Windows SMBGhost 취약점(CVE-2020-0796)

■ 취약점 개요

지난 2017년, 이터널블루(Eternal Blue) 취약점 발생으로 큰 이슈였던 Windows SMB에서 새로운 취약점이 발견되었다. SMBGhost라고 불리는 이 취약점(CVE-2020-0796)은 SMB 3.1.1 프로토콜에서 조작된 압축 패킷을 압축 해제하는 과정에서 Buffer Overflow가 발생한다. 클라이언트/서버 모두 공격이 가능하며, 공격자는 이를 통해 블루스크린(Blue Screen Of Death, BSOD)이나 임의의 명령을 실행할 수 있어 사용자들의 주의가 요구된다.

■ 영향받는 소프트웨어 버전

S/W 구분	취약 버전
Windows 10 Windows Server	1909, 1903 18362(3).720 이전 빌드

■ 대응 방안

CVE-2020-0796 취약점의 대응방안은 다음과 같다.

1. 최신 버전의 Windows Update를 받는다.

Product ▲	Platform	Article	Download	Impact	Severity	Supersedence
Windows 10 Version 1903 for 32-bit Systems		4551762	Security Update	Remote Code Execution	Critical	4540673
Windows 10 Version 1903 for ARM64-based Systems		4551762	Security Update	Remote Code Execution	Critical	4540673
Windows 10 Version 1903 for x64-based Systems		4551762	Security Update	Remote Code Execution	Critical	4540673
Windows 10 Version 1909 for 32-bit Systems		4551762	Security Update	Remote Code Execution	Critical	4540673
Windows 10 Version 1909 for ARM64-based Systems		4551762	Security Update	Remote Code Execution	Critical	4540673
Windows 10 Version 1909 for x64-based Systems		4551762	Security Update	Remote Code Execution	Critical	4540673
Windows Server, version 1903 (Server Core installation)		4551762	Security Update	Remote Code Execution	Critical	4540673
Windows Server, version 1909 (Server Core installation)		4551762	Security Update	Remote Code Execution	Critical	4540673

[Windows 취약점 패치]

2. 패치 적용이 어려울 경우 방화벽에서 445 Port 차단 및 PowerShell 스크립트로 SMBv3를 비활성화한다.

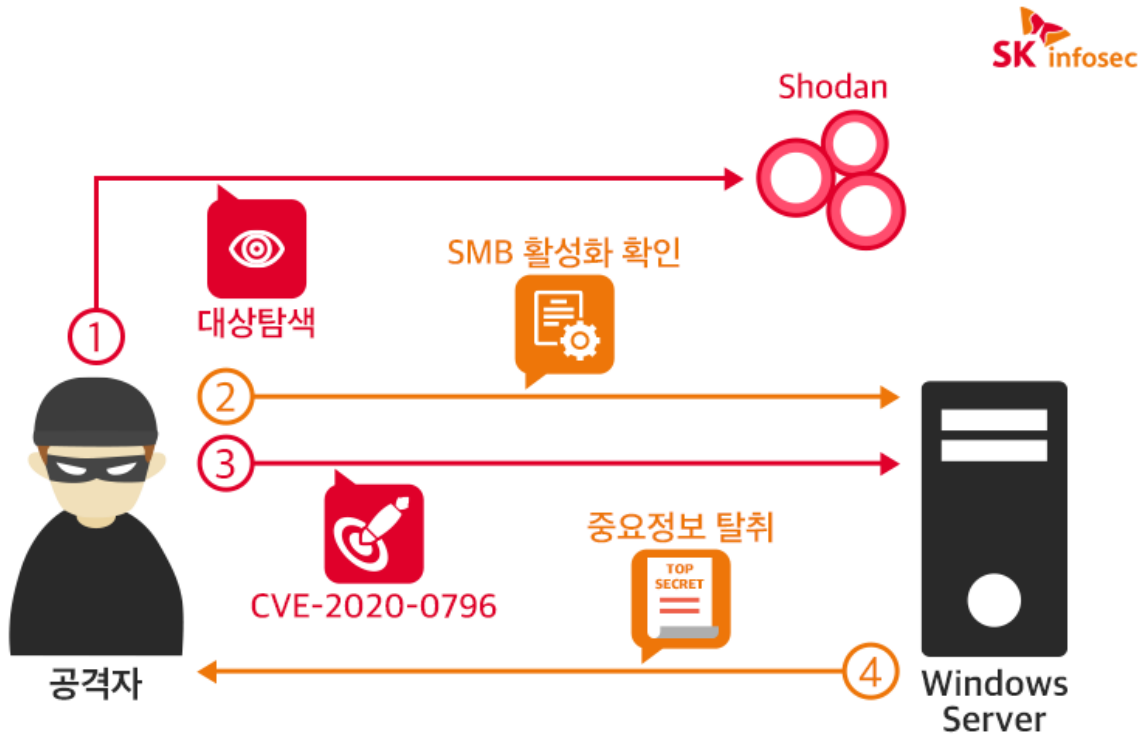
```
Set-ItemProperty -Path
"HKLM:₩SYSTEM₩CurrentControlSet₩Services₩LanmanServer₩Parameters"
DisableCompression -Type DWORD -Value 1 -Force
```

