다형성 바이러스 분석 방법

㈜ AhnLab / ASEC
한창규 선임 연구원

안철수연구소

목차

Ⅰ. 바이러스 개요

1. 바이러스 정의
2. 바이러스 분류
3. 바이러스 발전 단계
4. 감염 기법

1.바이러스 정의

❑악성코드 정의

✓ The Malware (for "malicious software") is any program or file that is harmful to a computer user.
Thus, malware includes computer Viruses, Worms, Trojan horses, and also SpyWare, programming that gathers information about a computer user without permission. (영문 Terms)

❑바이러스 정의

✓ We define a computer 'virus' as a program can 'infect' other programs by modifying them to include a possibly evolved copy of itself. [Fred Cohen 1984]

2.바이러스 분류

바이러스 분류

기 준	세 부 내 용
감염 대상	■ 부트바이러스 (부트 섹터 감염, Brain, Monkey, Anti-CMOS) ■ 파일 바이러스 (COM, EXE 실행파일 감염, 예루살렘, Sunday, Scorpion, Crow) ■ 부트/파일 바이러스 (부트섹터, 파일 모두 감염, Invader(1990), 안락사(Euthanasia)) ■ 매크로 바이러스 (XM/Laroux)
운영 체제	■ DOS 바이러스 (Brain, 예루살렘, 미켈란 젤로 등등) ■ Window 바이러스 (NE 계열 바이러스, PE 계열 바이러스) ■ Unix 바이러스 ■ 기타
감염 위치	■ 기생형 바이러스 (Parasitic Virus) ■ 겹쳐쓰기형 바이러스 (Overwriting Virus) ■ 산란형 바이러스 (Spawning) ■ 연결형 바이러스 (linking Virus)
동작 원리	■ 상주형 바이러스 (Resident Virus) ■ 비상주형 바이러스 (Non-Resident Virus)

3.바이러스 발전 단계

제 1세대 (Primitive)

- 단순한 구조, 분석 용이
- Stoned Virus, Jerusalem Virus

제 2세대 (Encryption)

- 암호화 기법 사용
- Cascade Virus, Slow Virus

제 3세대 (Stealth)

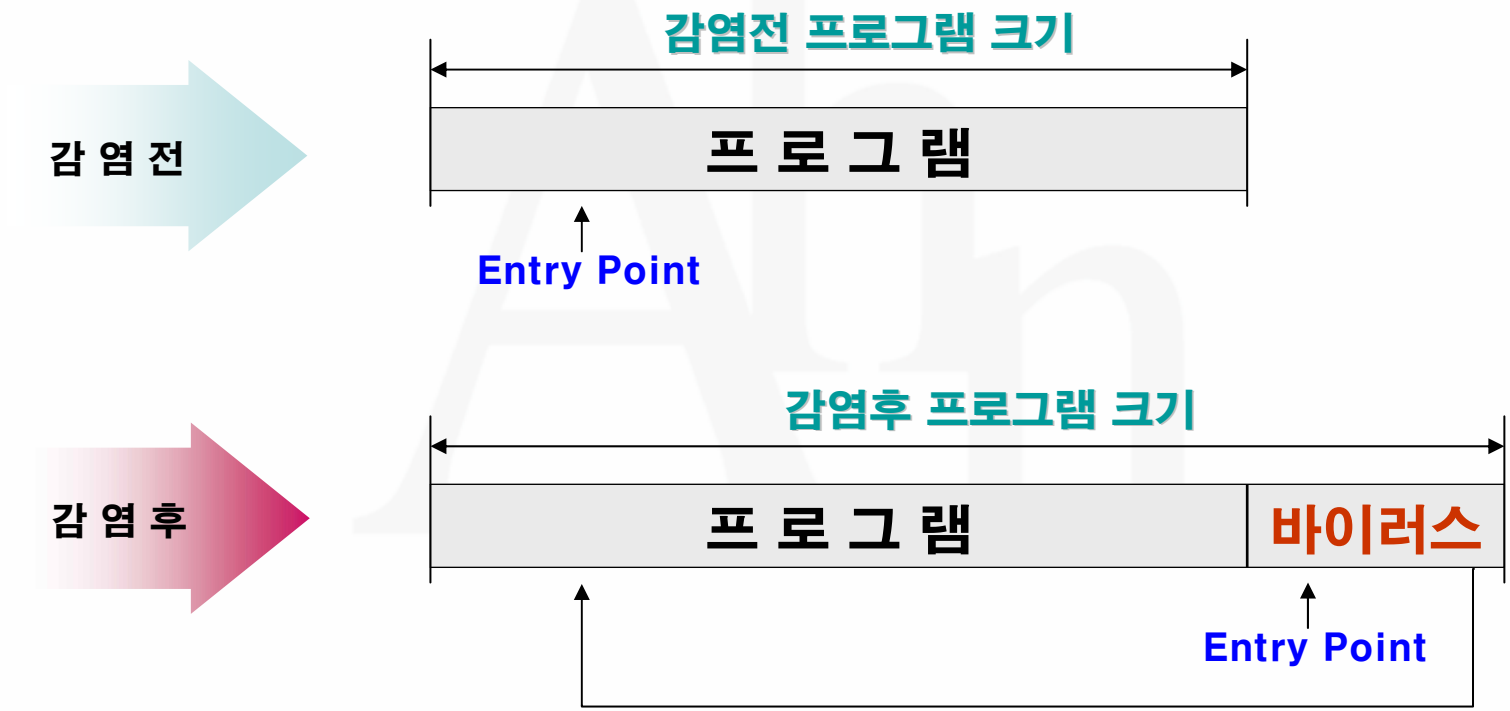
- 은폐 기법 사용
- Joshi Virus, Wanderer Virus

제 4세대 (Armour)

- 분석 방해 (Polymorphic, EPO 기법)
- Whale Virus

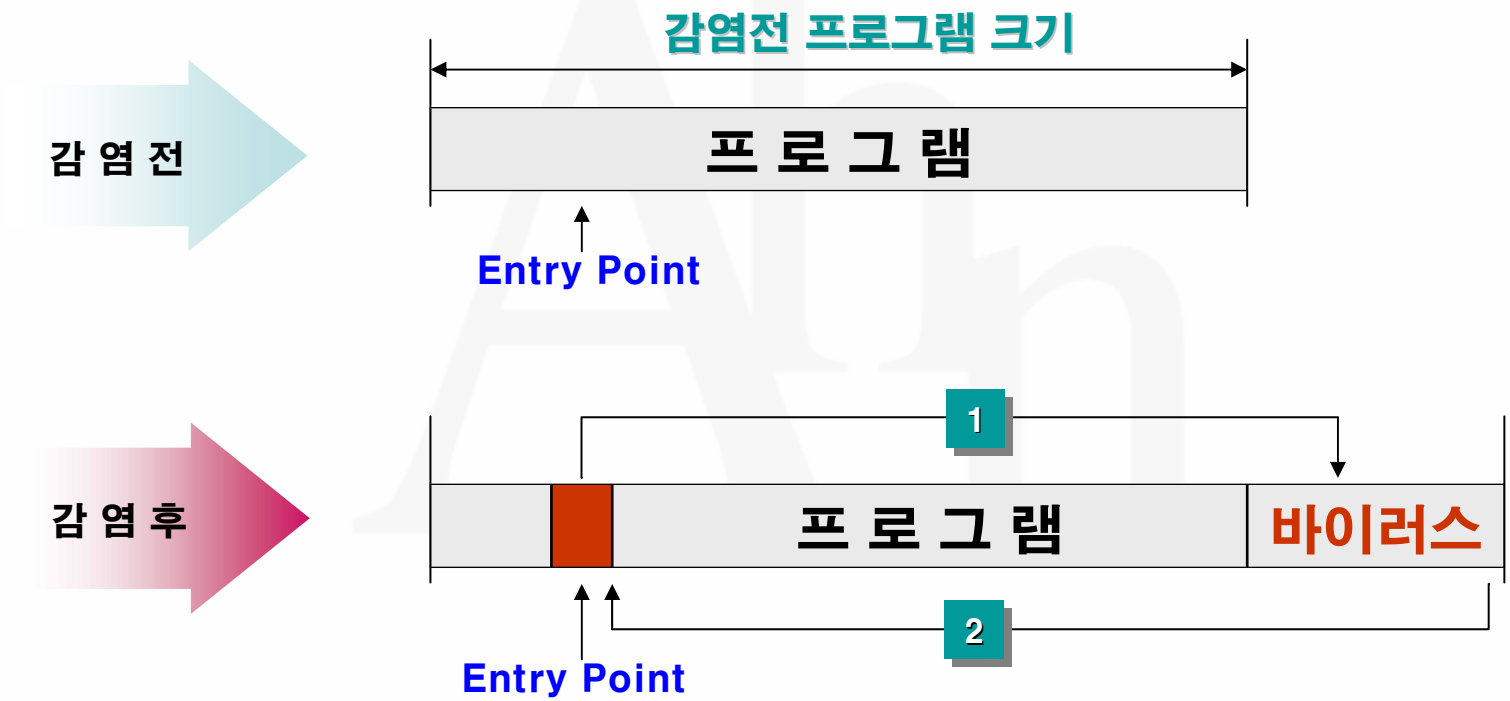
4.감염 기법 (1/4)

Entry Point를 수정하는 방법



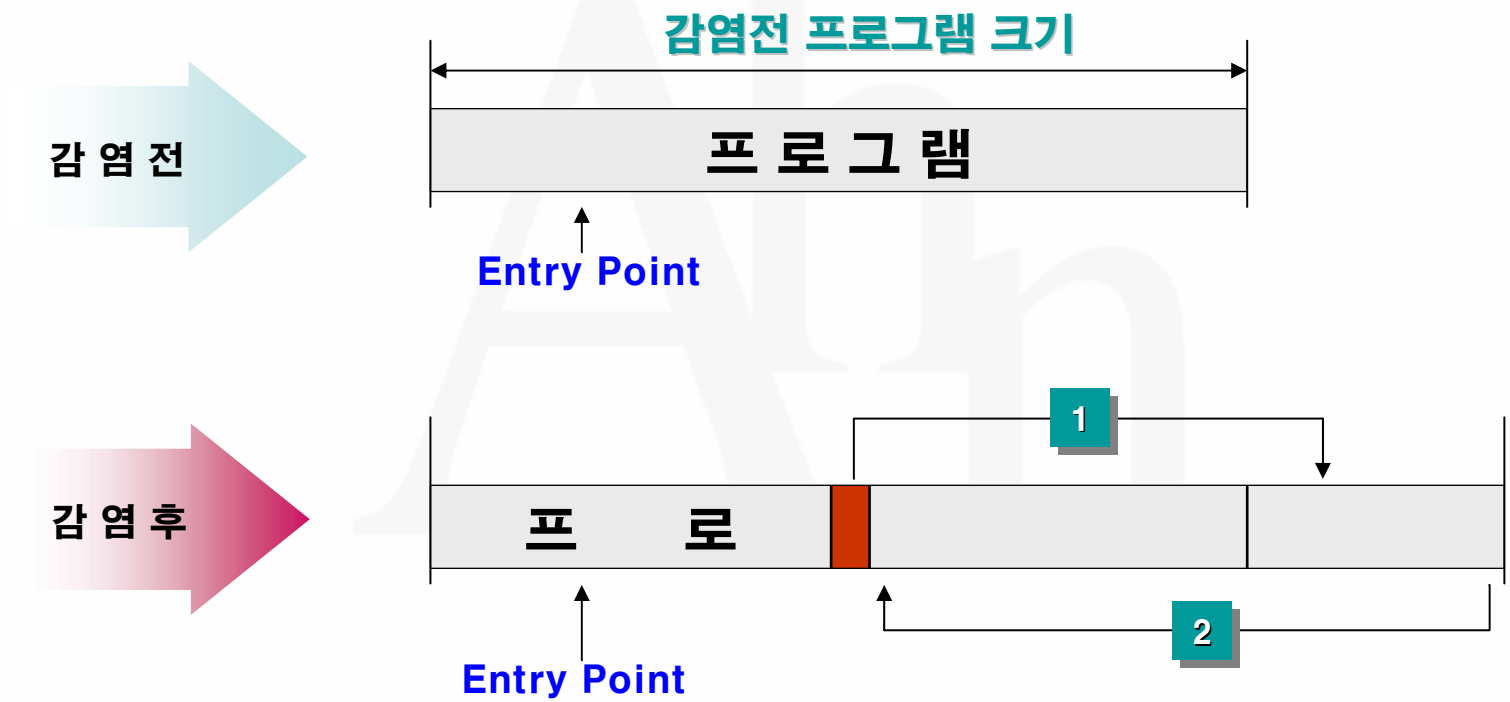
4.감염 기법 (2/4)

Entry Point 부분의 코드를 수정하는 방법



4.감염 기법 (3/4)

EPO (Entry Point Obscuring) 기법



4.감염 기법 (4/4)

EPO (Entry Point Obscuring) 기법

[감염전 코드]

```
////////////////////////////////////////////////////////////////////////
// 원본 코드
////////////////////////////////////////////////////////////////////////
0040681A    . FF15 E4114000  CALL DWORD PTR DS:[<&USER32.CreateWindow>
00406820    .^E9 45D3FFFF    JMP msn6.00403B6A
00406825    >-FF25 EC114000  JMP DWORD PTR DS:[<&USER32.DefWindowProc>
0040682B    > FF75 08        PUSH DWORD PTR SS:[EBP+8]
0040682E    . FF15 4C104000  CALL DWORD PTR DS:[<&KERNEL32.DeleteFile>
```

[감염후 코드]

```
////////////////////////////////////////////////////////////////////////
// 바이러스에 의해 수정된 코드
////////////////////////////////////////////////////////////////////////
0040681A    . FF15 E4114000  CALL DWORD PTR DS:[<&USER32.CreateWindow>
00406820    .^E9 45D3FFFF    JMP msn6.00403B6A
00406825    >-E9 E09A0100    JMP msn6.0042030A
0040682A      00             DB 00
0040682B    > FF75 08        PUSH DWORD PTR SS:[EBP+8]
0040682E    . FF15 4C104000  CALL DWORD PTR DS:[<&KERNEL32.DeleteFile>
```

Ⅱ. 바이러스 분석

1. 사전 지식
2. 분석 과정
3. 진단법

1.사전 지식(1/14)

분석 도구

- 기본 툴
  - Frhed HexaEditor, UltraEdit
  - Process Explorer
  - PEID, LordPE
  - Dependency Walker
- 모니터링 툴
  - FileMon
  - RegMon
  - TcpViewer, NetStat
  - Analyzer, Etherreal
- 디버깅 / 디어셈플 툴
  - Dos Debuger
  - OllyDbg
  - IDA Pro
  - WinDbg
  - SoftICE

