파일바이러스 분석 및 치료로직 개발

한정화 (daly25@gmail.com)

순서

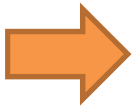
- 악성프로그램
- 파일바이러스
- PE 구조
- 파일 바이러스 감염 유형
- 파일 바이러스 분석 및 치료로직 개발

악성 프로그램

- Trojan
- Backdoor
- Worm
- Spyware
- Adware
- Virus

파일(감염형) 바이러스

- Trojan
- Backdoor
- Worm
- Spyware
- Adware
- Virus
  - Sality
  - Virut
  - Parite
  - Patched
  - Detnat



파일 감염형 바이러스
(파일 바이러스)

악성프로그램 통계

Current rank	Delta	Verdict
1	0	Net-Worm.Win32.Kido.ir
2	0	Virus.Win32.Sality.aa
3	6	HackTool.Win32.Kiser.zv
4	-1	Net-Worm.Win32.Kido.ih
5	2	Virus.Win32.Sality.bh
6	-2	Hoax.Win32.Screensaver.b
7	-2	AdWare.Win32.HotBar.dh
8	0	Virus.Win32.Virut.ce
9	-3	Trojan.JS.Agent.bhr
10	1	HackTool.Win32.Kiser.il

Kaspersky Lab - Monthly Malware Statistics, February 2011

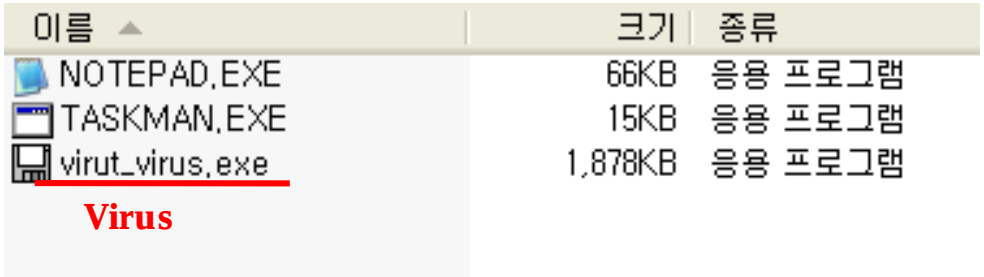
파일바이러스란?

- 파일 감염형 바이러스
- 특징(일반적)
  - 정상 파일에 기생하여 악의적인 행동을 함
  - 악성코드 자체로는 실행 수 없음
  - 실행 가능한 파일들을 감염시킴
  - 하나의 정상파일이 중복 감염될 수 있음
  - 숙주 파일이 존재 함
  - 다형성 바이러스로 진화 함

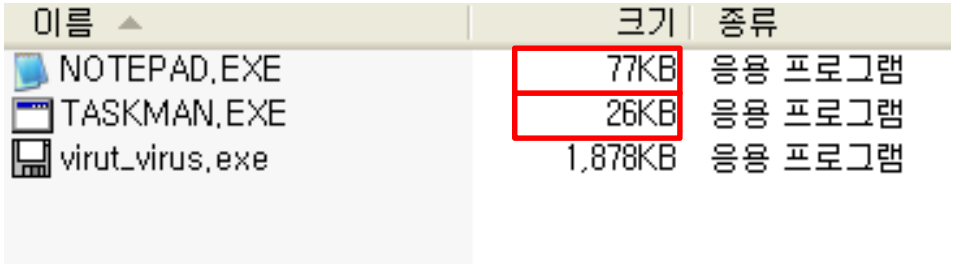
감염된 파일 비교

- 파일 사이즈 증가

감염 전

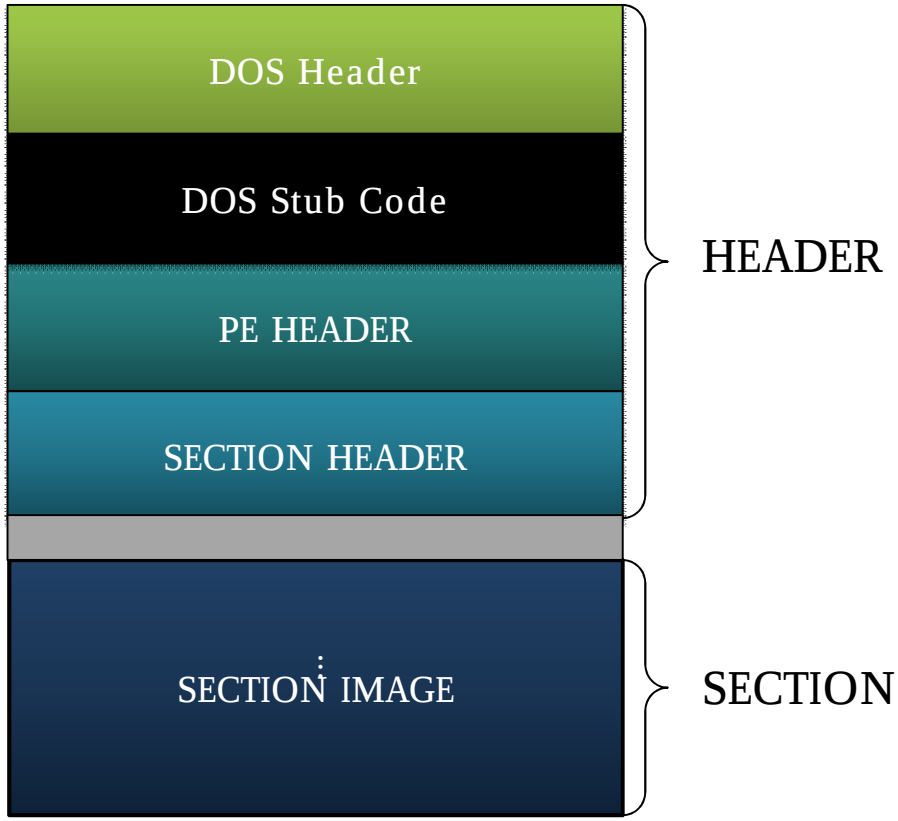


감염 후

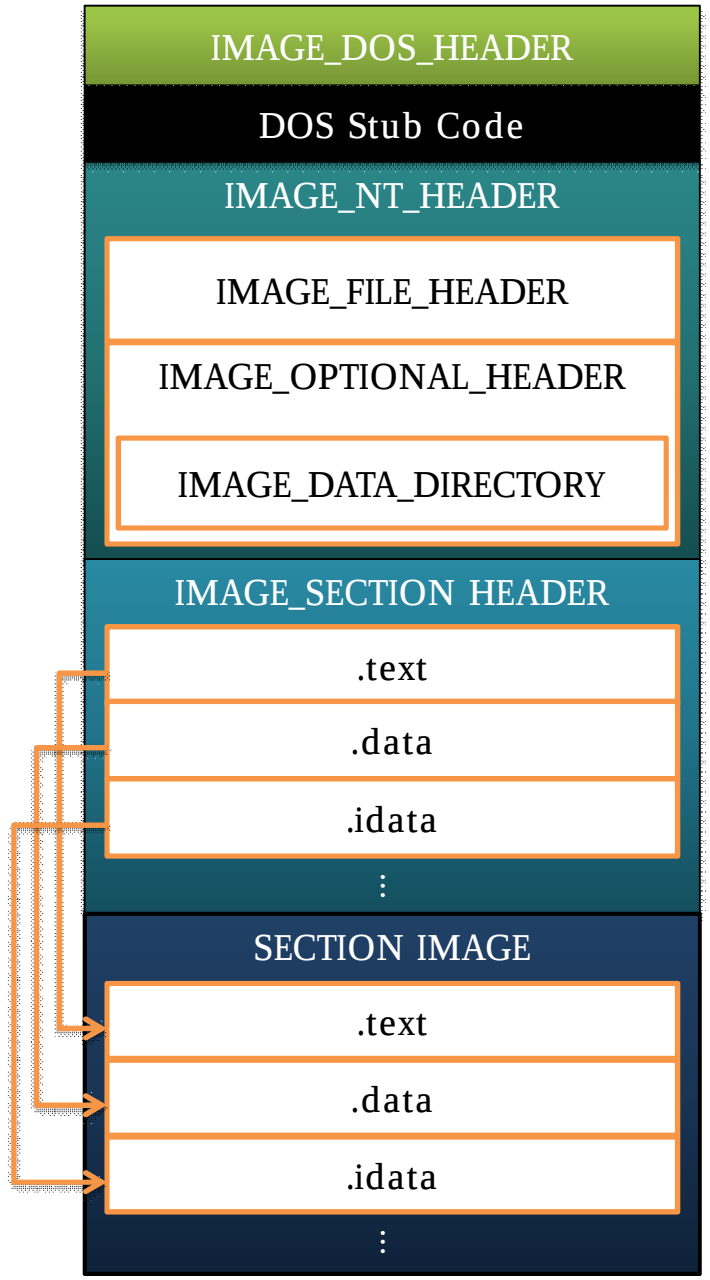


PE Format

PE 구조 (1)


DOS Header
DOS Stub Code
PE HEADER
SECTION HEADER
SECTION IMAGE
HEADER
SECTION

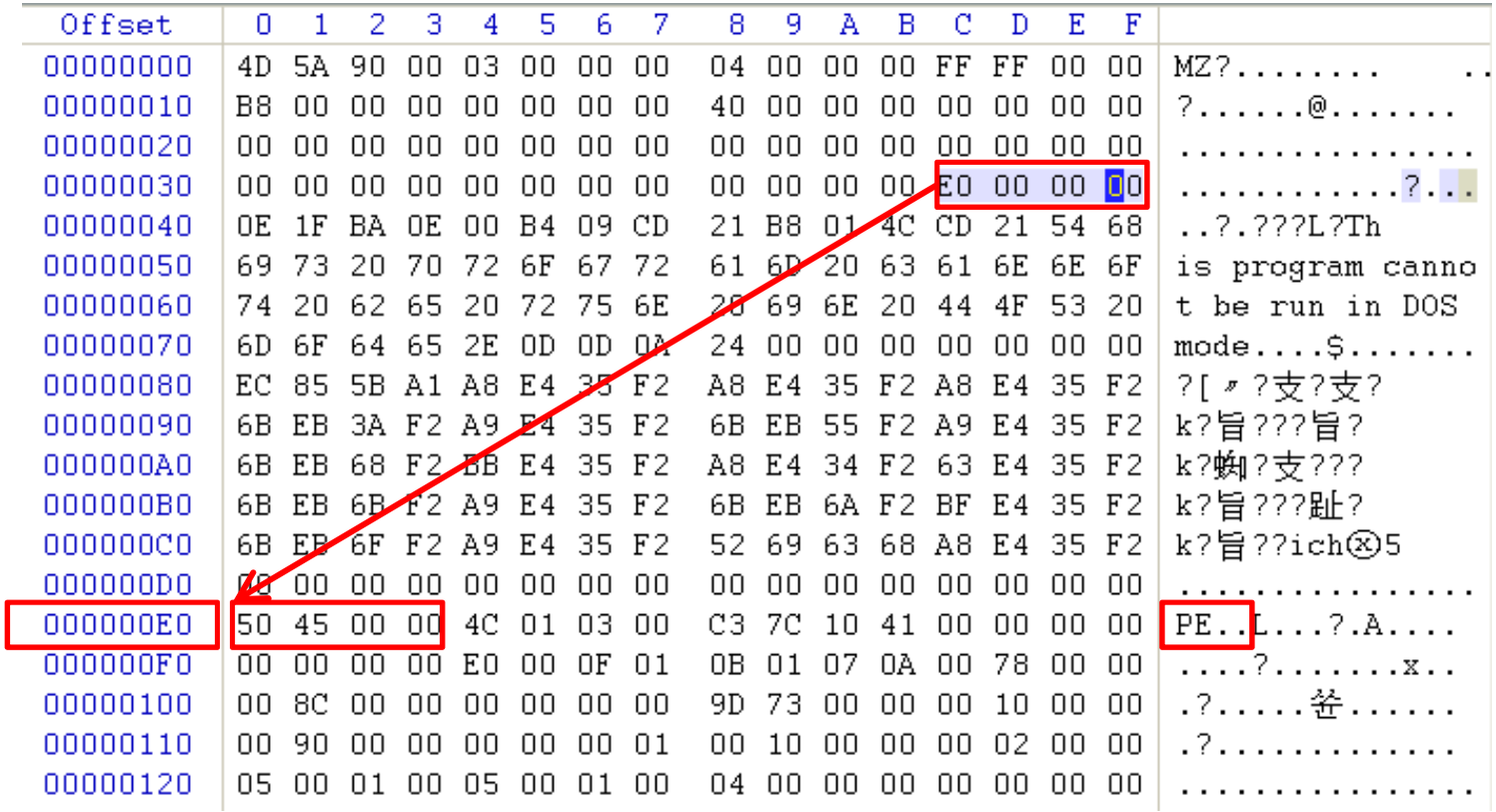
PE 구조도(2)


IMAGE_DOS_HEADER
DOS Stub Code
IMAGE_NT_HEADER
IMAGE_FILE_HEADER
IMAGE_OPTIONAL_HEADER
IMAGE_DATA_DIRECTORY
IMAGE_SECTION HEADER
.text
.data
.idata
SECTION IMAGE
.text
.data
.idata

DOS Header

```
typedef struct _IMAGE_DOS_HEADER {      // DOS .EXE header
    WORD   e_magic;                     // Magic number
    WORD   e_cblp;                      // Bytes on last page of file
    WORD   e_cp;                        // Pages in file
    WORD   e_crlc;                      // Relocations
    WORD   e_cparhdr;                   // Size of header in paragraphs
    WORD   e_minalloc;                  // Minimum extra paragraphs needed
    WORD   e_maxalloc;                  // Maximum extra paragraphs needed
    WORD   e_ss;                        // Initial (relative) SS value
    WORD   e_sp;                        // Initial SP value
    WORD   e_csum;                      // Checksum
    WORD   e_ip;                        // Initial IP value
    WORD   e_cs;                        // Initial (relative) CS value
    WORD   e_lfarlc;                    // File address of relocation table
    WORD   e_ovno;                      // Overlay number
    WORD   e_res[4];                    // Reserved words
    WORD   e_oemid;                     // OEM identifier (for e_oeminfo)
    WORD   e_oeminfo;                   // OEM information; e_oemid specific
    WORD   e_res2[10];                  // Reserved words
    LONG   e_lfanew;                    // File address of new exe header
  } IMAGE_DOS_HEADER, *PIMAGE_DOS_HEADER;
```

DOS Header – e_lfanwe



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ?..... ..
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	?.....@.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000030	00	00	00	00	00	00	00	00	00	00	00	00	E0	00	00	00	.....?...
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	..?.???L?Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode....$.....
00000080	EC	85	5B	A1	A8	E4	35	F2	A8	E4	35	F2	A8	E4	35	F2	?[〃?支?支?
00000090	6B	EB	3A	F2	A9	E4	35	F2	6B	EB	55	F2	A9	E4	35	F2	k?旨???旨?
000000A0	6B	EB	68	F2	BB	E4	35	F2	A8	E4	34	F2	63	E4	35	F2	k?蜘?支???
000000B0	6B	EB	6B	F2	A9	E4	35	F2	6B	EB	6A	F2	BF	E4	35	F2	k?旨???趾?
000000C0	6B	EB	6F	F2	A9	E4	35	F2	52	69	63	68	A8	E4	35	F2	k?旨??ich⓪5
000000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000E0	50	45	00	00	4C	01	03	00	C3	7C	10	41	00	00	00	00	PE..L...?.A....
000000F0	00	00	00	00	E0	00	0F	01	0B	01	07	0A	00	78	00	00	....?.......x..
00000100	00	8C	00	00	00	00	00	00	9D	73	00	00	00	10	00	00	.?.....쑵.....
00000110	00	90	00	00	00	00	00	01	00	10	00	00	00	02	00	00	.?.....
00000120	05	00	01	00	05	00	01	00	04	00	00	00	00	00	00	00	.....

