No. 35 | 2022.02

우크라이나 주요 기관 및 시설 파괴 HermeticWiper 악성코드

소만사 악성코드 분석 센터

SOMANSA

목 차

1. 개요

1.1 배경

2022년 02월 러시아는 우크라이나를 침공했다. 물리적 침공이 이루어지기전에 사이버 공격이 선제적으로 이루어졌다.

러시아는 우크라이나의 금융, 군수, 정부 사이트에 대해서 대규모 DDos 공격과 시설 파괴형 악성코드를 배포하고 이로 인하여 해당 시설은 심각한 피해를 입었다. 특히 시설 파괴형 악성코드인 와이퍼(Wiper)는 주요 산업 시설에 침투하여 데이터를 삭제하여 파괴하였다고 한다.

소만사는 DDoS 공격과 더불어 진행된 시설 파괴형 악성코드인 와이퍼(Wiper)를 지속적으로 모니터링하여 분석하였고, 본 보고서를 통해 와이퍼(Wiper)의 공격의 특성을 공유하고 동일한 공격을 예방 및 차단할 수 있도록 상세한 내용을 서술하였다.

[그림 1] 우크라이나-러시아 전쟁

[그림 2] 시설 파괴형 악성코드에 감염된 우크라이나의 화학 기업

2. 파일 정보

2.1 [HermeticWiper].exe

Name	[HermeticWiper].exe
Type	PE File
Behavior	System Destroyer
SHA-256	1bc44eef75779e3ca1eefb8ff5a64807dbc942b1e4a2672d77b9f6928d292591
Description	Russian Made System Destroyer Malware

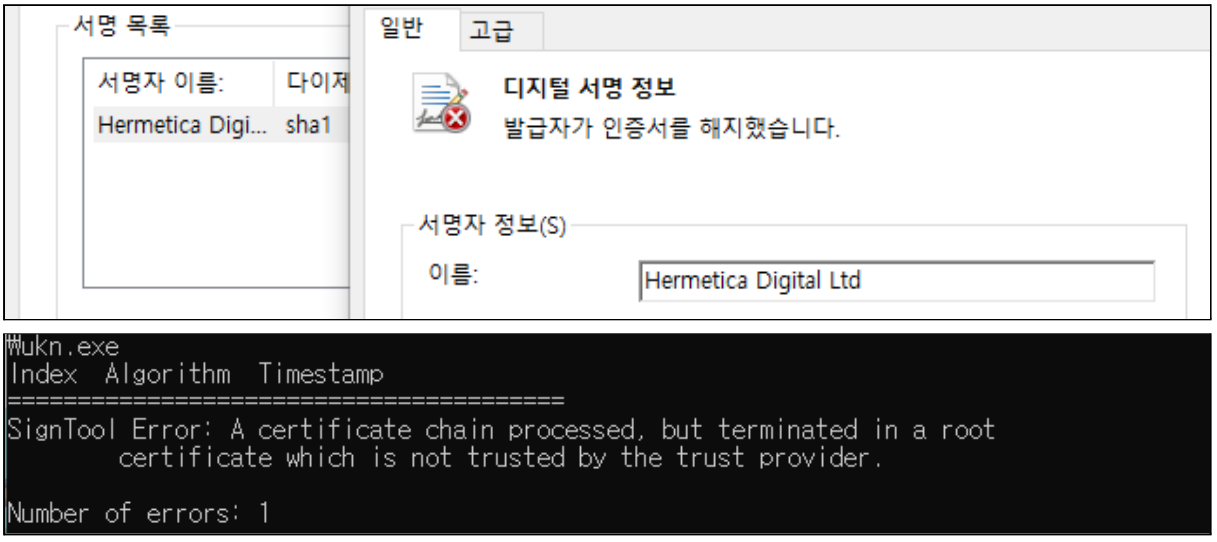
[파일 1] 러시아가 제작한 시스템 파괴형 Wiper 악성코드

2.2 [HermeticWiper].sys

Name	[HermeticWiper].sys
Type	Driver File
Behavior	System Destroyer
SHA-256	0385eeab00e946a302b24a91dea4187c1210597b8e17cd9e2230450f5ece21da
Description	Russian Made System Destroyer Malware

[파일 2] 시스템 파괴형 Wiper 악성코드가 사용하는 파티션 관리자 드라이버 사본

2.3 Hermetica Digital Ltd



[파일 3] 탈취 및 위장 서명된 Hermetica 회사명의 인증서 서명 파일

3. 분석

3.1 프로세스 토큰 획득

```
00793CBD    50              push eax
00793CBE    FF15 30507900   call dword ptr ds:[<&OpenProcessToken>]
00793CC4    85C0            test eax,eax
00793CC6    75 12           jne ukn.793CDA
00793CC8    8B3D 68507900   mov edi,dword ptr ds:[<&GetLastError>]
00793CCE    FFD7            call edi
00793CD0    53              push ebx
00793CD1    6A 00           push 0
00793CD3    FFD6            call esi
00793CD5    E9 CE000000     jmp ukn.793DA8
00793CDA    68 04010000     push 104
```

[그림 4] 프로세스 토큰 권한 획득

시스템을 파괴하기에 앞서, OpenProcessToken API를 사용하여 현재 프로세스의 토큰을 확인하여 시스템 파괴에 필요한 권한을 보유하고 있는지 확인하기 위해 프로세스의 토큰을 얻는다.

3.2 프로세스 토큰 권한 확인

```
00793D19    FF15 D8507900           call dword ptr ds:[<&FindFirstFileW>]
00793D1F    8B3D 68507900           mov edi,dword ptr ds:[<&GetLastError>]
00793D25    FFD7                    call edi
00793D27    8D8424 0C030000         lea eax,dword ptr ss:[esp+30C]
00793D2E    50                      push eax
00793D2F    FF15 6C517900           call dword ptr ds:[<&CharLowerW>]
00793D35    0FB78424 0C030000       movzx eax,word ptr ss:[esp+30C]
00793D3D    8B35 2C507900           mov esi,dword ptr ds:[<&LookupPrivilegeValueW>]
00793D43    C784C4 38FDFFFF 77006   mov dword ptr ss:[esp+eax*8-2C8],6E0077
00793D4E    C784C4 3CFDFFFF 50007   mov dword ptr ss:[esp+eax*8-2C4],720050
00793D59    8D43 04                 lea eax,dword ptr ds:[ebx+4]
```

[그림 5] 프로세스 토큰 권한 확인

FindFirstFileW API를 호출하여 현재 프로세스의 경로를 확인하고, LookupPrivilegeValueW API를 호출함으로 현재 프로세스의 토큰 권한을 확인한다. 이는, 현재 시스템 파괴에 필요한 권한이 미보유 상태 일시 향후 이를 획득하기 위함이다.

3.3 백업 권한 확인 및 상승

```
00793D5D    8D4424 44       lea eax,dword ptr ss:[esp+44]
00793D61    50              push eax
00793D62    6A 00           push 0
00793D64    FFD6            call esi
00793D66    8D43 10         lea eax,dword ptr ds:[ebx+10]
00793D69    50              push eax
00793D6A    68 A8557900     push ukn.7955A8                 7955A8:L"SeBackupPrivilege"
00793D6F    6A 00           push 0
00793D71    FFD6            call esi                        LookupPrivilegeValueW
00793D73    6A 00           push 0
00793D75    6A 00           push 0
00793D77    6A 00           push 0

00793D82    C743 0C 02000000    mov dword ptr ds:[ebx+C],2
00793D89    C743 18 02000000    mov dword ptr ds:[ebx+18],2
00793D90    FF7424 24           push dword ptr ss:[esp+24]
00793D94    FF15 28507900       call dword ptr ds:[<&AdjustTokenPrivileges>]
```

[그림 6] 프로세의 백업 권한 보유 여부 확인 및 상승

LookupPrivilegeValueW API를 호출하며, 주요 권한을 확인하는데 이 중 가장 중요한 백업 권한 보유를 확인한다. 이는 향후 있을 시스템 파괴 행위에 필수적으로 필요한 권한으로서 주요 데이터 백업이라는 긍적적인 기능이 아닌 시스템 파괴 후 백업이라는 기능을 무력화 시키기 위해 확인하는 권한이다. 만약 백업 권한이 없을 경우, 향후 시스템 파괴 행위에 필요한 권한을 획득하지 못하여 시스템 파괴가 불가 할 경우를 가정하여 AdjustTokenPrivileges API를 호출하여 필요한 시스템 백업 권한을 획득한다.