운영체제 / 통신

- 운영체제
  - Process / Thread, DLL
  - 가상메모리 / Memory Mapping
  - 드라이버, 서비스 프로세스
  - Kernel/Synchronization Object
  - Structured Exception Handling
- 통신
  - TCP / UDP 프로토콜
  - 통신 프로그램 구조
  - Socket API

분석 기본 지식

- StartUp Code
- 실행 파일 구조 (EXE, NE, PE)
- 실행 압축
- DLL Injection 기법
- Stealth 기법
- Windows File Protection

1.사전 지식(2/14)

StartUp Code

[메인 덧셈 프로그램]

```
int Add(int a, int b) { return a+b; }
int main(int argc, char* argv[])
{
    int nResult, nNum1 = 1, nNum2 = 2;
    nResult = Add(nNum1, nNum2);

    printf("%d + %d = %d", nNum1, nNum2, nResult);

    return 0;
}
```

```
CALL Add.00401005
ADD ESP,8
MOV DWORD PTR SS:[EBP-C],EAX
MOV EDX,DWORD PTR SS:[EBP-C]
PUSH EDX                                <%d>
MOV EAX,DWORD PTR SS:[EBP-8]
PUSH EAX                                <%d>
MOV ECX,DWORD PTR SS:[EBP-4]
PUSH ECX                                <%d>
PUSH OFFSET Add.??_C@_0N@PEGD@  format = "%d + %d = %d"
CALL Add.printf                         printf
ADD ESP,10
```

[StartUp Code]

```
CALL Add._ioinit
CALL DWORD PTR DS:[<&KERNEL32.  [GetCommandLineA
MOV DWORD PTR DS:[_acmdln],EAX
CALL Add.__crtGetEnvironmentSt
MOV DWORD PTR DS:[_aenvptr],EA
CALL Add._setargv
CALL Add._setenvp
CALL Add._cinit
MOV ECX,DWORD PTR DS:[_environ
MOV DWORD PTR DS:[__initenv],E
MOV EDX,DWORD PTR DS:[_environ
PUSH EDX
MOV EAX,DWORD PTR DS:[__argv]
PUSH EAX
MOV ECX,DWORD PTR DS:[__argc]
PUSH ECX
CALL Add.0040100A
ADD ESP,0C
MOV DWORD PTR SS:[EBP-1C],EAX
MOV EDX,DWORD PTR SS:[EBP-1C]
PUSH EDX                                [status
CALL Add.exit                           [exit
```

1.사전 지식(3/14)

실행파일 구조

- Test.dll
  - IMAGE_DOS_HEADER
  - MS-DOS Stub Program
  - IMAGE_NT_HEADERS
    - Signature
    - IMAGE_FILE_HEADER
    - IMAGE_OPTIONAL_HEADER
  - IMAGE_SECTION_HEADER .text
  - IMAGE_SECTION_HEADER .rdata
  - IMAGE_SECTION_HEADER .data
  - IMAGE_SECTION_HEADER .reloc
  - SECTION .text
  - SECTION .rdata
    - IMPORT Address Table
    - IMPORT Directory Table
    - IMPORT Name Table
    - IMPORT Hints/Names & DLL Nam
    - IMAGE_EXPORT_DIRECTORY
    - EXPORT Address Table
    - EXPORT Name Pointer Table
    - EXPORT Ordinal Table
    - EXPORT Names
  - SECTION .data
  - SECTION .reloc

pFile	Data	Description
000000F0	010B	Magic
000000F2	06	Major Linker Version
000000F3	00	Minor Linker Version
000000F4	00003000	Size of Code
000000F8	00003000	Size of Initialized Data
000000FC	00000000	Size of Uninitialized Data
00000100	00001109	Address of Entry Point
00000104	00001000	Base of Code
00000108	00004000	Base of Data
0000010C	10000000	Image Base
00000110	00001000	Section Alignment
00000114	00001000	File Alignment
00000118	0004	Major O/S Version
0000011A	0000	Minor O/S Version
0000011C	0000	Major Image Version
0000011E	0000	Minor Image Version
00000120	0004	Major Subsystem Version
00000122	0000	Minor Subsystem Version
00000124	00000000	Win32 Version Value
00000128	00007000	Size of Image
0000012C	00001000	Size of Headers
00000130	00000000	Checksum
00000134	0002	Subsystem

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
0000h: 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 ; MZ?.....
0010h: B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ; ?......@...
0020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ............
0030h: 00 00 00 00 00 00 00 00 00 00 00 00 D8 00 00 00 ; ............
0040h: 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 ; ..?.???L?Th
0050h: 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F ; is program
0060h: 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 ; t be run in
0070h: 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 ; mode....$...
0080h: 8E 17 EC CC CA 76 82 9F CA 76 82 9F CA 76 82 9F ; ?쏢?귲?귲?
0090h: 22 69 88 9F DC 76 82 9F 49 6A 8C 9F C3 76 82 9F ; "i뉮?귲Ij뷈
00a0h: CA 76 83 9F FB 76 82 9F A8 69 91 9F C9 76 82 9F ; ?걖?귲뛥뮴
00b0h: 22 69 89 9F C9 76 82 9F 22 69 86 9F CB 76 82 9F ; "i듐?귲"i늚
00c0h: 52 69 63 68 CA 76 82 9F 00 00 00 00 00 00 00 00 ; Rich?귲....
00d0h: 00 00 00 00 00 00 00 00 50 45 00 00 4C 01 04 00 ; ........PE..
00e0h: C7 3C 38 45 00 00 00 00 00 00 00 00 E0 00 0E 21 ; ?8E........
00f0h: 0B 01 06 00 00 30 00 00 00 30 00 00 00 00 00 00 ; .....0...0..
0100h: 09 11 00 00 00 10 00 00 00 40 00 00 00 00 10 00 ; .........@..
0110h: 00 10 00 00 00 10 00 00 04 00 00 00 00 00 00 00 ; ............
0120h: 04 00 00 00 00 00 00 00 00 70 00 00 00 10 00 00 ; .........p....
0130h: 00 00 00 00 02 00 00 00 00 00 10 00 00 10 00 00 ; ............
0140h: 00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00 ; ............
0150h: E0 47 00 00 5B 00 00 00 0C 44 00 00 28 00 00 00 ; ?..[....D..(...
0160h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ............
0170h: 00 00 00 00 00 00 00 00 00 60 00 00 A4 03 00 00 ; .........`....
0180h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ............
0190h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ............
01a0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ............
01b0h: 00 40 00 00 B0 00 00 00 00 00 00 00 00 00 00 00 ; .@..?.......
01c0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ............
01d0h: 2E 74 65 78 74 00 00 00 D6 2A 00 00 00 10 00 00 ; .text...?..
```

1.사전 지식(4/14)

실행파일 구조 (Import Directory Table)

pFile	Data	Description	Value
00000158	0000548C	RVA	IMPORT Table
0000015C	0000003C	Size	

[Import Directory Table]

pFile	Data	Description	Value
0000548C	0000556C	Import Name Table RVA	
00005490	00000000	Time Date Stamp	
00005494	00000000	Forwarder Chain	
00005498	00005594	Name RVA	Test.dll
0000549C	000050A4	Import Address Table RVA	

[Import Name Table]

0000556C	00005578	Hint/Name RVA	0001 SubInteger
00005570	00005586	Hint/Name RVA	0000 AddInteger
00005574	00000000	End of Imports	Test.dll

[Import Hints / Names]

pFile	Raw Data	Value
00005578	01 00 53 75 62 49 6E 74 65 67 65 72 00 00 00 00	..SubInteger....
00005588	41 64 64 49 6E 74 65 67 65 72 00 00 54 65 73 74	AddInteger..Test
00005598	2E 64 6C 6C 00 00 CA 00 47 65 74 43 6F 6D 6D 61	.dll....GetComma
000055A8	6E 64 4C 69 6E 65 41 00 74 01 47 65 74 56 65 72	ndLineA.t.GetVer
000055B8	73 69 6F 6E 00 00 7D 00 45 78 69 74 50 72 6F 63	sion..}.ExitProc
000055C8	65 73 73 00 9E 02 54 65 72 6D 69 6E 61 74 65 50	ess...TerminateP

1.사전 지식(5/14)

실행파일 구조 (Export Directory Table)

pFile	Data	Description	Value
00000150	000047E0	RVA	EXPORT Table
00000154	0000005B	Size	

pFile	Data	Description	Value
000047E0	00000000	Characteristics	
000047E4	45383CC7	Time Date Stamp	2006/10/20 03:04:39 UTC
000047E8	0000	Major Version	
000047EA	0000	Minor Version	
000047EC	0000481C	Name RVA	Test.dll
000047F0	00000001	Ordinal Base	
000047F4	00000002	Number of Functions	
000047F8	00000002	Number of Names	
000047FC	00004808	Address Table RVA	
00004800	00004810	Name Pointer Table RVA	
00004804	00004818	Ordinal Table RVA	

[Export Directory Table]

[Export Address Table]

pFile	Data	Description	Value
00004808	00001010	Function RVA	0001 AddInteger
0000480C	00001020	Function RVA	0002 SubInteger

[Export Name Table]

pFile	Data	Description	Value
00004810	00004825	Function Name RVA	0001 AddInteger
00004814	00004830	Function Name RVA	0002 SubInteger

1.사전 지식(6/14)

실행 압축 (분석지연, AV 진단우회, 다양한 변형 제작)

[실행압축 종류]

실행압축 비율				
No	실행압축 종류	샘플수	비율	누적비율
1	UPX	1070	19.48%	19.48%
2	FSG	764	13.91%	33.39%
3	PE_Patch	747	13.60%	47.00%
4	PECompact	315	5.74%	52.73%
5	PecBundle	296	5.39%	58.12%
6	PE_Patch,PECompact	296	5.39%	63.51%
7	ASPack	239	4.35%	67.86%
8	UPack	232	4.22%	72.09%
9	MEW	222	4.04%	76.13%
10	NSPack	182	3.31%	79.44%
11	MewBundle	156	2.84%	82.28%
12	Morphine	118	2.15%	84.43%
13	PE_Patch,Morphine	114	2.08%	86.51%
14	Petite	104	1.89%	88.40%
15	Armadillo	48	0.87%	89.28%
16	Molebox	37	0.67%	89.95%
17	Obsidium	37	0.67%	90.62%
18	PESpin	36	0.66%	91.28%
19	PE-Crypt,Sue	33	0.60%	91.88%
20	PE_Patch,Sue	30	0.55%	92.43%

실행압축 비율				
No	실행압축 종류	샘플수	비율	누적비율
21	Polyene	25	0.46%	0.46%
22	NTPacker	24	0.44%	0.89%
23	Yoda	22	0.40%	1.29%
24	Expressor	19	0.35%	1.64%
25	s0m	17	0.31%	1.95%
26	Packman	17	0.31%	2.26%
27	PE_Patch,Upolyx	15	0.27%	2.53%
28	UltraProtect	14	0.25%	2.79%
29	PE_Patch,UltraProtect	14	0.25%	3.04%
30	PE-Crypt,AntiDeb	13	0.24%	3.28%
31	PE_Patch,Stolen	13	0.24%	3.51%
32	PEBundle	12	0.22%	3.73%
33	ASProtect	11	0.20%	3.93%
34	PE-Crypt,Stone	11	0.20%	4.13%
35	PE_Patch,DotFix	10	0.18%	4.32%
36	NSAnti	10	0.18%	4.50%
37	PE_Patch,NSAnti	10	0.18%	4.68%
38	PE-Pack	9	0.16%	4.84%
39	PE-Diminisher	9	0.16%	5.01%
40	PE-Crypt,PFD	9	0.16%	5.17%

1.사전 지식(7/14)

실행 압축 (OEP)

[실행압축 이전 OEP]

```
0040101F    90              NOP
00401020  r$ 55             PUSH EBP
00401021  . 8BEC            MOV EBP,ESP
00401023  . 6A FF           PUSH -1
00401025  . 68 A0404000     PUSH MsgTest.004040A0
0040102A  . 68 541B4000     PUSH MsgTest.00401B54
0040102F  . 64:A1 0000000(  MOV EAX,DWORD PTR FS:[0]
00401035  . 50              PUSH EAX
00401036  . 64:8925 00000(  MOV DWORD PTR FS:[0],ESP
0040103D  . 83EC 58         SUB ESP,58
00401040  . 53              PUSH EBX
00401041  . 56              PUSH ESI
00401042  . 57              PUSH EDI
00401043  . 8965 E8         MOV DWORD PTR SS:[EBP-18],ESP
00401046  . FF15 14404000   CALL DWORD PTR DS:[<&KERNEL32.GetVersio  kernel32.GetVersion
0040104C  . 33D2            XOR EDX,EDX
0040104E  . 8AD4            MOV DL,AH
```

```
00407E10  $ 60              PUSHAD
00407E11  . BE 00604000     MOV ESI,Upx125Te.00406000
00407E16  . 8DBE 00B0FFFF   LEA EDI,DWORD PTR DS:[ESI+FFFFB000]
00407E1C  . 57              PUSH EDI
00407E1D  . 83CD FF         OR EBP,FFFFFFFF
00407E20  . EB 10           JMP SHORT Upx125Te.00407E32
00407E22    90              NOP
00407E23    90              NOP
00407E24    90              NOP
00407E25    90              NOP
00407E26    90              NOP
00407E27    90              NOP
00407E28  > 8A06            MOV AL,BYTE PTR DS:[ESI]
00407E2A  . 46              INC ESI
00407E2B  . 8807            MOV BYTE PTR DS:[EDI],AL
00407E2D  . 47              INC EDI
00407E2E  > 01DB            ADD EBX,EBX
00407E30  . 75 07           JNZ SHORT Upx125Te.00407E39
```

[실행압축 이후 OEP]

1.사전 지식(8/14)

DLL Injection 기법 1

❑ 레지스트리 등록 기법

- user32.dll 를 사용하는 프로그램에 Injection (NT 계열)
- user32.dll 이 다른 프로세스에 attach 될때 레지스트리에 등록된 dll를 로드하여 attach 시켜줌

[HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Windows]
"AppInit_DLLs"="d:\Work\Test\Debug\Test.dll "

❑ Windows Message Hook 기법

- 정상적인 DLL Injection 기법(정상 프로그램에서 주로 사용되며, Win9x/NT 계열)
- SetWindowsHookEx, UnhookWindowsHookEx API 사용