NT Header

```
typedef struct _IMAGE_NT_HEADERS {
    DWORD Signature;
    IMAGE_FILE_HEADER FileHeader;
    IMAGE_OPTIONAL_HEADER32 OptionalHeader;
} IMAGE_NT_HEADERS32, *PIMAGE_NT_HEADERS32;
```

File Header

```
typedef struct _IMAGE_FILE_HEADER {
    WORD    Machine;
    WORD    NumberOfSections;
    DWORD   TimeDateStamp;
    DWORD   PointerToSymbolTable;
    DWORD   NumberOfSymbols;
    WORD    SizeOfOptionalHeader;
    WORD    Characteristics;
} IMAGE_FILE_HEADER, *PIMAGE_FILE_HEADER;
```

Optional Header

```
typedef struct _IMAGE_OPTIONAL_HEADER {
    WORD    Magic;
    BYTE    MajorLinkerVersion;
    BYTE    MinorLinkerVersion;
    DWORD   SizeOfCode;
    DWORD   SizeOfInitializedData;
    DWORD   SizeOfUninitializedData;
    DWORD   AddressOfEntryPoint;
    DWORD   BaseOfCode;
    DWORD   BaseOfData;
    DWORD   ImageBase;
    DWORD   SectionAlignment;
    DWORD   FileAlignment;
    WORD    MajorOperatingSystemVersion;
    WORD    MinorOperatingSystemVersion;
    WORD    MajorImageVersion;
    WORD    MinorImageVersion;
    WORD    MajorSubsystemVersion;
    WORD    MinorSubsystemVersion;
    DWORD   Win32VersionValue;
    DWORD   SizeOfImage;
    DWORD   SizeOfHeaders;
    DWORD   CheckSum;
    WORD    Subsystem;
    WORD    DllCharacteristics;
    DWORD   SizeOfStackReserve;
    DWORD   SizeOfStackCommit;
    DWORD   SizeOfHeapReserve;
    DWORD   SizeOfHeapCommit;
    DWORD   LoaderFlags;
    DWORD   NumberOfRvaAndSizes;
    IMAGE_DATA_DIRECTORY DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];
} IMAGE_OPTIONAL_HEADER32, *PIMAGE_OPTIONAL_HEADER32;
```

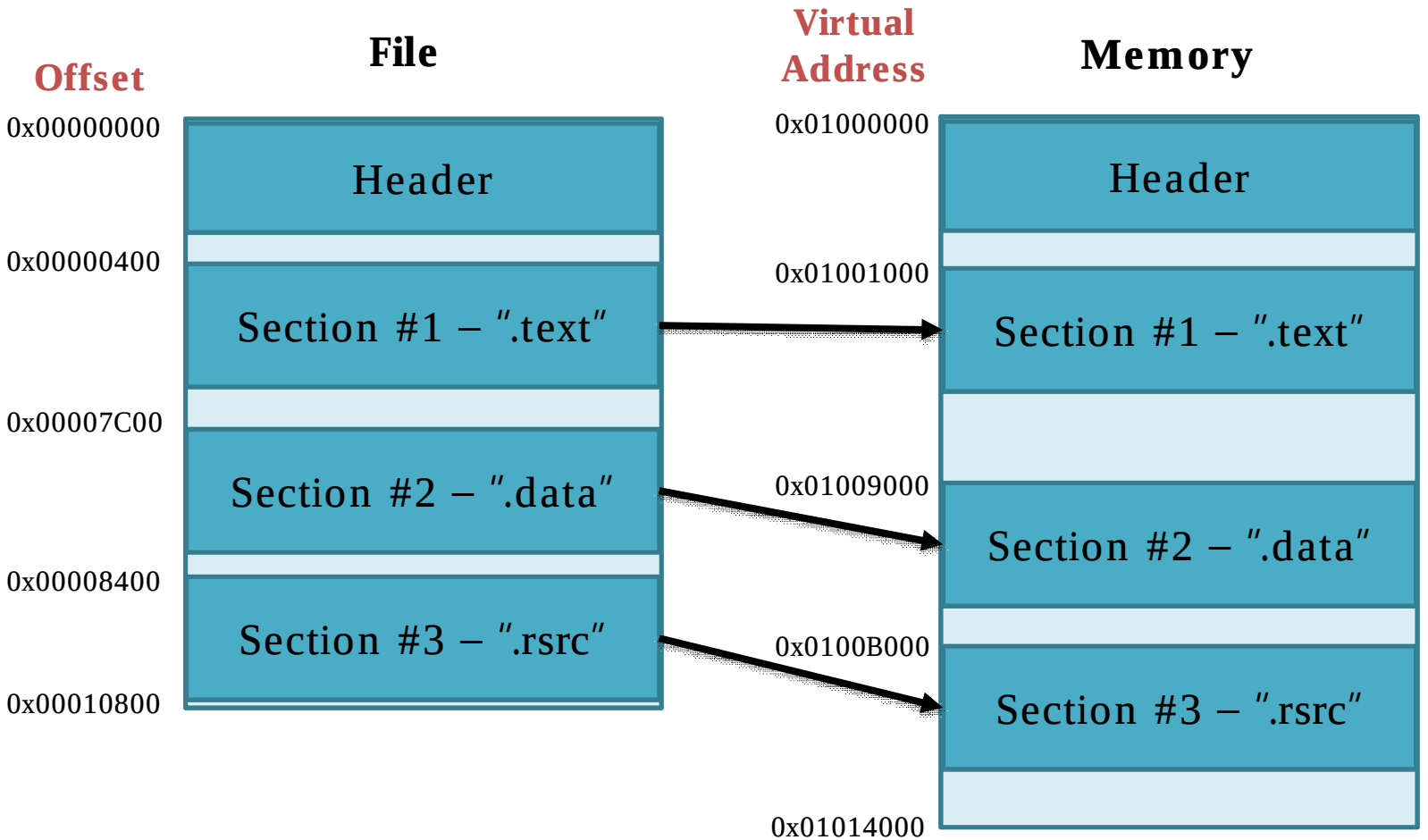
SECTION HEADER

```
typedef struct _IMAGE_ {
    BYTE    Name[IMAGE_SIZEOF_SHORT_NAME];
    union {
            DWORD   PhysicalAddress;
            DWORD   VirtualSize;
    } Misc;
    DWORD   VirtualAddress;
    DWORD   SizeOfRawData;
    DWORD   PointerToRawData;
    DWORD   PointerToRelocations;
    DWORD   PointerToLinenumbers;
    WORD    NumberOfRelocations;
    WORD    NumberOfLinenumbers;
    DWORD   Characteristics;
} IMAGE_SECTION_HEADER, *PIMAGE_SECTION_HEADER;
```

HEADERS (Coff+Optional)

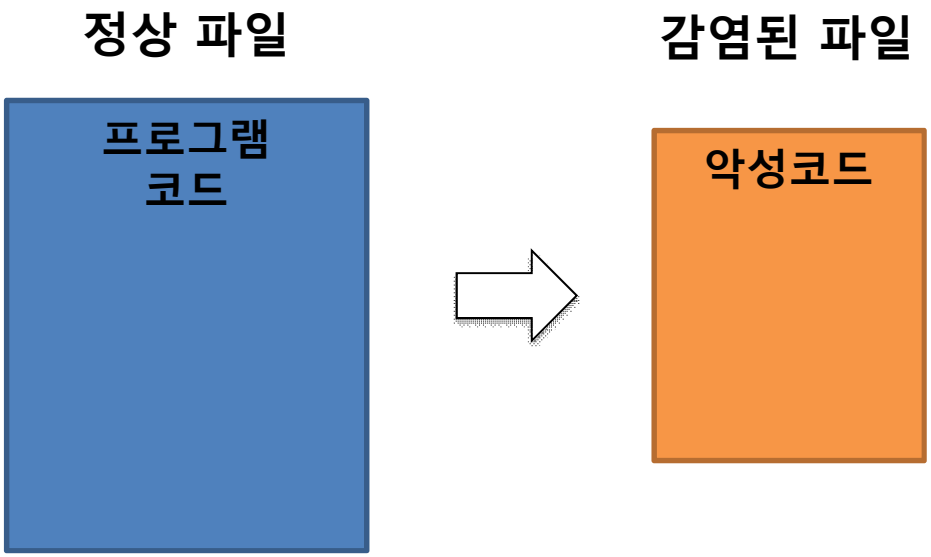
0000739D	EntryPoint (rva)
0000679D	EntryPoint (raw)
01000000	ImageBase
00014000	Size of Image
00001000	Sections Alignment
00000200	File Alignment
00000003	Number of sections

No	Name	VirtualSize	VirtualOffset	RawSize	RawOffset	Characteristics
01	.text	00007748	00001000	00007800	00000400	60000020
02	.data	00001BA8	00009000	00000800	00007C00	C0000040
03	.rsrc	00008304	0000B000	00008400	00008400	40000040