3.4 리다이렉션 여부 및 시스템 환경 확인

```
00792A34    85FF            test edi,edi
00792A36    74 33           je ukn.792A6B
00792A38    8B35 A8507900   mov esi,dword ptr ds:[<&GetProcAddress>]
00792A3E    68 44537900     push ukn.795344                 795344:"Wow64DisableWow64FsRedirection"
00792A43    57              push edi
00792A44    FFD6            call esi
00792A46    68 64537900     push ukn.795364                 795364:"Wow64RevertWow64FsRedirection"
00792A4B    57              push edi
00792A4C    8BD8            mov ebx,eax
00792A4E    FFD6            call esi
00792A50    68 84537900     push ukn.795384                 795384:"IsWow64Process"
```

[그림 7] 리다이렉션 여부 및 시스템 환경 확인

Wow64DisableWow64FsRedirection API를 호출하여 현재 시스템에서 리다이렉션 여부를 확인한다. 이는 특정 시스템 DLL 또는 API를 호출하였을 경우, 시스템 파괴 행위를 위해 목표로 한 대상 DLL 또는 API가 실행될 수 있는지를 확인하는 작업이다. 이와 더불어 IsWow64Process API를 호출하는데 이는 현재 시스템이 x86 또는 x64 환경인지를 확인하여 이에 적합한 시스템 파괴 행위를 하기 위함이다.

3.5 시스템 버전 정보 및 특정 확인

```
00792AA5    6A 03           push 3
00792AA7    6A 02           push 2
00792AA9    6A 00           push 0
00792AAB    6A 00           push 0
00792AAD    FFD6            call esi                        VerSetConditionMask
00792AAF    6A 03           push 3
00792AB1    6A 01           push 1
00792AB3    52              push edx
00792AB4    50              push eax
00792AB5    FFD6            call esi
00792AB7    52              push edx
00792AB8    50              push eax
00792AB9    6A 03           push 3
00792ABB    8D85 80FEFFFF   lea eax,dword ptr ss:[ebp-180]
00792AC1    50              push eax
00792AC2    FF15 B4507900   call dword ptr ds:[<&VerifyVersionInfoW>]
```

[그림 8] 시스템 버전 정보 및 특성 확인

시스템 파괴에 앞서, 다양하게 개발된 Windows OS 버전을 염두하여 완벽하고 치밀한 파괴를 하기 위해 Windows의 OS 버전과 그 버전에 대한 세부적인 특성을 확인한다. VerSetConditionMask API를 호출하여 Windows OS의 특성을 확인하고 그 특성에 따라 VerifyVersionInfoW API를 호출하여 버전을 확인한다. 이를 통해 향후 파괴할 시스템에 대한 세부 사항을 파악하여 더욱 더 교묘한 시스템 파괴를 할 수 있게 된다.

3.6 시스템 환경에 따른 악성 리소스 추출

```
00792AD0    68 94537900     push ukn.795394                 795394:L"RCDATA"
00792AD5    74 07           je ukn.792ADE
00792AD7    68 A4537900     push ukn.7953A4                 7953A4:L"DRV_X64"
00792ADC    EB 36           jmp ukn.792B14
00792ADE    68 B4537900     push ukn.7953B4                 7953B4:L"DRV_X86"
00792AE3    EB 2F           jmp ukn.792B14
00792AE5    FF15 68507900   call dword ptr ds:[<&GetLastError>]
00792AEB    3D 7E040000     cmp eax,47E
00792AF0    0F85 22040000   jne ukn.792F18
00792AF6    837D F8 00      cmp dword ptr ss:[ebp-8],0
00792AFA    C745 E4 01000000 mov dword ptr ss:[ebp-1C],1
00792B01    68 94537900     push ukn.795394                 795394:L"RCDATA"
00792B06    74 07           je ukn.792B0F
00792B08    68 C4537900     push ukn.7953C4                 7953C4:L"DRV_XP_X64"
00792B0D    EB 05           jmp ukn.792B14
00792B0F    68 DC537900     push ukn.7953DC                 7953DC:L"DRV_XP_X86"
00792B14    FF35 80737900   push dword ptr ds:[797380]
00792B1A    FF15 B8507900   call dword ptr ds:[<&FindResourceW>]
```

[그림 9] 시스템 환경에 따른 악성 리소스 추출

리소스 영역 내 포함되어있는 악성 리소스를 추출하는데, 이 때 이전에 구한 시스템 환경에 따라 악성 리소스를 추출한다. 추출한 리소스는 향후 시스템 파괴에 사용된다.

3.7 악성 리소스 추출 및 메모리 내 적재

```
00792B2B    FF35 80737900       push dword ptr ds:[797380]
00792B31    FF15 BC507900       call dword ptr ds:[<&LoadResource>]
00792B37    85C0                test eax,eax
00792B39    0F84 D9030000       je ukn.792F18
00792B3F    50                  push eax
00792B40    FF15 C0507900       call dword ptr ds:[<&LockResource>]
00792B46    8945 EC             mov dword ptr ss:[ebp-14],eax
00792B49    85C0                test eax,eax
00792B4B    0F84 C7030000       je ukn.792F18
00792B51    56                  push esi
00792B52    FF35 80737900       push dword ptr ds:[797380]
00792B58    FF15 C4507900       call dword ptr ds:[<&SizeofResource>]
00792B5E    837D F8 00          cmp dword ptr ss:[ebp-8],0
```

[그림 10] 악성 리소스 추출 및 메모리 내 적재

이전에 추출한 악성 리소스를 실질적으로 추출한다. 먼저, LoadResource와 LockResource API를 통해 메모리 내에 리소스를 추출하여 적재한다. 러시아가 제작한 본 악성코드는 프로그램 내 리소스만 70% 정도 보유하고 있을 정도로 다양한 환경에 대해 대처할 수 있는 악성 리소스를 보유하고 있다.

3.8 덤프 파일 생성 비활성화

```
00792B7B    50                  push eax
00792B7C    68 E0567900         push ukn.7956E0                 7956E0:L"SYSTEM\\CurrentControlSet\\Control\\CrashControl"
00792B81    68 02000080         push 80000002
00792B86    FF15 4C507900       call dword ptr ds:[<&RegOpenKeyW>]
00792B8C    85C0                test eax,eax
00792B8E    75 24               jne ukn.792BB4
00792B90    6A 04               push 4
00792B92    8945 F4             mov dword ptr ss:[ebp-C],eax
00792B95    8D45 F4             lea eax,dword ptr ss:[ebp-C]
00792B98    50                  push eax
00792B99    6A 04               push 4
00792B9B    6A 00               push 0
00792B9D    68 3C577900         push ukn.79573C                 79573C:L"CrashDumpEnabled"
00792BA2    FF75 FC             push dword ptr ss:[ebp-4]
00792BA5    FF15 54507900       call dword ptr ds:[<&RegSetValueExW>]
00792BAB    FF75 FC             push dword ptr ss:[ebp-4]
00792BAE    FF15 50507900       call dword ptr ds:[<&RegCloseKey>]
```

[그림 11] 덤프 파일 생성 비활성화

시스템은 만약의 비정상 상태를 대비하여 이의 원인을 확인할 수 있도록 덤프 파일을 생성하도록 구성되어 있다. 그러나 해당 시스템 파괴형 악성코드는 이러한 원인을 파악할 수 도 없을 정도로 시스템 구성을 변경시킨다. HKLM₩SYSTEM₩CurrentControlSet₩Control₩CrashControl 경로의 CrashDumpEnabled 값을 변경하여 시스템의 마지막 비정상 상태 파악 시도까지 무력화 시킨다.