```
HHOOK SetWindowsHookEx(
  int idHook,         // type of hook to install
  HOOKPROC lpfn,      // address of hook procedure
  HINSTANCE hMod,     // handle to application instance
  DWORD dwThreadId    // identity of thread to install hook for
);

BOOL UnhookWindowsHookEx(
  HHOOK hhk           // handle to hook procedure to remove
);
```

1.사전 지식(9/14)

DLL Injection 기법 2

❑ DLL Forwarding 기법 (Proxy DLL)

- 원본 DLL의 이름을 변경한후, Hooking DLL를 원본 DLL 과 동일한 이름으로 변경
- Hooking Dll 에서는 원본 Dll의 함수를 Export 해야 하며, 원하는 동작후에 원본 함수를 호출해야 함

원본 : ws2_32.dll → ws2_32_org.dll
Hook : hook.dll → ws2_32.dll

❑ Remote Thread 기법

- 타 프로세스에 Thread를 생성하는 API 이용 (NT 계열)
- DLL Injection를 사용하는 최근 악성코드의 대부분이 이 기법 사용
- CreateRemoteThread, WriteProcessMemory API 사용

```
HANDLE CreateRemoteThread(
  HANDLE hProcess,          // handle to process to create thread in
  LPSECURITY_ATTRIBUTES lpThreadAttributes,  // pointer to security attributes
  DWORD dwStackSize,        // initial thread stack size, in bytes
  LPTHREAD_START_ROUTINE lpStartAddress, // pointer to thread function
  LPVOID lpParameter,       // argument for new thread
  DWORD dwCreationFlags,    // creation flags
  LPDWORD lpThreadId        // pointer to returned thread identifier
);
```

1.사전 지식(10/14)

[Win32/Korgo.worm DLL Injection]

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; - Dll Injection core 루틴
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ------- 편집 ------
10005797   FF15 70100010     CALL DWORD PTR DS:[10001070]              ; kernel32.VirtualAllocEx
                        ------- 편집 ------
100057E1   8D45 EC           LEA EAX,DWORD PTR SS:[EBP-14]
100057E4   50                PUSH EAX                                   ; pBytesWritten = 0095F9E0
100057E5   FF75 F0           PUSH DWORD PTR SS:[EBP-10]                 ; BytesToWrite = 2E (46.)
100057E8   56                PUSH ESI                                   ; Buffer = 003E0000
100057E9   53                PUSH EBX                                   ; Address = EA0000
100057EA   FF75 08           PUSH DWORD PTR SS:[EBP+8]                  ; hProcess = 00000090 (window)
100057ED   FF15 6C100010     CALL DWORD PTR DS:[1000106C]               ; kernel32.WriteProcessMemory

        003E0000  83 EC FC 90 90 E8 57 D9 F9 76 E8 80 4A F9 76 43  껋?맗W黙v?J?C
        003E0010  3A 5C 57 49 4E 44 4F 57 53 5C 53 79 73 74 65 6D  :₩WINDOWS₩System
        003E0020  33 32 5C 72 65 6A 61 33 32 2E 64 6C 6C 00 00 00  32₩reja32.dll...

        003E0000   83EC FC           SUB ESP,-4
        003E0003   90                NOP
        003E0004   90                NOP
        003E0005   E8 57D9F976       CALL 7737D961           ; EA000A + 76F9D957 = 77E3D961 ==> Kernel32!LoadLibraryA
        003E000A   E8 804AF976       CALL 77374A8F           ; EA000F + 76F94A80 = 77E34A8F ==> Kernel32!ExitThread


100057F9   8D43 0F           LEA EAX,DWORD PTR DS:[EBX+F]
100057FC   56                PUSH ESI                                   ; dwCreationFlag = 0
100057FD   50                PUSH EAX                                   ; lpParameter = EA000F
100057FE   53                PUSH EBX                                   ; lpStartAddress = EA0000
100057FF   56                PUSH ESI                                   ; dwStackSize = 0;
10005800   56                PUSH ESI                                   ; lpThreadAttributes = 0
10005801   FF75 08           PUSH DWORD PTR SS:[EBP+8]                  ; hProcess = 90
10005804   FF15 68100010     CALL DWORD PTR DS:[10001068]               ; kernel32.CreateRemoteThread
```

1.사전 지식(11/14)

❑ Stealth 기법

- ✓ 사용자가 악성코드의 설치/ 감염 여부를 확인 못하도록 함
- ✓ 악성 코드 진단 회피

고전적 기법

- 유사한 파일명, 파일크기 변경, 파일명 숨김 등등
- 도스/BIOS 인터럽트 후킹
  - 인터럽트 백터를 수정하여 메모리에 상주한 바이러스 코드 호출

최신 기법

- IRP(I/O Request Packet) Hooking
- DKOM(Direct Kernel Object Manipulation)

1.사전 지식(12/14)

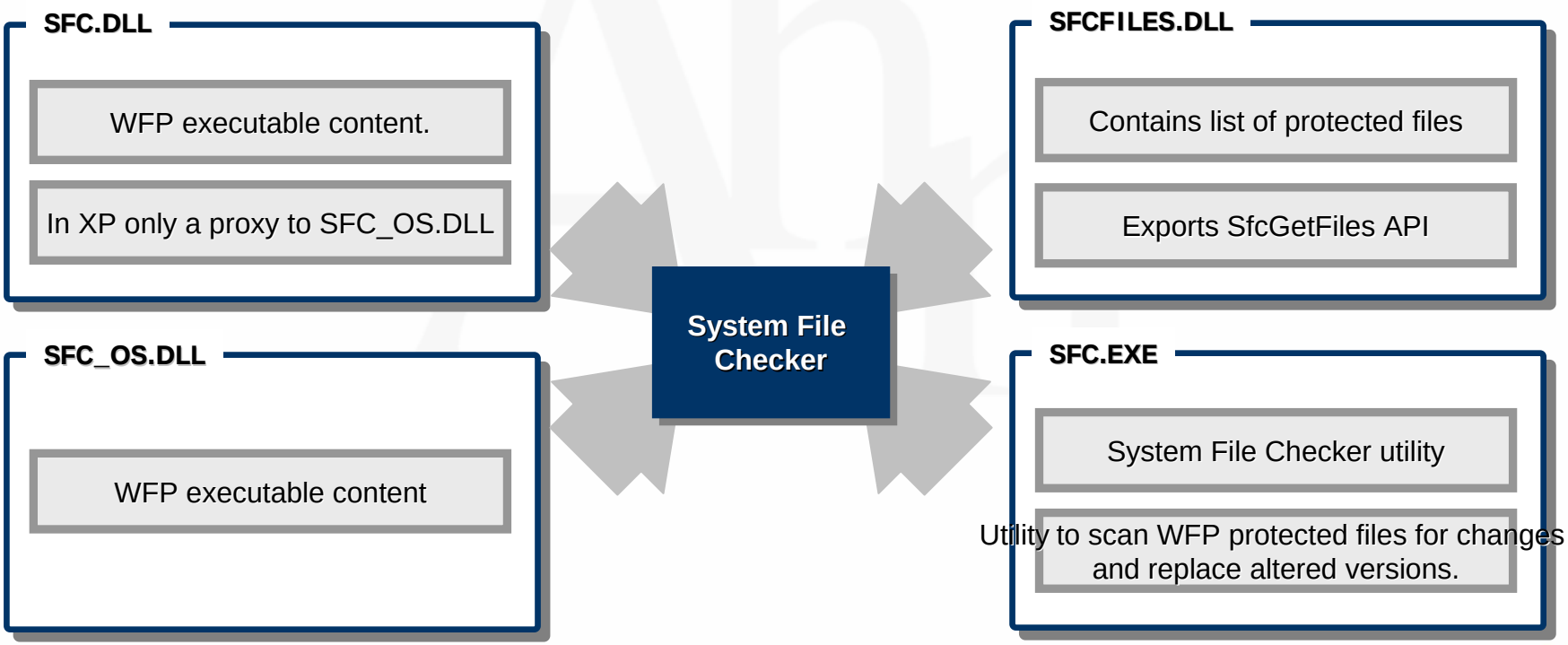
Stealth 기법 (SSDT 후킹)

```
mov     eax, ds:ZwQueryDirectoryFile
mov     ecx, [eax+1]                            ; ecx = ZwQueryDirectoryFile의  index
mov     edx, ds:KeServiceDescriptorTable
mov     eax, [edx]
mov     ecx, [eax+ecx*4]                        ; ecx = ZwQueryDirectoryFile의 함수 주소
mov     dword_13000, ecx                        ; ZwQueryDirectoryFile 함수 주소 백업
mov     edx, ds:ZwQuerySystemInformation
mov     eax, [edx+1]                            ; eax = ZwQuerySystemInformation 의 index
mov     ecx, ds:KeServiceDescriptorTable
mov     edx, [ecx]
mov     eax, [edx+eax*4]                        ; eax = ZwQuerySystemInformation의 함수 주소
mov     dword_13008, eax                        ; ZwQuerySystemInformation의 함수 주소 백업
cli
mov     eax, cr0
and     eax, 0FFFEFFFFh
mov     cr0, eax
mov     ecx, ds:ZwQueryDirectoryFile
mov     edx, [ecx+1]                            ; edx = ZwQueryDirectoryFile의  index
mov     eax, ds:KeServiceDescriptorTable
mov     ecx, [eax]
mov     dword ptr [ecx+edx*4], offset sub_11CC6 ; ZwQueryDirectoryFile 함수 hooking
mov     edx, ds:ZwQuerySystemInformation
mov     eax, [edx+1]                            ; eax = ZwQuerySystemInformation 의 index
mov     ecx, ds:KeServiceDescriptorTable
mov     edx, [ecx]
mov     dword ptr [edx+eax*4], offset sub_11ECF ; ZwQuerySystemInformation 함수 hooking
mov     eax, cr0
or      eax, 10000h
mov     cr0, eax
sti
```

1.사전 지식(13/14)

❑ WFP (Windows File Protection)

- ✓ A mechanism that protects system files from being modified or deleted
- ✓ 시스템 안정성 (Stability) 확보
- ✓ 시스템 무결성 (Integrity) 확보
- ✓ DLL HELL 문제 해결



1.사전 지식(14/14)

WFP (Windows File Protection) 무력화 기법

- Closing Directory Change Notification Handle
- Terminating SFC Watcher Thread via undocumented SFC API
- Disable WFP for 1 minute
- Disable WFP Permanently via patches and undocumented registry value
- Disable WFP Permanently for specific files via patching the protected file list

2.분석 과정(1/2)

기초 분석(외형 분석, 증상 분석, 자료 분석) ➔ 상세분석

- ❑ 파일유형 판단
- ❑ 문자열 추출 및 분석
- ❑ 파일 등록 정보 활용
- ❑ 샘플 접수 정보 활용

- ❑ 전 이미지 정보 수집
- ❑ 샘플 실행
- ❑ 후 이미지 정보 수집
- ❑ 증상 정보 분석

- ❑ 증상 정보 추가 분석
- ❑ 각종 정보 수집
- ❑ 로그 정보 수집/분석

- ❑ 디어셈블 분석
- ❑ 상세 디버깅

- ❑ 엔진 반영 여부 결정
- ❑ 진단시그니쳐 및 함수 작성
- ❑ 분석정보 작성

2.분석 과정(2/2)

바이러스 분석 과정

1. 기본 외형 분석
2. 샘플 셋 생성 및 기본 증상 확인
3. 바이러스의 감염 유형 파악(전위형, 후위형)
4. 디버깅을 통한 바이러스의 전반적인 기능 확인
   (감염 대상, 감염 조건, 감염 기법 등등)
5. 호스트 프로그램 실행 부분 분석
6. 바이러스 특징 분석 (일반, 암호화, 다형성 등등)
7. 진단명 명명
8. 진단 문자열 추출
9. 치료 데이터 작성 및 검증 작업

3.진단법 (1/3)

바이러스 진단법

1. Signature 기반 진단법
   - 진단 위치 (Entry Point / 프로그램의 전체 크기)
   - Signature (특정 영역 / 특정 코드)
   - 특정 위치 진단법 (by Ahnlab)

2. Non-Signature 기반 진단법
   - 전용진단 함수 제작
     선행조건 검사
     악성코드 각각의 특징을 검사

3. 행위 기반 진단법
   - 악성코드 행위에 대한 진단
     파일 이름, 인젝션, 파일 Drop, 통신 포트 오픈 등등

3.진단법 (2/3)

Signature 기반 진단법

```
2820  e1 e1 83 e7  8e a5 52 44  b2 c5 68 a6  4e a0 4d ba   áá.ç.¥RD²Åh¦N M°
2830  01 86 00 f4  68 21 cd ba  1e f3 93 e2  4f c2 a5 60   ...ôh!Í°.ó.âOÂ¥`
2840  77 ec 03 32  f1 47 18 7d  e4 73 dc ab  92 73 1b f8   wì.2ñG.}äsÜ«'s.ø
2850  95 55 36 ed  86 f4 08 ef  02 96 1f 50  be 62 59 3c   .U6í.ô.ï...P¾bY<
2860  dc 54 14 69  b1 ef 27 91  f7 fc 1b ee  cd 67 b8 98   ÜT.i±ï'`÷ü.îÍg¸.
2870  7b ea c9 b2  fe 8e 2f 78  65 f4 41 6b  61 a2 d4 52   {êÉ²þ./xeôAka¢ÔR
2880  00 7d 04 58  e7 a3 de 59  56 4e bf a9  a7 6c 9f 49   .}.Xç£ÞYVN¿©§l.I
2890  80 89 2a 0f  b6 89 2d e7  83 c0 b3 a3  1b 97 c1 d1   ..*.¶.-ç.À³£..ÁÑ
28a0  39 72 05 64  2d 57 be f2  dd 80 a7 72  e3 d6 75 de   9r.d-W¾òÝ.§rãÖuÞ
28b0  9d 49 b7 73  75 14 60 37  20 e7 19 f3  70 37 b1 df   .I·su.`7 ç.óp7±ß
28c0  fc 11 6e 0f  42 b8 5e f9  7c 0f 60 0d  33 ad 14 28   ü.n.B¸^ù|.`.3-.(
28d0  15 be 2c 3b  d5 5c 1d c7  1c e3 bf 04  0c 3b 9b da   .¾,;Õ\.Ç.ã¿..;.Ú
28e0  b7 88 5b a4  de b2 70 4a  b4 34 44 df  8c 15 cd 90   ·.[¤Þ²pJ´4Dß..Í.
28f0  01 55 ff 4d  6f b2 91 4f  c4 af 9d e3  9b 37 01 25   .UÿMo²'OÄ¯.ã.7.%
2900  3b 88 96 fb  cc ce 87 7f  ce e9 02 7a  85 ec c9 b2   ;..ûÌÎ..Îé.z.ìÉ²
```