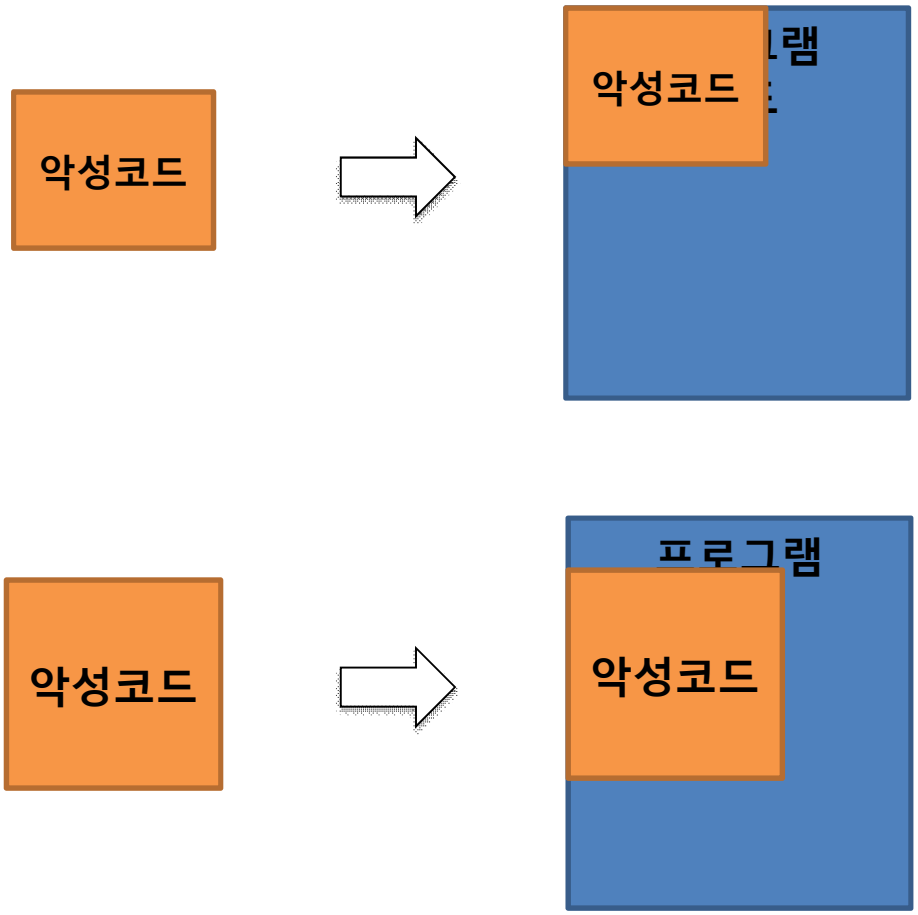


파일 바이러스 감염 유형

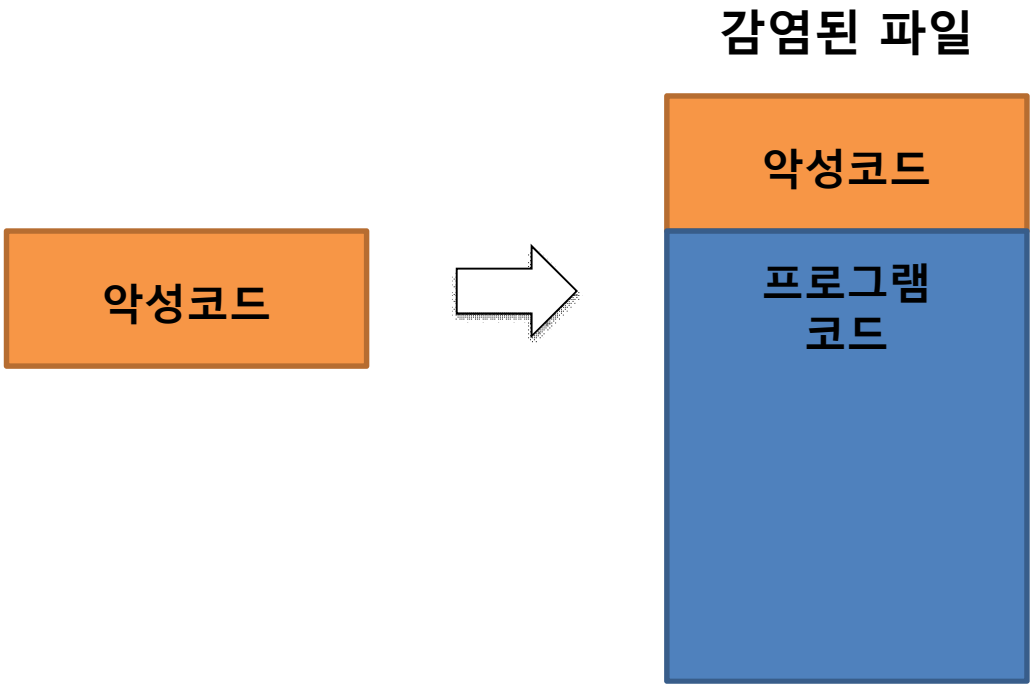
Overwriting 감염형



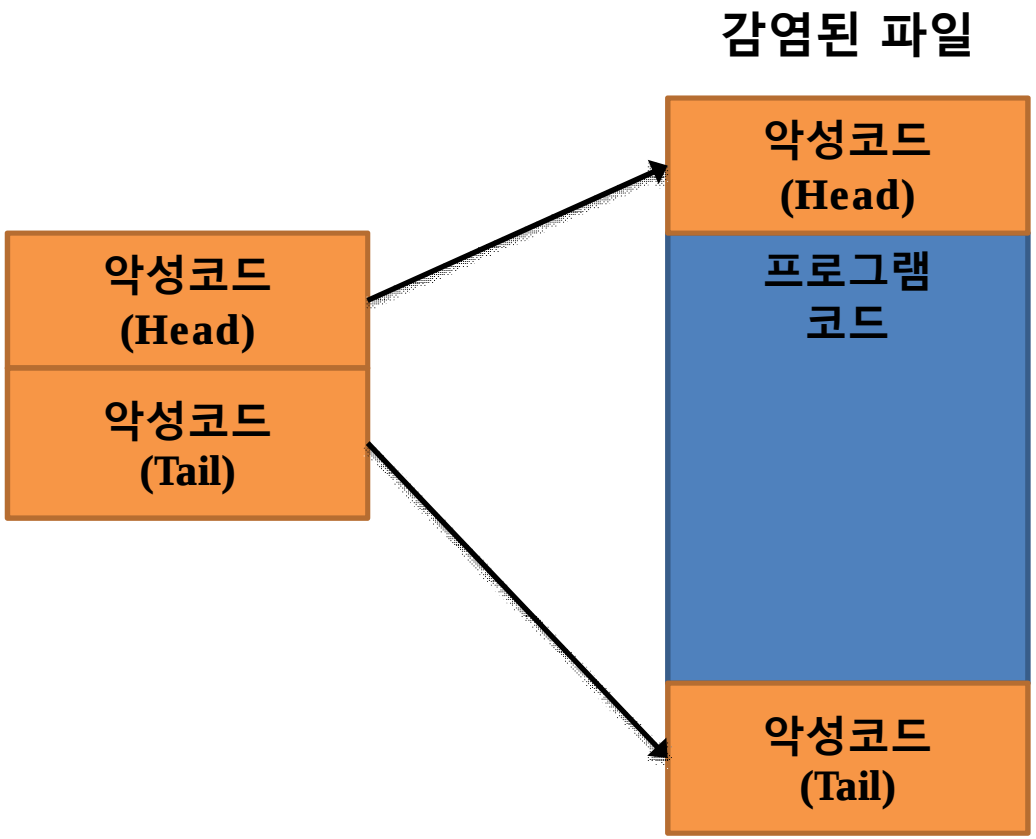
Overwriting 감염형



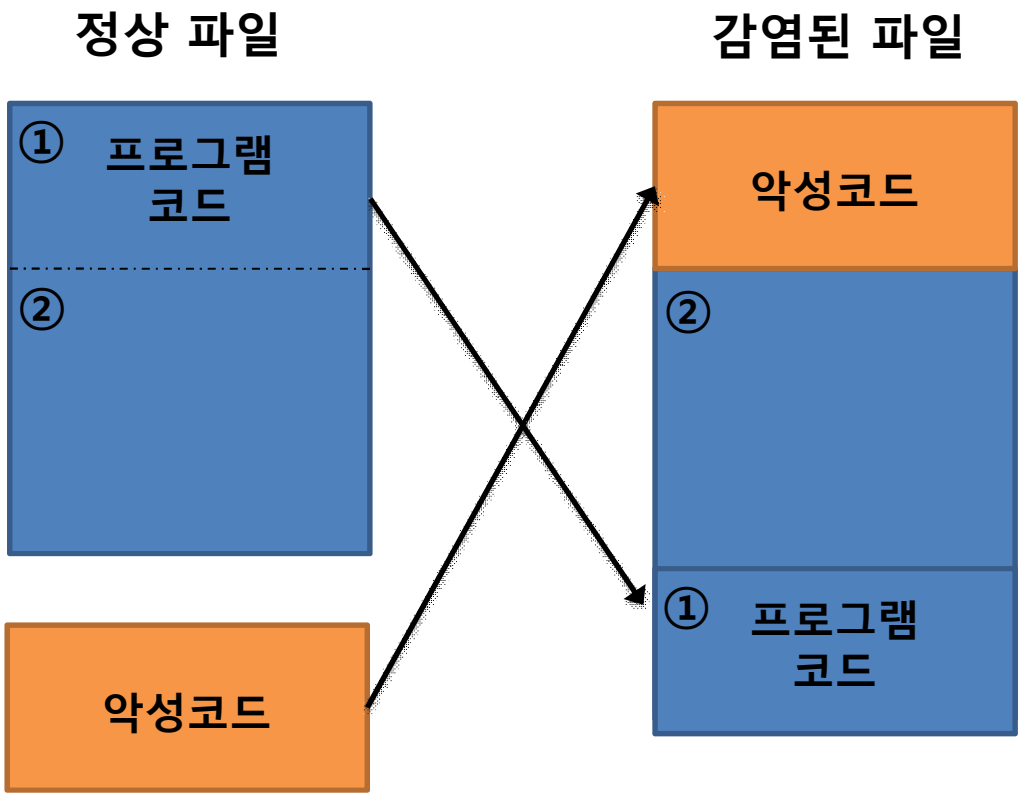
Prepending 감염형



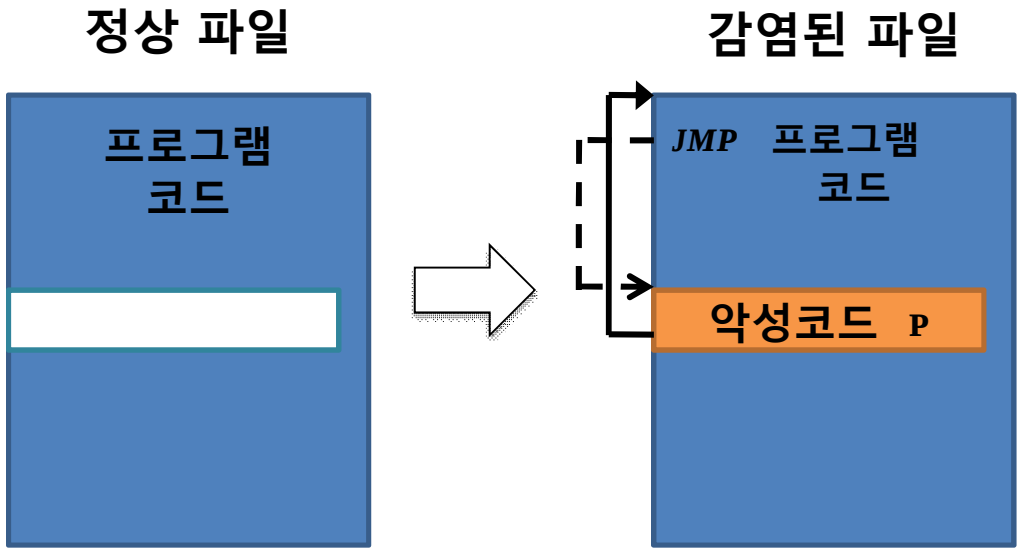
Amoeba 감염형



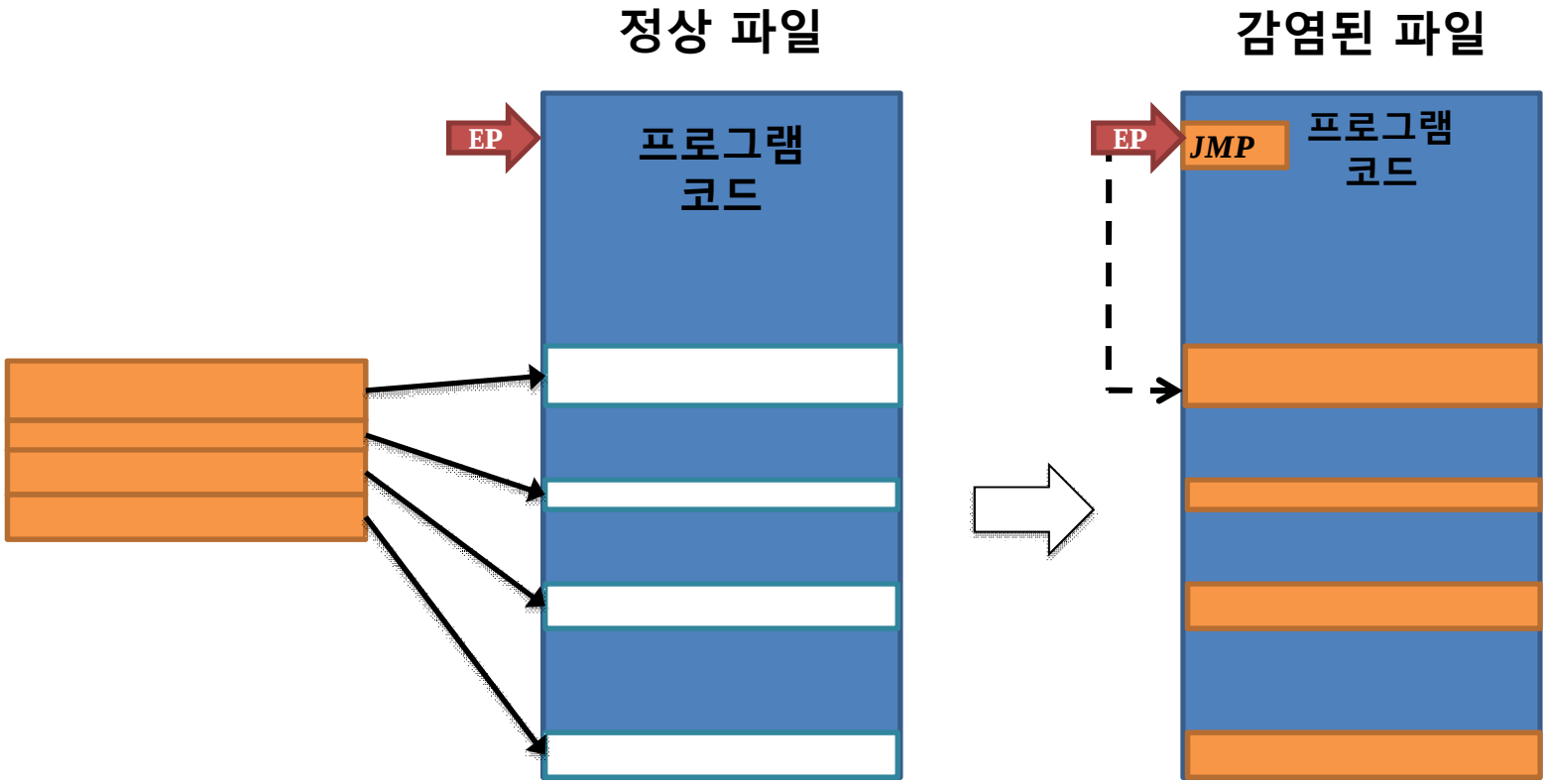
Classic 감염형



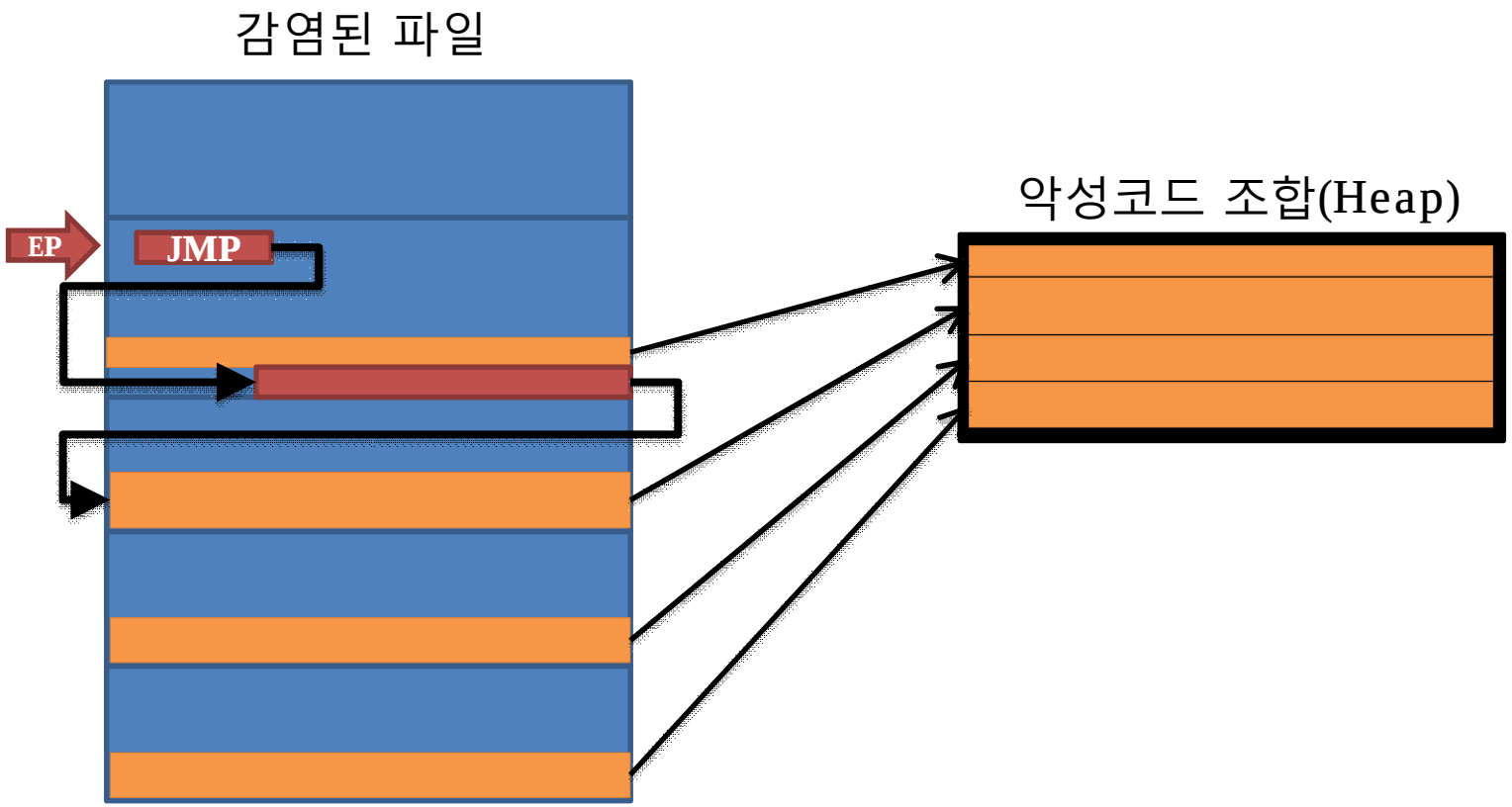
Cavity 감염형



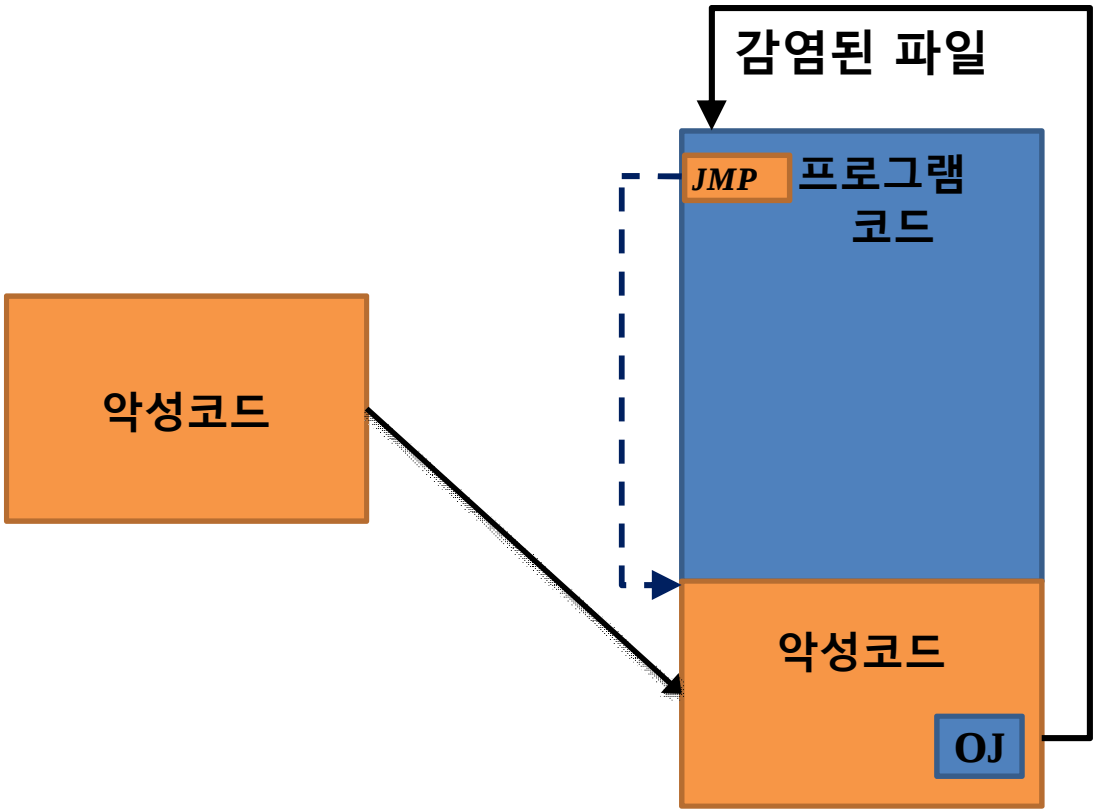
Cavity 감염형 – 변종



Cavity 감염형 – 변종 특징

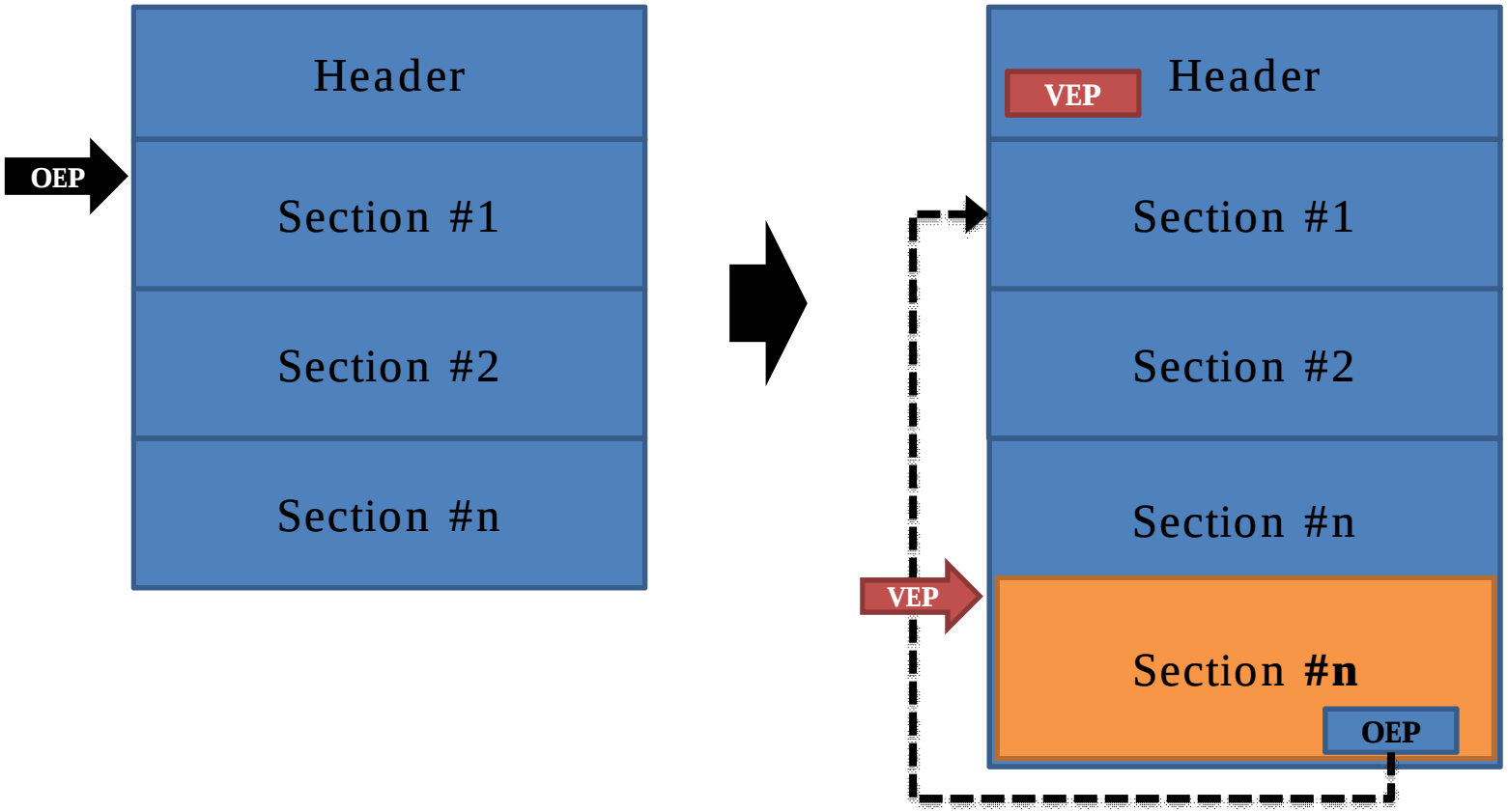


Appending 감염형



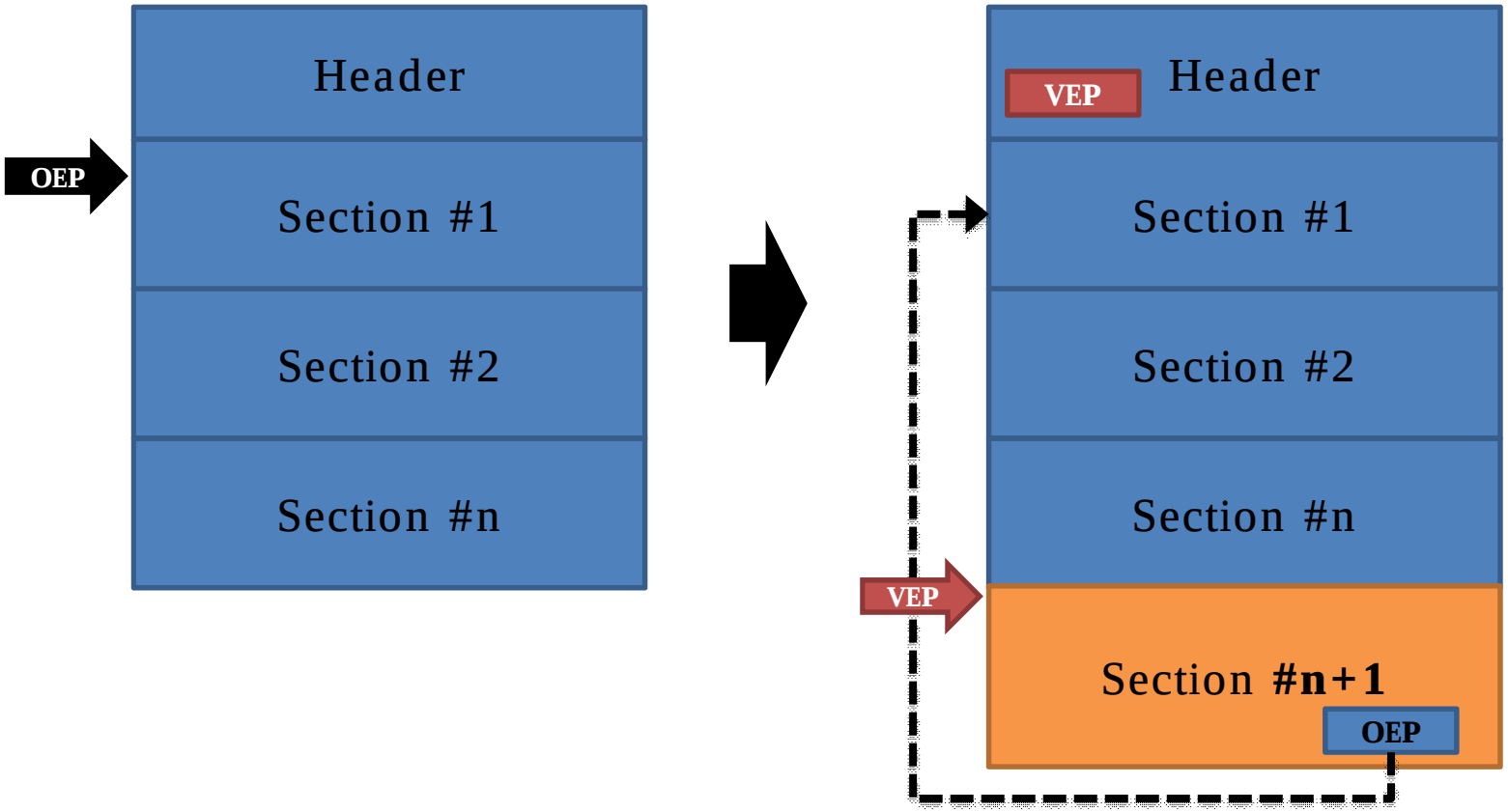
Appending 감염형 종류

- 악성코드가 마지막 섹션에 덧붙인 경우



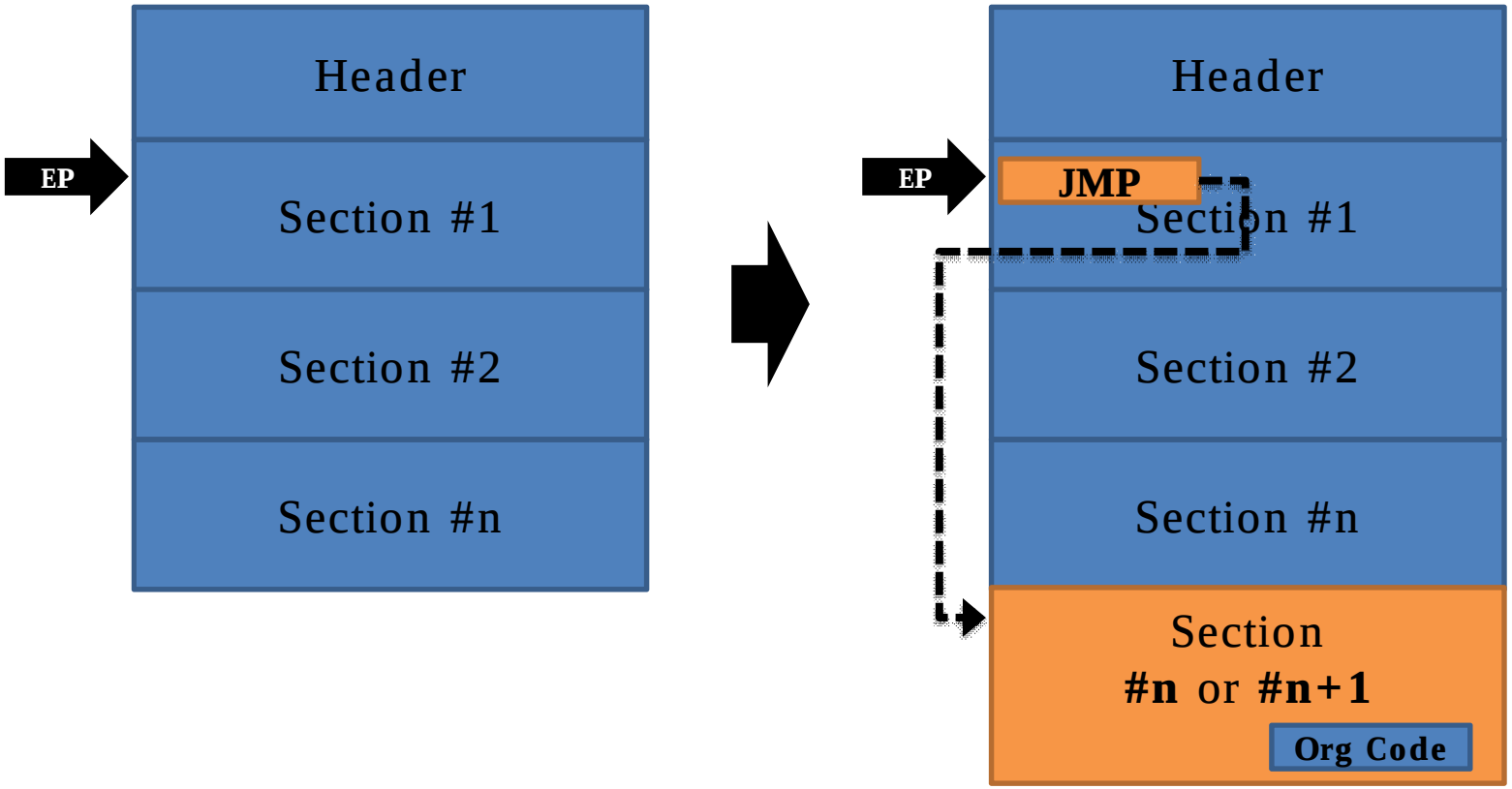
Appending 감염형 종류

- 악성코드가 섹션으로 덧붙은 경우



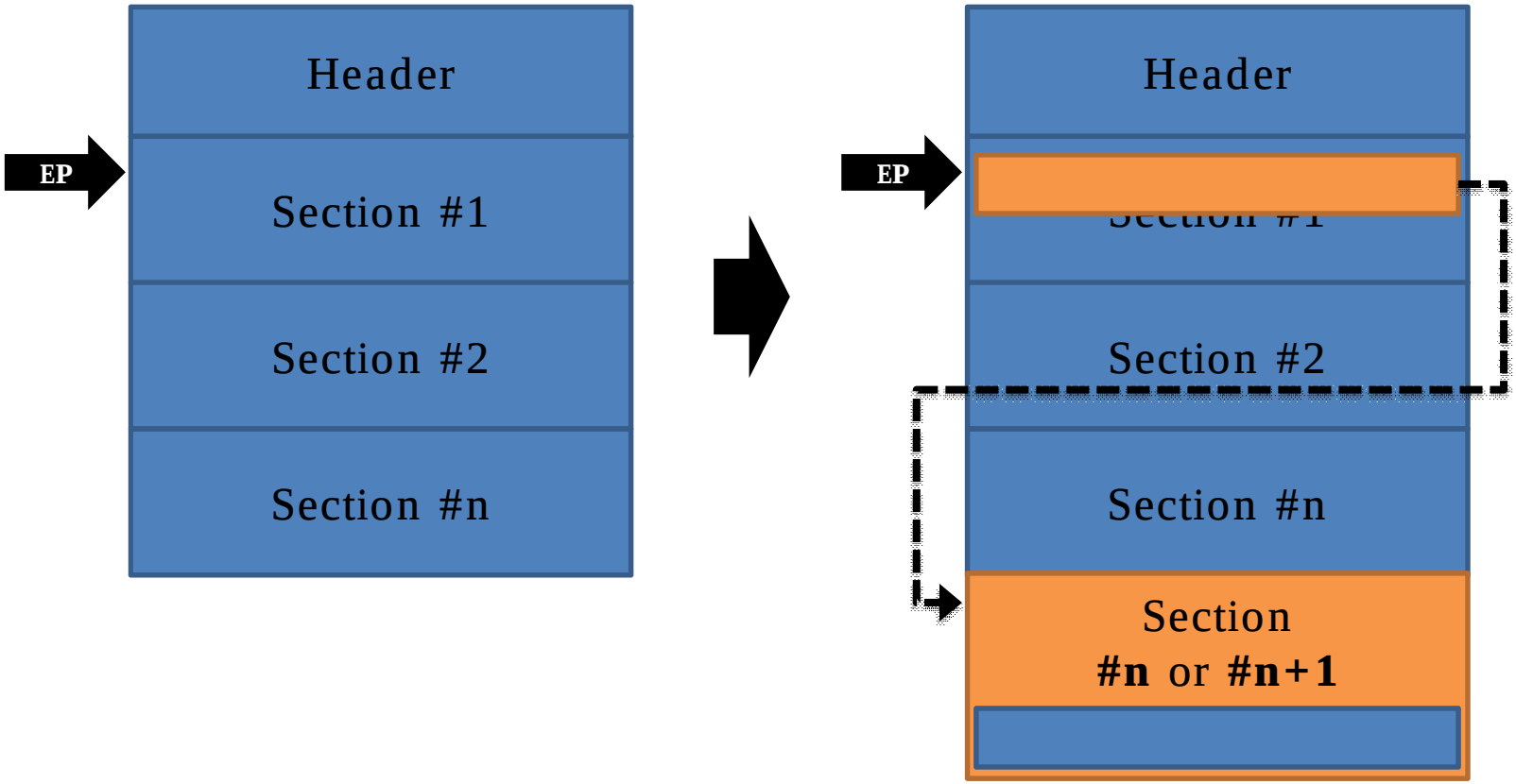
Appending 감염형 종류

- 악성코드로 JMP하는 코드가 EP에 패치된 경우



Appending 감염형 종류

- 악성코드 일부가 EP에 패치된 경우



파일 바이러스 감염유형 분석 및 치료로직 설계

Sality 분석 및 치료로직 설계

대상 샘플 정보

- 진단명 (VirusTotal)

Anti-Virus	Virus Name
AhnLab-V3	**Win32/Sality.S**
Avast	**Win32:Sality**
BitDefender	**Win32.Sality.L**
Kaspersky	**Virus.Win32.Sality.h**
Microsoft	**Worm:Win32/Sality.I**
Sophos	**W32/Sality-AL**
ViRobot	**Win32.Sality.O**

Sality PE 구조

PEview

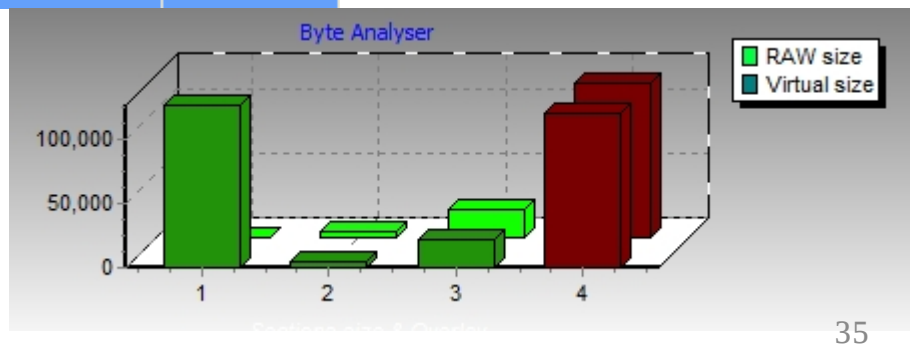
File View Go Help

Sality.han.exe
- IMAGE_DOS_HEADER
- MS-DOS Stub Program
- IMAGE_NT_HEADERS
- IMAGE_SECTION_HEADER 1
- IMAGE_SECTION_HEADER 2 1
- IMAGE_SECTION_HEADER 3 2
- SECTION 1
- SECTION 2 1
- SECTION 3 2

pFile	Raw Data	Value
00006E10	A2 35 2A F0 28 01 0F 14 50 33 ED FF EE B7 B6 0C	.5*.(...P3.....
00006E20	60 17 11 26 06 13 A5 1D 61 1F 5F 20 1E 14 10 0D	`..&....a._....
00006E30	0E 76 FB DF 0A 47 1D 1A 86 37 2C 44 07 17 1B 1D	.v...G...7,D....
00006E40	82 20 06 00 2A 34 5E 50 F0 B5 20 24 0D 77 B2 5B	...*4^P.. $.w.[
00006E50	EA 09 00 49 09 2C 03 32 6E B0 B1 B3 4F 0C 14 05	...I.,.2n...O...
00006E60	26 1A 0A 1C 53 58 FB 65 00 13 65 F0 B7 FD 00 3C	&...SX.e..e....<
00006E70	42 B2 B3 60 33 05 6F 17 B6 4E 36 60 BF 6F 0A 57	B..`3.o..N6`.o.W
00006E80	2A 1F 09 5E 0A 85 B7 09 FB 31 6D 48 29 02 F0 C3	*..^.....1mH)...
00006E90	05 75 C5 B7 A7 EC 0D 52 04 00 54 04 B9 04 6C 04	.u.....R..T...l.
00006EA0	00 9F 00 00 00 4A 42 00 09 00 00 FF	.....JB.....

Stud PE

No	Name	VirtualSize	VirtualOf...	RawSize	RawOffset	Charact...