Path	HKEY_LOCAL_MACHINE₩SYSTEM₩CurrentControlSet₩Control₩CrashControl
Value	CrashDumpEnabled (0)

[표 1] 비정상 상태 시 덤프 파일 생성을 비활성화 시키는 레지스트리 구성

3.9 Named Pipe 생성

```
00792BB4    6A 00           push 0
00792BB6    68 D0517900     push ukn.7951D0                  7951D0:L"\\\\.\\EPMNTDRV\\%u"
00792BBB    8D85 60F9FFFF   lea eax,dword ptr ss:[ebp-6A0]
00792BC1    68 04010000     push 104
00792BC6    50              push eax
00792BC7    FF15 5C517900   call dword ptr ds:[<&wnsprintfW>]
00792BCD    83C4 10         add esp,10
00792BD0    8D8D 60F9FFFF   lea ecx,dword ptr ss:[ebp-6A0]
```

[그림 12] Named Pipe 생성

이 후 Named Pipe를 생성하는데, [₩₩.₩₩EPMNTDRV₩*]와 같은 형태를 보인다. 해당 파이프는 향후 이전의 악성 리소스로 부터 추출된 데이터가 생성하는 악성 드라이버 파일과 통신하기 위해 생성한다.

3.10 파티션 관리자 사본 드라이버 압축 파일 생성

```
00872D69    53              push ebx                         ebx:L"zddr"
00872D6A    50              push eax                         eax:L"\\\\.\\EPMNTDRV\\0"
00872D6B    FF15 78518700   call dword ptr ds:[<&wcsncpy>]
00872D71    8B5D F4         mov ebx,dword ptr ss:[ebp-C]     [ebp-C]:L"C:\\Windows\\sy
00872D74    83C4 0C         add esp,C
00872D77    33C0            xor eax,eax                      eax:L"\\\\.\\EPMNTDRV\\0"
00872D79    66:898475 60F9FFFF mov word ptr ss:[ebp+esi*2-6A0],ax
00872D81    50              push eax                         eax:L"\\\\.\\EPMNTDRV\\0"
00872D82    50              push eax                         eax:L"\\\\.\\EPMNTDRV\\0"
00872D83    6A 02           push 2
00872D85    50              push eax                         eax:L"\\\\.\\EPMNTDRV\\0"
00872D86    50              push eax                         eax:L"\\\\.\\EPMNTDRV\\0"
00872D87    68 00000040     push 40000000
00872D8C    53              push ebx                         ebx:L"zddr"
00872D8D    FF15 7C508700   call dword ptr ds:[<&CreateFileW>]
00872D93    8B3D CC508700   mov edi,dword ptr ds:[<&DeleteFileW>]
00872D99    8BF0            mov esi,eax                      eax:L"\\\\.\\EPMNTDRV\\0"
00872D9B    85F6            test esi,esi
```

```
008837F0  53 5A 44 44 88 F0 27 33 41 00 48 44 00 00 FF 4D  SZDD.ð'3A.HD..ÿM
00883800  5A 90 00 03 00 00 00 7D 04 F5 F0 FF FF 00 00 B8  Z.......}.õðÿÿ...
00883810  F5 F0 A2 01 01 40 01 04 0F 0D 1C 09 F0 F5 F0 0E  õð¢..@......ðõð.
00883820  FF 1F BA 0E 00 B4 09 CD 21 FF B8 01 4C CD 21 54  ÿ.º..´.Í!ÿ..LÍ!T
00883830  68 69 FF 73 20 70 72 6F 67 72 61 FF 6D 20 63 61  hiÿs progrаÿm ca
00883840  6E 6E 6F 74 FF 20 62 65 20 72 75 6E 20 FF 69 6E  nnotÿ be run ÿin
00883850  20 44 4F 53 20 6D FF 6F 64 65 2E 0D 0D 0A 24 FE   DOS mÿode....$þ
00883860  01 04 8A CD 9C 54 CE AC F2 7D 07 74 05 E9 6A 72  ...Í.TÎ¬ò}.t.éjr
00883870  07 CF 7D 02 77 9C 07 CD 75 02 F3 07 D1 7D 02 DD  .Ï}.w..Íu.ó.Ñ}.Ý
00883880  89 8B 02 E9 6A 8F 9B 04 9F 07 D5 C0 7D 02 83 83  ...éj.....ÕÀ}...
00883890  04 8A 83 02 52 69 E3 63 68 74 01 1C 0D D0 05 50  ....Riãcht...Ð.P
008838A0  45 00 FF 00 64 86 06 00 B9 D7 A4 FD 48 24 07 22  E.ÿ.d...'x¤ýH$."
```

[그림 13] 파티션 관리자 사본 드라이버 압축 파일 생성

파티션 관리자 사본 드라이버 압축 파일(zddr)을 생성한다. 해당 행위는 파괴 행위에 앞서, WinAPI를 직접적으로 호출하지 않고 디스크 파티션에 접근하기 위함이다.

3.11 파티션 관리자 사본 드라이버 압축 해제 및 권한 획득

```
lea eax,dword ptr ss:[ebp-410]
push 2
push eax
push ebx                                    ebx:L"C:\\Windows\\system32\\Drivers\\zddr"
call dword ptr ds:[<&LZOpenFileW>]
mov esi,eax
test esi,esi
js ukn.872EF8
push ukn.875258                             875258:L".sys"
lea eax,dword ptr ss:[ebp-388]
push eax
call dword ptr ds:[<&PathAddExtensionW>]
push 1002
lea eax,dword ptr ss:[ebp-498]
push eax
push ebx                                    ebx:L"C:\\Windows\\system32\\Drivers\\zddr"
call dword ptr ds:[<&LZOpenFileW>]
mov dword ptr ss:[ebp-14],eax               [ebp-14]:"SZDD쟁'3A"
```

```
push eax
push ukn.795554                             795554:L"SeLoadDriverPrivilege"
push ebx
call dword ptr ds:[<&LookupPrivilegeValueW>]
push ebx
push ebx
push ebx
push edi
mov dword ptr ds:[edi],1
push ebx
```

[그림 14] 파티션 관리자 사본 드라이버 압축 해제 및 권한 획득

zddr로 명명된 파티션 관리자 사본 드라이버를 LZOpenFileW API 등을 통해 압축 해제하고, 드라이버 제어 수행에 필요한 SeLoadDriverPrivilege 권한을 LookupPrivilegeValueW API 호출을 통해 획득한다.