[추후에 Windows Update 후 SMBv3가 필요하면 PowerShell 스크립트로 SMBv3를 활성화할 수 있다.]

```
Set-ItemProperty -Path
"HKLM:₩SYSTEM₩CurrentControlSet₩Services₩LanmanServer₩Parameters"
DisableCompression -Type DWORD -Value 0 -Force
```

■ 예상 시나리오

CVE-2020-0796 취약점을 이용한 시나리오는 다음과 같다.



[공격 시나리오]

① 공격자는 Shodan과 같은 검색 엔진을 활용하여 취약한 Windows를 확인함
② 공격자는 공격 대상이 SMBv3가 활성화되어 있는지 확인함
③ 공격자는 공격 대상에 조작된 SMB 압축 패킷을 전달함
④ 공격을 통해 피해자 PC의 제어권을 탈취하고, 중요정보 탈취

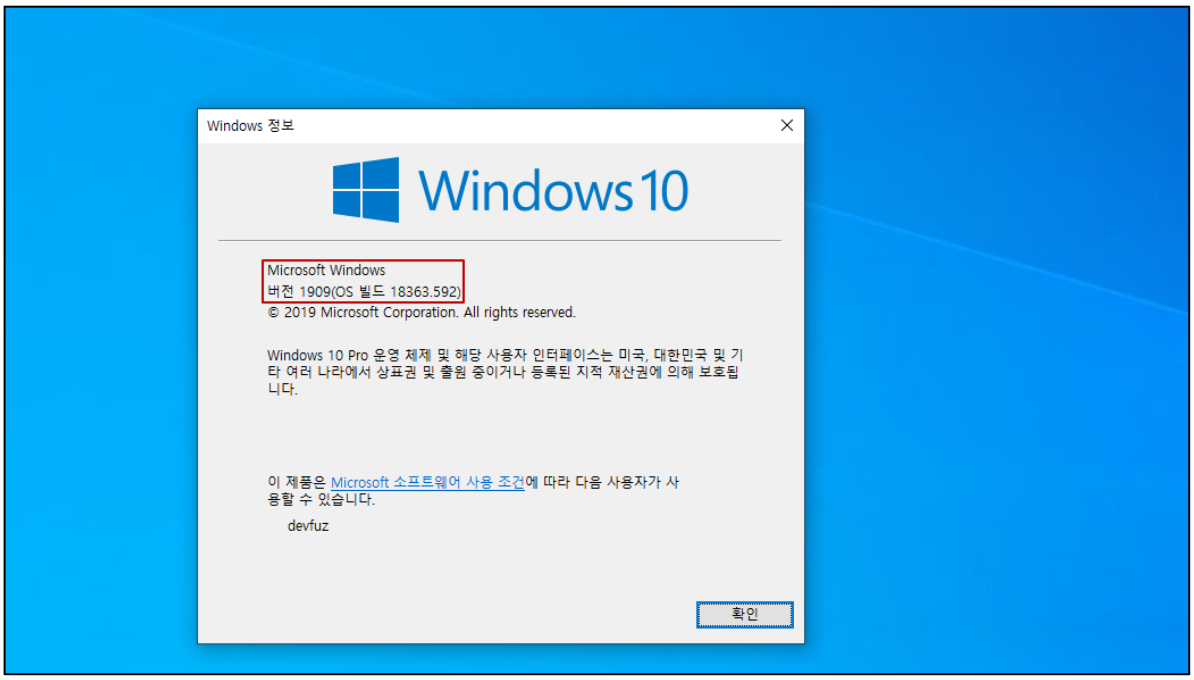
■ 테스트 환경 구성 정보

역할 구분	정보
공격자	Windows 10 1803
피해자	Windows 10 1909

■ 취약점 테스트

Step 1. 취약점 테스트 환경 구축

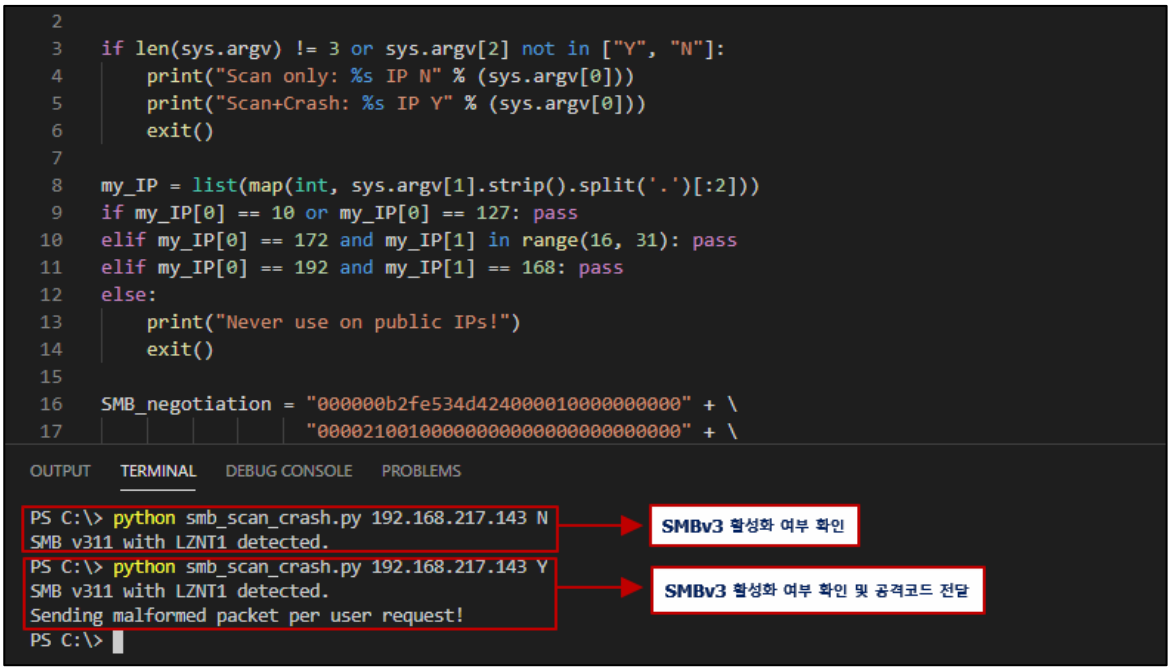
Windows 10 1909버전을 받은 후 취약한 버전인지 확인하고 공격 테스트를 위해 방화벽에서 445 Port를 열었다.



[피해자 Windows 버전 확인]

Step 2. PoC 테스트

공개된 PoC를 이용하여 공격을 수행했다. 스크립트는 피해자가 SMBv3를 사용하는지 확인하고 조작된 SMB 압축 패킷을 전달한다.