[특정 영역 Signature]

[특정 OP 코드]

```
B00D:FEB9 07          POP ES                          ; ES=FFFF
B00D:FEBA 8BDA        MOV BX,DX                       ; BX=DX=0321
B00D:FEBC BA  64FB    MOV DX,FB64                     ; DX=FB64
B00D:FEBF C707 50B4   MOV WORD PTR [BX],B450          ; [0000:0321] = B450
B00D:FEC3 C74702 19CD MOV WORD PTR [BX+02],CD19       ; [0000:0323] = CD19
B00D:FEC8 C74704 CC58 MOV WORD PTR [BX+04],58CC       ; [0000:0325] = 58CC
B00D:FECD E8  0B00    CALL    FEDB                    ;
B00D:FED0 BF  02FC    MOV DI,FC02
B00D:FED3 BB  1603    MOV BX,0316
B00D:FED6 2E          CS:
B00D:FED7 C415        LES DX,[DI]                     ; ES= C000, DX=0DFE
B00D:FED9 EB  04      JMP FEDF
```

3.진단법 (3/3)

Non-Signature 기반 진단법

[전용 진단 함수]

```
void ScanFile(char *szFileName, BOOL bSaveDecodedFile)
{
            ...

    // Before Decoding
    if (IsInfectedByGoldBug(buf, dwToRead) == TRUE)
        ExitFunc(szFileName, "infected by GoldBug !")

    DecodeBuf(buf, dwToRead, (WORD) dwFileSize);

    // After Decoding
    if (IsInfectedByGoldBug(buf, dwToRead) == TRUE)
        ExitFunc(szFileName, "infected by GoldBug !")

            ...
}
```

```
void DecodeBuf(BYTE *pBuf, int nMaxIndex, WORD wKey)
{
    BYTE loKey = LOBYTE(wKey);
    BYTE hiKey = HIBYTE(wKey);

    // 1'th Decoding
    int nStartIndex = 0xC;
    pBuf[nStartIndex] = pBuf[nStartIndex] ^ loKey;
    nStartIndex++;
    for (int i=nStartIndex; i<nMaxIndex; i++)
        pBuf[i] = pBuf[i] ^ loKey ^ hiKey;

    // 2'th Decoding
    BYTE byTemp = 0;
    for (i=0; i<nMaxIndex; i++)
    {
        byTemp = byTemp + pBuf[i];
        byTemp++;

        int nDestIndex = 0x1C + i;
        if (nDestIndex == nMaxIndex)
            break;

        pBuf[nDestIndex] = pBuf[nDestIndex] ^ byTemp;
    }
}
```

Ⅲ. 다형성 바이러스

1. 다형성 바이러스 개요

2. 변형 기법

3. 다형성 바이러스 진단

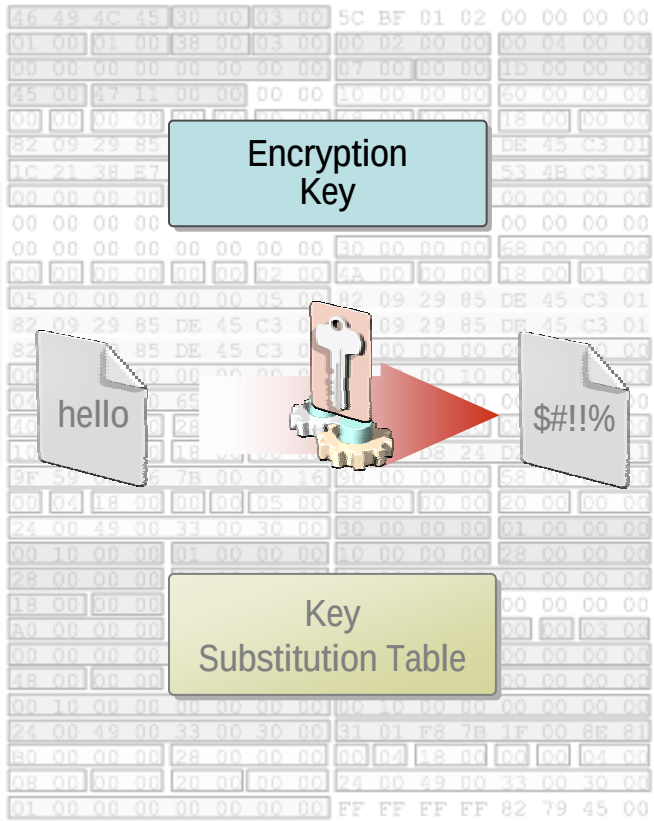
1.다형성 바이러스 개요 (1/6)

암호화 바이러스

1. 암호화 Key
   - 동일한 Decryption 루틴
   - 가변 Key 값

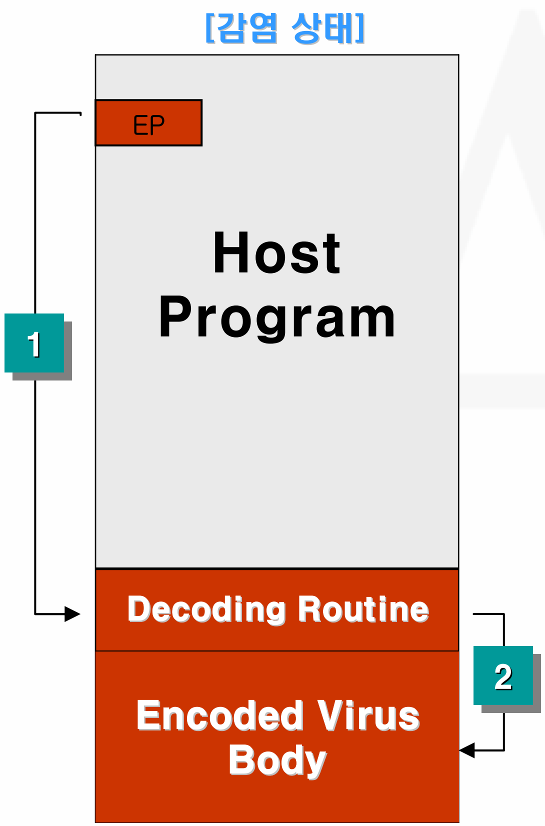
2. Decoding Buffer
   - Overwritting
   - Stack Memory
   - Allocated Memory

3. 암호화 기법
   - Linear / Non-Linear
   - Multiple Layer
   - RDA



1.다형성 바이러스 개요 (2/6)

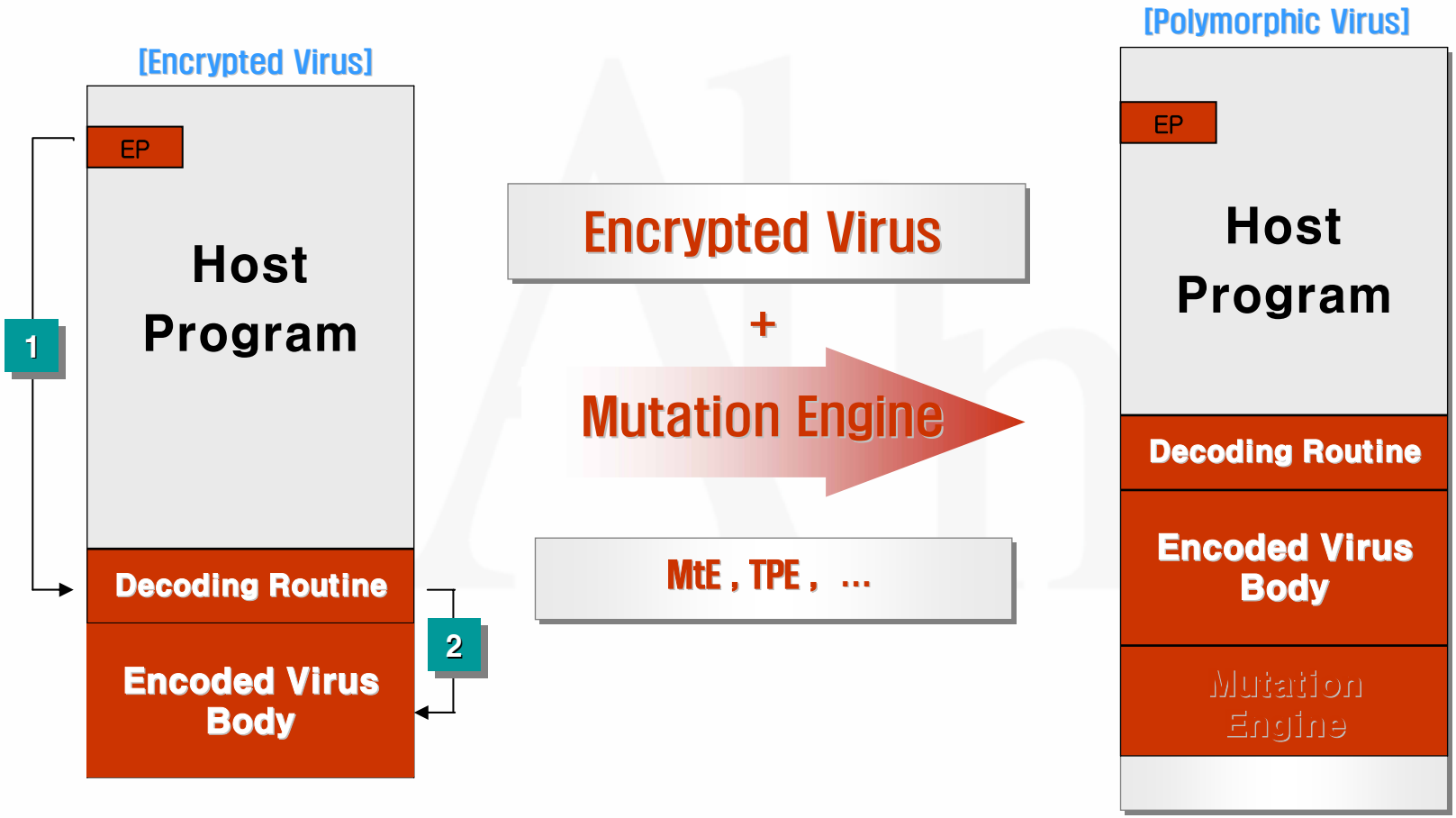
암호화 바이러스




```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; CX = 1A72  (파일 사이즈)
; AX = 8000
; SI = 0
; while(CX)
; {
;     AL = AL + FFFF:[SI + FB00]
;     INC AX
;     8000:[SI + 1C] = 8000:[SI+1C] xor AL
;     INC SI
;
;     DEC CX
; }
;
; AX = 1A72  (파일 사이즈)
; CX = 1A72  (파일 사이즈)
; BX = 000C
; while (CX)
; {
;     [8000:BX] = [8000:BX] xor AX
;     INC BX
;     DEC CX
; }
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

B00D:FC56 2E           CS:
B00D:FC57 028400FB     ADD AL,[SI+FB00]
B00D:FC5B 40           INC AX
B00D:FC5C 30441C       XOR [SI+1C],AL
B00D:FC5F 46           INC SI
B00D:FC60 E2F4         LOOP    FC56
B00D:FC62 61           POPA

                       ...

B00D:FC6B 3107         XOR [BX],AX
B00D:FC6D 43           INC BX
B00D:FC6E E2FB         LOOP    FC6B
```

1.다형성 바이러스 개요 (3/6)

Polymorphic Virus



1.다형성 바이러스 개요 (4/6)