01	1	0001F000	00001000	00000000	00000400	E0000040
02	2 1	000010D0	00020000	00001200	00000400	C0000040
03	3 2	000058AC	00022000	000058AC	00001600	E0000020
*	ExtraDat			00012958	00006EAC	



FileMon 툴을 이용한 분석

Sality.han.exe:2472	QUERY INFORMATION	C:\Documents and Settir	\Sality.han.~01	NOT FOUND	Attributes: Error
Sality.han.exe:2472	OPEN	C:\Documents and Settir	\Sality.han.~01	NOT FOUND	Options: Open Access: 00100100
Sality.han.exe:2472	READ	C:\WINDOWS\system32		SUCCESS	Offset: 246784 Length: 4096
Sality.han.exe:2472	OPEN	C:\Documents and Settir	\Sality.han.~01	NOT FOUND	Options: Open Access: 00010080
Sality.han.exe:2472	OPEN	C:\Documents and Settir	\Sality.han.~01	NOT FOUND	Options: Open Access: 00100100
Sality.han.exe:2472	QUERY INFORMATION	C:\Documents and Settir	\Sality.han.~01	NOT FOUND	Attributes: Error
Sality.han.exe:2472	OPEN	C:\Documents and Settir	\Sality.han.exe	SUCCESS	Options: Open Sequential Access: Read
Sality.han.exe:2472	QUERY INFORMATION	C:\Documents and Settir	\Sality.han.exe	SUCCESS	FileAttributeTagInformation
Sality.han.exe:2472	QUERY INFORMATION	C:\Documents and Settir	\Sality.han.exe	SUCCESS	Length: 104452
Sality.han.exe:2472	QUERY INFORMATION	C:\Documents and Settir	\Sality.han.exe	SUCCESS	Attributes: A
Sality.han.exe:2472	QUERY INFORMATION	C:\Documents and Settir	\Sality.han.exe	SUCCESS	FileStreamInformation
Sality.han.exe:2472	QUERY INFORMATION	C:\Documents and Settir	\Sality.han.exe	SUCCESS	Attributes: A
Sality.han.exe:2472	QUERY INFORMATION	C:\Documents and Settir	\Sality.han.exe	SUCCESS	FileEaInformation
Sality.han.exe:2472	CREATE	C:\Documents and Settir	\Sality.han.~01	SUCCESS	Options: OverwriteIf Sequential Access: 001301
Sality.han.exe:2472	OPEN	C:\Documents and Settir	\	SUCCESS	Options: Open Directory Access: 00100000
Sality.han.exe:2472	QUERY INFORMATION	C:\Documents and Settir	\Sality.han.~01	SUCCESS	FileFsAttributeInformation
Sality.han.exe:2472	QUERY INFORMATION	C:\Documents and Settir	\Sality.han.~01	SUCCESS	Attributes: A
Sality.han.exe:2472	QUERY INFORMATION	C:\Documents and Settir	\Sality.han.exe	SUCCESS	FileFsAttributeInformation
Sality.han.exe:2472	SET INFORMATION	C:\Documents and Settir	\Sality.han.~01	SUCCESS	Length: 104452
