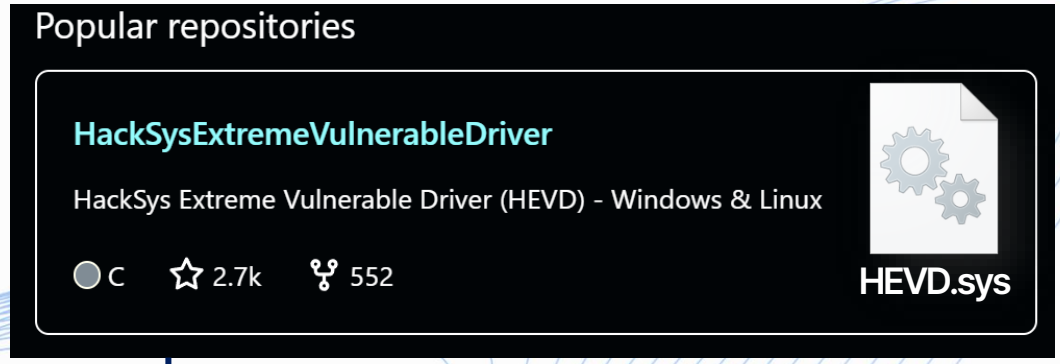
Voidsec

Windows Drivers Reverse Engineering Methodology

- 커널 디버깅 환경 설정
- Windows 드라이버 핵심 이론
- 리버스 엔지니어링 방법론 및 도구 활용
- 취약한 코드 경로 탐지

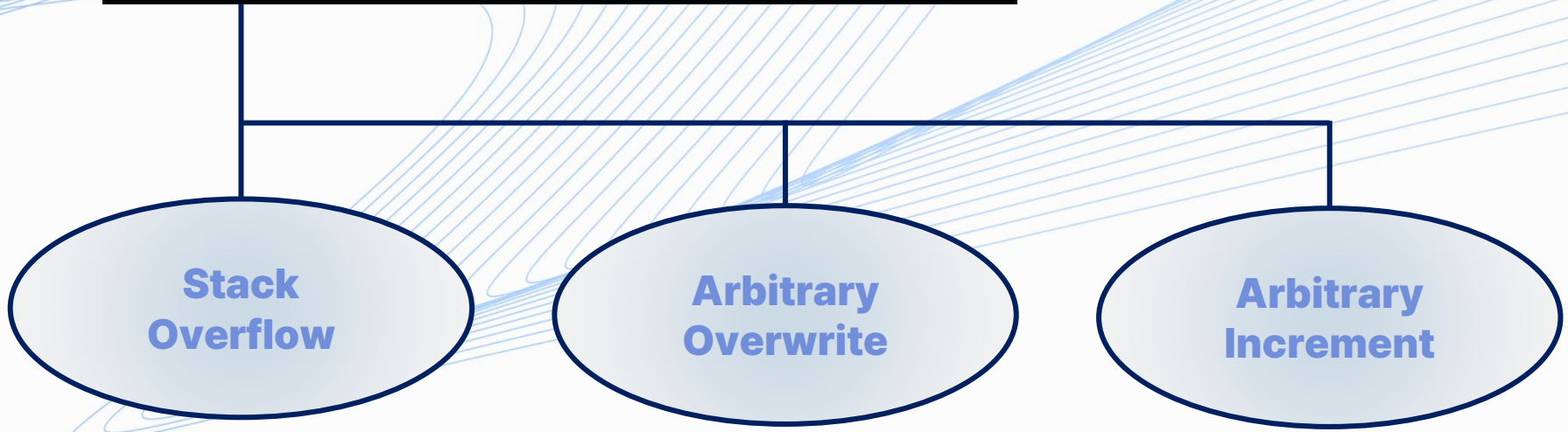
HEVD

HackSys Extreme Vulnerable Driver



Windows Kernel Exploit 기술을 학습하기 위한
여러가지 보안 취약점을 포함시켜 만들어진 실습용 드라이버

- 커널 취약점 분석
- 익스플로잇 개발 및 Privilege Escalation



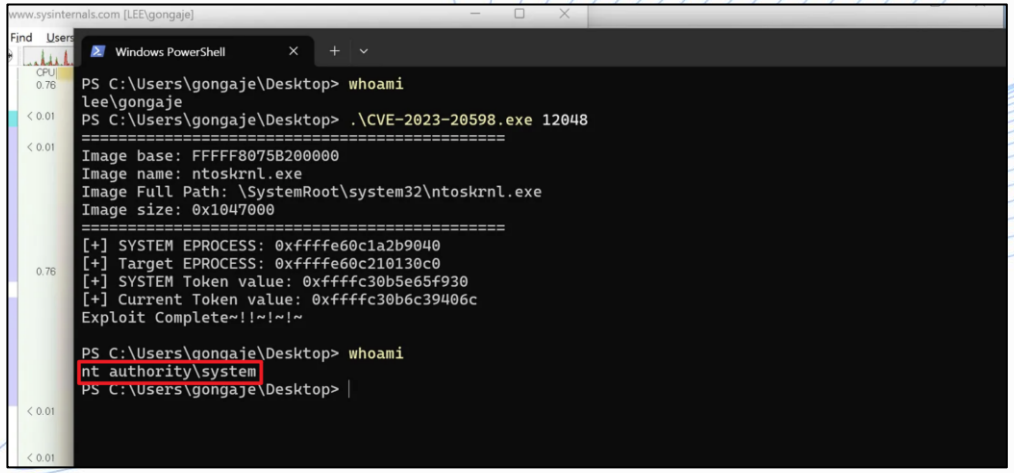
and so on ...

1-day

CVE-2023-20598

AMD RYZEN

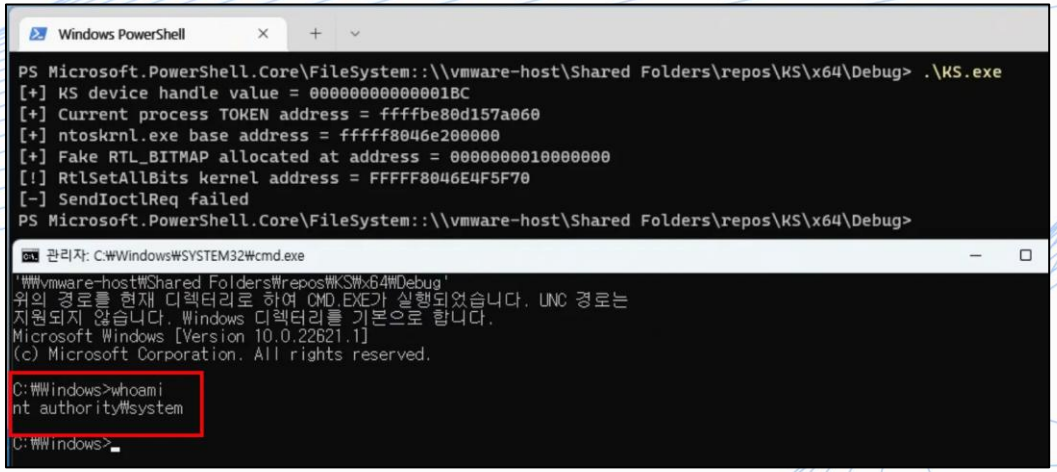
AMD Radeon™ 그래픽 드라이버에서 발생한
Arbitrary Write으로 인해 발생한 LPE



CVE-2024-35250

Windows

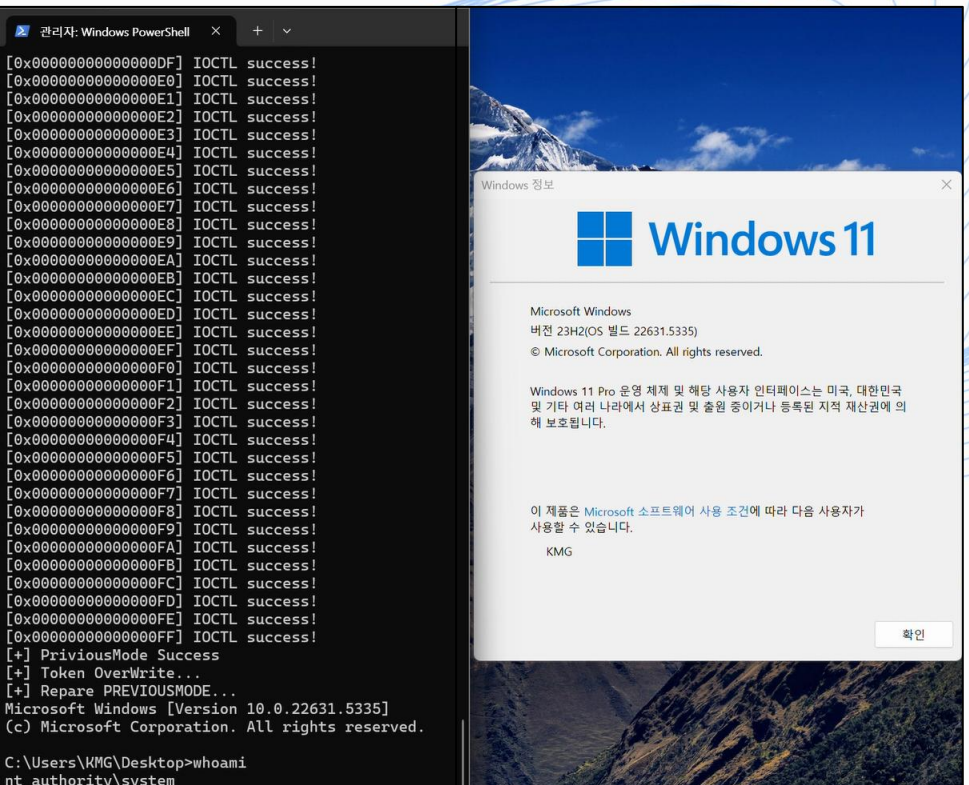
Windows 커널 스트리밍 서비스에서 발생한
신뢰할 수 없는 포인터 역참조로 인해 발생한 LPE

