“软件安全”实验报告

**.**

**班 级：　　 　　　.**

**姓 名： 　　 　　.**

**学 号：　 　 　　 　.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **项目** | **撰写规范** | **实验过程** | **问题分析与小结** | **总分** | **教师签字** |
| 分值 | 20 | 50 | 30 | 100 |  |
| 评分 |  |  |  |  |  |

# 实验4 恶意代码查杀实验

## 实验目的

* 理解恶意代码查杀流程
* 掌握查杀基本算法
* 基本掌握特征码、校验和、简单启发查杀技术

## 实验要求

* 在断网、安全的虚拟机环境中进行相关实验；
* 选用一种高级语言，例如C、C++语言为主要编程语言；
* 设计并实现恶意代码查杀的命令行程序MiniAntiVirus;
* 通过命令行方式，能对某一文件、某一文件夹下文件进行查毒；
* 能对自己前面编写的病毒模拟程序进行识别，并报出该病毒名称、染毒文件名、文件大小、文件位置、（被感染文件样本的校验和：可选项）；
* 对特征串，采用1）给定偏移比较；2）BM算法全文检索比较；可选算法3）校验和比较；
* 独自开发、独立运行（提交源代码、可执行程序、其他运行时需要的文件，例如已灭活的被感染样本、特征串文件）

## 实验环境

* 操作系统：Microsoft windows XP
* 开发工具：C

## 实验内容

* 学习杀毒软件的基本流程；
* 学习基本杀毒步骤及相关操作和算法；
* 利用C语言编写简单查杀程序；
* 验证结果

## 注意事项

* (1)注意如果有病毒样本程序，必须灭活，在安全环境中进行实验，切断网络连接；
* (2) 注意及时保存实验程序；遇到问题查阅资料解决时，记录资料的详细来源，列入参考文献中。

### 实验过程记录

首先介绍一下我的恶意代码程序，它是我第三次实验中编写的程序，主要功能有：自我复制、通过修改注册表的键盘映射改为空来实现屏蔽键盘字母输入以及令电脑在一分钟内关机等功能。由于制作查杀软件的开发者得到exe文件源码的可能性较低，以下过程是模拟未知源码的情况下的查杀程序编写过程，本次实验使用特征码查杀方法。

**实验的第一步是获得病毒的特征码。**使用ollydbg对病毒程序cpf.exe进行反汇编，首先在程序开头，出现了3个程序内的函数，如图1所示：



图1

三个函数分别为：main，copyfile，RegKeyBoardForbidden函数，通过函数名便能知道这些函数的大概功能，这里给了我一个启示就是，写恶意代码的时候，尽可能不要把函数名写的太容易理解，这会使得程序很容易在反汇编阶段就被理解清楚。

接下来分别研究这三个函数跳转到的内容，首先是main函数，如图2和图3所示：



图2

图2已经写出，在004012A8处程序调用了00401005的函数，通过图1我们知道这个函数正是RegKeyBoardForbidden函数。接着进行了sleep(3000)的操作,然后在004012D9进行了调用0040100A处函数的操作，该函数就是copyfile函数。



图3

图3显示在调用完copyfile之后，进行了sleep(1000)，然后执行system shutdown程序，准备关机，弹出关机提示，再向下看代码，出现retn指令，函数结束。容易注意到的一点是，在004012CA到004012F2，存在一个循环，在循环里不停调用着0040100A,这个循环使用累加的方式，达到0xf次终止调用copyfile，我认为对同一个文件对同一个路径进行若干次复制这并不是一个寻常的行为，