Polymorphic Virus

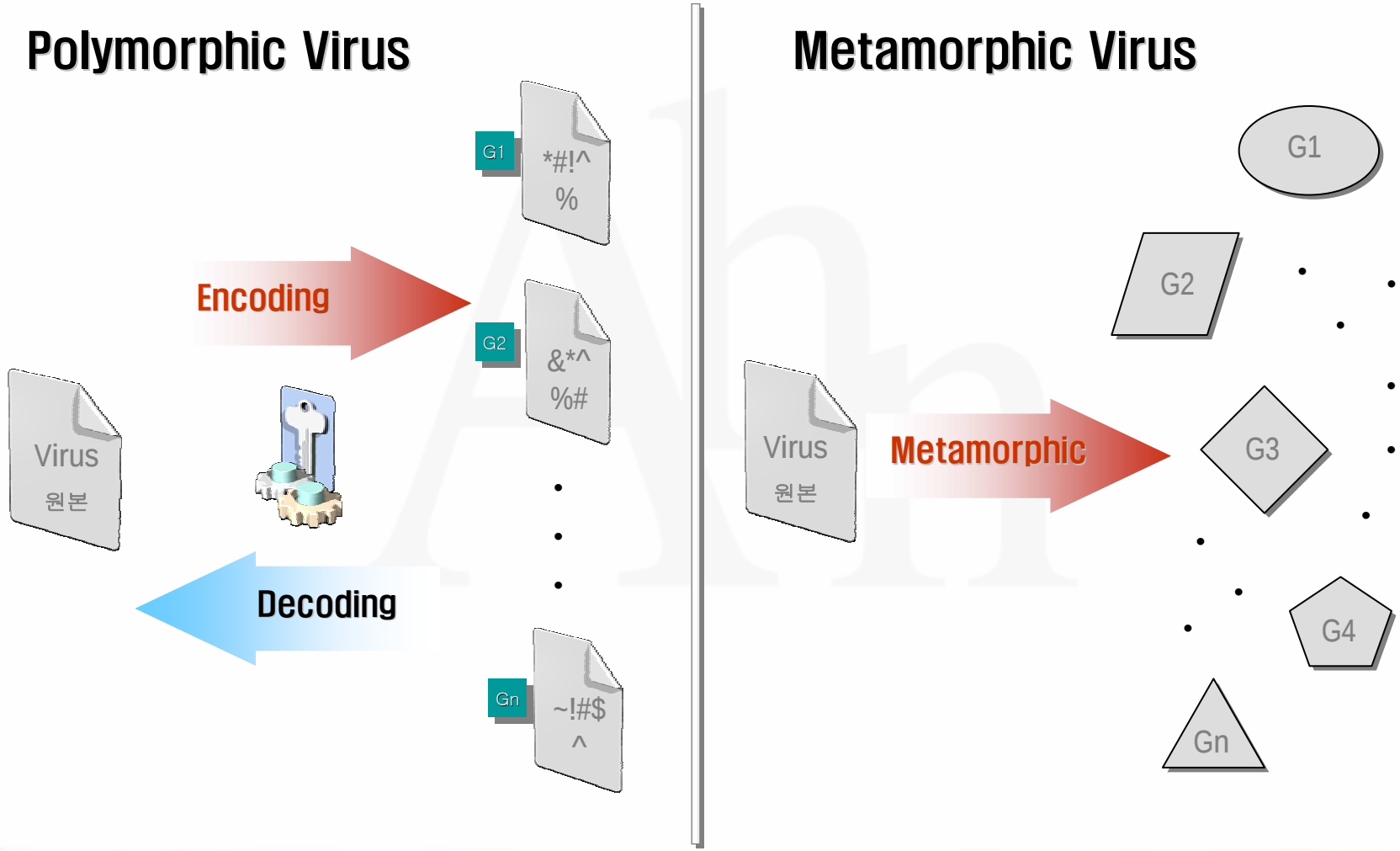
[MtE]

```
2299:0100 E9FF0F        JMP     1102
                ...

2299:1102 52            PUSH    DX
2299:1103 51            PUSH    CX
2299:1104 50            PUSH    AX
2299:1105 56            PUSH    SI
2299:1106 55            PUSH    BP
2299:1107 53            PUSH    BX
2299:1108 B80294        MOV     AX,9402
2299:110B BA1BAB        MOV     DX,AB1B
2299:110E F7E2          MUL     DX
2299:1110 96            XCHG    SI,AX
2299:1111 36            SS:
2299:1112 8B84001F      MOV     AX,[SI+1F00]
2299:1116 BAEDC0        MOV     DX,C0ED
2299:1119 F7E2          MUL     DX
2299:111B 36            SS:
2299:111C 8784001F      XCHG    AX,[SI+1F00]
2299:1120 8BEE          MOV     BP,SI
2299:1122 81C5BE1A      ADD     BP,1ABE
2299:1126 8BDD          MOV     BX,BP
2299:1128 81EBBC1A      SUB     BX,1ABC
2299:112C 8BF3          MOV     SI,BX
2299:112E 75E1          JNZ     1111
```

```
2299:0100 E9FF0F        JMP     1102
                ...

2299:1102 57            PUSH    DI
2299:1103 53            PUSH    BX
                ...
2299:1109 B85244        MOV     AX,4452
2299:110C BAF1A3        MOV     DX,A3F1
2299:110F F7E2          MUL     DX
2299:1111 BAD392        MOV     DX,92D3
2299:1114 F7E2          MUL     DX
2299:1116 95            XCHG    BP,AX
2299:1117 B8AA84        MOV     AX,84AA
2299:111A F7ED          IMUL    BP
2299:111C B107          MOV     CL,07
                ...
2299:1125 D3C8          ROR     AX,CL52
2299:1127 96            XCHG    SI,AX
2299:1128 8B86261F      MOV     AX,[BP+1F26]
2299:112C BA319F        MOV     DX,9F31
                ...
2299:112F F7E2          MUL     DX
2299:1135 8786261F      XCHG    AX,[BP+1F26]
2299:113E 2BEB          SUB     BP,BX
                ...
2299:114F 2BF7          SUB     SI,DI
2299:1151 8BEE          MOV     BP,SI
2299:1153 75C2          JNZ     1117
```

1.다형성 바이러스 개요 (5/6)

Metamorphic Virus



1.다형성 바이러스 개요 (6/6)

Metamorphic Virus

[ZMIST]

```
Notepad.vxe (Zmist 감염)

004010EC MOV ESI,EAX
004010EE MOV AL,BYTE PTR DS:[EAX]
004010F0 TEST AL,AL
004010F2 JE NOTEPAD.00401488

004010F8 PUSH EBX            ; EP of Virus
004010F9 POP DWORD PTR DS:[40F974]
004010FF RCR EBX,CL
00401101 BSWAP EBX
00401103 PUSH NOTEPAD.0040F42C
00401108 POP EBX
00401109 MOV DWORD PTR DS:[EBX],EAX
0040110B INC EBX
0040110C BSR EAX,EDX
0040110F TEST EAX,DC78A946
00401115 MOV EAX,EDX
00401117 PUSH EDX
00401118 MOV DH,86
0040111A MOV BL,27
0040111C MOV EAX,7FA1FA7C
00401122 JMP SHORT NOTEPAD.00401125
00401124 DB C8
00401125 BSF EAX,EDX
00401128 MOV DWORD PTR DS:[4188FC],0
00401132 SUB EAX,B9E80D21
00401137 IMUL EBX,EDX,9DD477E5
```

```
ACCSTAT.vxe (Zmist 감염)

004025DC JNZ SHORT ACCSTAT.004025E1
004025DE MOV DWORD PTR DS:[ESI+10],EDI
004025E1 MOV DWORD PTR DS:[40D0B8],EDI
004025E7 JMP ACCSTAT.004029C1

004025EC PUSH ECX            ; EP of Virus
004025ED XCHG ECX,ECX
004025EF XADD CL,CH
004025F2 TEST ECX,74FE3615
004025F8 PUSH EBX
004025F9 AND BL,DH
004025FB IMUL EBX,EDI
004025FE SUB BH,CH
00402600 PUSH EAX
00402601 BTS EAX,4F
00402605 XADD EAX,EBX
00402608 LEA ECX,DWORD PTR DS:[41D458]
0040260E BT EBX,EAX
00402611 MOV AH,41
00402614 BSWAP EAX
00402616 PUSH ESI
00402617 XCHG AL,AL
00402619 POP DWORD PTR DS:[ECX]
0040261B LEA EAX,DWORD PTR DS:[6F512E7B]
00402621 DEC BL
00402623 TEST CL,CH
```

2.변형 기법 (1/4)

사용 명령어 변경

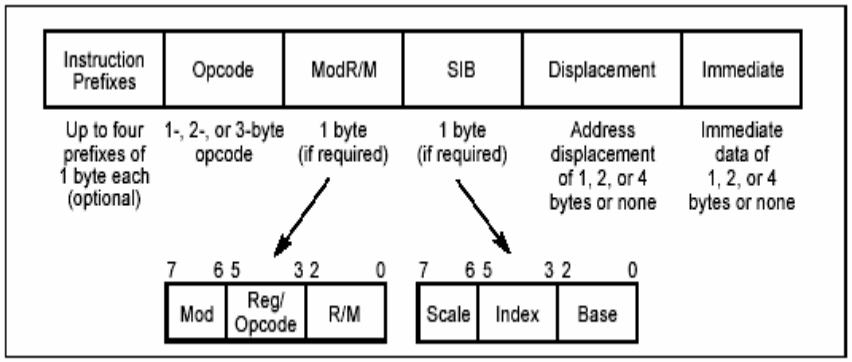


Figure 2-1. IA-32 Instruction Format

E : ModR/M follows the opcode and specifies the operand.

G : The reg field of ModR/M select a general register.

01C1 → ADD ECX, EAX 1100 0001

11C1 → ADC ECX, EAX 1100 0001

21C1 → AND ECX, EAX 1100 0001

31C1 → XOR ECX, EAX 1100 0001

31C2 → XOR EDX, EAX 1100 0010

	0	1	2	3	4	5	6	7
0	ADD Eb, Gb	ADD Ev, Gv	ADD Gb, Eb	ADD Gv, Ev	ADD AL, Ib[1D]	ADD eAX, Iv[1D]	PUSH ES[1D]	POP ES[1D]
1	ADC Eb, Gb	ADC Ev, Gv	ADC Gb, Eb	ADC Gv, Ev	ADC AL, Ib[1D]	ADC eAX, Iv[1D]	PUSH SS[1D]	POP SS[1D]
2	AND Eb, Gb	AND Ev, Gv	AND Gb, Eb	AND Gv, Ev	AND AL, Ib[1D]	AND eAX, Iv[1D]	SEG=ES Prefix	DAA[1D]
3	XOR Eb, Gb	XOR Ev, Gv	XOR Gb, Eb	XOR Gv, Ev	XOR AL, Ib[1D]	XOR eAX, Iv[1D]	SEG=SS Prefix	AAA[1D]
4	INC general register eAX[1D]	INC general register eCX[1D]	INC general register eDX[1D]	INC general register eBX[1D]	INC general register eSP[1D]	INC general register eBP[1D]	INC general register eSI[1D]	INC general register eDI[1D]
5	PUSH general register[1D] eAX	PUSH general register[1D] eCX	PUSH general register[1D] eDX	PUSH general register[1D] eBX	PUSH general register[1D] eSP	PUSH general register[1D] eBP	PUSH general register[1D] eSI	PUSH general register[1D] eDI
6	PUSHA/ PUSHAD[1D]	POPA/ POPAD[1D]	BOUND Gv, Ma	ARPL Ew, Gw	SEG=FS Prefix	SEG=GS Prefix	Opd Size Prefix	Addr Size Prefix
7	Jcc, Jb - Short-displacement jump on condition O[1D]	Jcc, Jb - Short-displacement jump on condition NO[1D]	Jcc, Jb - Short-displacement jump on condition B/NAE/C[1D]	Jcc, Jb - Short-displacement jump on condition NB/AE/NC[1D]	Jcc, Jb - Short-displacement jump on condition Z/E[1D]	Jcc, Jb - Short-displacement jump on condition NZ/NE[1D]	Jcc, Jb - Short-displacement jump on condition BE/NA[1D]	Jcc, Jb - Short-displacement jump on condition NBE/A[1D]
8	Immediate Grp 1[1A] Eb, Ib	Immediate Grp 1[1A] Ev, Iv	Immediate Grp 1[1A] Eb, Ib	Immediate Grp 1[1A] Ev, Ib	TEST Eb, Gb	TEST Ev, Gv	XCHG Eb, Gb	XCHG Ev, Gv
9	NOP[1D]	XCHG word or double-word register with eAX[1D] eCX	XCHG word or double-word register with eAX[1D] eDX	XCHG word or double-word register with eAX[1D] eBX	XCHG word or double-word register with eAX[1D] eSP	XCHG word or double-word register with eAX[1D] eBP	XCHG word or double-word register with eAX[1D] eSI	XCHG word or double-word register with eAX[1D] eDI
A	MOV[1D] AL, Ob	MOV[1D] eAX, Ov	MOV[1D] Ob, AL	MOV[1D] Ov, eAX	MOVS/ MOVSB Yb, Xb[1D]	MOVS/ MOVSW/ MOVSD Yv, Xv[1D]	CMPS/ CMPSB Yb, Xb[1D]	CMPS/ CMPSW/ CMPSD Xv, Yv[1D]
B	MOV immediate byte into byte register[1D] AL	MOV immediate byte into byte register[1D] CL	MOV immediate byte into byte register[1D] DL	MOV immediate byte into byte register[1D] BL	MOV immediate byte into byte register[1D] AH	MOV immediate byte into byte register[1D] CH	MOV immediate byte into byte register[1D] DH	MOV immediate byte into byte register[1D] BH
C	Shift Grp 2[1A] Eb, Ib	Shift Grp 2[1A] Ev, Ib	RET Iw[1D]	RET[1D]	LES Gv, Mp	LDS Gv, Mp	Grp 11[1A] - MOV Eb, Ib	Grp 11[1A] - MOV Ev, Iv
D	Shift Grp 2[1A] Eb, 1	Shift Grp 2[1A] Ev, 1	Shift Grp 2[1A] Eb, CL	Shift Grp 2[1A] Ev, CL	AAM Ib[1D]	AAD Ib[1D]		XLAT/ XLATB[1D]
E	LOOPNE/ LOOPNZ Jb[1D]	LOOPE/ LOOPZ Jb[1D]	LOOP Jb[1D]	JCXZ/ JECXZ Jb[1D]	IN AL, Ib[1D]	IN eAX, Ib[1D]	OUT Ib, AL[1D]	OUT Ib, eAX[1D]
F	LOCK Prefix		REPNE Prefix	REP/ REPE Prefix	HLT[1D]	CMC[1D]	Unary Grp 3[1A] Eb	Unary Grp 3[1A] Ev

2.변형 기법 (2/4)

Garbage 명령 삽입

```
LEA     BX,[SI+4D]   ; BX = 104D
CLC
MOV     CX,03B4      ; Loop Count
CLC
MOV     DX,[BX]      ; DX = EC2E
CLC
PUSH    BP           ; PUSH BP
CLC
MOV     BP,SI        ; BP = 1000
NOP
MOV     AX,000D      ; AX = D
CLD
ADD     BP,AX        ; BP = 100D
STC
PUSH    [BP+00]      ; PUSH 10E0
CLD
POP     AX           ; AX = 10E0
STC
POP     BP           ; BP = 0
STI
ADD     DX,AX        ; DX = 3FCC
NOT     DX           ; DX = C033
STI
MOV     [BX],DX      ; [104D] = C033
CLD
INC     BX           ; BX = 104E
STC
INC     BX           ; BX = 104F
LOOP    1822
```

```
LEA EAX,DWORD PTR DS:[F0FCE513]
SHL EAX,44
INC EAX
MOV EAX,NOTEPAD.0040F424          ; EAX = 40F424
ADC EBX,12317708
MOV EBX,EDX
TEST EDX,EBX
MOV EBX,EAX                       ; EBX = 40F424
MOV DWORD PTR DS:[EBX],EDX        ; DS:[40F424] = 9DE5CFBF
IMUL EAX,EDX
DEC EAX
MOV EDX,NOTEPAD.00403BC8          ; EDX = 403BC8
ADC EAX,EDX
MOV EBX,DWORD PTR DS:[EDX]        ; EBX = E14C383C
INC EAX
BSF EAX,EDX
LEA EAX,DWORD PTR DS:[40F428]     ; EAX = 40F428
AND DL,DH
IMUL EDX,EDX
PUSH EBX                          ; PUSH E14C383C
ADD DH,DH
OR EDX,EDX
IMUL EDX,EDX
SAL EDX,68
MOV DL,DL
POP DWORD PTR DS:[EAX]            ; DS:[40F428] = E14C383C
XOR EDX,C48729A2
CMP EDX,EDX
```

2.변형 기법 (3/4)