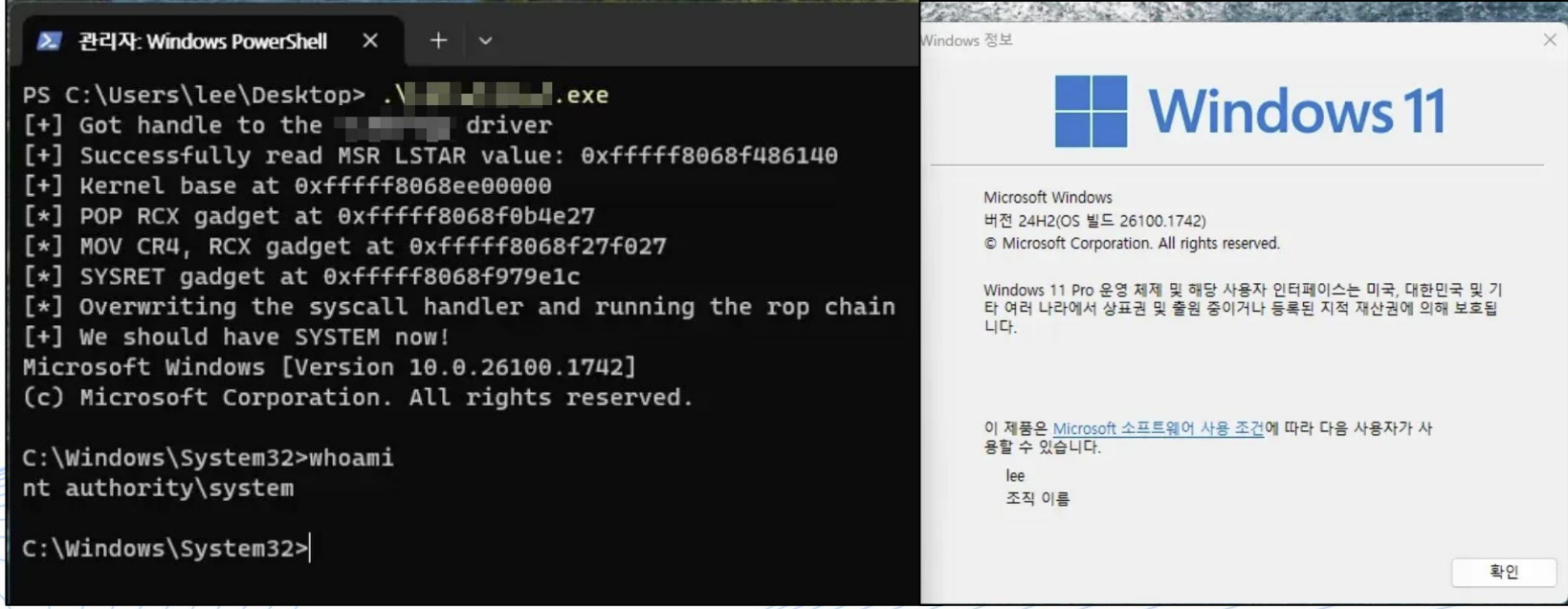


Result

그래서 LPE 찾았나요?

Arbitrary Write LPE





MSR Read/Write LPE

어떻게 찾았나요?