Sality.han.exe:2472	QUERY INFORMATION	C:\Documents and Settir	\Sality.han.exe	SUCCESS	Length: 104452
Sality.han.exe:2472	WRITE	C:\Documents and Settir	\Sality.han.~01	SUCCESS	Offset: 0 Length: 65536
Sality.han.exe:2472	READ	C:\Documents and Settir	\Sality.han.exe	SUCCESS	Offset: 32768 Length: 32768
Sality.han.exe:2472	READ	C:\Documents and Settir	\Sality.han.exe	SUCCESS	Offset: 65536 Length: 32768
Sality.han.exe:2472	READ	C:\Documents and Settir	\Sality.han.exe	SUCCESS	Offset: 98304 Length: 4096
Sality.han.exe:2472	WRITE	C:\Documents and Settir	\Sality.han.~01	SUCCESS	Offset: 65536 Length: 38916
Sality.han.exe:2472	SET INFORMATION	C:\Documents and Settir	\Sality.han.~01	SUCCESS	FileBasicInformation
Sality.han.exe:2472	CLOSE	C:\Documents and Settir	\Sality.han.exe	SUCCESS	
Sality.han.exe:2472	CLOSE	C:\Documents and Settir	\Sality.han.~01	SUCCESS	
Sality.han.exe:2472	OPEN	C:\Documents and Settir	\Sality.han.~01	SUCCESS	Options: Open Access: 00100100
Sality.han.exe:2472	SET INFORMATION	C:\Documents and Settir	\Sality.han.~01	SUCCESS	FileBasicInformation
Sality.han.exe:2472	CLOSE	C:\Documents and Settir	\Sality.han.~01	SUCCESS	
Sality.han.exe:2472	OPEN	C:\Documents and Settir	\Sality.han.~01	SUCCESS	Options: Open Access: 0012019F
Sality.han.exe:2472	QUERY INFORMATION	C:\Documents and Settir	\Sality.han.~01	SUCCESS	Attributes: H
Sality.han.exe:2472	QUERY INFORMATION	C:\Documents and Settir	\Sality.han.~01	SUCCESS	Length: 104452
Sality.han.exe:2472	READ	C:\Documents and Settir	\Sality.han.~01	SUCCESS	Offset: 0 Length: 28332
Sality.han.exe:2472	READ	C:\Documents and Settir	\Sality.han.~01	SUCCESS	Offset: 768 Length: 2
Sality.han.exe:2472	READ	C:\Documents and Settir	\Sality.han.~01	SUCCESS	Offset: 73728 Length: 28332
Sality.han.exe:2472	SET INFORMATION	C:\Documents and Settir	\Sality.han.~01	SUCCESS	Length: 73728
Sality.han.exe:2472	SET INFORMATION	C:\Documents and Settir	\Sality.han.~01	SUCCESS	Length: 73728
Sality.han.exe:2472	WRITE	C:\Documents and Settir	\Sality.han.~01	SUCCESS	Offset: 0 Length: 28332
Sality.han.exe:2472	SET INFORMATION	C:\Documents and Settir	\Sality.han.~01	SUCCESS	FileBasicInformation
Sality.han.exe:2472	CLOSE	C:\Documents and Settir	\Sality.han.~01	SUCCESS	