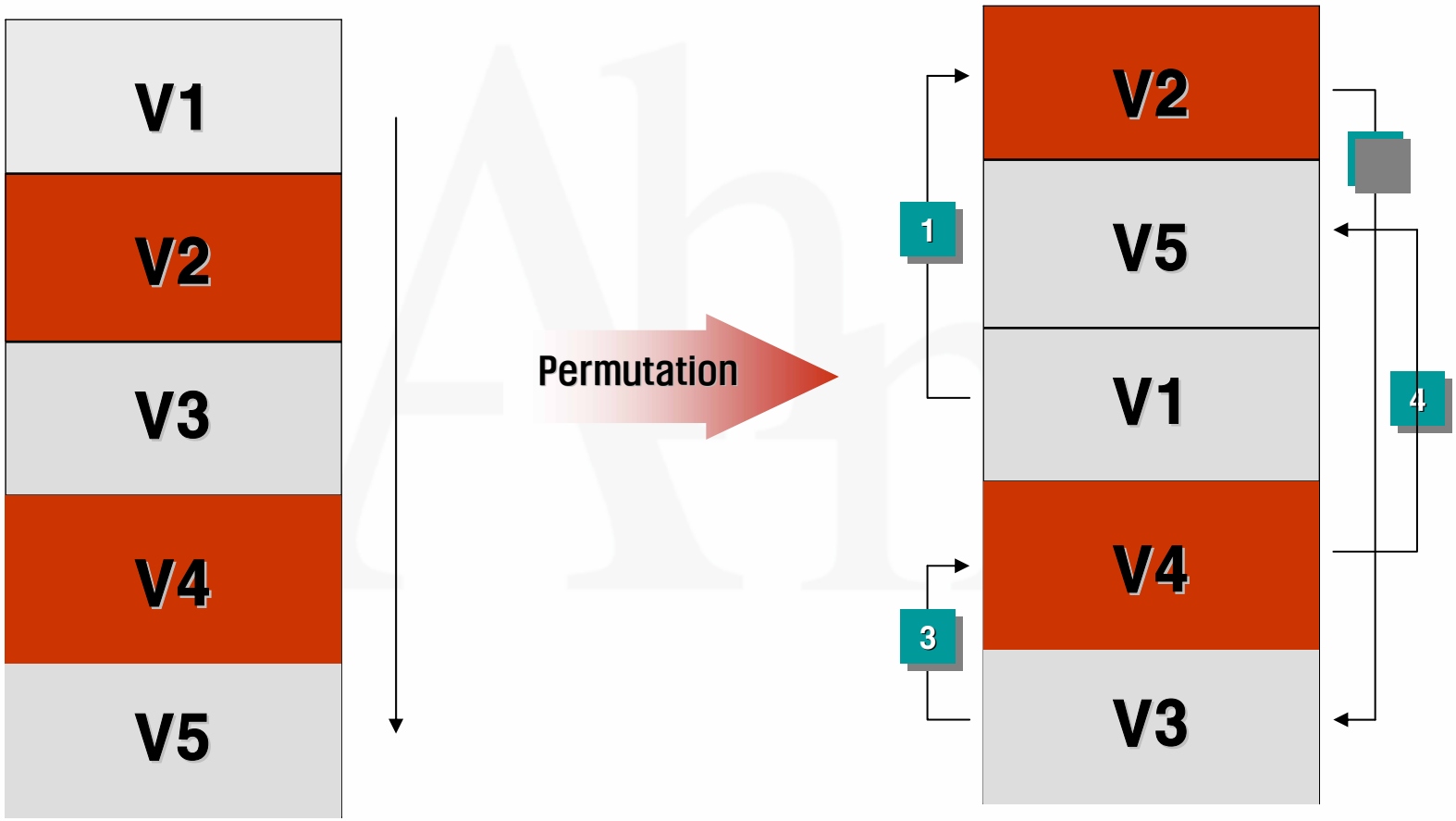
사용 Register 변경

1. Same code pattern

```
G1)
    5A              pop   edx
    BF04000000      mov   edi,0004h
    8BF5            mov   esi,ebp
    B80C000000      mov   eax,000Ch
    81C288000000    add   edx,0088h
    8B1A            mov   ebx,[edx]
    899C8618110000  mov   [esi+eax*4+00001118],ebx

G2)
    58              pop   eax
    BB04000000      mov   ebx,0004h
    8BD5            mov   edx,ebp
    BF0C000000      mov   edi,000Ch
    81C088000000    add   eax,0088h
    8B30            mov   esi,[eax]
    89B4BA18110000  mov   [edx+edi*4+00001118],esi
```

2.변형 기법 (4/4)

Permutation



3.다형성 바이러스 진단 (1/3)

다형성 바이러스 진단

1. Signature 기반 진단
   - 다형성 엔진에 대한 시그니쳐 진단
   - 오진 / 비정상 치료

2. 전용진단 함수
   - 선행 조건
   - 사용된 명령어 패턴 기반 진단

3. Emulation
   - Emulation 시작 위치 (EPO)
   - Emulation 종료
   - 진단 속도 / 예외 처리
   - Anti-Emulation 기법에 대한 대비

3.다형성 바이러스 진단 (2/3)

전용 진단 함수 (명령어 패턴)

[Win32/Zmist]

```
BOOL ScanFunc(X86INST *px86Inst, DWORD dwEip, int* pZmistPatternArray, DWORD *pdwPattern)
{
    const DISASMINFO *pDisasmInfo = px86Inst->pEntry->pDisasmInfo;

    if (0x70 <= px86Inst->byOpcode[0] && px86Inst->byOpcode[0] <= 0x7F)     // Jcc, jb xx
        goto Start;

    if (px86Inst->byOpcode[0] == 0xE9 && px86Inst->dwDisp32 != 1)           // JMP xxxxxxxx
        goto Start;

    if (px86Inst->byOpcode[0] == 0xC2 || px86Inst->byOpcode[0] == 0xC3)     // RETN or RETN xx
        goto Clear;

    if (0xC8 <= px86Inst->byOpcode[0] && px86Inst->byOpcode[0] <= 0xCF)     // ENTER ~ IRET
        goto Clear;

    if (0xE0 <= px86Inst->byOpcode[0] && px86Inst->byOpcode[0] <= 0xE3)     // LOOPNE ~ JCXZ
        goto Clear;

    if (0xA4 <= px86Inst->byOpcode[0] && px86Inst->byOpcode[0] <= 0xA7)     // MOVS ~ JCXZ
        goto Clear;

    if (0xAA <= px86Inst->byOpcode[0] && px86Inst->byOpcode[0] <= 0xAF)     // STOS ~ SCAS
        goto Clear;

    if (px86Inst->byOpcode[0] == 0xE8)                                      // CALL xxxxxxxx
        goto Clear;


    if (0x50 <= px86Inst->byOpcode[0] && px86Inst->byOpcode[0] <= 0x57)     // PUSH ERX
    {
        if (px86Inst->byOpcode[0] == 0x54 || px86Inst->byOpcode[0] == 0x55) // PUSH ESP, EBP
            goto Clear;
    }

                        ....
                        ....
```

3.다형성 바이러스 진단 (3/3)

Anti-Emulation Technique

1. Co-Processor / MMX instruction usage
2. Structured Exception Handling Usage
3. Random Virus Code Execution
4. Use Of Brute Force Decryption of Virus Code (RDA)
5. Use of API

Ⅳ. Case Study

1. Win32/Polip 분석

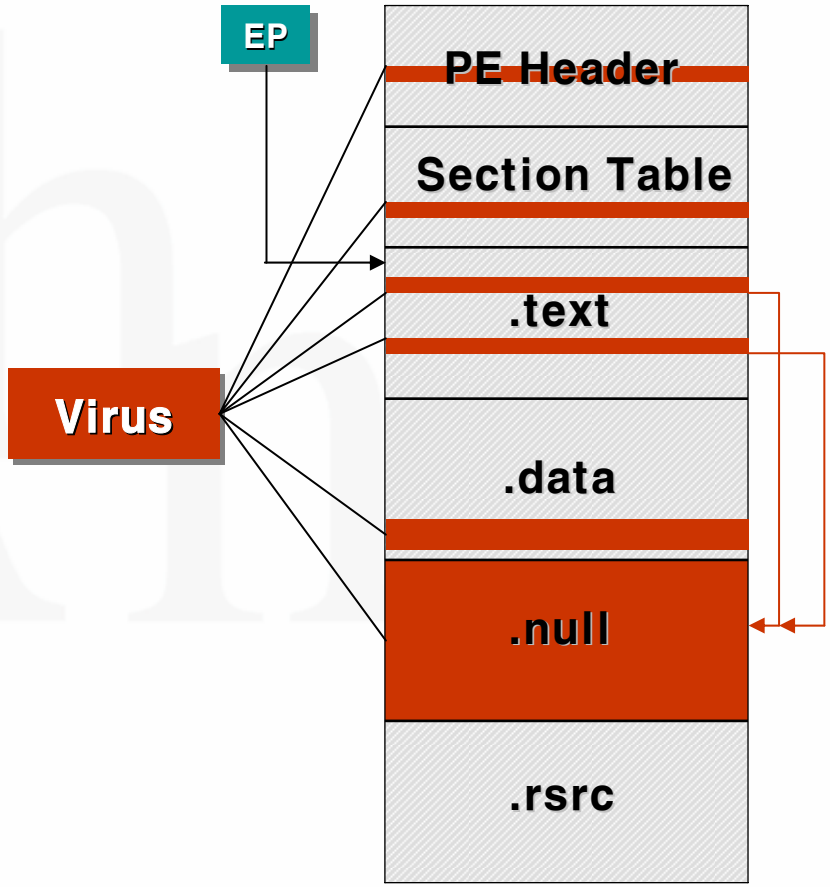
2. 진단

3. 치료

3.Win32/Polip 분석 (1/9)

Win32/Polip Virus

1. EPO 기법 사용
2. 다형성 엔진 + XTEA 변형
3. 섹션 빈 공간에 코드 추가
4. NULL 섹션 추가
5. 파일 크기 증가 (60 ~ 70 KB)
6. Win32.Polipos 1.2 by Joseph



3.Win32/Polip 분석 (2/9)

Polymorphic Engine – EPO (Entry Point Obscruing)

[Pinball.exe 감염]

```
////////////////////////////////////////////////////
// 원본 코드
////////////////////////////////////////////////////
010063A1  CMP DWORD PTR SS:[EBP+C],3BD
010063A8  JE SHORT Pinball.010063B1
010063AA  POP EBP
010063AB  JMP DWORD PTR DS:[<&USER32.DefWindowProcA>]
010063B1  MOV EDX,DWORD PTR DS:[10261E8]
010063B7  PUSH EBX


////////////////////////////////////////////////////
// 바이러스에 의해 수정된 코드
////////////////////////////////////////////////////
010063A1  CMP DWORD PTR SS:[EBP+C],3BD
010063A8  JE SHORT PINBALL.010063B1
010063AA  POP EBP
010063AB  JMP PINBALL.01050BE1          ; EP of Virus
010063B0  DB 01
010063B1  MOV EDX,DWORD PTR DS:[10261E8]
010063B7  PUSH EBX
```

[Msn6.exe 감염]

```
////////////////////////////////////////////////////
// 원본 코드
////////////////////////////////////////////////////
0040681A   CALL DWORD PTR DS:[<&USER32.CreateWindow>
00406820   JMP msn6.00403B6A
00406825   JMP DWORD PTR DS:[<&USER32.DefWindowProc>
0040682B   PUSH DWORD PTR SS:[EBP+8]
0040682E   CALL DWORD PTR DS:[<&KERNEL32.DeleteFile>


////////////////////////////////////////////////////
// 바이러스에 의해 수정된 코드
////////////////////////////////////////////////////
0040681A   CALL DWORD PTR DS:[<&USER32.CreateWindow>
00406820   JMP msn6.00403B6A
00406825   JMP msn6.0042030A        ; EP of Virus
0040682A   DB 00
0040682B   PUSH DWORD PTR SS:[EBP+8]
0040682E   CALL DWORD PTR DS:[<&KERNEL32.DeleteFile>
```

3.Win32/Polip 분석 (3/9)

Polymorphic Engine – Random Memory Access

```
////////////////////////////////////////////////////////////////////////////////
// EP of Virus
//
//  1. .text 섹션 : 1001000 ~ 1020400
//  2. .data 섹션 : 1021000 ~ 1022A00
//  3. .rsrc 섹션 : 1027000 ~ 1048800
//  1. 추가된 섹션: 1049000 ~ 1056E00
////////////////////////////////////////////////////////////////////////////////
01050BE1   55               PUSH EBP                                ; EP of Virus
01050BE2   8BEC             MOV EBP,ESP
01050BE4   83EC 34          SUB ESP,34
01050BE7   60               PUSHAD
01050BE8   8105 CB690201 FD>ADD DWORD PTR DS:[10269CB],156A9CFD     ; .data 섹션과 .rsrc 섹션 사이
01050BF2   69F2 1DBDFA18    IMUL ESI,EDX,18FABD1D

01050BF8   B9 28000000      MOV ECX,28
01050BFD   BF 4D630201      MOV EDI,PINBALL.0102634D                ; EDI = 0102634D
01050C02   F3:AA            REP STOS BYTE PTR ES:[EDI]
01050C04   4A               DEC EDX
01050C05   0FB63D 0E290201  MOVZX EDI,BYTE PTR DS:[102290E]         ; EDI = 0
01050C0C   3305 23270201    XOR EAX,DWORD PTR DS:[1022723]          ; EAX = C0004900
01050C12   88D7             MOV BH,DL
01050C14   8335 F2650201 73 XOR DWORD PTR DS:[10265F2],73           ; [10265F2] = 00000073
01050C1B   B9 10000000      MOV ECX,10                              ; ECX = 10
01050C20   BF 61640201      MOV EDI,PINBALL.01026461                ; EDI = 01026461
01050C25   F2:AF            REPNE SCAS DWORD PTR ES:[EDI]           ; EDI 에서 EAX를 찾음
01050C27   75 26            JNZ SHORT PINBALL.01050C4F

                 ....
```

3.Win32/Polip 분석 (4/9)

Polymorphic Engine – Garbage 명령 삽입

[Pinball.exe 감염]

```
01050BE1   PUSH EBP                ; EP of Virus
01050BE2   MOV EBP,ESP
01050BE4   SUB ESP,34
01050BE7   PUSHAD
01050BE8   ADD DWORD PTR DS:[10269CB],156A9CFD
01050BF2   IMUL ESI,EDX,18FABD1D
01050BF8   MOV ECX,28
01050BFD   MOV EDI,PINBALL.0102634D
01050C02   REP STOS BYTE PTR ES:[EDI]
01050C04   DEC EDX
01050C05   MOVZX EDI,BYTE PTR DS:[102290E]
01050C0C   XOR EAX,DWORD PTR DS:[1022723]
01050C12   MOV BH,DL
01050C14   XOR DWORD PTR DS:[10265F2],73
01050C1B   MOV ECX,10
01050C20   MOV EDI,PINBALL.01026461
01050C25   REPNE SCAS DWORD PTR ES:[EDI]
01050C27   JNZ SHORT PINBALL.01050C4F
01050C29   ROL EDX,18
01050C2C   AND EAX,57
01050C2F   AND EBX,EAX
01050C31   XOR DWORD PTR DS:[102642C],EDX
01050C37   AND EBX,EDX
01050C39   ADC EBX,DWORD PTR DS:[1022816]
01050C3F   MOV DWORD PTR DS:[1026ACF],48B6DF44
01050C49   RCL DWORD PTR DS:[10264F5],1
```