1. Kernel Driver

STEP 01
설치된 드라이버 식별

STEP 02
자동 로드
드라이버 식별

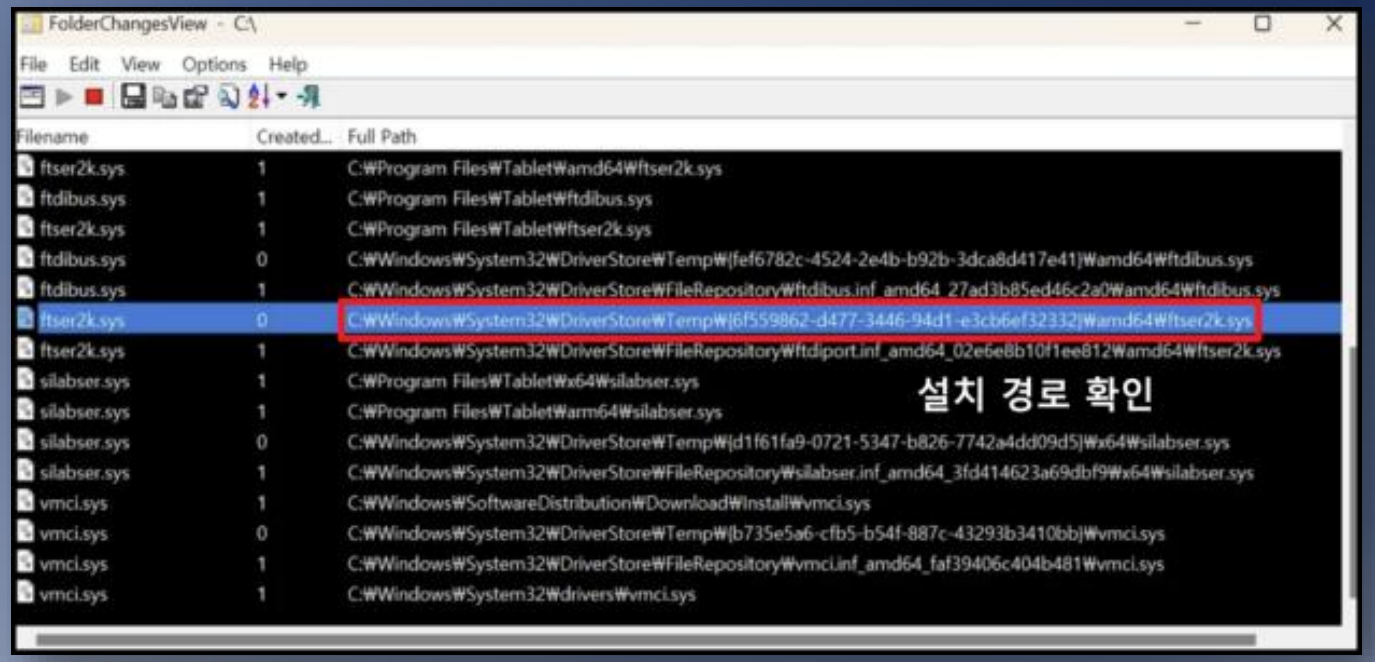
STEP 03
수집 자동화
스크립트

STEP 04
설치된 수집 결과

STEP 05
분석 시작

Target Collection Process

STEP 01



FolderChangesView

Driver 설치 시 파일 시스템 변화를 추적해 생성된 .sys 파일 식별

1. Kernel Driver

STEP 01
설치된 드라이버 식별

STEP 02
자동 로드
드라이버 식별

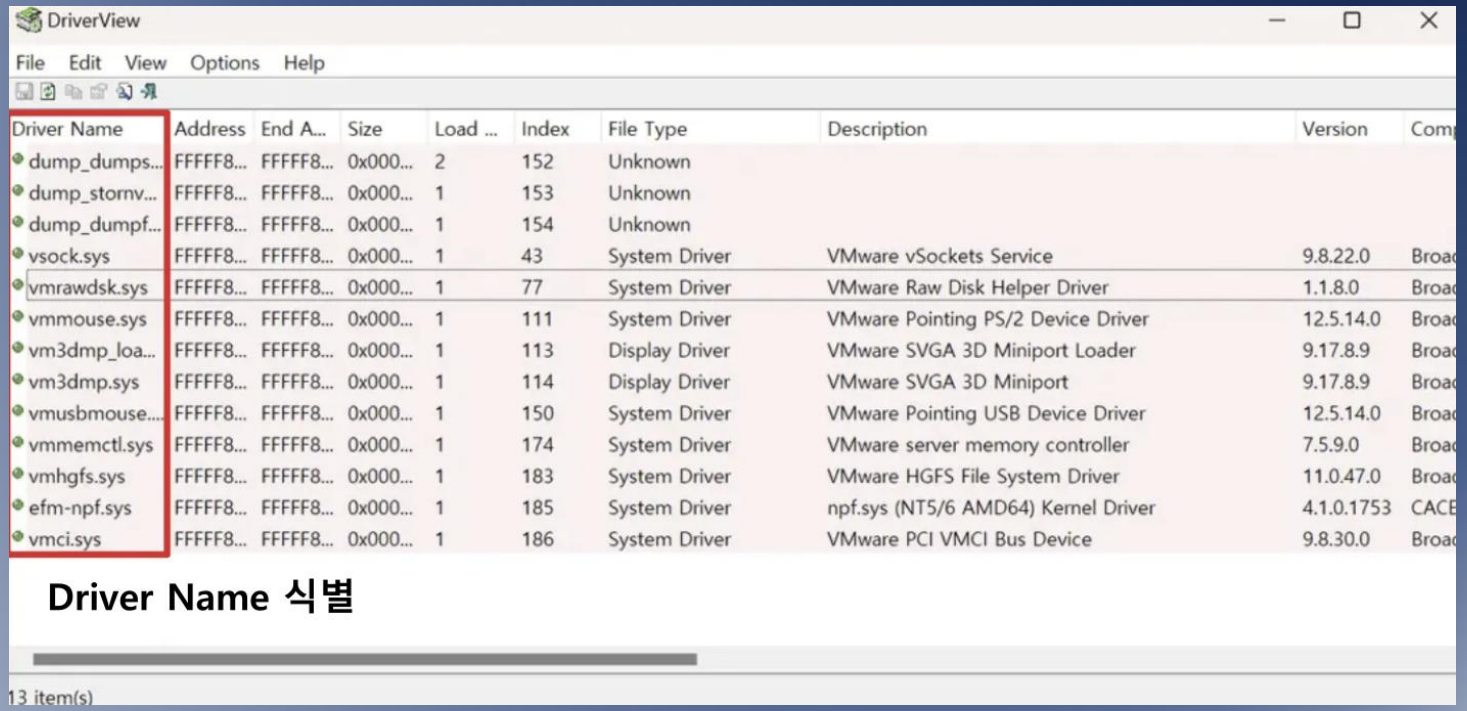
STEP 03
수집 자동화
스크립트

STEP 04
설치된 수집 결과

STEP 05
분석 시작

Target Collection Process

STEP 02



DriverView

SW 설치 및 실행 시 자동으로 로드되는 KERNEL Driver 탐색

1. Kernel Driver

STEP 01
설치된 드라이버 식별

STEP 02
자동 로드
드라이버 식별

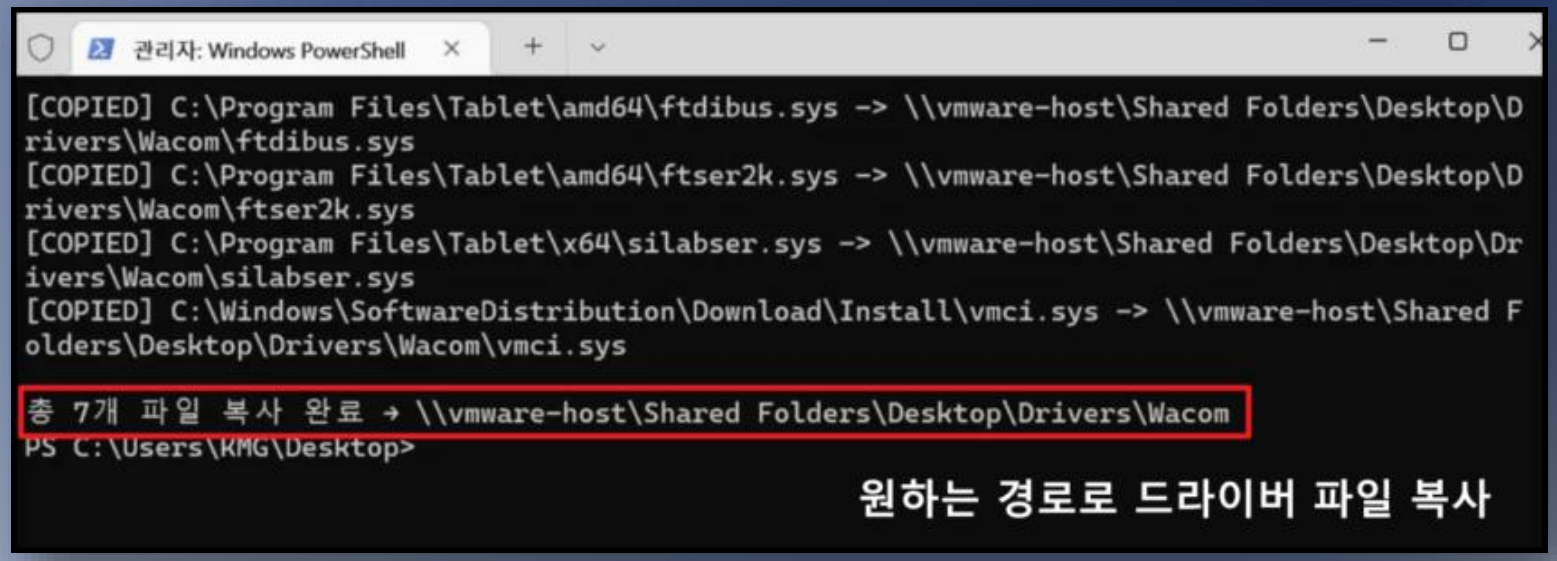
STEP 03
수집 자동화
스크립트

STEP 04
설치된 수집 결과

STEP 05
분석 시작

Target Collection Process

STEP 03



수집 자동화 스크립트 제작

스크립트를 통해 드라이버 파일을 자동으로 수집하여
지정한 공유 폴더에 복사

1. Kernel Driver

STEP 01
설치된 드라이버 식별

STEP 02
자동 로드
드라이버 식별

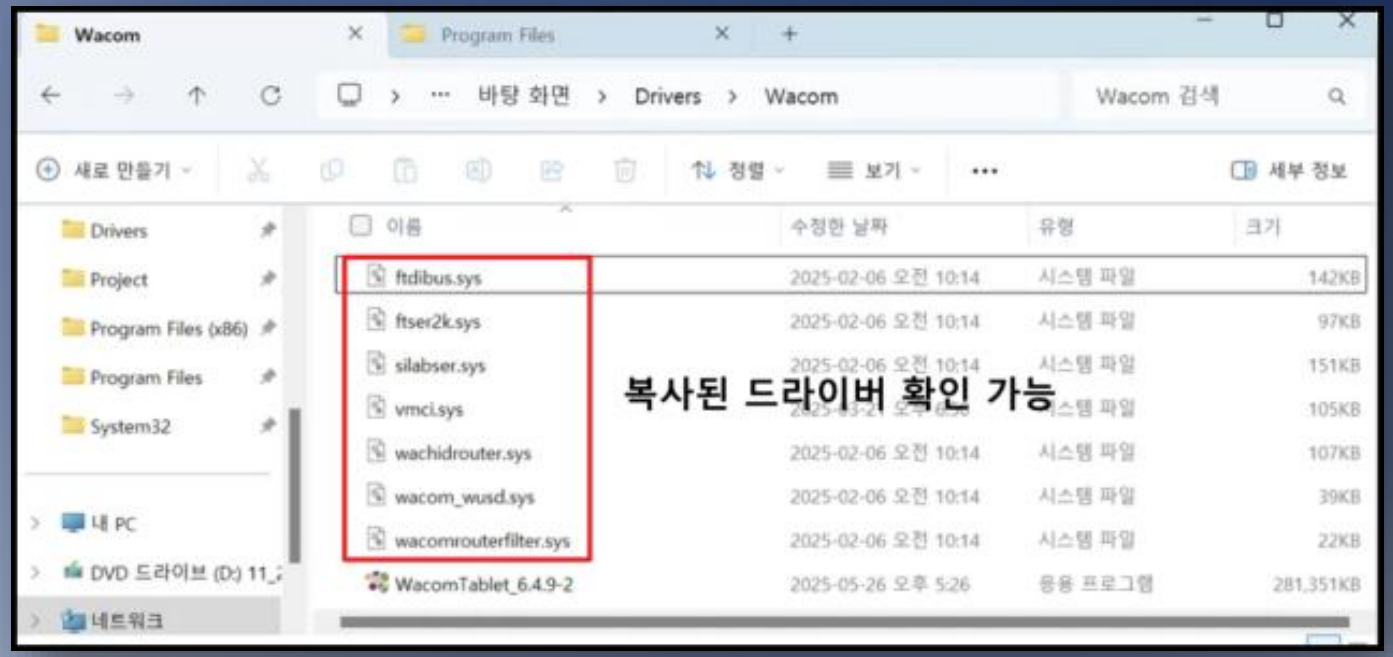
STEP 03
수집 자동화
스크립트

STEP 04
설치된 수집 결과

STEP 05
분석 시작

Target Collection Process

STEP 04



드라이버 수집 결과 확인

스크립트를 통해 복사된 드라이버 파일 확인

STEP 01
설치된 드라이버 식별

STEP 02
자동 로드
드라이버 식별

STEP 03
수집 자동화
스크립트

STEP 04
설치된 수집 결과

STEP 05
분석 시작

Target Collection Process

STEP 05

타겟 수집 완료

분석 시작

2. Named Pipe

STEP 01
소프트웨어 Service 확인

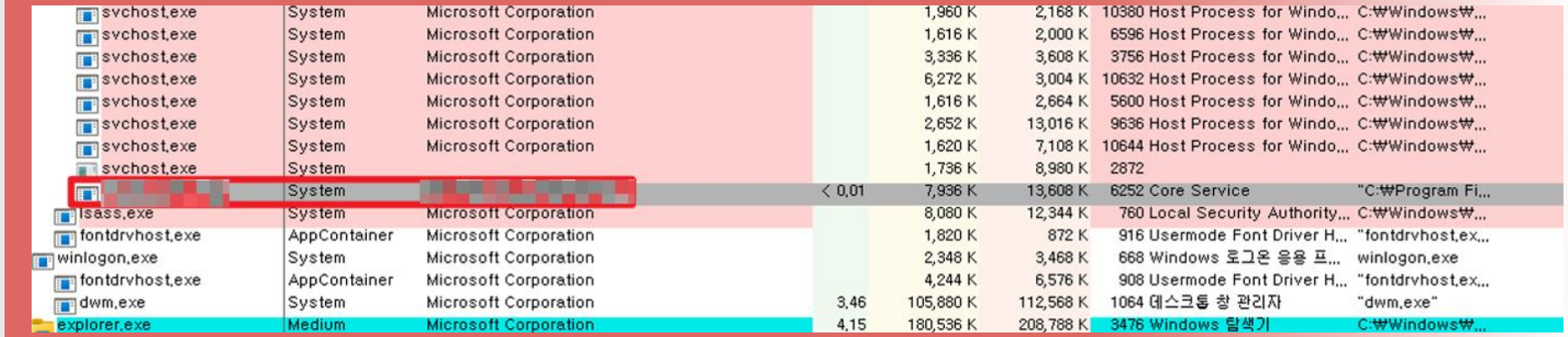
STEP 02
Named Pipe 식별

STEP 03
Named Pipe 권한 확인

STEP 04
Named Pipe 관련 API 확인

STEP 05
분석 시작

Target Collection Process
STEP 01



svchost.exe	System	Microsoft Corporation		1,960 K	2,168 K	10380 Host Process for Windo...	C:₩Windows₩...
svchost.exe	System	Microsoft Corporation		1,616 K	2,000 K	6596 Host Process for Windo...	C:₩Windows₩...
svchost.exe	System	Microsoft Corporation		3,336 K	3,608 K	3756 Host Process for Windo...	C:₩Windows₩...
svchost.exe	System	Microsoft Corporation		6,272 K	3,004 K	10632 Host Process for Windo...	C:₩Windows₩...
svchost.exe	System	Microsoft Corporation		1,616 K	2,664 K	5600 Host Process for Windo...	C:₩Windows₩...
svchost.exe	System	Microsoft Corporation		2,652 K	13,016 K	9636 Host Process for Windo...	C:₩Windows₩...
svchost.exe	System	Microsoft Corporation		1,620 K	7,108 K	10644 Host Process for Windo...	C:₩Windows₩...
svchost.exe	System			1,736 K	8,980 K	2872	
[redacted]	System	[redacted]	< 0.01	7,936 K	13,608 K	6252 Core Service	"C:₩Program Fi...
lsass.exe	System	Microsoft Corporation		8,080 K	12,344 K	760 Local Security Authority...	C:₩Windows₩...