接下来观察copyfile函数，由图4、图5可知它的确为一个文件复制函数，分为对文件随机命名和创建文件然后复制粘贴文件两部分。

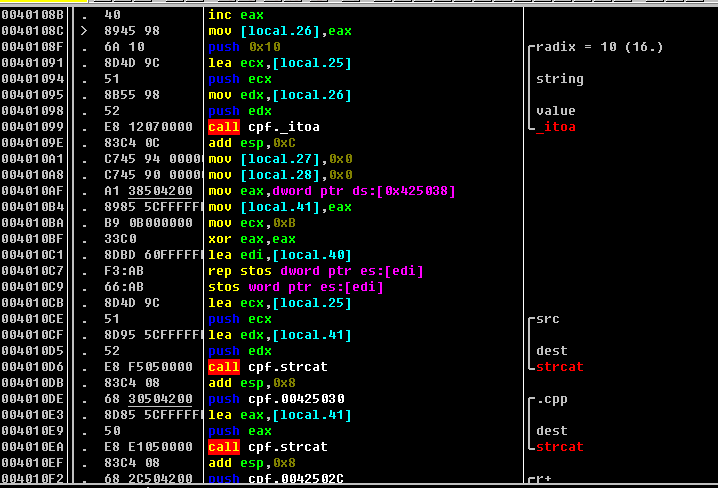


图4

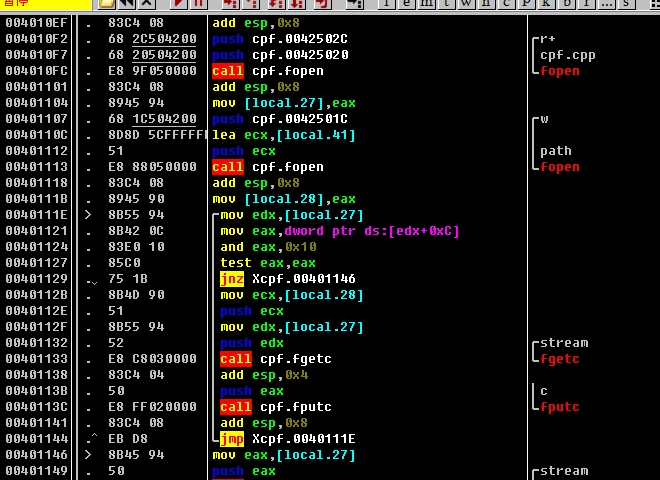


图5

其中，被复制文件由004010F7可知为代码本身:cpf.cpp，进行复制了0xf次。在这里，可以把main函数里面004012CA到004011F2的二进制代码作为特征码，因为把某个文件在一个路径复制0xf次的行为的确反常。但我们仍需研究一下RegKeyBoardForbidden函数。

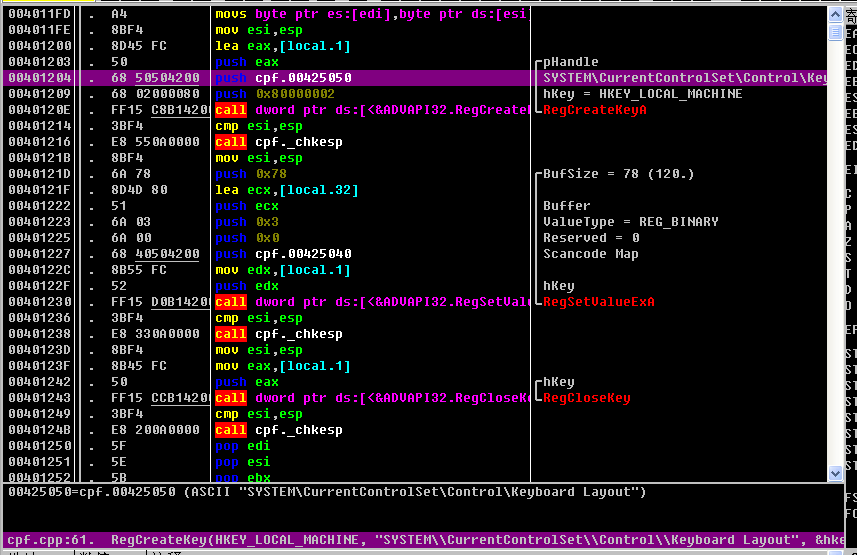


图6

由图6所示，程序在注册表的Keyboard Layout区域进行了修改，也就是创建了注册表，修改了映射值然后关闭注册表，单步调试恶意代码运行到此区域然后观察注册表中的值，发现添加了一个键盘映射，把a~z的输入都变成了空。尝试从00401218处的二进制代码8BF4开始，一直到00401238的330A开始，这32位数当作特征码。