[피해자의 SMBv3 활성화 여부 확인 및 공격코드 전달]

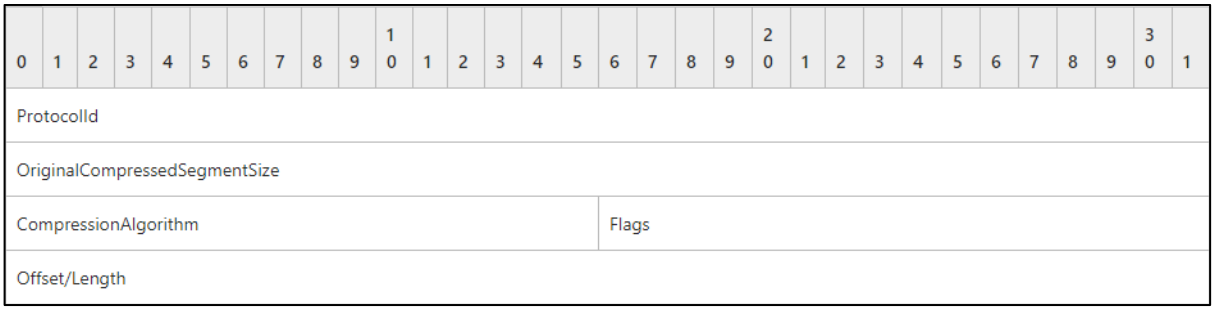
공격이 성공하면 피해자 화면에서 블루스크린이 발생한다.

[피해자 화면 블루스크린 발생]

■ 취약점 상세 분석

1. SMB2 Compression Transform Header

Microsoft에서 제공한 SMB2 Compression Transform Header를 확인하여 압축 패킷을 보낼 때 전달하는 값을 살펴본다. 여기서 공격자가 Buffer Overflow를 위해 조작하는 값은 OriginalCompressedSegmentSize와 Offset/Length이다.



[SMB2 Compression Transform Header]

이름	설명	
ProtocolId (4Bytes)	프로토콜 식별자, 0x424D53FC로 고정	
OriginalCompressedSegmentSize (4Bytes)	압축되지 않은 Data Segment 크기	
CompressionAlgorithm (2Bytes)	메시지 압축 알고리즘	
Flags (2Bytes)	Chain 압축 여부	
Offset/Length (4Bytes)	Flags 1	Length로 해석되며 압축된 페이로드의 길이
	Flags 2	Offset으로 해석되며 구조체 끝에서 압축된 Data Segment 시작까지의 길이

2. 취약점 동작 원리

SMBGhost 취약점이 발생하는 SMB Client는 mrxsmb.sys에서, SMB Server는 srv2.sys에서 수행되지만 둘 다 동일한 코드를 호출한다. 여기서는 srv2.sys 파일과 블루스크린이 발생하는 PoC를 이용하여 분석하였다.

먼저 취약한 srv2.sys와 패치된 srv2.sys를 비교하여 어느 함수에서 취약점이 발생하고 수정되었는지 확인한다.

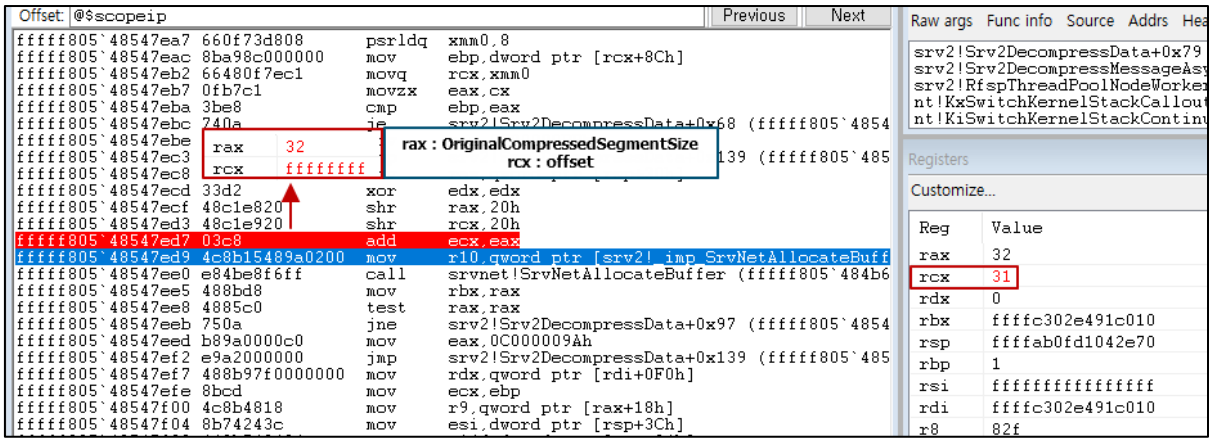
Similarity	Confi	Change	EA Primary	Name Primary	EA Secondary	Name Secondary
0,49	0,85	GI-····	00000001C001···	Srv2DecompressData	00000001C001···	Srv2DecompressData
0,51	0,90	GI-····	00000001C001···	sub_00000001C0017EE5	00000001C001···	sub_00000001C0018068
0,52	0,73	-I--E--	00000001C002···	WPP_SF_LL	00000001C000···	WPP_SF_LLL
0,99	0,99	-I----··	00000001C000···	sub_00000001C0009445	00000001C000···	sub_00000001C0009445
0,99	0,99	-I----··	00000001C005···	sub_00000001C0056C1C	00000001C005···	sub_00000001C0056C1C
0,99	0,99	-I----··	00000001C005···	sub_00000001C005D359	00000001C005···	sub_00000001C005D359
0,99	0,99	-I----··	00000001C005···	sub_00000001C005D38E	00000001C005···	sub_00000001C005D38E
0,99	0,99	-I----··	00000001C005···	sub_00000001C0056A7F	00000001C005···	sub_00000001C0056A7F