fontdrvhost.exe	AppContainer	Microsoft Corporation		1,820 K	872 K	916 Usermode Font Driver H...	"fontdrvhost.ex...
winlogon.exe	System	Microsoft Corporation		2,348 K	3,468 K	668 Windows 로그온 응용 프...	winlogon.exe
fontdrvhost.exe	AppContainer	Microsoft Corporation		4,244 K	6,576 K	908 Usermode Font Driver H...	"fontdrvhost.ex...
dwm.exe	System	Microsoft Corporation	3.46	105,880 K	112,568 K	1064 데스크톱 창 관리자	"dwm.exe"
explorer.exe	Medium	Microsoft Corporation	4.15	180,536 K	208,788 K	3476 Windows 탐색기	C:₩Windows₩...

설치된 소프트웨어의 System Service 확인

Process Explorer를 통해 System 권한으로 실행되는 서비스 확인

2. Named Pipe

STEP 01
소프트웨어 Service
확인

STEP 02
Named Pipe
식별

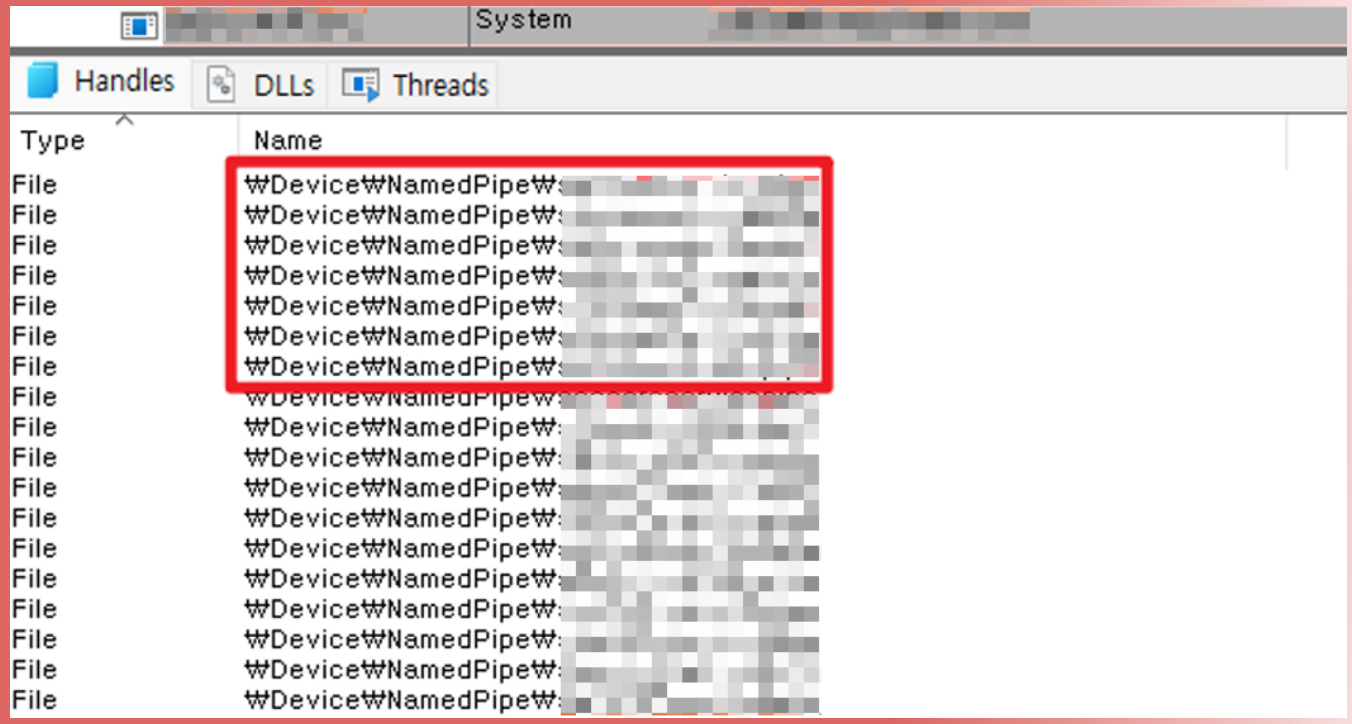
STEP 03
Named Pipe
권한 확인

STEP 04
Named Pipe 관련
API 확인

STEP 05
분석 시작

Target Collection Process

STEP 02



Named Pipe name 식별

해당 서비스의 Handles에서 Named Pipe 식별

STEP 01
소프트웨어 Service 확인

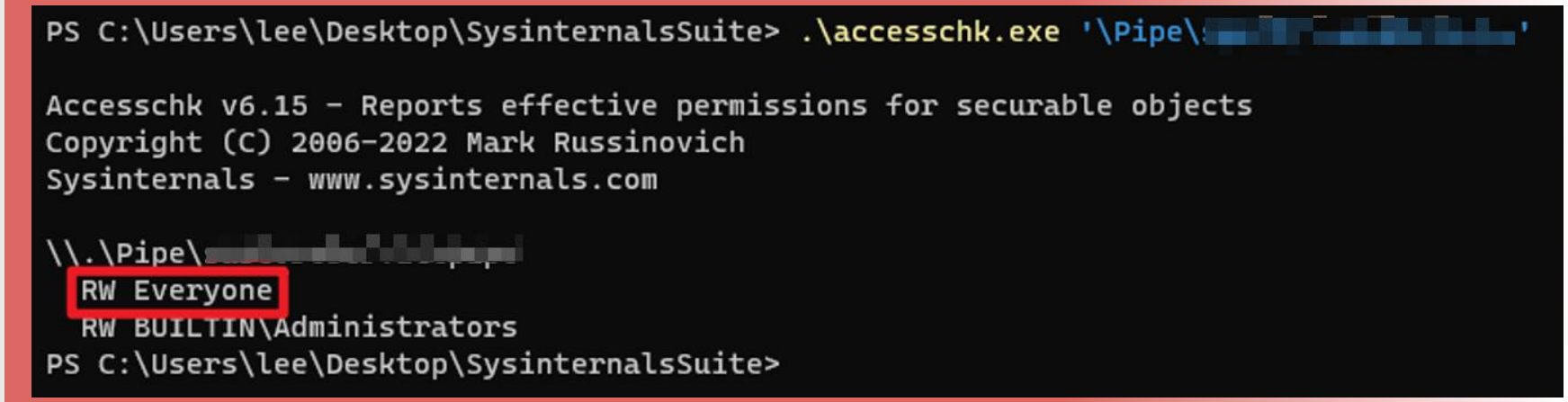
STEP 02
Named Pipe 식별

STEP 03
Named Pipe 권한 확인

STEP 04
Named Pipe 관련 API 확인

STEP 05
분석 시작

Target Collection Process
STEP 03



Named Pipe 권한 확인

Everyone 권한으로 Named Pipe에 접근이 가능한지 확인

2. Named Pipe

STEP 01
소프트웨어 Service 확인

STEP 02
Named Pipe 식별

STEP 03
Named Pipe 권한 확인

STEP 04
Named Pipe 관련 API 확인

STEP 05
분석 시작

Target Collection Process
STEP 04

Address	Ordinal	Name	Library
KERNEL32			
00007FF652C5…		CreateNamedPipeA	KERNEL32
00007FF652C5…		DisconnectNamedPipe	KERNEL32
00007FF652C5…		ConnectNamedPipe	KERNEL32
ADVAPI32			
00007FF652C5…		ImpersonateNamedPipeClient	ADVAPI32

namedPipe

CreateNamedPipe, ConnectNamedPipe API 검색

해당 서비스의 바이너리를 추출하여
Named Pipe 관련 API가 있는지 확인

2. Named Pipe

STEP 01
소프트웨어 Service 확인

STEP 02
Named Pipe 식별

STEP 03
Named Pipe 권한 확인

STEP 04
Named Pipe 관련 API 확인

STEP 05
분석 시작

Target Collection Process
STEP 05

```
if ( !v1[24] )
{
  while ( 1 )
  {
    NumberOfBytesRead = 0;
    memset(Buffer, 0, 0xA098u);
    if ( !ReadFile(lpThreadParameter[1], Buffer, 0xA098u, &NumberOfBytesRead, 0) || !NumberOfBytesRead )
      break;
    if ( !(unsigned int)sub_7FF652C2F7E0((__int64)*lpThreadParameter, (__int64)Buffer, lpThreadParameter[1])
      || *((_DWORD *)*lpThreadParameter + 24) )
    {
      goto LABEL_10;
    }
  }
  (*(void (__fastcall **)(_QWORD, _QWORD, const char *))(**((_QWORD **)*lpThreadParameter + 29) + 8LL))(
    *((_QWORD *)*lpThreadParameter + 29),
    0,
    "Read File - Failed");
}
```