**第二步便是获得特征码的偏移地址**

使用lordpe打开程序，如图7所示

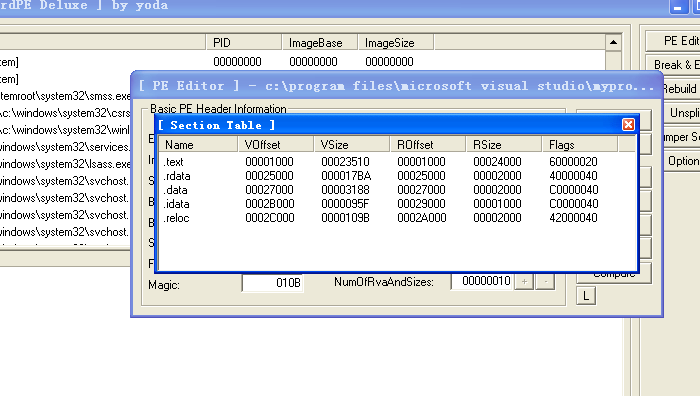


图7

再使用ultraedit观察8BF4所在的偏移地址，

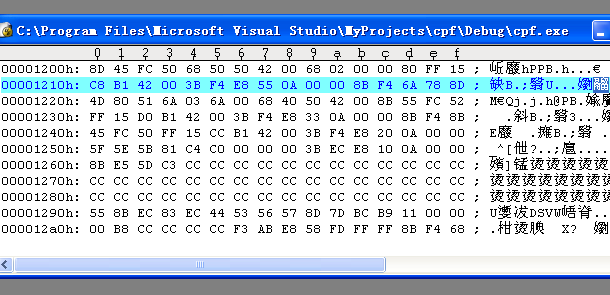


图8

可以知道所选取的代码在text区域，偏移地址是121b。至此，知道了特征码为： 8B F4 6A 78 8D 4D 80 51 6A 03 6A 00 68 40 50 42

00 8B 55 FC 52 FF 15 D0 B1 42 00 3B F4 E8 33 0A

偏移地址为：121b

**第三步便是编写有关的恶意代码查杀的程序**，原理是对exe文件特定偏移地址的特征码进行比对，如果特征码一致，便是恶意程序，否则不是。获得代码如下：

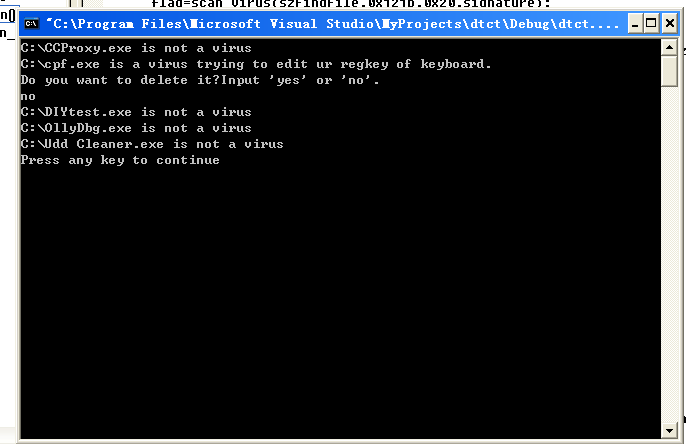
1. #include <stdio.h>
2. #include <string.h>
3. #include <windows.h>
4. **bool** scan\_virus(**const** **char** \*FileName,**long** offset, **int** length, **void** \*signature);
5. **void** main()
6. {   WIN32\_FIND\_DATA stFindFile;
7. **HANDLE** hFindFile;
8. **char** \*szFilter = "\*.exe";   // 保存搜索的筛选条件（所有exe文件）
9. **char** szFindFile[MAX\_PATH];  // 保存欲检测的程序的路径
10. **char** szSearch[MAX\_PATH];    // 保存完整筛选路径
11. **int** ret = 0;                // 搜索的返回值
12. lstrcpy(szFindFile,"C:\\");
13. lstrcpy(szSearch,"C:\\");
14. lstrcat(szSearch, szFilter);
15. hFindFile = FindFirstFile(szSearch, &stFindFile);
16. **bool** flag=**false**;
17. **char** input[10];
18. **char** yes[10]='yes';
19. **char** no[10]='no';
20. unsigned **char** signature[]=
21. {0x8B,0xF4,0x6A,0x78,0x8D,0x4D,0x80,0x51,0x6A,0x03,0x6A,0x00,0x68,0x40,0x50,0x42,
22. 0x00,0x8B,0x55,0xFC,0x52,0xFF,0x15,0xD0,0xB1,0x42,0x00,0x3B,0xF4,0xE8,0x33,0x0A,0x00,0x00};  //特征根据恶意代码反汇编选取
23. **if**(hFindFile!=INVALID\_HANDLE\_VALUE)
24. {
25. **do**
26. {
27. // 组成完整的待检测程序的路径
28. lstrcat(szFindFile, stFindFile.cFileName);
29. //调用scan\_virus判断是否为病毒
30. flag=scan\_virus(szFindFile,0x121b,0x20,signature);
31. **if**(flag)
32. {
33. printf("%s is a virus trying to edit ur regkey of keyboard.\n",szFindFile);  //需要输出病毒详细信息
34. printf("Do you want to delete it?Input 'yes' or 'no'.\n");//交互功能，如果想要删除，就会删除。
35. scanf("%s",input);
36. **if**((strcmp(input,yes)==0)||(strcmp(input,no)==0)))
37. {
38. **if**((strcmp(input,yes))==0)
39. {
40. remove(szFindFile);
41. }
42. }
43. // 删除程序名称，只保留“C:\”
44. szFindFile[3] = '\0';
45. ret = FindNextFile(hFindFile, &stFindFile);
46. }
47. **if**(!flag)
48. {
49. printf("%s is not a virus\n",szFindFile);
50. szFindFile[3] = '\0';
51. ret = FindNextFile(hFindFile, &stFindFile);
52. }
53. }**while**(ret!=0);//do
54. }//if(hFindhandle)
55. FindClose(hFindFile);
56. }//main
57. **bool** scan\_virus(**const** **char** \*FileName,**long** offset, **int** length, **void** \*signature)
58. { **FILE** \*fp = NULL;
59. **int** result=0;
60. **char** rvir[255]={ 0 };
61. **bool** Flag=**false**;
62. fp=fopen(FileName,"rb"); //以2进制打开指定文件
63. **if**(fp==NULL)
64. { **return** Flag; }
65. fseek(fp,offset,SEEK\_SET);   //把文件指针指向偏移地址
66. fread(rvir,length,1,fp);         //读取length长度的代码
67. result=memcmp(signature,rvir,length);  //与我们提取的代码比较，返回值放到result中
68. **if**(result==0)
69. { Flag=**true**;}
70. fclose(fp);   **return** Flag;
71. }