파일관련 행위

1. 파일 생성

QUERY INFORMATION	C: \ Doc... \ Sality.han.~01	NOT FOUND	Attributes: Error
QUERY INFORMATION	C: \ Doc... \ Sality.han.~01	SUCCESS	Length: 104452
CREATE	C: \ Doc... \ Sality.han.~01	SUCCESS	Options: Overwrite If Sequential Access: 00130196

파일관련 행위

2. 새로 생성한 파일에 데이터 쓰기

WRITE	C:\Doc...\Sality.han.~01	SUCCESS	Offset: 0 Length: 65536
READ	C:\Doc...\Sality.han.~01	SUCCESS	Offset: 32768 Length: 32768
READ	C:\Doc...\Sality.han.~01	SUCCESS	Offset: 65536 Length: 32768
READ	C:\Doc...\Sality.han.~01	SUCCESS	Offset: 98304 Length: 4096
WRITE	C:\Doc...\Sality.han.~01	SUCCESS	Offset: 65536 Length: 38916
SET INFORMATION	C:\Doc...\Sality.han.~01	SUCCESS	Length: 104452

파일관련 행위

3. 정상코드 복구

OPEN	C:\Doc...\Sality.han.~01	SUCCESS	Options: Open Access: 0012019F
READ	C:\Doc...\Sality.han.~01	SUCCESS	Offset: 0 Length: 28332
READ	C:\Doc...\Sality.han.~01	SUCCESS	Offset: 768 Length: 2
READ	C:\Doc...\Sality.han.~01	SUCCESS	Offset: 73728 Length: 28332
SET INFORMATION	C:\Doc...\Sality.han.~01	SUCCESS	Length: 73728
WRITE	C:\Doc...\Sality.han.~01	SUCCESS	Offset: 0 Length: 28332
CLOSE	C:\Doc...\Sality.han.~01	SUCCESS	

3.1) 복원할 코드 정보

ㄱ. FileOffset : 0, Size : 0x62AC(28332)

ㄴ. FileOffset : 0x300(768), Size : 2

ㄷ. FileOffset : 0x12000(73728), Size : 0x52AC(28332)

3.2) FileSize : 0x12000(73728)

파일 관련 행위 정리

1. [파일경로].~01 파일 생성
2. 새로 생성한 파일(.~01)에 정상코드 복원
3. 정상파일(.~01) 실행
4. 실행 완료 후, 생성한파일(.~01) 삭제
5. 다른 파일들을 계속 감염시킴

악성코드 분석

- UPX 해제 부분

```
00422037    60              PUSHAD
00422038    E8 00000000     CALL    0042203D
0042203D    58              POP     EAX
0042203E    83E8 3D         SUB     EAX, 3D
00422041    50              PUSH    EAX
00422042    8DB8 00F0FDFF   LEA     EDI, DWORD PTR DS:[EAX+FFFDF000]
```

```
004221A2  v 74 22           JE      SHORT 004221C6
004221A4    3C EF           CMP     AL, 0EF
004221A6  v 77 11           JA      SHORT 004221B9
004221A8    01C3            ADD     EBX, EAX
004221AA    8B03            MOV     EAX, DWORD PTR DS:[EBX]
004221AC    86C4            XCHG    AH, AL
004221AE    C1C0 10         ROL     EAX, 10
004221B1    86C4            XCHG    AH, AL
004221B3    01F0            ADD     EAX, ESI
004221B5    8903            MOV     DWORD PTR DS:[EBX], EAX
004221B7  ^ EB E2           JMP     SHORT 0042219B
004221B9    24 0F           AND     AL, 0F
004221BB    C1E0 10         SHL     EAX, 10
004221BE    66:8B07         MOV     AX, WORD PTR DS:[EDI]
004221C1    83C7 02         ADD     EDI, 2
004221C4  ^ EB E2           JMP     SHORT 004221A8
004221C6    61              POPAD
004221C7  - E9 9642FEFF     JMP     00406462
004221CC    55              PUSH    EBP
```

악성코드 분석

- 새로운 파일 생성

```
00403936   E8 6D580000      CALL    004091A8          JMP to kernel32.CopyFileA
0040393B   85C0             TEST    EAX, EAX
0040393D   75 52            JNZ     SHORT 00403991
```

```
0012F358   0040393B  CALL to CopyFileA from Sality_h.00403936
0012F35C   009103E4  ExistingFileName = "C:₩Documents and Settings₩Administrator
0012F360   00910444  NewFileName = "C:₩Documents and Settings₩Administrator₩바탕
0012F364   00000000  FailIfExists = FALSE
0012F368   0012FE9C  Pointer to next SEH record
0012F36C   00403C4B  SE handler
```

이름	크기	종류
Sality.han.exe	103KB	응용 프로그램
Sality.han.zip	61KB	ALZip ZIP File
Sality.han.~01	103KB	~01 파일

파일 생성

악성코드 분석

- 코드 복호화 부분

```
00403693  A1 45B74000     MOV     EAX, DWORD PTR DS:[40B745]
00403698  8A1C06          MOV     BL, BYTE PTR DS:[ESI+EAX]
0040369B  8BC6            MOV     EAX, ESI
0040369D  48              DEC     EAX
0040369E  B9 0A000000     MOV     ECX, 0A
004036A3  99              CDQ
004036A4  F7F9            IDIV    ECX
004036A6  42              INC     EDX
004036A7  329A 2AB74000   XOR     BL, BYTE PTR DS:[EDX+40B72A]
004036AD  0FB6DB          MOVZX   EBX, BL
004036B0  8B45 F0         MOV     EAX, DWORD PTR SS:[EBP-10]
004036B3  F7EE            IMUL    ESI
004036B5  33D8            XOR     EBX, EAX
004036B7  A1 45B74000     MOV     EAX, DWORD PTR DS:[40B745]
004036BC  881C06          MOV     BYTE PTR DS:[ESI+EAX], BL
004036BF  46              INC     ESI
004036C0  8B45 C4         MOV     EAX, DWORD PTR SS:[EBP-3C]
004036C3  3BC6            CMP     EAX, ESI
004036C5  7D CC           JGE     SHORT 00403693
```

악성코드 분석

- 코드 복호화

악성코드 분석

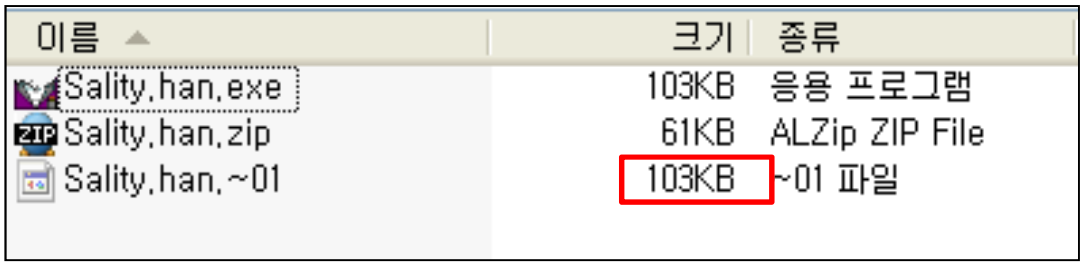
- 생성한 파일에 정상코드 복원

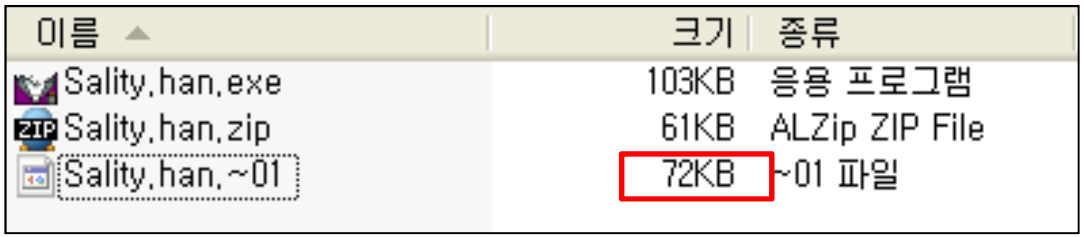
```
0040371A   E8 335C0000          CALL    00409352                JMP to kernel32._llseek

0012F2F0   00000044  hFile = 00000044 (window)
0012F2F4   00000000  Offset = 0
0012F2F8   00000000  Origin = FILE_BEGIN
```

```
00403733   E8 4A5C0000          CALL    00409382                JMP to kernel32._lwrite

0012F2F0   00000044  hFile = 00000044 (window)
0012F2F4   0014B8B0  Buffer = 0014B8B0
0012F2F8   00006EAC  BufSize = 6EAC (28332.)
```