Binary 분석 시작

사용자가 사용 할 수 있는 Named Pipe의 기능 분석

수집한 Target

벤더사	상태	분석 상황	사람	전체	WDM	Major	볼만할 듯라이버
	의미없음	설치 불가	김태영	0			
	Anti Virus	WDM 있음 볼만할듯 고고	경재 이	16	10	☑	7
	취약점 식별	WDM 있음	김태영	6	3		
	의미없음	패스 WDM 있음 Named P	김태영	2	1		
	Anti Virus	드라이버 없음	김명규				0
	의미없음	드라이버 없음	김태영	0			
	POC 작성 완료	Poc 제작 WDM 있음	경재 이	6	4		
	NonActiveX		방세연				
	의미없음	드라이버 없음	김태영	0			
	취약점 식별	취약점 식별 WDM 있음	김태영	7	5		
	시작 전		방세연				
	의미없음	설치 불가	김태영	0			
	의미없음	패스 디바이스네임 없음 W	경재 이	1	1		
	의미없음	드라이버 없음 패스	경재 이				
	시작 전	드라이버 없음	김명규	0	0		
	Anti Virus	WDM 있음 패스	김태영	15	6	☑	0
	Anti Virus	패스 WDM 있음	김태영	17	17	☑	0
	Anti Virus	WDM 없음 Named Pipe 존재	방세연	11	0	☑	0
	Anti Virus	패스 드라이버 없음	경재 이	0	0	☑	0
	시작 전	Named Pipe 없음 드라이버 없	김명규				
	의미없음	드라이버 없음	김태영	0			
	Anti Virus	WDM 있음 패스	김태영	11	3	☑	0
	Anti Virus	드라이버 없음 패스	경재 이	0	0		0
	의미없음	WDM 있음 디바이스네임 없음	김명규	1	1		
	시작 전	패스	방세연				
	Anti Virus	패스 드라이버 없음	경재 이	0	0		0
	시작 전	드라이버 없음	김명규	0	0		
	NonActiveX	드라이버 없음	방세연				
	의미없음	드라이버 없음	김태영	0			
	Anti Virus	패스 무료 버전 메일 컴퓨 필요	김태영				0
	Anti Virus	진행 중	임한글			☑	
	의미없음	WDM 없음	방세연				
	Anti Virus	패스 WDM 없음	경재 이	6	0	☑	0
	취약점 제보	익스플로잇 제작 WDM 있음	경재 이	3	1		
	의미없음	패스 WDM 있음	김태영	4	2		
	의미없음	패스 WDM 있음	김태영	6	1		
	의미없음	드라이버 없음	김태영	0			
	시작 전	WDM 없음	김명규	1	0		
	의미없음	드라이버 없음	김태영	0			
	NonActiveX	진행 중	방세연				
	Anti Virus	패스 무료 버전 메일 컴퓨 필요	김태영			☑	
	시작 전	WDM 없음	김명규	2	0		
	시작 전	패스 WDM 있음	경재 이	2	2		

이제 취약점 찾으러 ㄱㄱ

Target Analyze Process

STEP 01

```
Driver Buddy Reloaded Auto-analysis
--------------------------------------------
[+] `DriverEntry` found at: 0x00015008
[>] Searching for `Device Names`...
          - \DosDevices\WinRing0_1_2_0
          - \Device\WinRing0_1_2_0
[>] Searching for `Pooltags`...
[>] Searching for interesting opcodes...
          - Found rdpmc in sub_114D0 at 0x000114d2
          - Found wrmsr in sub_1149C at 0x000114ac
          - Found rdmsr in sub_11468 at 0x0001146a
```

1. DriverEntry Address

2. Device & SymbolicLink Name

```
[+] Possible `DispatchDeviceControl` at 0x000110d8
[>] Based off basic CFG analysis, potential dispatch functions are:
          - Possible_DispatchDeviceControl_0
          - DriverEntry
[+] Driver type detected: WDM
[>] Searching for IOCTLs found by IDA...
[!] Unable to automatically find any IOCTLs
[+] Analysis Completed!
```