[srv2.sys 변경부분 확인]

Srv2DecompressData에 수정된 부분이 존재한다. 해당 함수를 살펴보면 SrvNetAllocateBuffer()를 호출하여 전달한 SMB2 Compression Transform Header의 OriginalCompressedSegmentSize값과 Offset값을 더한 크기만큼 Buffer를 할당한다. 여기서 전달한 값은 OriginalCompressedSegmentSize = 0x32, Offset = 0xffffffff으로 두 값을 더하면 0x100000031이 된다. 이 값은 unsigned int 타입이기 때문에 의도한 값보다 작은 0x31이 된다.

```
"00000042 fc534d42 32000000 0100 0000" + \
# netbios session + protocolId   OriginalCompressedSegmentSize + CompressionAlgorithm + flags
'ffffffff 41414141 4141414141414141" + \
"4141414141414141 4141414141414141" + \
"4141414141414141 4141414141414141" + \
"414141414141"
# offset/length + payload
```

[전달한 SMBv3 압축 헤더 및 데이터]



[OriginalCompressedSegmentSize + Offset 코드]

Buffer가 생성되면 srvnet.sys의 SmbCompressionDecompress()를 호출하여 압축 알고리즘에 따라 해당하는 압축 해제 함수를 호출한다. 그리고 순차적으로 ntoskrnl.exe의 RtlDecompressBufferEx2() 함수를 호출한다. 해당 함수는 압축된 데이터를 압축 해제하는데, Header에서 CompressionAlgorithm 값을 0x0001로 전달했기 때문에 LZNT1압축 해제를 수행한다.

```
; __int64 (__fastcall *RtlDecompressBufferProcs[5])( QWORD,  QWORD,  QWORD, _QWORD, _DWORD, _QWORD, _QWORD)
RtlDecompressBufferProcs dq 2 dup(0), offset RtlDecompressBufferLZNT1, offset RtlDecompressBufferXpressLz
                                        ; DATA XREF: RtlDecompressBufferEx+31↑o
                                        ; RtlDecompressBufferEx2+31↑o ...
                dq offset RtlDecompressBufferXpressHuff
RtlWorkSpaceProcs db    0               ; DATA XREF: RtlGetCompressionWorkSpaceSize+20↑o
```

[알고리즘에 따른 압축 해제 함수 호출]

호출된 RtlDecompressBufferLZNT1()은 위의 overflow로 인해 잘못된 주소에 접근하게 되어 Crash가 발생하게 된다.