3.12 시스템 복구 서비스 비활성화

```
00793DCC    56              push esi
00793DCD    FF15 24507900   call dword ptr ds:[<&OpenSCManagerW>]
00793DD3    894424 10       mov dword ptr ss:[esp+10],eax       [esp+10]:EntryPoint
00793DD7    85C0            test eax,eax
00793DD9    75 06           jne ukn.793DE1
00793DDB    FFD7            call edi
00793DDD    8BF0            mov esi,eax
00793DDF    EB 5D           jmp ukn.793E3E
00793DE1    6A 22           push 22
00793DE3    68 B4587900     push ukn.7958B4                     7958B4:L"vss"
00793DE8    50              push eax
00793DE9    FF15 20507900   call dword ptr ds:[<&OpenServiceW>]
```

```
00793E11    6A 04           push 4
00793E13    6A 10           push 10
00793E15    53              push ebx                            ebx:"8셺"
00793E16    FF15 14507900   call dword ptr ds:[<&ChangeServiceConfigW>]
00793E1C    85C0            test eax,eax                        eax:"8셺"
00793E1E    75 04           jne ukn.793E24
00793E20    FFD7            call edi
00793E22    8BF0            mov esi,eax                         eax:"8셺"
00793E24    6A 00           push 0
00793E26    6A 01           push 1
00793E28    53              push ebx                            ebx:"8셺"
00793E29    FF15 04507900   call dword ptr ds:[<&ControlService>]
00793E2F    8B3D 08507900   mov edi,dword ptr ds:[<&CloseServiceHandle>]
```

[그림 15] 시스템 복구 서비스 비활성화

시스템은 시스템 손상 등 훼손 시 시스템을 일정 기간 이전으로 복구 시키는 서비스를 보유하고 있는데, 본 시스템 파괴형 악성코드는 이러한 시도조차 무력화 시키기 위해 해당 서비스를 비활성화시킨다. 위 그림의 VSS로 표기된 서비스는 Volume Shadow Copy Service로서 볼륨의 복제본을 만들어 시스템을 복구할 수 있는 서비스이며, 해당 서비스조차 비활성화하여 완전한 시스템 파괴를 시도한다.

3.13 디스크 유휴 공간 확인

```
007924A5    50                  push eax                                eax:L"C:"
007924A6    8D8424 A0000000     lea eax,dword ptr ss:[esp+A0]
007924AD    50                  push eax                                eax:L"C:"
007924AE    FF15 8C507900       call dword ptr ds:[<&GetDiskFreeSpaceW>]
007924B4    85C0                test eax,eax                            eax:L"C:"
007924B6    0F84 9C010000       je ukn.792658
007924BC    6A 00               push 0
```

[그림 16] 디스크 유휴 공간 확인

GetDiskFreeSpaceW API 호출을 통해 디스크의 남은 유휴 공간을 확인하는데, 이는 이 후 수행 될 시스템 파괴 행위를 위해 시스템의 정보를 획득하는 행위이다.

3.14 드라이브 정보 확인

```
0079251F    6A 00               push 0
00792521    68 00005600         push 560000
00792526    56                  push esi
00792527    FF15 64507900       call dword ptr ds:[<&DeviceIoControl>]
0079252D    85C0                test eax,eax
```

```
00792574    50                  push eax
00792575    68 73000900         push 90073
0079257A    57                  push edi
0079257B    FF15 64507900       call dword ptr ds:[<&DeviceIoControl>]
00792581    FF15 68507900       call dword ptr ds:[<&GetLastError>]
```

[그림 17] 드라이브 정보 확인

DeviceIoControl API를 2회 호출한다. 이 때, 사용되는 인자는 드라이브의 확장 정보를 확인과 드라이브의 데이터를 확인하는 인자로 아래와 같다. 이는 드라이브의 종류에 따라 FAT, NTFS 유형을 구분하여 각기 다른 파괴를 하기 위함이다. 만약, FAT 유형의 드라이브일 경우 비트 피들러를 호출하여 파티션을 손상하며 NTFS 유형의 드라이브일 경우 MFT(Master File Table)을 구문 분석하여 이를 파괴한다.

3.15 드라이브 데이터 암호화 데이터 생성

```
007916A4    C745 F8 00000000    mov dword ptr ss:[ebp-8],0
007916AB    FF15 40507900       call dword ptr ds:[<&CryptAcquireContextW>]
007916B1    85C0                test eax,eax
007916B3    74 31               je ukn.7916E6
007916B5    57                  push edi
007916B6    53                  push ebx
007916B7    FF75 F8             push dword ptr ss:[ebp-8]
007916BA    FF15 3C507900       call dword ptr ds:[<&CryptGenRandom>]
007916C0    85C0                test eax,eax
007916C2    75 17               jne ukn.7916DB
007916C4    85DB                test ebx,ebx
007916C6    74 13               je ukn.7916DB
007916C8    0F1F8400 00000000   nop dword ptr ds:[eax+eax],eax
007916D0    C607 00             mov byte ptr ds:[edi],0
007916D3    8D7F 01             lea edi,dword ptr ds:[edi+1]
007916D6    83EB 01             sub ebx,1
007916D9    75 F5               jne ukn.7916D0
007916DB    6A 00               push 0
007916DD    FF75 F8             push dword ptr ss:[ebp-8]
007916E0    FF15 38507900       call dword ptr ds:[<&CryptReleaseContext>]
```

[그림 18] 드라이브 데이터 암호화 데이터 생성

드라이브 암호화를 수행하여 파괴하기 위해 암호화 API를 호출한다. 이를 통해 연산된 암호화 데이터는 향후 드라이브를 암호화하고 파괴하는데 사용된다.

3.16 드라이브 수 확인

```
007918CC    8D4D F8         lea ecx,dword ptr ss:[ebp-8]
007918CF    51              push ecx
007918D0    6A 0C           push C
007918D2    8D4D EC         lea ecx,dword ptr ss:[ebp-14]
007918D5    51              push ecx
007918D6    6A 00           push 0
007918D8    6A 00           push 0
007918DA    68 80102D00     push 2D1080
007918DF    56              push esi
007918E0    FFD0            call eax                    DeviceIoControl
007918E2    85C0            test eax,eax
```

[그림 19] 드라이브 수 확인

암호화 및 파괴 과정에 앞서, 시스템 내 드라이브의 수를 확인하는데, 해당 과정은 DeviceIoControl API를 호출하며 수행된다. 이 때, [0x2D1080 : IOCTL_STORAGE_GET_DEVICE_NUMBER] 인자를 주어 위 API를 호출하는데 이를 통해 파괴 대상이 되는 드라이브 저장 공간의 개수를 파악한다.

3.17 드라이브(저장매체)의 상세 정보 획득

```
00791924    8D4D C4         lea ecx,dword ptr ss:[ebp-3C]   [ebp-3C]:L"\\\\.
00791927    51              push ecx                        ecx:&L"\\\\.\\PH
00791928    6A 00           push 0
0079192A    6A 00           push 0
0079192C    68 A0000700     push 700A0
00791931    56              push esi
00791932    FFD0            call eax                    DeviceIoControl

00791DF1    6A 00           push 0
00791DF3    68 50000700     push 70050
00791DF8    53              push ebx
00791DF9    FF15 64507900   call dword ptr ds:[<&DeviceIoControl>]
```

[그림 20] 드라이브 상세 정보 획득

이전의 드라이브 저장 공간의 개수를 파악한 뒤, 해당 드라이브에 대해 상세 정보를 획득한다. 해당 작업은 이전처럼 DeviceIoControl API를 호출하여 [0x700A0 : IOCTL_DISK_GET_DRIVE_GEOMETRY_EX]와 [0x70050 : IOCTL_DISK_GET_DRIVE_LAYOUT_EX] 등을 인자로 주어 수행된다. 해당 작업을 통해 본 시스템 파괴형 악성코드는 자신이 파괴할 드라이브에 대해 상세한 정보를 파악하여 철저한 파괴를 수행할 수 있게 된다.