3. DispatchDeviceControl Address

4. Driver Type

Driver Buddy / Plugin 활용해 WDM 드라이버 분석 시작

STEP 01
WDM 드라이버
분석 시작

STEP 02
Device 노드
접근 확인

STEP 03
유저 접근성 파악

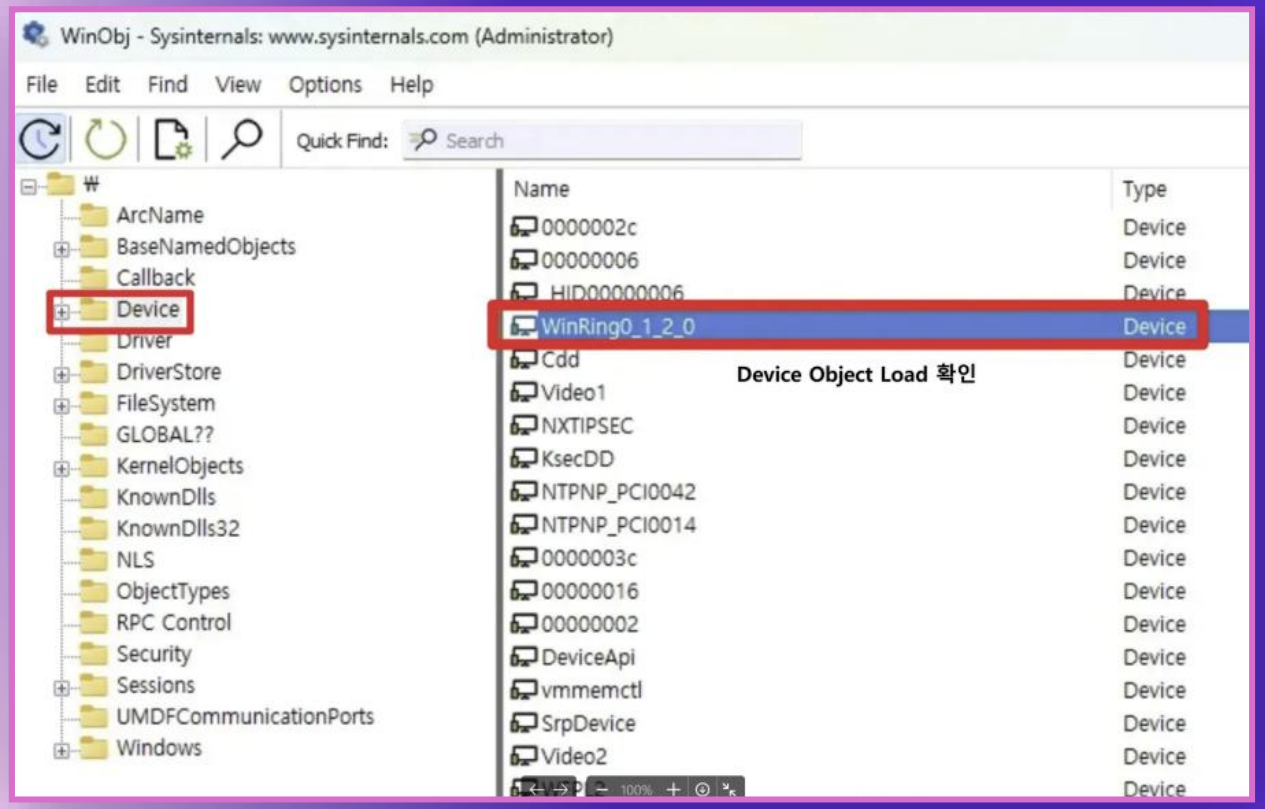
STEP 04
IOCTL 테이블 식별

STEP 05
디스패치 루틴 및
취약점 분석

STEP 06
동적 추적 및 분석

Target Analyze Process

STEP 02



WinObj로 \Device 노드 접근 가능 여부 확인

STEP 01
WDM 드라이버
분석 시작

STEP 02
Device 노드
접근 확인

STEP 03
유저 접근성 파악

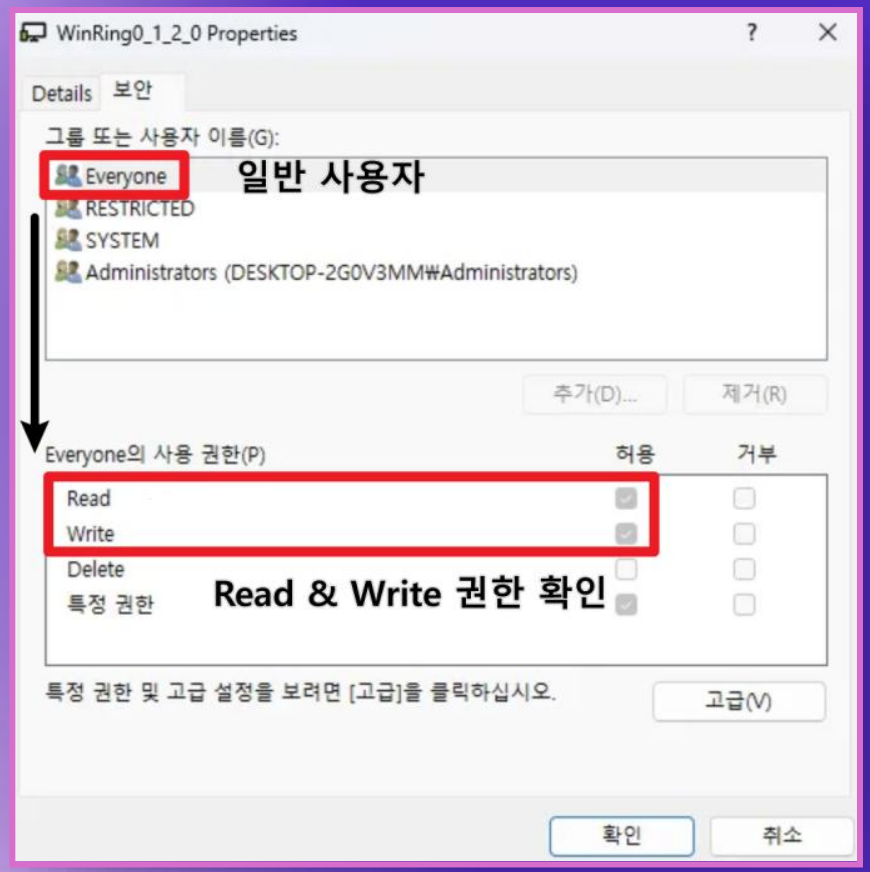
STEP 04
IOCTL 테이블 식별

STEP 04
디스패치 루틴 및
취약점 분석

STEP 04
동적 추적 및 분석

Target Analyze Process

STEP 03



노출된 DeviceName -> 유저 접근 가능 여부 판단

STEP 01
WDM 드라이버
분석 시작

STEP 02
Device 노드
접근 확인

STEP 03
유저 접근성 파악

STEP 04
IOCTL 테이블 식별

STEP 04
디스패치 루틴 및
취약점 분석

STEP 04
동적 추적 및 분석

Target Analyze Process

STEP 04

```
DeviceObject = 0;
RtlInitUnicodeString(&DeviceName, L"\\Device\\WinRing0_1_2_0");
result = IoCreateDevice(DriverObject, 0, &DeviceName, 0x9C40u,
```

Device Name 확인

```
if ( result >= 0 )
{
  dword_140003110 = 0;
  DriverObject->MajorFunction[0] = (PDRIVER_DISPATCH)DispatchDeviceControl;
  DriverObject->MajorFunction[2] = (PDRIVER_DISPATCH)DispatchDeviceControl;
  DriverObject->MajorFunction[14] = (PDRIVER_DISPATCH)DispatchDeviceControl;
  DriverObject->DriverUnload = (PDRIVER_UNLOAD)sub_140001424;
```

DispatchDeviceControl 함수 확인

IDA, FLOSS, strings 등으로 IOCTL 테이블 식별

STEP 01
WDM 드라이버
분석 시작

STEP 02
Device 노드
접근 확인

STEP 03
유저 접근성 파악

STEP 04
IOCTL 테이블 식별

STEP 05
디스패치 루틴 및
취약점 분석

STEP 06
동적 추적 및 분석

Target Analyze Process

STEP 05

```
      goto LABEL_24;
    case 0x9C402004:
      *(_DWORD *)Irp->AssociatedIrp.SystemBuffer = dword_140003110;
LABEL_24:
      *(_QWORD *)p_Information = 4;
LABEL_64:
      v5 = 0;
      break;
    case 0x9C402084:        IOCTL 코드 확인
      v11 = sub_140001468(
              (unsigned int *)Irp->AssociatedIrp.SystemBuffer,
              CurrentStackLocation->Parameters.DeviceIoControl.InputBufferLength,
              (unsigned __int64 *)Irp->AssociatedIrp.SystemBuffer,
              CurrentStackLocation->Parameters.DeviceIoControl.OutputBufferLength,
              p_Information);
      goto LABEL_58;
    case 0x9C402088:
      v11 = sub_14000149C(
              (__int64)Irp->AssociatedIrp.SystemBuffer,
              CurrentStackLocation->Parameters.DeviceIoControl.InputBufferLength,
              (__int64)Irp->AssociatedIrp.SystemBuffer,
              CurrentStackLocation->Parameters.DeviceIoControl.OutputBufferLength,
              p_Information);
      goto LABEL_58;
```

디스패치 루틴 분석 -> 권한 검증 누락 · 구조체 오염 가능성 판단

STEP 01
WDM 드라이버
분석 시작

STEP 02
Device 노드
접근 확인

STEP 03
유저 접근성 파악

STEP 04
IOCTL 테이블 식별

STEP 05
디스패치 루틴 및
취약점 분석

STEP 06
동적 추적 및 분석

Target Analyze Process

STEP 06

```
6: kd> pt
Breakpoint 1 hit                                    ← 브레이크 포인트 설정
HEVD!TriggerBufferOverflowStack+0xca:
fffff804`4bf1667e e83dabf7ff      call    HEVD!memcpy (fffff804`4be911c0)
6: kd> pt
HEVD!TriggerBufferOverflowStack+0x10b:              ← 버퍼 오버플로우 발생
fffff804`4bf166bf c3              ret
4: kd> dq rsp
ffffb78a`a8dff748  41414141`41414141 41414141`41414141
ffffb78a`a8dff758  41414141`41414141 41414141`41414141
ffffb78a`a8dff768  41414141`41414141 41414141`41414141
ffffb78a`a8dff778  41414141`41414141 41414141`41414141   ← 메모리 레이아웃 확인
ffffb78a`a8dff788  41414141`41414141 41414141`41414141
ffffb78a`a8dff798  41414141`41414141 41414141`41414141
ffffb78a`a8dff7a8  41414141`41414141 41414141`41414141
ffffb78a`a8dff7b8  41414141`41414141 41414141`41414141
4: kd> g
Access violation - code c0000005 (!!! second chance !!!)
HEVD!TriggerBufferOverflowStack+0x10b:
fffff804`4bf166bf c3              ret
```

실행 중인 함수 흐름 및 디바이스 요청 처리 과정을
추적하며, 동적 분석 수행

STEP 01
WDM 드라이버
분석 시작

STEP 02
Device 노드
접근 확인

STEP 03
유저 접근성 파악

STEP 04
IOCTL 테이블 식별

STEP 05
디스패치 루틴 및
취약점 분석

STEP 06
동적 추적 및 분석

Target Vender 330개

그래서 어떤 취약점을 찾았나요?

MSR Read/Write

Named Pipe

Arbitrary Write

Logical Bug

1. MSR Read/Write LPE (Token Stealing) - Driver

```
v5 = __readmsr(*SystemBuffer);
*SystemBuffer_1 = ((unsigned __int64)HIDWORD(v5) << 32) | (unsigned int)v5;
*a5 = 8;
return 0;
```

MSR 레지스터를 Read 하여 Kernel base 주소를 leak 가능

```
__writemsr(*(_DWORD *)SystemBuffer, *(_QWORD *)(SystemBuffer + 4));
*a5 = 0;
return 0;
```

LSTAR 레지스터(SystemCall)에 원하는 값을 Write 하여 ROP Chain 구성 가능

2. Arbitrary Write LPE (PriviousMode) - Driver

```
if ( IoControlCode == 0x8200E810 )
{
  SystemBuffer = IRP->AssociatedIrp.SystemBuffer;
  SystemBuffer_value = *(_QWORD *)SystemBuffer;
  v14 = KeAcquireSpinLockRaiseToDpc(&SpinLock);
  HIDWORD(DeviceExtension[10].Header.WaitListHead.Flink) = 0;
  if ( SystemBuffer_value )
  {
    if ( SystemBuffer[2] != (unsigned int)Check_Value((__int64)&dword_FFFFF8020CF74058) )
    {
      v5 = 0xC000009C;
      goto LABEL_32;
    }
    *(_DWORD *)(SystemBuffer_value + 0x18) = Increment_Value((__int64)&dword_FFFFF8020CF74058);
  }
```

일반 사용자가 제어할 수 있는 SystemBuffer_value + 0x18을 포인터로 사용하여

Increment_Value 함수의 Return 값을 할당하므로 임의 커널 주소에 임의 값 쓰기 가능

3-1. CreateFile / WriteFile LPE (Logical) - Driver

```
else
{
  SystemBuffer_value_plus10 = SystemBuffer[4];
  CreateDisposition = 3;
  if ( SystemBuffer_value_plus10 == 1 )
    CreateDisposition = 2;
  if ( SystemBuffer_value_plus10 == 2 )
    CreateDisposition = 0;
  if ( SystemBuffer_value_plus10 == 3 )
    CreateDisposition = 1;
  if ( SystemBuffer_value_plus10 == 4 )
    CreateDisposition = 3;
  v8 = ZwCreateFile(
         &FileHandle,
         SystemBuffer[2] | 0x100000,
         &ObjectAttributes,
         &IoStatusBlock,
         0,
         SystemBuffer[5],
         SystemBuffer[3],
         CreateDisposition,
         0x20u,
         0,
         0);
}
```

```
NTSTATUS __fastcall ZwWriteFileHandler(__int64 SystemBuffer)
{
  ULONG_PTR Information; // rbx
  NTSTATUS result; // eax
  struct _IO_STATUS_BLOCK v4; // [rsp+50h] [rbp-18h] BYREF

  Information = 0;
  result = ZwWriteFile(
             *(HANDLE *)(SystemBuffer + 0x18),
             0,
             0,
             0,
             &v4,
             *(PVOID *)(SystemBuffer + 0x20),
             *(_DWORD *)(SystemBuffer + 0x28),
             0,
             0);
  if ( result >= 0 )
    Information = v4.Information;
  *(_QWORD *)(SystemBuffer + 0x30) = Information;
  return result;
```