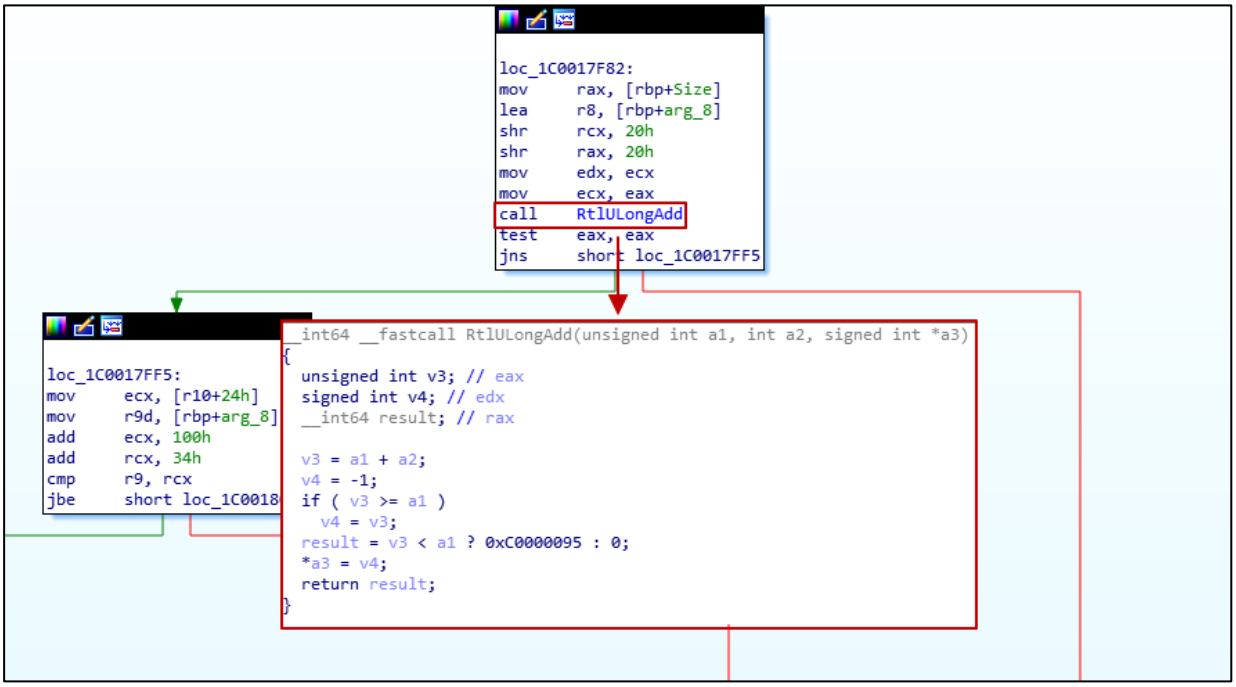
```
Offset: @$scopeip                                                                    Previous  Next
fffff802`192eb6dd 8bd3            mov     edx,ebx
fffff802`192eb6df 488d4de0        lea     rcx,[rbp-20h]
fffff802`192eb6e3 4533c0          xor     r8d,r8d
fffff802`192eb6e6 418bfc          mov     edi,r12d
fffff802`192eb6e9 f30f7f45e8      movdqu  xmmword ptr [rbp-18h],xmm0
fffff802`192eb6ee e8cd5894ff      call    nt!KeInitializeEvent (fffff802`18c30fc0)
fffff802`192eb6f3 48895df8        mov     qword ptr [rbp-8],rbx
fffff802`192eb6f7 0fb71e          movzx   ebx,word ptr [rsi] ds:002b:ffffb907`ad1025cf=????
fffff802`192eb6fa 8bcb            mov     ecx,ebx
fffff802`192eb6fc 44896548        mov     dword ptr [rbp+48h],r12d
fffff802`192eb700 eb73            jmp     nt!RtlDecompressBufferLZNT1+0xd5 (fffff802`192eb775)
fffff802`192eb702 448be1          mov     r12d,ecx
fffff802`192eb705 6685db          test    bx,bx

Customize...
Reg     Value
rax     fffffe8f4eb5ad58
rcx     fffffe8f4eb5ad50
rdx     1
rbx     1
rsp     fffffe8f4eb5ad20
rbp     fffffe8f4eb5ad70
rsi     ffffb907ad1025cf
rdi     0
r8      0

Command
nt!RtlDecompressBufferLZNT1+0x57:
fffff802`192eb6f7 0fb71e          movzx   ebx,word ptr [rsi]
3: kd> r
rax=fffffe8f4eb5ad58 rbx=0000000000000001 rcx=fffffe8f4eb5ad50
rdx=0000000000000001 rsi=ffffb907ad1025cf rdi=0000000000000000
rip=fffff802192eb6f7 rsp=fffffe8f4eb5ad20 rbp=fffffe8f4eb5ad70
 r8=0000000000000000  r9=0000000000000033 r10=fffff802192eb6a0
r11=0000000000000000 r12=0000000000000000 r13=ffffb907ad102602
r14=ffffb907ae35904f r15=ffffb907ae359081
iopl=0         nv up ei pl zr na po nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00040246
nt!RtlDecompressBufferLZNT1+0x57:
fffff802`192eb6f7 0fb71e          movzx   ebx,word ptr [rsi] ds:002b:ffffb907`ad1025cf=????
3: kd> dd @rsi
ffffb907`ad1025cf  ???????? ???????? ???????? ????????
ffffb907`ad1025df  ???????? ???????? ???????? ????????
ffffb907`ad1025ef  ???????? ???????? ???????? ????????
ffffb907`ad1025ff  ???????? ???????? ???????? ????????
ffffb907`ad10260f  ???????? ???????? ???????? ????????
ffffb907`ad10261f  ???????? ???????? ???????? ????????
ffffb907`ad10262f  ???????? ???????? ???????? ????????
ffffb907`ad10263f  ???????? ???????? ???????? ????????
```