3.18 파일 시스템 영역 획득

```
00791EDF    6A 00           push 0
00791EE1    6A 00           push 0
00791EE3    FF70 0C         push dword ptr ds:[eax+C]
00791EE6    FF70 08         push dword ptr ds:[eax+8]
00791EE9    53              push ebx
00791EEA    FF15 84507900   call dword ptr ds:[<&SetFilePointerEx>]
00791EF0    85C0            test eax,eax
00791EF2    0F84 A5000000   je ukn.791F9D
00791EF8    6A 00           push 0
00791EFA    8D45 F4         lea eax,dword ptr ss:[ebp-C]
00791EFD    50              push eax
00791EFE    FF75 D0         push dword ptr ss:[ebp-30]
00791F01    57              push edi
00791F02    53              push ebx
00791F03    FF15 88507900   call dword ptr ds:[<&ReadFile>]
00791F09    85C0            test eax,eax
```

```
008EFDC8  EB 58 90 4D 53 44 4F 53 35 2E 30 00 02 02 FE 19  ëX.MSDOS5.0...þ.
008EFDD8  02 00 00 00 00 F8 00 00 3F 00 FF 00 00 08 00 00  .....ø..?.ÿ.....
008EFDE8  00 20 03 00 01 03 00 00 00 00 00 00 02 00 00 00  . ..............
008EFDF8  01 00 06 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
008EFE08  80 00 29 C0 F3 B4 82 4E 4F 20 4E 41 4D 45 20 20  ..)Àó´.NO NAME
008EFE18  20 20 46 41 54 33 32 20 20 20 33 C9 8E D1 BC F4    FAT32   3É.Ñ¼ô
008EFE28  7B 8E C1 8E D9 BD 00 7C 88 56 40 88 4E 02 8A 56  {.Á.Ù½.|.V@.N..V
008EFE38  40 B4 41 BB AA 55 CD 13 72 10 81 FB 55 AA 75 0A  @´A»ªUÍ.r..ûUªu.
008EFE48  F6 C1 01 74 05 FE 46 02 EB 2D 8A 56 40 B4 08 CD  öÁ.t.þF.ë-.V@´.Í
008EFE58  13 73 05 B9 FF FF 8A F1 66 0F B6 C6 40 66 0F B6  .s.'ÿÿ.ñf.¶Æ@f.¶
008EFE68  D1 80 E2 3F F7 E2 86 CD C0 ED 06 41 66 0F B7 C9  Ñ.â?÷â.ÍÀí.Af.·É
008EFE78  66 F7 E1 66 89 46 F8 83 7E 16 00 75 39 83 7E 2A  f÷áf.Fø.~..u9.~*
008EFE88  00 77 33 66 8B 46 1C 66 83 C0 0C BB 00 80 B9 01  .w3f.F.f.À.»..'.
008EFE98  00 E8 2C 00 E9 A8 03 A1 F8 7D 80 C4 7C 8B F0 AC  .è,.é¨.¡ø}.Ä|.ð¬
008EFEA8  84 C0 74 17 3C FF 74 09 B4 0E BB 07 00 CD 10 EB  .Àt.<ÿt.´.»..Í.ë
```

[그림 21] 파일 시스템 영역 획득

SetFilePointerEx API와 ReadFile API를 통해 파일 시스템 영역의 첫 번째 영역부터 메모리 내로 적재하며, 파일 시스템 영역의 데이터를 획득한다.

3.19 파일 시스템 영역 암호화

```
007916A4    C745 F8 00000000    mov dword ptr ss:[ebp-8],0
007916AB    FF15 40507900       call dword ptr ds:[<&CryptAcquireContextW>]
007916B1    85C0                test eax,eax
007916B3    74 31               je ukn.7916E6
007916B5    57                  push edi
007916B6    53                  push ebx
007916B7    FF75 F8             push dword ptr ss:[ebp-8]
007916BA    FF15 3C507900       call dword ptr ds:[<&CryptGenRandom>]
007916C0    85C0                test eax,eax
007916C2    75 17               jne ukn.7916DB
007916C4    85DB                test ebx,ebx
007916C6    74 13               je ukn.7916DB
007916C8    0F1F8400 00000000   nop dword ptr ds:[eax+eax],eax
007916D0    C607 00             mov byte ptr ds:[edi],0
007916D3    8D7F 01             lea edi,dword ptr ds:[edi+1]
007916D6    83EB 01             sub ebx,1
007916D9    75 F5               jne ukn.7916D0
007916DB    6A 00               push 0
007916DD    FF75 F8             push dword ptr ss:[ebp-8]
007916E0    FF15 38507900       call dword ptr ds:[<&CryptReleaseContext>]
007916E6    8B56 08             mov edx,dword ptr ds:[esi+8]
```

[그림 22] 파일 시스템 영역 암호화

파일 시스템 영역 파괴에 앞서, 암호화를 수행한다. 위처럼 암호화 관련 API를 호출하여 이전에 획득한 파일 시스템 영역의 데이터를 암호화한다. 이를 통해 부팅 및 주요 시스템을 구성하는 드라이브의 데이터 영역은 완전히 암호화되어 정상적인 역할을 수행할 수 없게 된다. 즉, 해당 시점부터 시스템은 부팅 및 여러가지의 과정을 수행할 수 없는 파괴를 입은 상태이다.

3.20 MFT(Master File Table) 삭제

```
push 0
push 2000000
push 3
push 0
push 1
push 80000000
push esi
call dword ptr ds:[<&CreateFileW>]
```

esi:L"\\\\?\\C:\\System Volume Information::$INDEX_ALLOCATION"

```
lea eax,dword ptr ss:[ebp-E4]
push eax
push esi
call dword ptr ds:[<&GetFileInformationByHandle>]
```

[그림 23] MFT(Master File Table) 삭제

MFT(Master File Table)은 드라이브의 파일 시스템 구조를 DB(DataBase)와 같이 구성하고 있는 테이블로 표현할 수 있다. 지금까지 진행된 시스템 파괴 과정에서 드라이브 파일 시스템 구조를 가지고 있는 MFT를 암호화 및 완전한 파괴를 수행하여 드라이브가 가지고 있는 본질적인 파일 구조 마저 복구할 수 없도록 파괴한다.

식별값	이름	설명
16 (0x10)	$STANDARD_INFORMATION	파일의 최근 생성,접근,수정 시간, 소유자 등의 일반적인 정보
32 (0x20)	$ATTRIBUTE_LIST	속성들에 대한 리스트
48 (0x30)	$FILE_NAME	파일 이름(유니코드), 최근 생성, 접근. 수정 시간
64 (0x40)	$VOLUME_VERSION	볼륨 정보 (윈도우 NT 1.2 버전에만 존재)
64 (0x40)	$OBJECT_ID	파일 및 디렉터리의 16바이트 고유값 (윈도우 2000+)
80 (0x50)	$SECURITY_DESCRIPTOR	파일의 접근 제어와 보안 속성
96 (0x60)	$VOLUME_NAME	볼륨 이름
112 (0x70)	$VOLUME_INFORMATION	파일시스템 버전과 플래그 정보
128 (0x80)	$DATA	파일 내용
144 (0x90)	$INDEX_ROOT	인덱스 트리의 루트 노드 정보
160 (0xA0)	$INDEX_ALLOCATION	인덱스 트리의 루트와 연결된 하위 노드 정보
176 (0xB0)	$BITMAP	$MFT의 비트맵 정보
192 (0xC0)	$SYMBOLIC_LINK	심볼릭 링크 정보 (윈도우 2000+)
192 (0xC0)	$REPARSE_POINT	심볼릭 링크에서 사용하는 Reparse point 정보 (윈도우 2000+)
208 (0xD0)	$EA_INFORMATION	OS/2 응용프로그램과 호환성을 위해 존재 (HPFS)
224 (0xE0)	$EA	OS/2 응용프로그램과 호환성을 위해 존재 (HPFS
256 (0xF0)	$LOGGED_UTILITY_STREAM	암호화된 속성의 정보와 키 값 (윈도우 2000+)

[표 2] NTFS 드라이브 내 MFT(Master File Table) 속성

3.21 Thread 생성 및 우선 순위 변경

```
00793EE0    8B35 9C507900       mov esi,dword ptr ds:[<&CreateThread>]
00793EE6    8D4424 38           lea eax,dword ptr ss:[esp+38]
00793EEA    6A 00               push 0
00793EEC    6A 00               push 0
00793EEE    50                  push eax
00793EEF    68 403B7900         push ukn.793B40
00793EF4    6A 00               push 0
00793EF6    6A 00               push 0
00793EF8    897C24 50           mov dword ptr ss:[esp+50],edi
00793EFC    FFD6                call esi
00793EFE    6A 00               push 0

00793F14    894424 70           mov dword ptr ss:[esp+70],eax
00793F18    8D4424 70           lea eax,dword ptr ss:[esp+70]
00793F1C    50                  push eax
00793F1D    68 D0347900         push ukn.7934D0
00793F22    6A 00               push 0
00793F24    6A 00               push 0
00793F26    FFD6                call esi
00793F28    8B3D D4507900       mov edi,dword ptr ds:[<&SetThreadPriority>]
00793F2E    8BD8                mov ebx,eax
```

[그림 24] Thread 생성 및 우선 순위 변경

특정 Thread를 생성하여 이전의 시스템 파괴 작업 후, 남은 시스템 구조마저 파괴를 시작한다.