ZwCreateFile 함수를 통해 이미 존재하는 파일의 핸들 획득

ZwWriteFile 함수를 통해 악성 Payload 주입 가능

3-2. CreateFile / WriteFile LPE (Logical) – Named Pipe

```
if ( v6 == 0x1000 )
{
  FileW = CreateFileW(
            (LPCWSTR)(input + 8),
            *(_DWORD *)(input + 0x1048),
            *(_DWORD *)(input + 0x104C),
            0,
            *(_DWORD *)(input + 0x1050),
            *(_DWORD *)(input + 0x1054),
            0);
  LastError = GetLastError();
  *((_QWORD *)&Buffer[0] + 1) = FileW;
LABEL_45:
  LODWORD(Buffer[0]) = LastError;
LABEL_46:
  NumberOfBytesWritten[0] = 0;
  p_TokenHandle = NumberOfBytesWritten;
  goto LABEL_14;
}
```

```
case 0x1005u:
  v51 = j__malloc_base(*(unsigned int *)(input + 16));
  NumberOfBytesWritten[0] = 0;
  if ( ReadFile(a3, v51, *(_DWORD *)(input + 16), NumberOfBytesWritten, 0) )
  {
    v52 = *(_DWORD *)(input + 16);
    if ( NumberOfBytesWritten[0] == v52 )
    {
      LODWORD(lpdwDisposition) = 0;
      DWORD2(Buffer[0]) = WriteFile(*(HANDLE *)(input + 8), v51, v52, (LPDWORD)&lpdwDisposition, 0);
      HIDWORD(Buffer[0]) = (_DWORD)lpdwDisposition;
    }
  }
  free(v51);
  LODWORD(Buffer[0]) = GetLastError();
  LODWORD(TokenHandle) = 0;
  p_TokenHandle = (DWORD *)&TokenHandle;
  goto LABEL_14;
```

CreateFile 함수를 통해 이미 존재하는 파일의 핸들 획득

WriteFile 함수를 통해 악성 Payload 주입 가능

4-1. Registrey Set LPE (Logical) - Driver

```
__int64 __fastcall ZwOpenKeyHandler(__int64 SystemBuffer)
{
  ACCESS_MASK SystemBuffer_value_plus8; // edx
  void **SystemBuffer_value_plus48; // rcx
  struct _UNICODE_STRING UnicodeString; // [rsp+20h] [rbp-48h] BYREF
  struct _OBJECT_ATTRIBUTES ObjectAttributes; // [rsp+30h] [rbp-38h] BYREF

  Select_path(*(_QWORD *)SystemBuffer, *(const char **)(SystemBuffer + 0x10), &UnicodeString);
  SystemBuffer_value_plus8 = *(_DWORD *)(SystemBuffer + 8);
  ObjectAttributes.RootDirectory = 0;
  ObjectAttributes.SecurityDescriptor = 0;
  ObjectAttributes.SecurityQualityOfService = 0;
  SystemBuffer_value_plus48 = *(void ***)(SystemBuffer + 0x48);
  ObjectAttributes.Length = 48;
  ObjectAttributes.Attributes = 64;
  ObjectAttributes.ObjectName = &UnicodeString;
  *(_DWORD *)(SystemBuffer + 0x58) = ZwOpenKey(SystemBuffer_value_plus48, SystemBuffer_value_plus8, &ObjectAttributes);
  RtlFreeUnicodeString(&UnicodeString);
  return *(unsigned int *)(SystemBuffer + 0x58);
```

ZwOpenKey 함수를 통해 레지스트리 핸들 획득

```
else
{
  Data = *(PWSTR *)(SystemBuffer + 0x28);
  DataSize = **(_DWORD **)(SystemBuffer + 0x30);
}
*(_DWORD *)(SystemBuffer + 0x58) = ZwSetValueKey(
  *(HANDLE *)SystemBuffer,
  &Destination,
  0,
  **(_DWORD **)(SystemBuffer + 0x20),
  Data,
  DataSize);
```

ZwSetValueKey 함수를 통해 레지스트리 변조

4-2. Registrey Set LPE (Logical) – Named Pipe

```
{
  if ( v83 == HKEY_CURRENT_USER )
    v83 = (HKEY)TlsGetValue(*(_DWORD *)(a1 + 144));
  v84 = RegOpenKeyExW(
          v83,
          v82,
          *(_DWORD *)(input + 4176),
          *(_DWORD *)(input + 4180),
          (PHKEY)NumberOfBytesWritten);
}
LODWORD(Buffer[0]) = GetLastError();
DWORD2(Buffer[0]) = v84;
*(_QWORD *)&Buffer[1] = *(_QWORD *)NumberOfBytesWritten;
LODWORD(TokenHandle) = 0;
p_TokenHandle = (DWORD *)&TokenHandle;
goto LABEL_14;
```

RegOpenKeyExW 함수를 통해 레지스트리 핸들 획득

RegSetValueExA 함수를 통해 레지스트리 변조

```
  if ( v169 == HKEY_CURRENT_USER )
    v169 = (HKEY)TlsGetValue(*(_DWORD *)(a1 + 144));
  InfoKeyA = RegSetValueExA(
               v169,
               v168,
               0,
               *(_DWORD *)(input + 2096),
               (const BYTE *)(input + 2100),
               *(_DWORD *)(input + 34868));
}
goto LABEL_164;
```

5. Create Service LPE (Logical) - Named Pipe

```
ServiceW = CreateServiceW(
             *(SC_HANDLE *)(input + 8),
             v227,
             v228,
             *(_DWORD *)(input + 0x2090),
             *(_DWORD *)(input + 0x2094),
             *(_DWORD *)(input + 0x2098),
             *(_DWORD *)(input + 0x209C),
             v229,
             v230,
             NumberOfBytesWritten,
             v231,
             v232,
             lpPassword);
LODWORD(Buffer[0]) = GetLastError();
*((_QWORD *)&Buffer[0] + 1) = ServiceW;
LODWORD(Buffer[1]) = NumberOfBytesWritten[0];
LODWORD(TokenHandle) = 0;
p_TokenHandle = (DWORD *)&TokenHandle;
goto LABEL_14;
```

```
case 0x1040u:
  FileAttributesW = StartServiceW(*(SC_HANDLE *)(input + 8), 0, 0);
  goto LABEL_93;
```

System 권한으로 임의 파일을 서비스로 실행

6. Service Hijacking LPE (Logical) - Named Pipe

임의 서비스의 BinaryPath를 악성 파일로 변조

```
case 0x103Du:
  v219 = (const WCHAR *)(input + 16);
  if ( !*(_WORD *)(input + 16) )
    v219 = 0;
  EventW = OpenServiceW(*(SC_HANDLE *)(input + 8), v219, *(_DWORD *)(input + 4176));
```

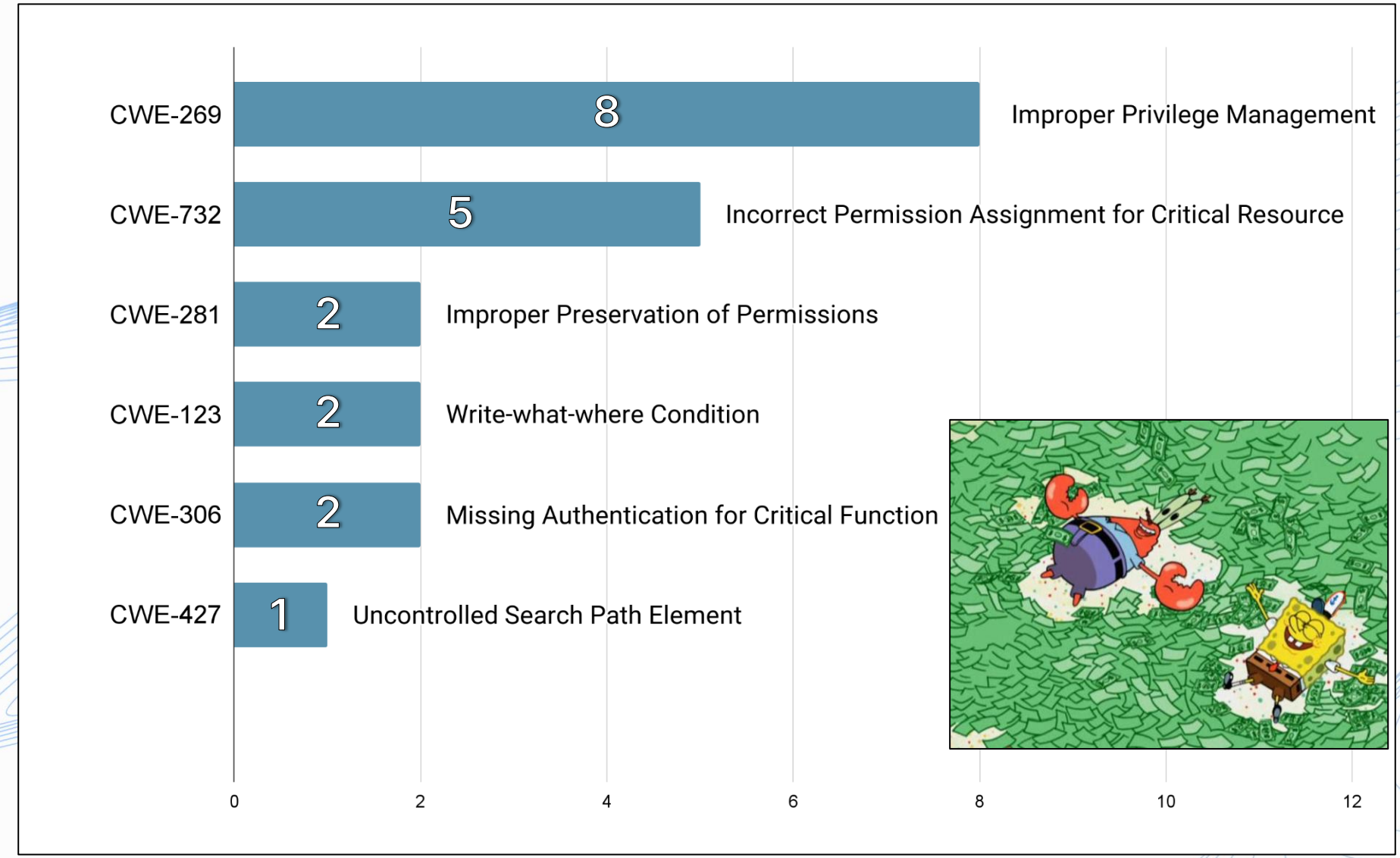
```
v226 = ChangeServiceConfigW(
         *(SC_HANDLE *)(input + 8),
         *(_DWORD *)(input + 16),
         *(_DWORD *)(input + 20),
         *(_DWORD *)(input + 24),
         v220,
         v221,
         NumberOfBytesWritten,
         v222,
         v223,
         v224,
         v225);
LODWORD(Buffer[0]) = GetLastError();
*((_QWORD *)&Buffer[0] + 1) = __PAIR64__(NumberOfBytesWritten[0], v226);
LODWORD(TokenHandle) = 0;
p_TokenHandle = (DWORD *)&TokenHandle;
goto LABEL_14;
```