[Msn6.exe 감염]

```
0042030A   PUSH EBP                ; EP of Virus
0042030B   MOV EBP,ESP
0042030D   SUB ESP,34
00420310   PUSHAD
00420311   IMUL EDI,DWORD PTR DS:[40BC27],53543809
0042031B   XOR BYTE PTR DS:[40BD63],89
00420322   SHLD DWORD PTR DS:[40BF53],EDX,16
0042032A   MOV DWORD PTR DS:[40BD11],18AB316F
00420334   DEC DWORD PTR DS:[40BD40]
0042033A   SHR DWORD PTR DS:[40BDF3],1
00420340   MOV EAX,EDX
00420342   CALL msn6.0041DF3A
00420347   MOVSX ECX,AX
0042034A   OR EDX,EAX
0042034C   MOV EDX,DWORD PTR DS:[40BBE4]
00420352   SHL BL,0
00420355   RCL BL,7
00420358   SHRD DWORD PTR DS:[40BD62],EAX,1F
00420360   ADD ECX,46DA011F
00420366   OR DWORD PTR DS:[40BD6D],FFFFFFD4
0042036D   MOV ECX,EAX
0042036F   SBB DWORD PTR DS:[40BEBF],-5
00420376   MOV DWORD PTR DS:[40BEA6],794BFF2D
00420380   PUSH DWORD PTR DS:[40BF6E]
00420386   INC EDI
```

3.Win32/Polip 분석 (5/9)

Polymorphic Engine – Random Code Execution

```
                        ...

01054EAC   2315 DF2D0201        AND EDX,DWORD PTR DS:[1022DDF]
01054EB2   75 36                JNZ SHORT PINBALL.01054EEA
01054EB4   41                   INC ECX
01054EB5   0F9E05 1E650201      SETLE BYTE PTR DS:[102651E]
01054EBC   66:8B0D 06280201     MOV CX,WORD PTR DS:[1022806]

                        ...

01054EEA   037D 0C              ADD EDI,DWORD PTR SS:[EBP+C]
01054EED   66:850D CD2A0201     TEST WORD PTR DS:[1022ACD],CX
01054EF4   75 5A                JNZ SHORT PINBALL.01054F50
01054EF6   C0D8 01              RCR AL,1

                        ...

010550B9   81C1 6791653F        ADD ECX,3F659167
010550BF   3905 DD660201        CMP DWORD PTR DS:[10266DD],EAX
010550C5   76 32                JBE SHORT PINBALL.010550F9
010550C7   41                   INC ECX
010550C8   42                   INC EDX
010550C9   830D 17680201 CE     OR DWORD PTR DS:[1026817],FFFFFFCE
010550D0   0FAF05 242B0201      IMUL EAX,DWORD PTR DS:[1022B24]
010550D7   FE0D 4D630201        DEC BYTE PTR DS:[102634D]
010550DD   1B15 DC290201        SBB EDX,DWORD PTR DS:[10229DC]
010550E3   C705 28640201 EC     MOV DWORD PTR DS:[1026428],5102E7EC
010550ED   8A35 D4260201        MOV DH,BYTE PTR DS:[10226D4]
010550F3   0B05 132B0201        OR EAX,DWORD PTR DS:[1022B13]
010550F9   66:29DE              SUB SI,BX
010550FC   1205 BA250201        ADC AL,BYTE PTR DS:[10225BA]
```

3.Win32/Polip 분석 (6/9)

XTEA (eXtended Tiny Encryption Algorithm) 원형

```
/* XTEA is a version of slightly improved tea.
   The plain or cypher text is in v[0], v[1].
   The key is in k[n], where n = 0 - 3,
   The number of coding cycles is given by N and
   the number of decoding cycles given by -N     */

tean(long *v, long *k, long N)
{   unsigned long y = v[0],
                  z = v[1],
                  DELTA = 0x9e3779b9,
                  limit,
                  sum;

    if (N > 0)              /* coding */
    {
        limit = DELTA * N;
        sum = 0;
        while (sum != limit)
        {   y   += (z << 4 ^ z >> 5) + z ^ sum + k[sum & 3];
            sum += DELTA;
            z   += (y << 4 ^ y >> 5) + y ^ sum + k[sum >> 11 & 3];
        }
    }
    else                    /* decoding */
    {
         sum = DELTA * (-N);
         while (sum)
         {   z   -= (y << 4 ^ y >> 5) + y ^ sum + k[sum >> 11 & 3];
             sum -= DELTA;
             y   -= (z << 4 ^ z >> 5) + z ^ sum + k[sum & 3];
         }
    }

    v[0] = y;
    v[1] = z;
    return;
}
```

```
XTEA_Decipher   proc
    PUSHAD                  ; Save registers
    MOV EAX, v[0 * 4]       ; EAX -> y
    MOV EBX, v[1 * 4]       ; EBX -> z
    MOV ECX, N              ; ECX -> n
    MOV EDX, DIS            ; EDX -> sum

DeStart:
                        ...
    MOV ESI, EDX
    SHR ESI, 11             ; sum >> 11
    AND ESI, 3              ; sum >> 11 & 3
    MOV ESI, k[ESI * 4]     ; k[sum >> 11 & 3]
    ADD ESI, EDX            ; sum + k[sum >> 11 & 3]
    XOR ESI, EDI            ; (y << 4 ^ y >> 5) + y ^ sum + k[sum >> 11 & 3]
    SUB EBX, ESI            ; z -= (y << 4 ^ y >> 5) + y ^ sum + k[sum >> 11 & 3]

    SUB EDX, Delta          ; sum -= delta

                        ...
    MOV ESI, EDX
    AND ESI, 3              ; sum & 3
    MOV ESI, k[ESI * 4]     ; k[sum & 3]
    ADD ESI, EDX            ; sum + k[sum & 3]
    XOR ESI, EDI            ; (z << 4 ^ z >> 5) + z ^ sum + k[sum & 3]
    SUB EAX, ESI            ; y -= (z << 4 ^ z >> 5) + z ^ sum + k[sum & 3]

    LOOP    DeStart

    MOV w[0 * 4], EAX
    MOV w[1 * 4], EBX

    POPAD                   ; Restore registers
    RET
XTEA_Decipher   endp
```

3.Win32/Polip 분석 (7/9)

XTEA (eXtended Tiny Encryption Algorithm) 변형

```
void DecodeBuf(DWORD dwData, DWORD *pdwDecodeData, DWORD dwKey, DWORD dwSum, DWORD dwDelta)
{
    DWORD y = LOWORD(dwData);
    DWORD z = HIWORD(dwData);

    for (int i=0; i<5; i++)
    {
        DWORD dwTemp1 = (y << 4 ^ y >> 5);              // (y << 4 ^ y >> 5)
        WORD t1 = LOWORD(dwTemp1) + LOWORD(y);
        memcpy(&dwTemp1, &t1, 2);

        DWORD dwTemp2 = dwTemp1 ^ (dwSum + dwKey);      // (y << 4 ^ y >> 5) + y ^ dwSum + dwKey
        WORD t2 = LOWORD(z) - LOWORD(dwTemp2);
        memcpy(&z, &t2, 2);                             // z   -= (y << 4 ^ y >> 5) + y ^ dwSum + dwKey

        dwSum -= dwDelta;

        dwTemp1 = (z << 4 ^ z >> 5);
        t1 = LOWORD(dwTemp1) + LOWORD(z);               // (z << 4 ^ z >> 5)
        memcpy(&dwTemp1, &t1, 2);

        dwTemp2 = dwTemp1 ^ ((dwSum + dwKey) >> 16);    // (z << 4 ^ z >> 5) + z ^ ( (dwSum + dwKey) >> 16 )
        t2 = LOWORD(y) - LOWORD(dwTemp2);
        memcpy(&y, &t2, 2);                             // y   -= (z << 4 ^ z >> 5) + z ^ dwSum + dwKey;

    }

    DWORD dwResult = y | (z << 16);
    *pdwDecodeData = dwResult;
}
```

3.Win32/Polip 분석 (8/9)

XTEA (eXtended Tiny Encryption Algorithm) 변형

```
01052261D   8B45 08              MOV EAX,DWORD PTR SS:[EBP+8]          ; EAX = [00127654] = 010487A8   ==> arg1
01052623    8B00                 MOV EAX,DWORD PTR DS:[EAX]            ; EAX = 4FD5D982
0105262B    89C6                 MOV ESI,EAX                           ; ESI = 4FD5D982              ==> 디코딩할 데이터
01052637    C745 FC D54FD7F9     MOV DWORD PTR SS:[EBP-4],F9D74FD5      ; [00127648] = F9D74FD5
0105264C    C1EE 10              SHR ESI,10                            ; ESI = 4FD5
01052662    25 FFFF0000          AND EAX,0FFFF                         ; EAX = D982
0105267B    C745 F8 DFA5FC05     MOV DWORD PTR SS:[EBP-8],5FCA5DF      ; [00127644] = 5FCA5DF        ==> SUM 값
01052685    BA 05000000          MOV EDX,5                             ; EDX = 5                     ==> 반복할 회수
010526C1    8175 F8 4245EB12     XOR DWORD PTR SS:[EBP-8],12EB4542     ; [00127644]=1717E09D         ==> SUM 값
010526D4    8175 FC 6CB6E067     XOR DWORD PTR SS:[EBP-4],67E0B66C     ; [00127648] = 9E37F9B9       ==> Delta 값
                        @Loop:
01052700    56                   PUSH ESI                              ; PUSH ESI = 00004FD5
01052750    89C2                 MOV EDX,EAX                           ; EDX = EAX = 0000D982
01052768    C1E2 04              SHL EDX,4                             ; EDX = 0000D982 ==> 000D9820
01052772    C1EE 05              SHR ESI,5                             ; ESI = 0000D982 ==> 000006CC
01052783    31F2                 XOR EDX,ESI                           ; EDX = 000D9EEC
01052792    66:01C2              ADD DX,AX                             ; EDX = 000D786E
01052795    8B75 F8              MOV ESI,DWORD PTR SS:[EBP-8]          ; ESI = [00127644] = 1717E09D   ==> SUM 값
0105279F    0375 0C              ADD ESI,DWORD PTR SS:[EBP+C]          ; ESI = 1717E09D + 1092A940 = 27AA89DD
010527B1    31F2                 XOR EDX,ESI                           ; EDX = 000D786E ^ 27AA89DD = 27A7F1B3
010527BC    5E                   POP ESI                               ; ESI = 00004FD5
010527C0    66:29D6              SUB SI,DX                             ; ESI = 4FD5 - F1B3 = 5E22
010527C3    8B55 FC              MOV EDX,DWORD PTR SS:[EBP-4]          ; EDX = SS:[00127648]=9E37F9B9  ==> Delta 값
010527C6    2955 F8              SUB DWORD PTR SS:[EBP-8],EDX          ; [00127644]= 78DFE6E4  ==> sum -= delta
010527E1    50                   PUSH EAX                              ; PUSH EAX = 0000D982
010527EC    89F2                 MOV EDX,ESI                           ; EDX = ESI = 00005E22
010527F0    89D0                 MOV EAX,EDX                           ; EAX = 00005E22
010527F9    C1E8 05              SHR EAX,5                             ; EAX = 00005E22 ==> 000002F1
01052803    C1E2 04              SHL EDX,4                             ; EDX = 00005E22 ==> 0005E220
0105281E    31C2                 XOR EDX,EAX                           ; EDX = 0005E220 ^ 000002F1 = 0005E0D1
01052821    8B45 F8              MOV EAX,DWORD PTR SS:[EBP-8]          ; EAX = [00127644]=78DFE6E4
0105282E    0345 0C              ADD EAX,DWORD PTR SS:[EBP+C]          ; EAX = 78DFE6E4 + 1092A940 ==> 89729024
01052837    66:01F2              ADD DX,SI                             ; EDX = E0D1 + 5E22 = 00053EF3
01052846    C1E8 10              SHR EAX,10                            ; 유효 명령임
01052856    31C2                 XOR EDX,EAX                           ; EDX = 00053EF3 ^ 00008972 = 0005B781
01052858    58                   POP EAX                               ; EAX = 0000D982
01052861    66:29D0              SUB AX,DX                             ; EAX = D982 - B781 = 00002201
01052865    5A                   POP EDX                               ; EDX = 5
0105287B    4A                   DEC EDX                               ; EDX = 4
0105287C   ^0F85 63FEFFFF        JNZ @Loop(010526E5)

0105288E    8B55 10              MOV EDX,DWORD PTR SS:[EBP+10]         ; EDX = 00127680
01052892    C1E6 10              SHL ESI,10                            ; ESI = 00000D10 ==> 0D100000
01052897    09F0                 OR EAX,ESI                            ; EAX = 0000F074 | 0D100000 = 0D10F074
0105289F    8902                 MOV DWORD PTR DS:[EDX],EAX            ; [00127680] = 0D10F074
```

3.Win32/Polip 분석 (9/9)

Win32/Polip XTEA – Decoding 루틴 구조

```
Subroutine1:
    ...
    call    Subroutine2
    ...
    ret

Subroutine2:
    ...
    call    Subroutine1
    ...
    ret
```

```
Function10519A6
{
    ...

    Function10525F8 (XTEA 디코딩 함수)
    ...

    Function1052275
    {
        ...

        Function105359F (XTEA 디코딩 함수)

        ...

        Function1055CCB
        {
            ...

            Function1054DA9 (XTEA 디코딩 함수)

            ...

            Function10519A6 (Recursive call)
        }
    }

    ...
}
```

2.Win32/Polip 진단 (1/2)