3.22 암호화 한 시스템 파일 구조 덮어쓰기

```
00792740    6A 00               push 0
00792742    6A 00               push 0
00792744    52                  push edx
00792745    57                  push edi
00792746    53                  push ebx
00792747    C74424 24 00000000  mov dword ptr ss:[esp+24],0
0079274F    FF15 84507900       call dword ptr ds:[<&SetFilePointerEx>]
00792755    85C0                test eax,eax
00792757    75 06               jne ukn.79275F
00792759    FF15 68507900       call dword ptr ds:[<&GetLastError>]
0079275F    6A 00               push 0
00792761    8D4424 14           lea eax,dword ptr ss:[esp+14]
00792765    50                  push eax
00792766    FF7424 1C           push dword ptr ss:[esp+1C]
0079276A    FF7424 28           push dword ptr ss:[esp+28]
0079276E    53                  push ebx
0079276F    FF15 94507900       call dword ptr ds:[<&WriteFile>]
```

[그림 25] 암호화 한 시스템 파일 구조 덮어쓰기

이전의 암호화한 시스템 파일 구조인 MBR(Master Boot Record) 및 MFT(Master File Table)를 덮어쓴다. 이 때 SetFilePointerEx API를 호출하여 지정된 부분만 덮어쓰는 치밀하고 계산적인 모습을 보인다.

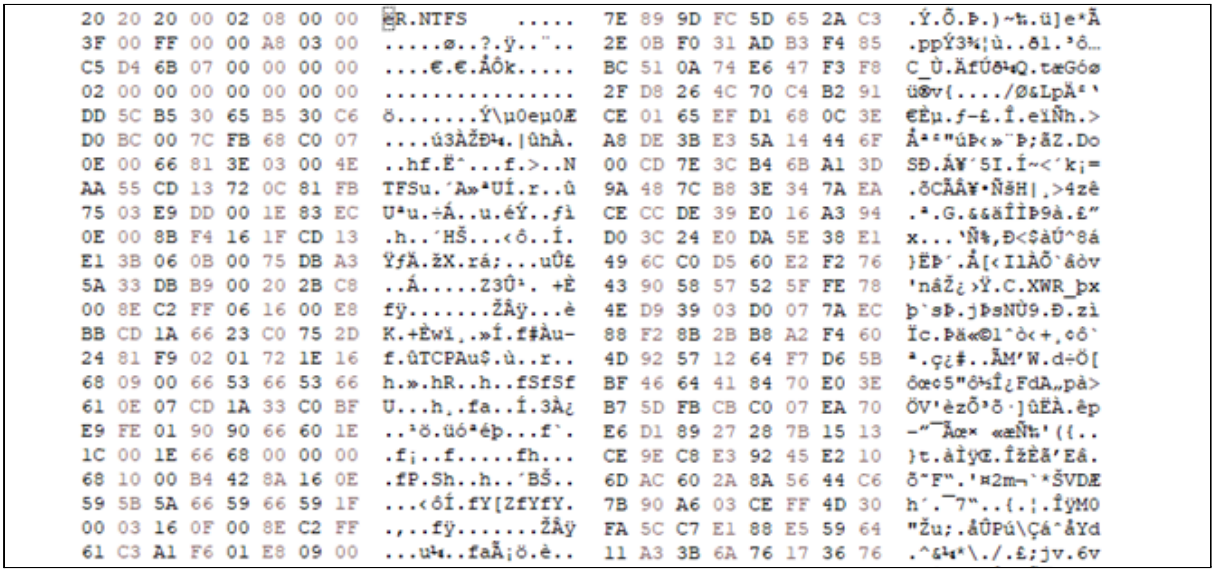
3.23 시스템 재부팅 대기

```
00793B40    55                  push ebp
00793B41    8BEC                mov ebp,esp
00793B43    8B45 08             mov eax,dword ptr ss:[ebp+8]
00793B46    FF30                push dword ptr ds:[eax]
00793B48    FF15 0C517900       call dword ptr ds:[<&Sleep>]
00793B4E    68 03000280         push 80020003
00793B53    6A 01               push 1
00793B55    6A 01               push 1
00793B57    6A 00               push 0
00793B59    6A 00               push 0
00793B5B    6A 00               push 0
00793B5D    FF15 00507900       call dword ptr ds:[<&InitiateSystemShutdownExW>]
```

[그림 26] 시스템 재부팅 대기

시스템 파괴 후, 시스템을 재부팅하여 파괴 작업을 마무리 하기 위해 Thread를 할당하여 대기한다. 모든 작업이 끝나면 InitiateSystemShutdownExW API를 호출하여 시스템을 재부팅한다.