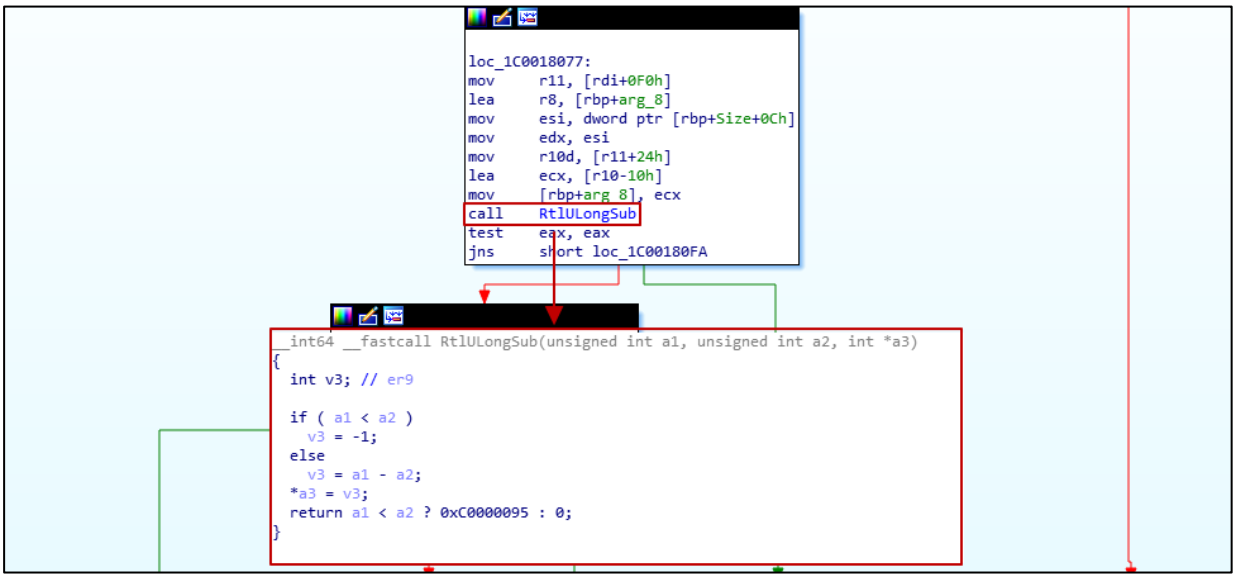
[Crash 발생 부분]

패치된 srv2.sys파일을 확인해보면 RtlULongAdd() 함수와 RtlULongSub() 함수가 추가되었다. RtlULongAdd()는 offset과 OriginalCompressedSegmentSize를 더한 값이 OriginalCompressedSegmentSize보다 큰지 확인한다.



[RtlULongAdd() 취약점 패치 코드]

RtlULongSub()는 압축 버퍼의 크기가 offset보다 큰지 확인한다. 두 조건에 부합하지 않으면 압축 해제를 수행하지 않는다.



[RtlULongSub() 취약점 패치 코드]