Emulation (Pre-Condition 체크)

```
BOOL Scan()
{
    Initialize();

    if (FindEmulationLoc() == FALSE)
        return FALSE;

    while(true)
    {
        if (FetchInstruction() == FALSE)
            return FALSE;

        DisamInstruction();

        call pEntry->pfnOpcode()

        if (CheckException() == TRUE)
            return FALSE;

        if (CheckEndOfEmulation() == TRUE)
            break;
    }

    if (IsInfectedByPolip() == TRUE)
        return TRUE;

AtExit:
    Uninitialize();
}
```

```
static const OPTABLE g_Opcode32Table[256*2] =
{
                    ...
  /* 24 */  { 0, NULL_TABLE, &Ia_andb_AL_Ib,    NULL_FN_OPCODE  },
  /* 25 */  { 0, NULL_TABLE, &Ia_andl_EAX_Id,   AND32_EAX_Id    },
  /* 26 */  { 0, NULL_TABLE, &Ia_prefix_es,     NULL_FN_OPCODE  },
  /* 27 */  { 0, NULL_TABLE, &Ia_daa,           NULL_FN_OPCODE  },
  /* 28 */  { 0, NULL_TABLE, &Ia_subb_Eb_Gb,    NULL_FN_OPCODE  },
  /* 29 */  { 0, NULL_TABLE, &Ia_subl_Ed_Gd,    SUB32_EdGd      },
  /* 2A */  { 0, NULL_TABLE, &Ia_subb_Gb_Eb,    NULL_FN_OPCODE  },
  /* 2B */  { 0, NULL_TABLE, &Ia_subl_Gd_Ed,    SUB32_GdEd      },
  /* 2C */  { 0, NULL_TABLE, &Ia_subb_AL_Ib,    NULL_FN_OPCODE  },
  /* 2D */  { 0, NULL_TABLE, &Ia_subl_EAX_Id,   SUB32_EAXId     },
  /* 2E */  { 0, NULL_TABLE, &Ia_prefix_cs,     NULL_FN_OPCODE  },
  /* 2F */  { 0, NULL_TABLE, &Ia_das,           NULL_FN_OPCODE  },
  /* 30 */  { 0, NULL_TABLE, &Ia_xorb_Eb_Gb,    XOR32_EbGb      },
  /* 31 */  { 0, NULL_TABLE, &Ia_xorl_Ed_Gd,    XOR32_EdGd      },
  /* 32 */  { 0, NULL_TABLE, &Ia_xorb_Gb_Eb,    NULL_FN_OPCODE  },
  /* 33 */  { 0, NULL_TABLE, &Ia_xorl_Gd_Ed,    XOR32_GdEd      },
  /* 34 */  { 0, NULL_TABLE, &Ia_xorb_AL_Ib,    NULL_FN_OPCODE  },
  /* 35 */  { 0, NULL_TABLE, &Ia_xorl_EAX_Id,   XOR32_EAXId     },
  /* 36 */  { 0, NULL_TABLE, &Ia_prefix_ss,     NULL_FN_OPCODE  },
  /* 37 */  { 0, NULL_TABLE, &Ia_aaa,           NULL_FN_OPCODE  },
  /* 38 */  { 0, NULL_TABLE, &Ia_cmpb_Eb_Gb,    NULL_FN_OPCODE  },
  /* 39 */  { 0, NULL_TABLE, &Ia_cmpl_Ed_Gd,    CMP32_EdGd      },
  /* 3A */  { 0, NULL_TABLE, &Ia_cmpb_Gb_Eb,    NULL_FN_OPCODE  },
  /* 3B */  { 0, NULL_TABLE, &Ia_cmpl_Gd_Ed,    CMP32_GdEd      },
                    ...
}
```

2.Win32/Polip 진단 (2/2)

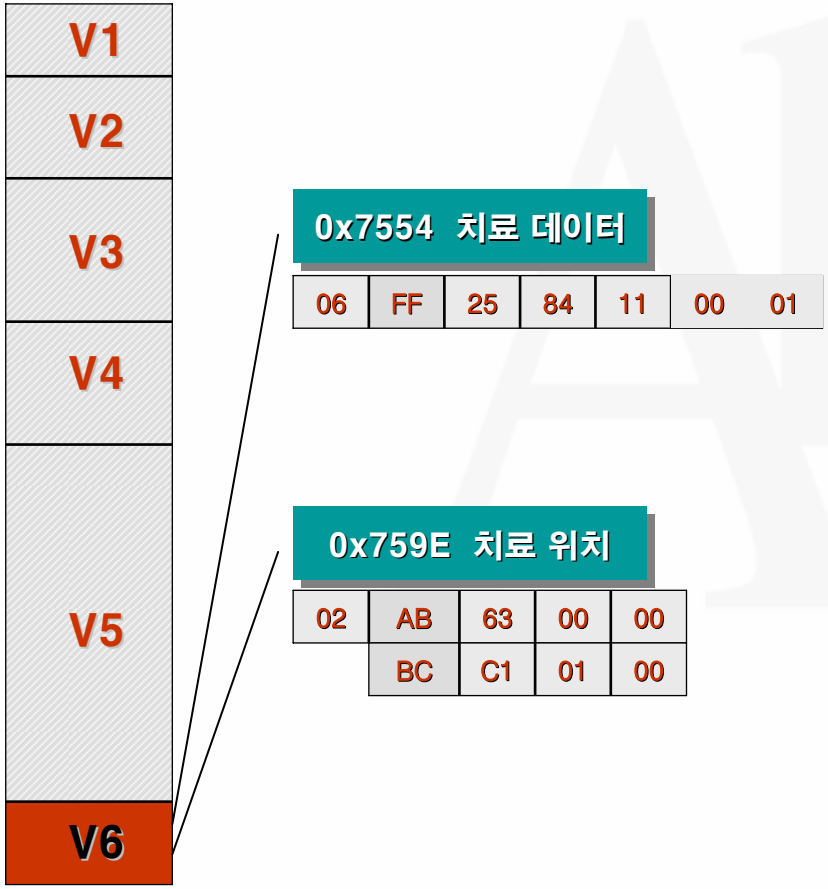
진단문자열 (Virus Body)

```
////////////////////////////////////////////////////////////////////////////////////////////////
// 암호화 해제 10505DD 부터 1050AC8 까지 10F07688 키값을 사용해서 XOR 디코딩
////////////////////////////////////////////////////////////////////////////////////////////////
010505C4   60               PUSHAD
010505C5   E8 00000000      CALL PINBALL.010505CA
010505CA   5A               POP EDX                                  ; EDX = 010505CA
010505CB   B9 3B010000      MOV ECX,13B                              ; ECX = 13B (0x4EC 바이트)
010505D0   81B48A 0F000000 >XOR DWORD PTR DS:[EDX+ECX*4+F],10F07688  ; [01050AC5] = 09F0B57A ^ 10F07688 = 1900C3F2
010505DB  ^E2 F3            LOOPD SHORT PINBALL.010505D0             ; 10505DD 부터 1050AC8 까지 XOR 디코딩

////////////////////////////////////////////////////////////////////////////////////////////////
// 스택에 있는 00127BB3 부터 7145 만큼 XOR 연산 수행 함 (key 값 증가 시키면서 01050551)
////////////////////////////////////////////////////////////////////////////////////////////////
010505F3   64:FF35 00000000 PUSH DWORD PTR FS:[0]
010505FA   64:8925 00000000 MOV DWORD PTR FS:[0],ESP                 ; FS:[0] = ESP = 0012766C
01050601   8B7424 2C        MOV ESI,DWORD PTR SS:[ESP+2C]            ; ESI = SS:[00127698]=0012769C
01050605   8BFA             MOV EDI,EDX                              ; EDI = 105050CA
01050607   8D86 17050000    LEA EAX,DWORD PTR DS:[ESI+517]           ; EAX = 00127BB3
0105060D   B9 45710000      MOV ECX,7145                             ; ECX = 7145
01050612   B2 51            MOV DL,51                                ; DL = 51   ==> EDX = 01050551
                    @L1:
01050614   3010             XOR BYTE PTR DS:[EAX],DL                 ; [00127BB3] = C5 ^ 51 = 94
01050616   42               INC EDX                                  ; EDX = 01050552
01050617   40               INC EAX                                  ; EAX = 00127BB4
01050618  ^E2 FA            LOOPD @L1(01050614)                      ; 루프 반복

0105061A   E8 AD010000      CALL PINBALL.010507CC                    ; ==> kernel32 주소를 구하고 'MZ' 시그니쳐를 확인
0105061F   8BEB             MOV EBP,EBX                              ; EBP = 77E20000
01050621   68 1EDADF1A      PUSH 1ADFDA1E
01050626   E8 E8020000      CALL @GetApiAddress(01050913)            ; ==> GetModuleHandleA 주소 구함
0105062B   85C0             TEST EAX,EAX                             ; EAX = 77E3AD86 (kernel32.GetModuleHandleA)
0105062D   0F84 D6000000    JE PINBALL.01050709
01050633   8987 D9030000    MOV DWORD PTR DS:[EDI+3D9],EAX
```

3.Win32/Polip 치료 (1/2)

치료 정보

[복호화 후 Stack]

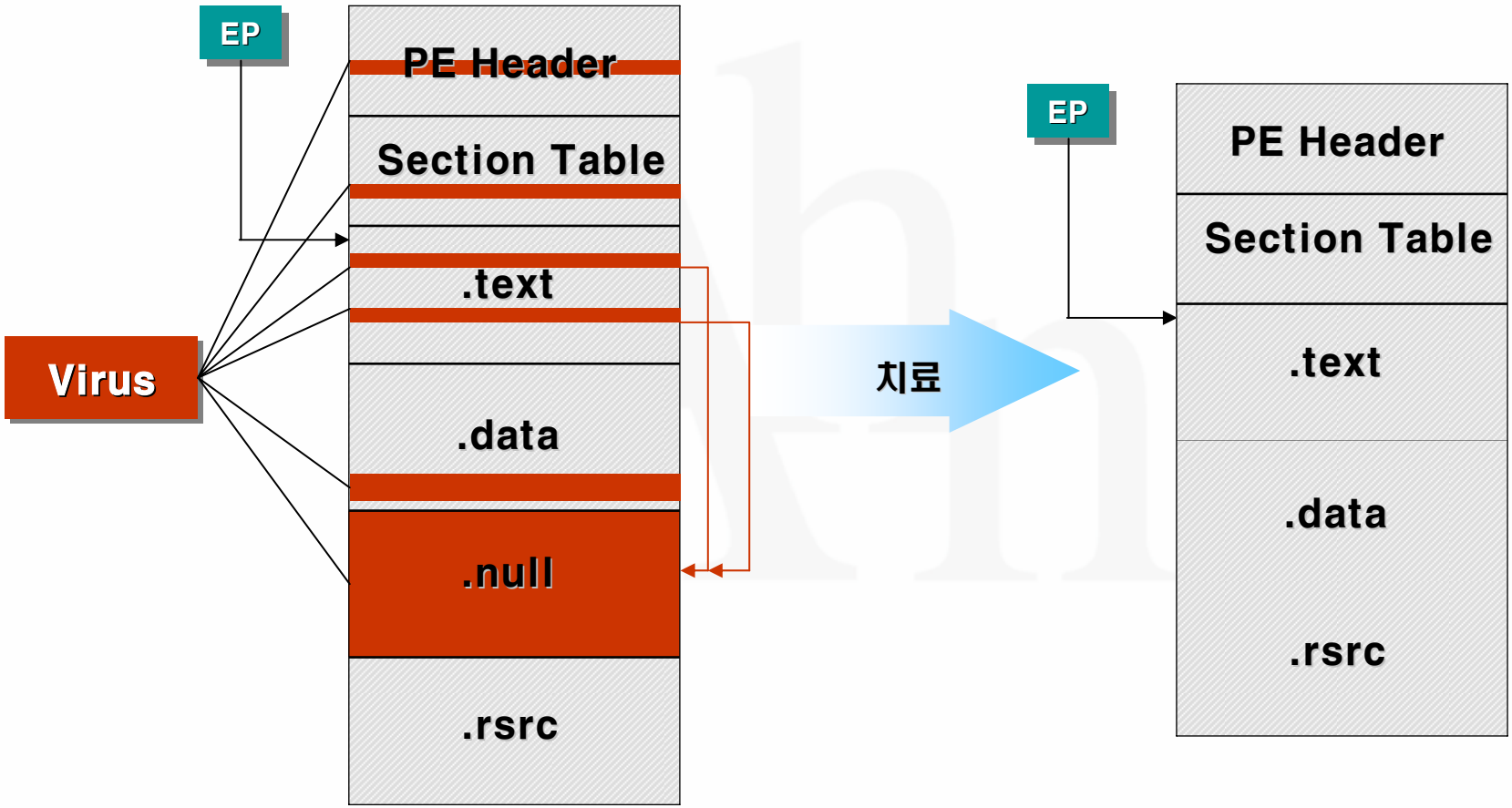


[Pinball.exe 감염]

```
////////////////////////////////////////////////////////////
// 원본 코드
////////////////////////////////////////////////////////////
010063A1  817D 0C BD0300    CMP DWORD PTR SS:[EBP+C],3BD
010063A8  74 07             JE SHORT Pinball.010063B1
010063AA  5D                POP EBP
010063AB  FF25 84110001     JMP PTR DS:[<&USER32.DefWindowProcA>]
010063B1  8B15 E8610201     MOV EDX,DWORD PTR DS:[10261E8]
010063B7  53                PUSH EBX



////////////////////////////////////////////////////////////
// 바이러스에 의해 수정된 코드
////////////////////////////////////////////////////////////
010063A1  817D 0C BD0300    CMP DWORD PTR SS:[EBP+C],3BD
010063A8  74 07             JE SHORT PINBALL.010063B1
010063AA  5D                POP EBP
010063AB  E9 31A80400       JMP PINBALL.01050BE1    ; EP of Virus
010063B0  01                DB 01
010063B1  8B15 E8610201     MOV EDX,DWORD PTR DS:[10261E8]
010063B7  53                PUSH EBX
```

3.Win32/Polip 치료 (2/2)

치료 (Entry Point 치료 Vs 원본 파일에 가깝게)



Ⅴ. 결 론

결 론

참고 문헌

http://www.ahnlab.com
http://www.ncsc.go.kr
http://www.kisa.or.kr
http://www.sysinternals.com
http://www.rootkit.com
http://www.microsoft.com/korea/technet/security
http://msdn.microsoft.com/default.aspx
http://vx.netlux.org/lib/vmd03.html
http://www.bitdefender.com/VIRUS-1000066-en--Win32.Polip.A.html
http://secunia.com/virus_information
http://www.virusbtn.com/index
The Windows 2000 Device Driver Book
Inside Microsoft Windows2000 Third Edition (Mark E. Russinovich)
Programming Applications for Microsoft Windows Fourth Edition (Jeffrey Ritcher)
Subverting The Windows Kernel Rootkits (Greg Hoglund)
Programming the Microsoft Windows Driver Model (Walter Oney)
The Art of Computer Virus Research and Defender (Peter Szor)
Hacker Dissembling Uncovered (Kris Kaspersky)

Q/A

경청해 주셔서 감사합니다