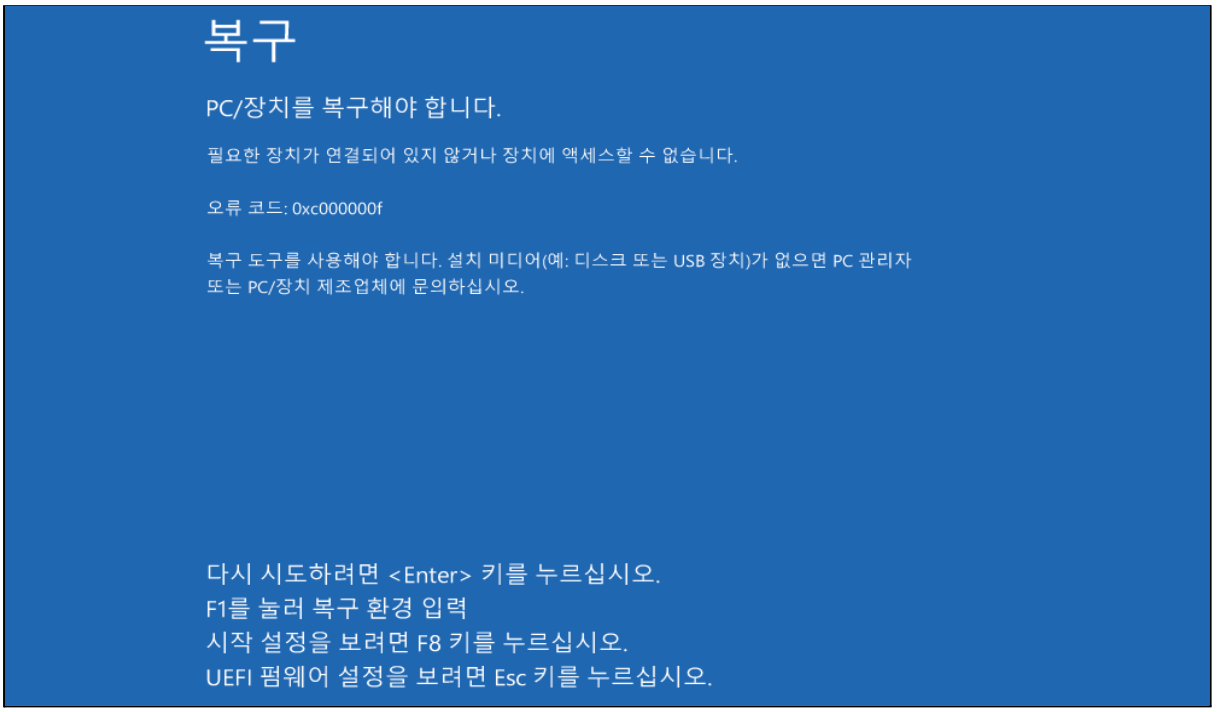
3.24 시스템 파괴



```
20 20 20 00 02 08 00 00  R.NTFS     .....   7E 89 9D FC 5D 65 2A C3  .Ý.Õ.Þ.)~‰.ü]e*Ã
3F 00 FF 00 00 A8 03 00  .....ø..?.ÿ..¨..   2E 0B F0 31 AD B3 F4 85  .ppÝ3¾¦ù..ð1.³ô…
C5 D4 6B 07 00 00 00 00  ....€.€.ÅÔk.....   BC 51 0A 74 E6 47 F3 F8  C_Ù.ÄfÚð¼Q.tæGóø
02 00 00 00 00 00 00 00  ................   2F D8 26 4C 70 C4 B2 91  ü®v{..../Ø&LpÄ²'
DD 5C B5 30 65 B5 30 C6  ö.......Ý\µ0eµ0Æ   CE 01 65 EF D1 68 0C 3E  €Ëµ.ƒ-£.Î.eïÑh.>
D0 BC 00 7C FB 68 C0 07  ....ú3ÀŽÐ¼.|ûhÀ.   A8 DE 3B E3 5A 14 44 6F  Å²³"úÞ‹»¨Þ;ãZ.Do
0E 00 66 81 3E 03 00 4E  ..hf.Ë^...f.>..N   00 CD 7E 3C B4 6B A1 3D  SÐ.Á¥´5I.Í~<´k¡=
AA 55 CD 13 72 0C 81 FB  TFSu.´A»ªUÍ.r..û   9A 48 7C B8 3E 34 7A EA  .õCÃÅ¥•ÑšH|¸>4zê
75 03 E9 DD 00 1E 83 EC  Uªu.÷Ã..u.éÝ..ƒì   CE CC DE 39 E0 16 A3 94  .ª.G.&&äÎÌÞ9à.£"
0E 00 8B F4 16 1F CD 13  .h..´HŠ...‹ô..Í.   D0 3C 24 E0 DA 5E 38 E1  x...'Ñ¼,Ð<$àÚ^8á
E1 3B 06 0B 00 75 DB A3  ŸƒÄ.žX.rá;...uÛ£   49 6C C0 D5 60 E2 F2 76  }ÉÐ´.Å[<IlÀÕ`âòv
5A 33 DB B9 00 20 2B C8  ..Á.....Z3Û¹. +È   43 90 58 57 52 5F FE 78  'náŽ¿›Ÿ.C.XWR_þx
00 8E C2 FF 06 16 00 E8  fÿ.......ŽÂÿ...è   4E D9 39 03 D0 07 7A EC  þ`sÞ.jÞsNÙ9.Ð.zì
BB CD 1A 66 23 C0 75 2D  K.+È»Í¸.»Í.f#Àu-   88 F2 8B 2B B8 A2 F4 60  Ïc.Þä«®l^ò‹+¸¢ô`
24 81 F9 02 01 72 1E 16  f.ûTCPAu$.ù..r..   4D 92 57 12 64 F7 D6 5B  ª.ç¿#..ÃM'W.d÷Ö[
68 09 00 66 53 66 53 66  h.».hR..h..fSfSf   BF 46 64 41 84 70 E0 3E  ôœ¢5"ö¼Î¿FdA„pà>
61 0E 07 CD 1A 33 C0 BF  U...h,.fa..Í.3À¿   B7 5D FB CB C0 07 EA 70  ÖV'èzÕ'ð·]ûËÀ.êp
E9 FE 01 90 90 66 60 1E  ..³ö.üó³éþ...f`.   E6 D1 89 27 28 7B 15 13  –"¯Ãœ× «æÑ‰'({..
1C 00 1E 66 68 00 00 00  .f¡..f.....fh...   CE 9E C8 E3 92 45 E2 10  }t.àÌÿŒ.ÎžÈã'Eâ.
68 10 00 B4 42 8A 16 0E  .fP.Sh..h..´BŠ..   6D AC 60 2A 8A 56 44 C6  õ~F".'¤2m¬`*ŠVDÆ
59 5B 5A 66 59 66 59 1F  ...‹ôÍ.fY[ZfYfY.   7B 90 A6 03 CE FF 4D 30  h´.¯7"..{.¦.ÎÿM0
00 03 16 0F 00 8E C2 FF  .,..fÿ.......ŽÂÿ   FA 5C C7 E1 88 E5 59 64  "Žu;.åÛPú\Çá^åYd
61 C3 A1 F6 01 E8 09 00  ...u¼..faÃ¡ö.è..   11 A3 3B 6A 76 17 36 76  .^&¼*\./.£;jv.6v
```

[그림 27] 파괴된 시스템 파일 구조

시스템 파괴 전과 후를 확인하면 위 그림처럼 복구가 불가능하도록 MBR(Master Boot Record)가 암호화 및 특정 데이터로 변경된 것을 확인할 수 있으며 MFT(Master File Table) 또한 파괴된 것을 확인할 수 있다.

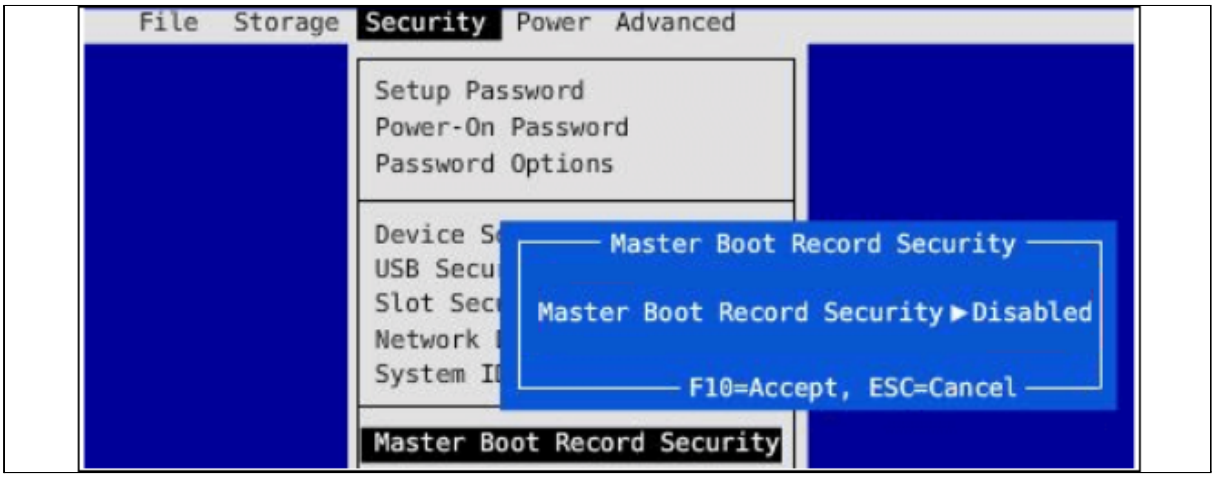


[그림 28] 시스템 파괴

재부팅 후에는 시스템 내 MBR(Master Boot Record) 및 MFT(Master File Table) 등이 모두 파괴되어 정상적인 부팅이 불가능하며, 위와 같이 시스템 복구가 필요하다는 화면이 생성된다. 그러나 이미 모든 과정 안에서 시스템의 구조가 파괴되어 복구는 불가능하다.