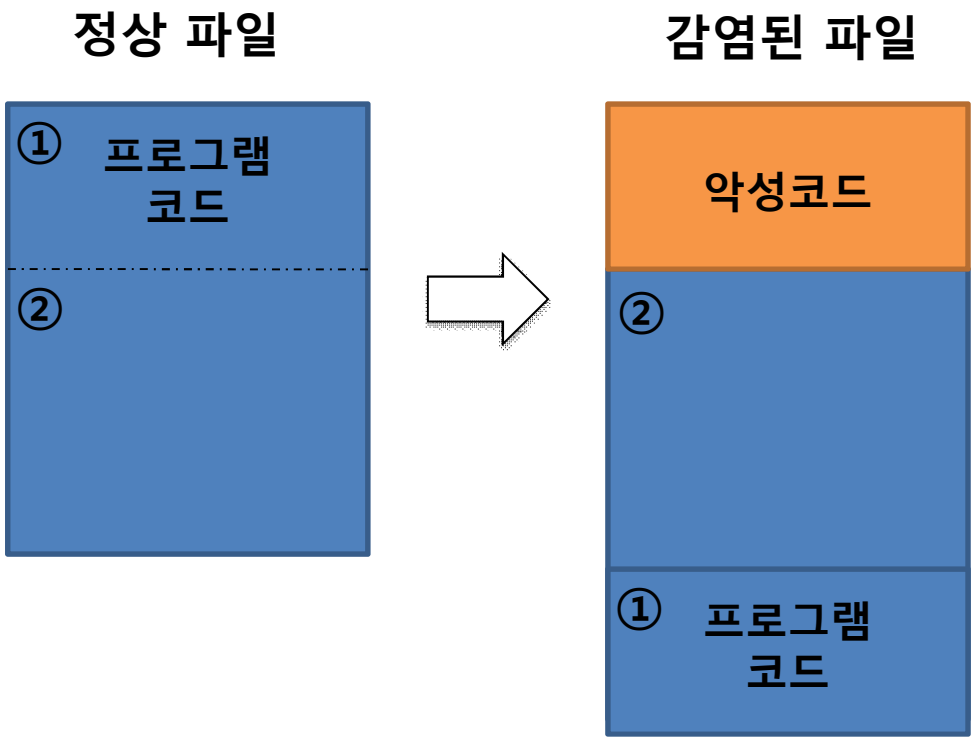
악성코드 분석

- 복원된 정상파일

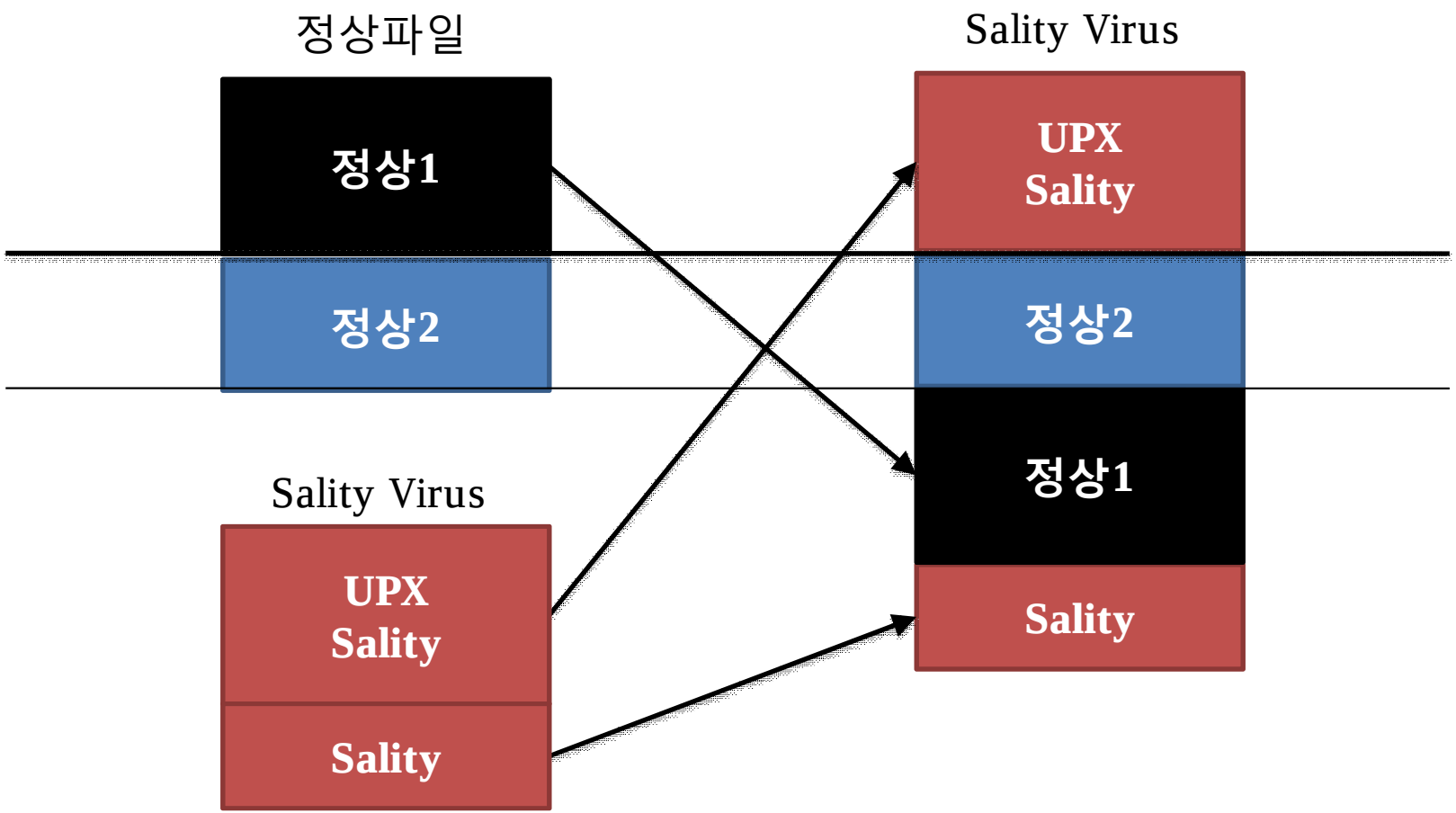




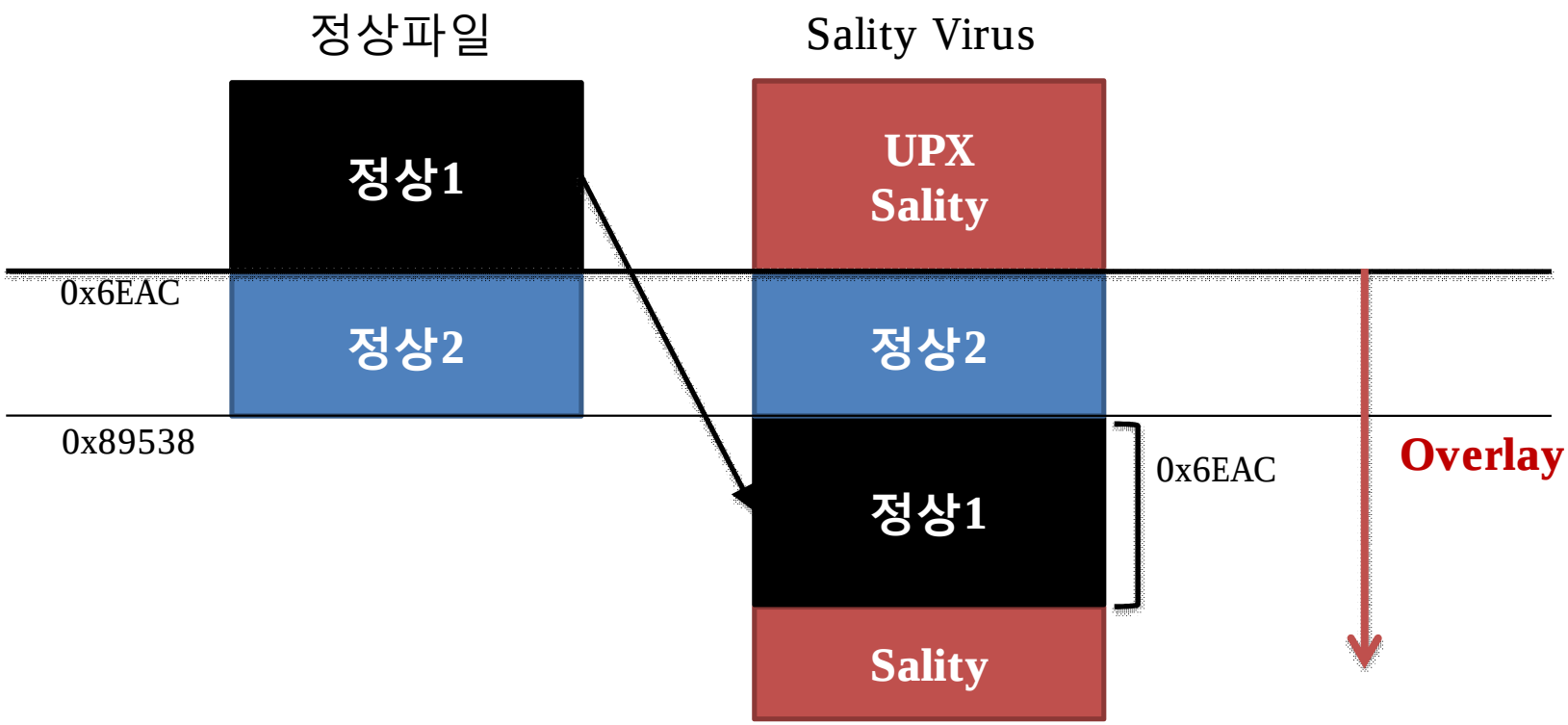
감염구조



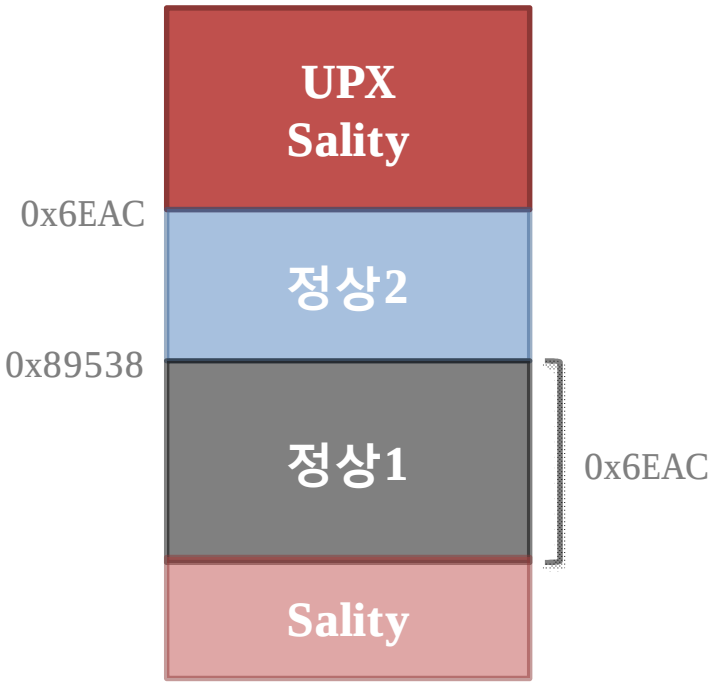
감염 구조


정상파일
정상1
정상2
Sality Virus
UPX
Sality
Sality
Sality Virus
UPX
Sality
정상2
정상1
Sality

감염 구조


정상파일
Sality Virus
정상1
UPX
Sality
0x6EAC
정상2
정상2
0x89538
정상1
0x6EAC
Overlay
Sality

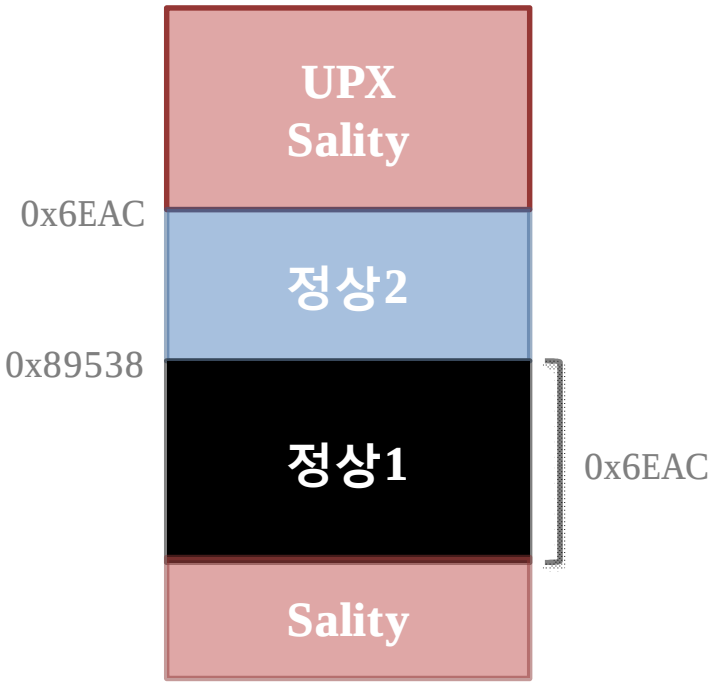
Sality 구조 분석



UPX Sality

- UPX 실행압축 해제
  - 첫번째 섹션(메모리)에 실행 압축 해제
  - 압축 해제 후, 해제된 코드(첫번째 섹션)로 JMP 해서 정상파일 생성 및 악성코드 실행
- 압축 해제 후 악성코드 수행
  - 정상파일 생성
  - 정상파일 실행
  - 정상파일 삭제
  - 악성코드 실행

Sality 구조 분석



정상1

- 정상파일 앞부분
- 암호화
- 정상1 정보
  - Offset = 정상파일
    - 원래의 정상파일 뒷부분에 정상1이 덧붙었기 때문
  - Size = 패치된 UPX Sality 크기 (0x6EAC)

치료로직 개발

정상코드 복구과정

1. 전체 복호화 – 복호화 로직 필요
   - 0 ~ 0x6EAB 까지 복호화(UPX_Sality Size)
2. MZ 복구 – 정적 복구
   - 앞부분 2byte(“MZ”)
3. 추가 복호화 – 복호화 로직 필요
   - 특정 위치(0xBB9 ~ 0xBCC) 복호화

전체 복호화

악성코드 분석

- 코드 복호화 부분

```
00403693  A1 45B74000     MOV     EAX, DWORD PTR DS:[40B745]
00403698  8A1C06          MOV     BL, BYTE PTR DS:[ESI+EAX]
0040369B  8BC6            MOV     EAX, ESI
0040369D  48              DEC     EAX
0040369E  B9 0A000000     MOV     ECX, 0A
004036A3  99              CDQ
004036A4  F7F9            IDIV    ECX
004036A6  42              INC     EDX
004036A7  329A 2AB74000   XOR     BL, BYTE PTR DS:[EDX+40B72A]
004036AD  0FB6DB          MOVZX   EBX, BL
004036B0  8B45 F0         MOV     EAX, DWORD PTR SS:[EBP-10]
004036B3  F7EE            IMUL    ESI
004036B5  33D8            XOR     EBX, EAX
004036B7  A1 45B74000     MOV     EAX, DWORD PTR DS:[40B745]
004036BC  881C06          MOV     BYTE PTR DS:[ESI+EAX], BL
004036BF  46              INC     ESI
004036C0  8B45 C4         MOV     EAX, DWORD PTR SS:[EBP-3C]
004036C3  3BC6            CMP     EAX, ESI
004036C5  7D CC           JGE     SHORT 00403693
```

복호 Key 정보

- **KeyTable**

  unsigned char KeyTable[0x0A] =
  {0xF7, 0xD5, 0x28, 0xD3, 0xCD, 0x33, 0xDB, 0xF6, 0x74, 0x21 };

- **ExtKey**

  = SUM(0xF7, 0xD5, 0x28, 0xD3, 0xCD, 0x33, 0xDB, 0xF6, 0x74, 0x21)
  + 0x0A(SUM 개수=KeyTable Size)

복호화 로직(C)

- CodeBuff : 암호화된 코드
- CodeSize : CodeBuff 의 Size(복호화 Size)
- KeyTable : 복호화 KeyTable
- ExtKey : KeyTable값들의 합
  + KeyTable Size
- KeyTableSize : 0x0A

```
for(Cnt = 0; Cnt < CodeSize; Cnt++)
{
    if( Cnt == 0)
    {
        EBX = 0;
    }
    else
    {
        KeyPoint = (nCnt -1) % KeyTableSize;
        EBX = KeyTable[KeyPoint];
    }

    EBX ^= (ExtKey * nCnt);
    CodeBuff[nCnt] ^= (BYTE)EBX;
}
```