这里检查的是C://路径下的所有exe文件，我在其中放置了5个文件，其中4个为正常文件1一个为恶意程序。



图9

运行dtct程序，结果如下：与预期完全一致



如果选择yes，显然cpf.exe会被删除，在此不再展示。

## 实验遇到的难点与问题分析

本次实验的难点之一在于特征码的提取，在不知道恶意程序源码的情况下，需要使用反汇编工具静态动态结合研究程序的功能，对于自己编写的简易病毒程序还好观察，但倘若是复杂一些的软件，会出现一些困难，需要我们对汇编语言有一定的熟悉程度，了解一些常见的系统功能，以及熟悉od调试。即使是在这次尝试中，也在提取特征码的环节里耗费了很多的时间，不过现在回顾起来，有了这次实验，踩了一些坑，便可以按照经验查杀其他复杂的软件。

另外便是程序的编写部分，程序主体按照实验指导书里的样例进行，添加了一些额外的交互功能等，比如询问删除病毒文件以及搜索给定路径的特定类型文件。这里需要对一些api比较熟悉，虽然是比较简单的代码但也体现了自己在编程方面的不熟练，希望以后能勤加练习，努力改善。

## 实验小结

完成本次实验后，我会觉得它并没有一开始想像的那么难，不过它也让我抽出时间来重新回顾了课堂上讲过的知识细节，也让我重新熟悉了lordpe,ultraedit,vc6.0,ollydbg等软件的具体操作。这次的实验我并没有研究复杂的病毒程序，而只是研究了自己写的简单的恶意代码，但是它让我对这方面的技术产生了兴趣，做完这次实验，我会去想在以后继续研究一些曾经遇到过或者在身边传播过的程序，这可能会踩坑或者花费一些时间，但的确让我很感兴趣。

这次实验给我的启发有两方面，首先是作为病毒编写而言的：我理解了课堂上所讲的那个“复杂编程比赛”的例子，作为病毒程序，不能写的太简单，函数不能用显而易见的名字命名，否则对方反汇编轻松便知道了程序的功能。

同时也可以通过查杀过程进一步理解一些免杀方法，比如修改特征码、修改字符串大小写、等价替换、通用跳转和指令顺序调换等。

其次是作为查杀软件方面的，我觉得自己编写的特征码法并不实用，它只能对特定的某个行为在特定的偏移地址进行查杀，但可能特定的行为也有不同的特征码，特征码的偏移地址也未必一样，就很容易漏报。如果是真正的查杀软件，特征码的数据库十分复杂，也降低了效率。我觉得这也是通用检测技术可以弥补的地方，两者互相结合，病毒查杀效率会更高。

完成这次实验，我也认识到自己在一些方面的欠缺，比如C编程还不熟练，病毒相关算法知识也没有应用过等，希望自己在接下来的时间里能在实践中练习它们。

## 参考文献

[通过特征码查杀病毒.(小段c程序)](https://blog.csdn.net/huabihan/article/details/480889)

[病毒木马查杀实战第018篇：病毒特征码查杀之基本原理](https://blog.csdn.net/ioio_jy/article/details/51198544)

[病毒木马查杀实战第019篇：病毒特征码查杀之编程实现](https://blog.csdn.net/ioio_jy/article/details/51198566)

[Window文件目录遍历 和 WIN32\_FIND\_DATA 结构](https://blog.csdn.net/qq2399431200/article/details/11878611)