4. 대응

4.1 MBR 파괴를 방지할 수 있는 보안 설정을 유지한다.



[그림 29] MBR(Master Boot Record) 보안 설정

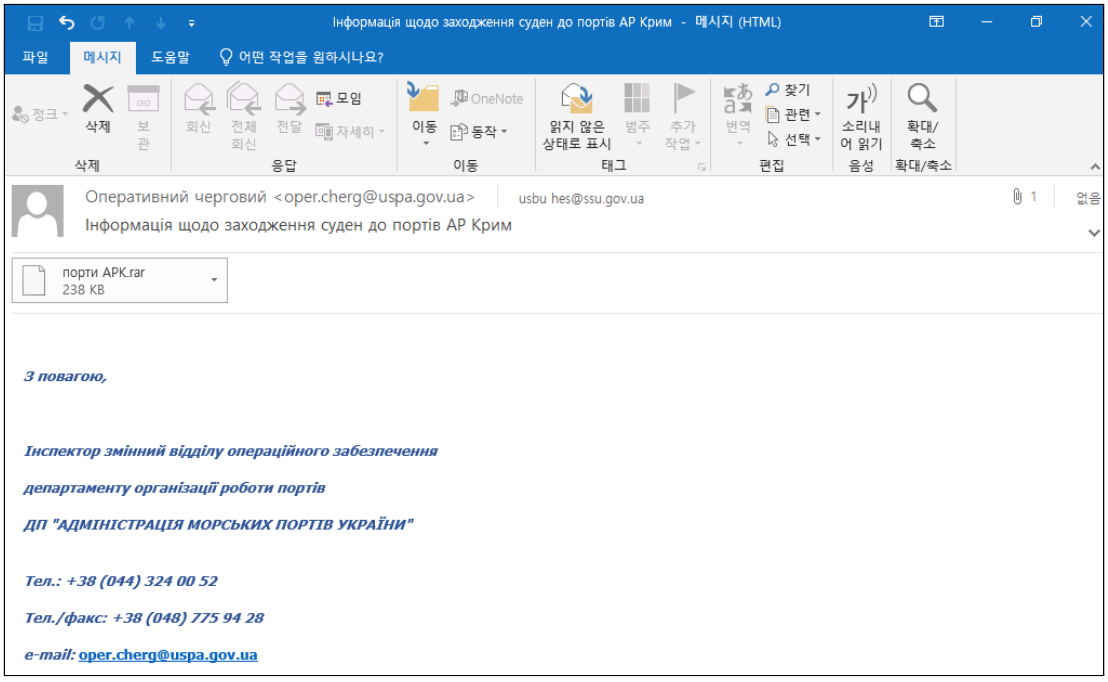
각 BIOS 제조사는 MBR(Master Boot Record) 손상에 대해 위와 같이 보호 설정을 지원하고 있다. 설정은 BIOS에 진입 후 보안 설정의 MBR(Master Boot Record) Security를 활성화시키면 되는데, BIOS 제조사 별 BIOS 진입 방법은 PC 재부팅 후 [F2] 등 제조사가 지정한 키를 눌러 진입할 수 있다.

1. PC 켜기 또는 다시시작 2. BIOS 메뉴 진입 (제조사 별 상이) 3. BIOS 메뉴 내 보안 탭으로 이동 4. Master Boot Record 보안 설정으로 이동 5. 사용(Enabled)로 변경 6. Master Boot Record 저장 선택 7. 변경 사항 저장 8. BIOS 메뉴 종료 및 재부팅

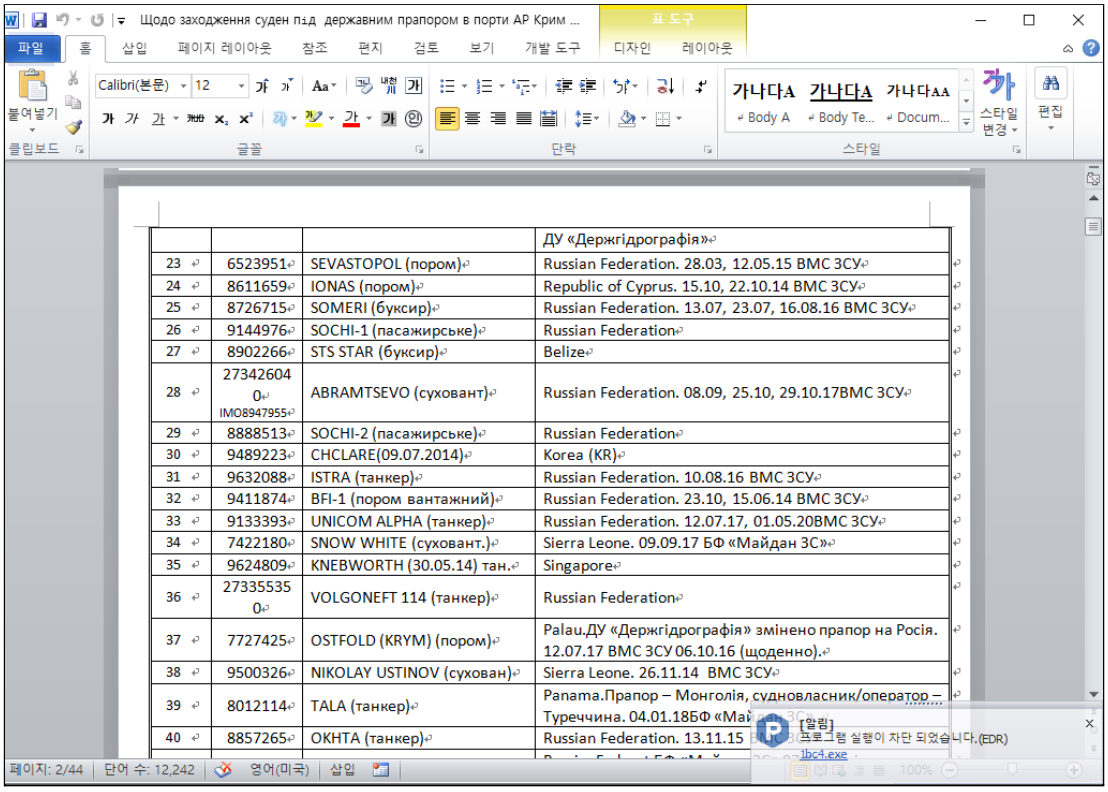
[표 3] 바이오스 진입 및 MBR(Master Boot Record) 보안 설정 방법

위 과정을 수행하면 MBR(Master Boot Record)를 파괴하는 악성코드에 대한 보안 수준을 높일 수 있다. 이와 더불어 PC를 시작할 때마다 BIOS는 현재 부팅 가능한 드라이브의 MBR(Master Boot Record)와 이전에 저장된 MBR(Master Boot Record)를 비교하여 변경 사항이 감지 될 시 오류 메시지가 표시된다. 이 때, 사용자는 현재 MBR(Master Boot Record)의 상태를 저장하거나, 이전에 저장된 상태를 복원할 수 있고 보안 기능을 비활성화할 수 있다.

4.2 Anti-Virus 및 EDR 제품을 사용하여 해당 악성코드를 탐지하여 대응 한다.



[그림 30] HermeticWiper 악성코드가 첨부된 이메일



[그림 32] Privacy-i EDR 안티바이러스 엔진을 통한 악성코드 차단

탐지 일시	탐지 종류	검사 유형	파일 경로	파일 해시	탐지 정보
2022-02-28 15:50:31	안티 바이러스	실시간	C:\Users\test\Desktop\1bc4.exe	1bc44eef75779e3ca1eefb8ff5a64807dbc942b1e4a2672d77b9f6928d292591	Trojan.HermeticWiper.A

[그림 32] Privacy-i EDR 안티바이러스 엔진을 통한 악성코드 탐지 로그

4.3 주요 문서는 주기적으로 백업하고 물리적으로 분리하여 관리한다.

4.4 비 업무 사이트 및 신뢰 할 수 없는 웹사이트의 연결을 차단한다.

궁금하신 사항이 있으실 경우 malware@somansa.com 으로 문의주십시오.