복호화

- 암호화된 코드

```
0014B4A8 F9 CD 2B 8D 0C DE 79 5A 4A 9B 07 AA BE 1C D1 F4 鶴+??ZJ?ぞ 捺
0014B4B8 FB 7C 28 61 6D 74 6F D9 BB 92 A5 16 F2 4F 53 5E ?(amto某뭔■?S^
0014B4C8 35 3F 9D 48 8F 28 DC 15 B9 38 D3 15 A7 66 D1 C2 5?쒱????쑸豈
0014B4D8 A6 F3 9F 02 F9 4B 49 1C 3B E4 80 D9 15 EC 87 A1 ???I ;??? ¢
0014B4E8 1D 25 A7 B0 6A 13 22 7B 8C 3F 34 9C AA 71 60 E5 ■%㎞j■"{?4쑸q` ¢
0014B4F8 78 E3 6B 8D AD E1 2E 18 7F 06 57 19 70 BD 8F CB x?뜭?■■₩W■p쫵 ¢
0014B508 E7 2C 9A 54 7D 36 4A 67 AB 2B 9B 06 86 50 D0 0E ?쉏}6Jg??낉?
0014B518 A8 00 C9 1D F1 F5 A1 EF CD 08 E3 45 77 16 21 92 ??拯ᄂ??w■! ¢
0014B528 79 7E 41 27 C2 67 F9 6A 40 68 B0 0F 7E 60 B7 37 y~A'헸?@h?~`?
0014B538 EB 3E D0 E8 7E AB FB 20 36 2B 24 26 90 BC E4 FB ?矜~? 6+$&맻液
0014B548 84 B4 A4 8B B0 62 79 9C 26 EF 19 EC 6B DF 51 F2 꽩쩈캼y???? ¢
0014B558 2B 08 72 A7 C7 C8 2F DF 09 1B A6 9E D9 33 13 58 +■r₭V??■쨋?■X
```

- 복호화한 코드

```
0014B4A8 F9 0D 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 ?? ...■...ÿÿ..
0014B4B8 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ?......@.......
0014B4C8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0014B4D8 00 00 00 00 00 00 00 00 00 00 00 00 D0 00 00 00 ............?..
0014B4E8 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 ■■?.???L?Th
0014B4F8 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F is program canno
0014B508 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS
0014B518 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 mode....$.......
0014B528 0F BD 8E F5 4B DC E0 A6 4B DC E0 A6 4B DC E0 A6 ■쫵?馥쩘馥쩘馥 ¢
0014B538 C8 D4 BD A6 44 DC E0 A6 4B DC E1 A6 27 DC E0 A6 쮈쉐D馥쩘鰒?馥 ¢
0014B548 C5 D4 BF A6 5F DC E0 A6 C8 D4 BE A6 4A DC E0 A6 덧엡_馥 ┏禿쩸馥 (
0014B558 C8 D4 BA A6 4A DC E0 A6 52 69 63 68 4B DC E0 A6 쮈벧J馥쩸ichK馥 (
```

MZ 복구

MZ 복구

- MZ 복구 로직

```
004036C7   A1 45B74000          MOV     EAX, DWORD PTR DS:[40B745]
004036CC   C600 4D              MOV     BYTE PTR DS:[EAX], 4D
004036CF   A1 45B74000          MOV     EAX, DWORD PTR DS:[40B745]
004036D4   C640 01 5A           MOV     BYTE PTR DS:[EAX+1], 5A
```

- 복구된 모습

```
0014B4A8 F9 0D 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 ??  ...■...ÿÿ..
0014B4B8 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ?.......@.......
0014B4C8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0014B4D8 00 00 00 00 00 00 00 00 00 00 00 00 D0 00 00 00 ............?...
0014B4E8 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 ■■?.???L?Th
```

```
0014B4A8 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ?  ...■...ÿÿ..
0014B4B8 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ?.......@.......
0014B4C8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0014B4D8 00 00 00 00 00 00 00 00 00 00 00 00 D0 00 00 00 ............?...
0014B4E8 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 ■■?.???L?Th
0014B4F8 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F is program canno
```

추가 복호화

추가 복호화 로직(ASM)

```
004036DD   8B45 EC          MOV     EAX, DWORD PTR SS:[EBP-14]
004036E0   05 61020000      ADD     EAX, 261
004036E5   03C6             ADD     EAX, ESI
004036E7   8A1C38           MOV     BL, BYTE PTR DS:[EAX+EDI]
004036EA   8BC6             MOV     EAX, ESI
004036EC   48               DEC     EAX
004036ED   B9 0A000000      MOV     ECX, 0A
004036F2   99               CDQ
004036F3   F7F9             IDIV    ECX
004036F5   42               INC     EDX
004036F6   329A 2AB74000    XOR     BL, BYTE PTR DS:[EDX+40B72A]
004036FC   8B0D 45B74000    MOV     ECX, DWORD PTR DS:[40B745]
00403702   8BC6             MOV     EAX, ESI
00403704   05 B80B0000      ADD     EAX, 0BB8
00403709   881C08           MOV     BYTE PTR DS:[EAX+ECX], BL
0040370C   46               INC     ESI
0040370D   83FE 14          CMP     ESI, 14
00403710   7E CB            JLE     SHORT 004036DD
```

추가 복호화 로직 정보

- 복원할 위치 정보
  - Offset : 0xBB9
  - Size : 0x14
- 정상코드 정보
  - Offset : 0x264
  - Size : 0x14
- KeyTable
  - Offset : 0x2AF
  - Size : 0x0A

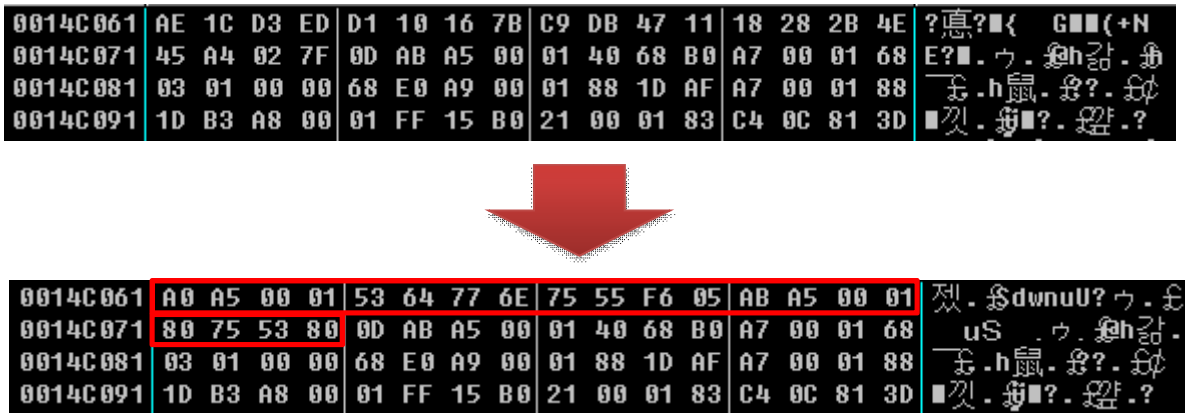
추가 복호화 로직(C)

- CodeBuff : 추가 암호화된 코드
- CodeSize : CodeBuff 의 Size(복호화 Size)
  0x14
- KeyTableSize : 0x0A

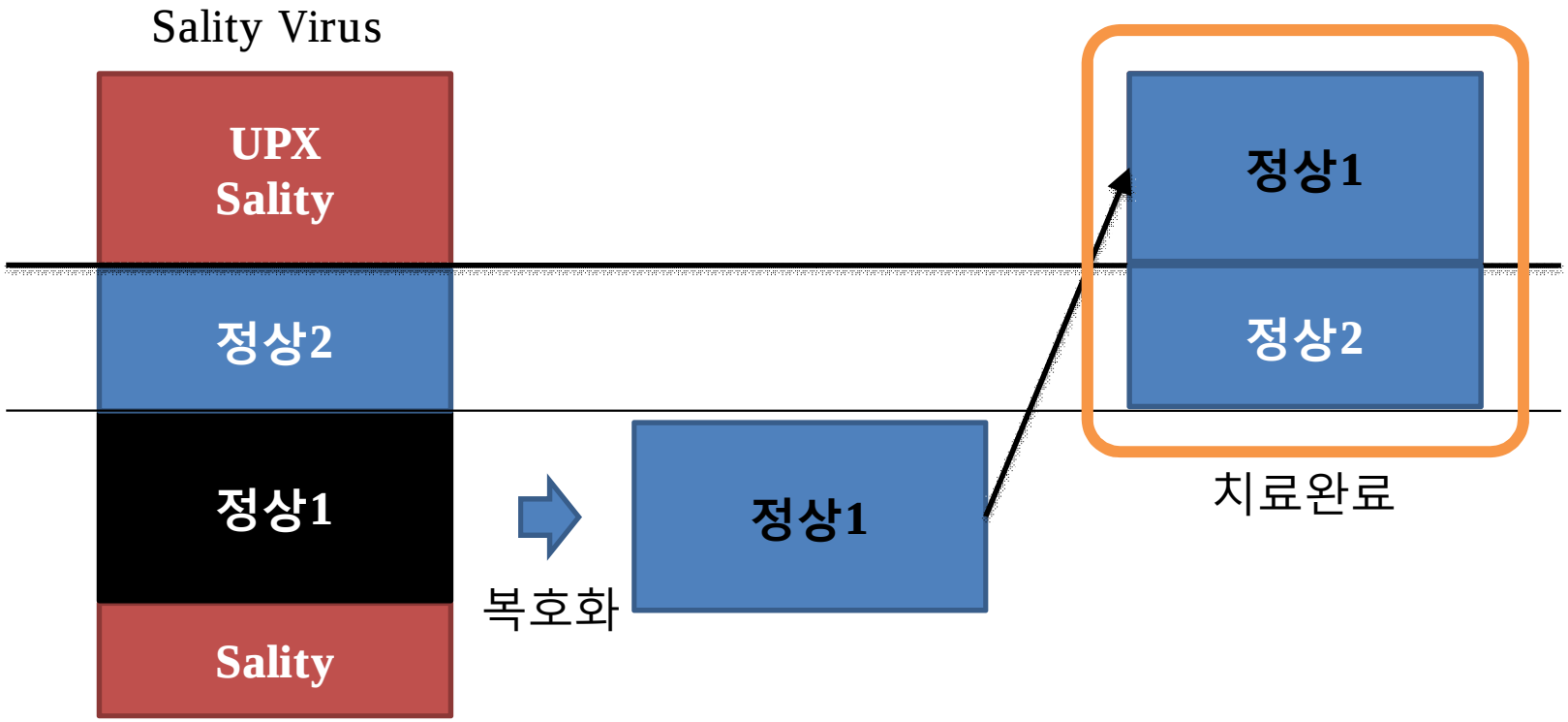
```
KeyPoint = 0;

for(Cnt = 0; Cnt < CodeSize ; Cnt++)
{
    KeyPoint = Cnt % KeyTableSize ;
     CodeBuff [Cnt] ^= KeyTable[KeyPoint];
}
```

추가 복호화한 모습

- 복구된 모습

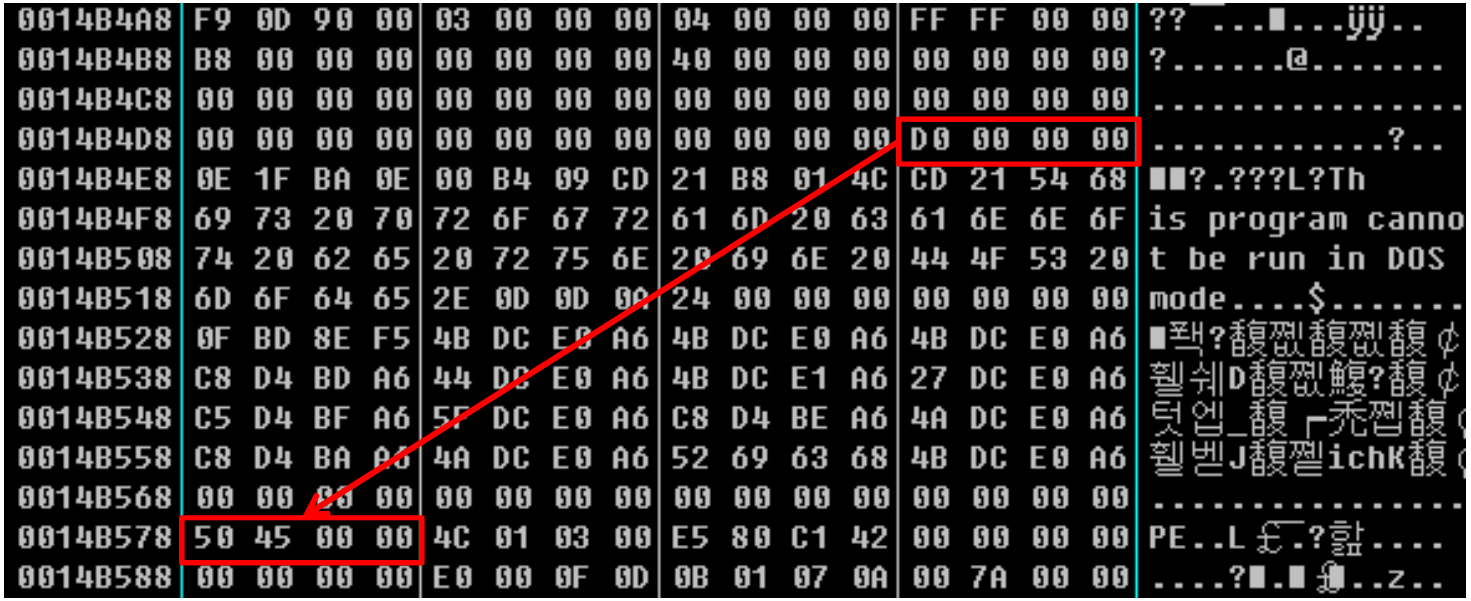


정상파일 복구



진단 조건 연구

- 복호화한 코드



진단&치료 테스트

향후 연구 방향

- 코드 다형성
- 변종
- 패킹(실행압축)

- 로직에서의 정확한 코드 진단

- 파일 바이러스 감염 및 확산 차단

참고문헌

- Peter Szor, The Art of Computer Virus Research and Defense, 2005.02.25

Questions?

한정화 (daly25@gmail.com)