7. SetNamedSecurityInfo LPE (Logical) - Named Pipe

```
case 0x1061u:
  sub_7FF652C33430(a1, (WCHAR *)(input + 8));
  v238 = SetNamedSecurityInfoW(
           (LPWSTR)(input + 8),
           *(SE_OBJECT_TYPE *)(input + 0x1048),
           *(_DWORD *)(input + 0x104C),
           0,
           0,
           0,
           0);
  sub_7FF652C33430(a1, (WCHAR *)(input + 8));
  v239 = GetLastError();
  DWORD2(Buffer[0]) = v238;
  LODWORD(Buffer[0]) = v239;
  LODWORD(TokenHandle) = 0;
  p_TokenHandle = (DWORD *)&TokenHandle;
  goto LABEL_14;
case 0x1062u:
  v20 = 0;
```

임의 파일의 보안 설정을 조작하여 NULL DACL로 변조

지금까지 찾은 취약점


CWE-269
8
Improper Privilege Management
CWE-732
5
Incorrect Permission Assignment for Critical Resource
CWE-281
2
Improper Preservation of Permissions
CWE-123
2
Write-what-where Condition
CWE-306
2
Missing Authentication for Critical Function
CWE-427
1
Uncontrolled Search Path Element
0
2
4
6
8
10
12

소감

- 취약점은 존재한다
- 끈기를 통한 성취감을 느껴보자
- 포기하지 않는 것이 중요하다!

느낀 점

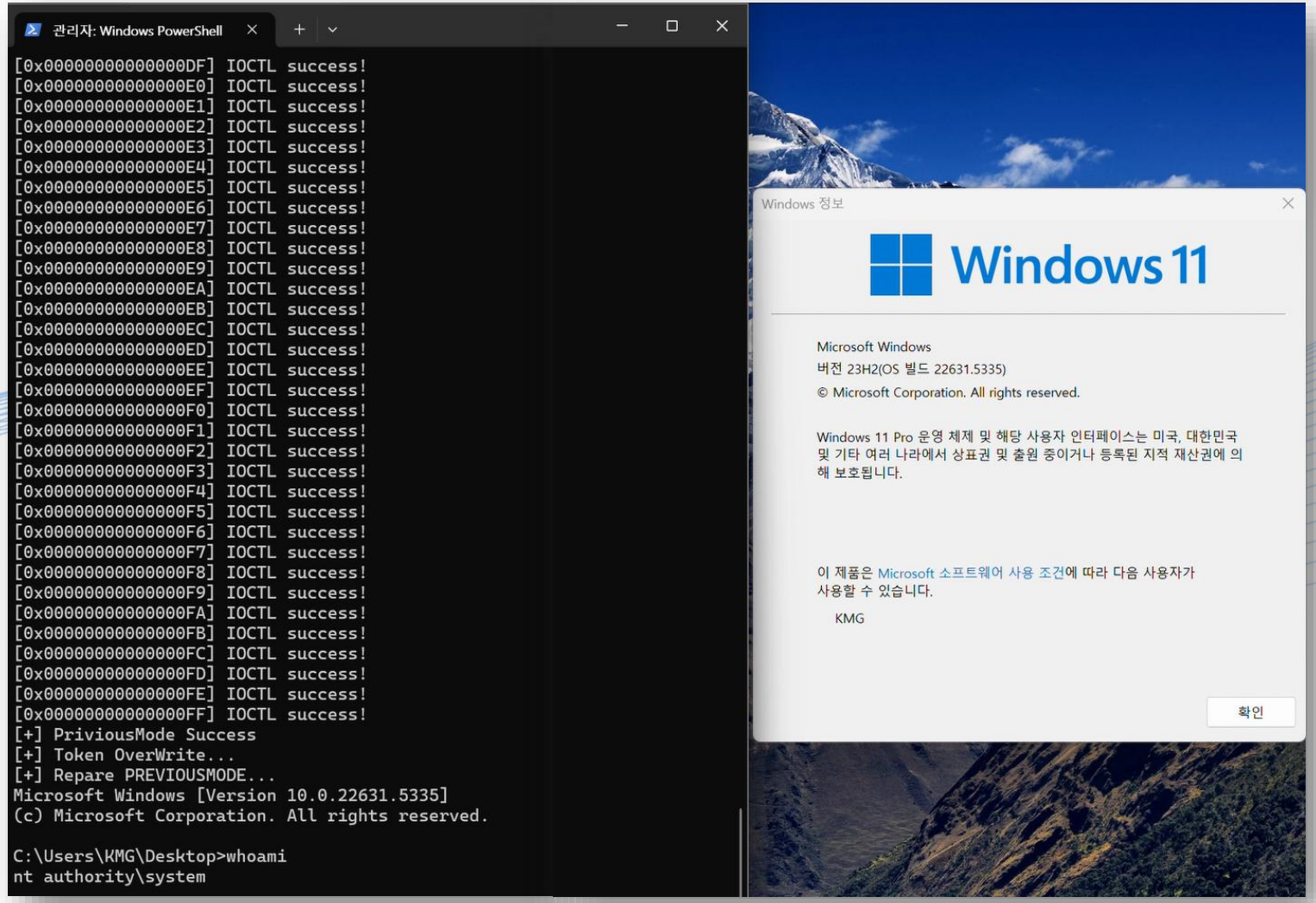
3개월이라는 기간이
너무 짧았다
좀 더 다양한 취약점
범위를 공부하고 싶다

일단 도전하는 것이
중요하고, 포기하지
않는다면 무조건
찾을 수 있다!!

살면서 이렇게 다같이
모여서 공부했던 시간은
값진 경험이 될 것

어려움이 많을수록
끝까지 파고드는
끈기가 얼마나
중요한지 알게 되었다

마치며


관리자: Windows PowerShell
[0x00000000000000DF] IOCTL success!
[0x00000000000000E0] IOCTL success!
[0x00000000000000E1] IOCTL success!
[0x00000000000000E2] IOCTL success!
[0x00000000000000E3] IOCTL success!
[0x00000000000000E4] IOCTL success!
[0x00000000000000E5] IOCTL success!
[0x00000000000000E6] IOCTL success!
[0x00000000000000E7] IOCTL success!
[0x00000000000000E8] IOCTL success!
[0x00000000000000E9] IOCTL success!
[0x00000000000000EA] IOCTL success!
[0x00000000000000EB] IOCTL success!
[0x00000000000000EC] IOCTL success!
[0x00000000000000ED] IOCTL success!
[0x00000000000000EE] IOCTL success!
[0x00000000000000EF] IOCTL success!
[0x00000000000000F0] IOCTL success!
[0x00000000000000F1] IOCTL success!
[0x00000000000000F2] IOCTL success!
[0x00000000000000F3] IOCTL success!
[0x00000000000000F4] IOCTL success!
[0x00000000000000F5] IOCTL success!
[0x00000000000000F6] IOCTL success!
[0x00000000000000F7] IOCTL success!
[0x00000000000000F8] IOCTL success!
[0x00000000000000F9] IOCTL success!
[0x00000000000000FA] IOCTL success!
[0x00000000000000FB] IOCTL success!
[0x00000000000000FC] IOCTL success!
[0x00000000000000FD] IOCTL success!
[0x00000000000000FE] IOCTL success!
[0x00000000000000FF] IOCTL success!
[+] PriviousMode Success
[+] Token OverWrite...
[+] Repare PREVIOUSMODE...
Microsoft Windows [Version 10.0.22631.5335]
(c) Microsoft Corporation. All rights reserved.
C:\Users\KMG\Desktop>whoami
nt authority\system
Windows 정보
Windows 11
Microsoft Windows
버전 23H2(OS 빌드 22631.5335)
© Microsoft Corporation. All rights reserved.
Windows 11 Pro 운영 체제 및 해당 사용자 인터페이스는 미국, 대한민국 및 기타 여러 나라에서 상표권 및 출원 중이거나 등록된 지적 재산권에 의해 보호됩니다.
이 제품은 Microsoft 소프트웨어 사용 조건에 따라 다음 사용자가 사용할 수 있습니다.
KMG
확인


ZERO DAY INITIATIVE
CASE OPENED
2025-06-28 02:17 GMT-6
A case has been opened and added to the queue for review.
CASE ASSIGNED
2025-06-30 11:09 GMT-6
A case has been opened and added to the queue for review.
CASE INVESTIGATED
2025-07-02 12:04 GMT-6
This case has been investigated.
CASE CONTRACTED
2025-07-15 10:27 GMT-6
This case has been officially contracted to the ZDI.
CASE REVIEWED
2025-07-22 10:46 GMT-6
This case has been reviewed.
VENDOR DISCLOSURE
2025-07-22 17:55 GMT-6
The details of this case have been submitted to the vendor as ZDI-CAN-27540.


ZDI-CAN-27540.

Thank you for listening!

Stop code : Q&A