台灣二００五年國際科學展覽會

科　　別：電腦科學

作品名稱：攔截未知病毒之基礎技術-參數監控

學　　校：弘道國中

作　　者：黃煜翔

作者介紹

姓名：黃煜翔

興趣：閱讀、程式開發、哲學

專長：電腦、鋼琴、桌球

簡介：

自國小三年級對電腦產生濃厚的興趣，特別是資訊安全議題。而在實際電腦操作中遇到許多問題，學習過程中像無頭蒼蠅亂飛亂撞，卻也得到相當寶貴的經驗，也不斷從雜誌、書籍涉獵相關的電腦知識。在學習過程中，最喜愛的便是防毒技術與網路系統管理。所以在國一、國二時，分別取得了微軟的 MCP 和 MCSA 證照。在電腦科學日新月異的發展下，希望能隨時充實自我，更上一層樓！

中文摘要

在網際網路蓬勃發展的時代，電腦病毒也更加的猖獗。雖說防毒軟體百家爭鳴，但對於未知病毒的防治技術仍然束手無策。本研究的目的在追溯出電腦惡意程式運行模式，利用 IDA 靜態反編譯軟體與病毒原始碼尋找出病毒共同的特徵，阻斷其滋生途徑。

本研究在實驗中，研究者利用 Borland C++ Builder 6 作爲應用程式開發環境，配合 WinAPI 函數與 TRegistry 類別設計出參數監控系統。研究顯示不論是修改於 Window 9x 架構下的 win.ini, system.ini，或是 Run 與 Run Service 之下的資料項，本系統都能有效運作。此系統能成功的攔截到惡意程式對 Registry 鍵值的讀寫，亦即當使用者否決其修改鍵值時，惡意程式無法在重新開機後運行；易言之，惡意程式失去執行效力。

實驗結果發現，參數監控系統可攔截惡意程式之修改命令，防止未知病毒與惡意程式開機自動運行，以此達到未知病毒的攔截基礎技術。本研究之結果可補強市售防毒系統，對於未知病毒的攔截，能達到完整的防護效果。

ABSTRACT

Computer viruses become more rampant in the rapidly-developing era of Internet. With the various kinds of Anti-Virus software, people still can do nothing about detecting the unknown viruses. This research aims to trace back the computer virus function mode. By using IDA Pro Disassembler and Virus Source, we can find out the common characteristics of viruses. Thus, we can stop viruses from thriving.

The experiment uses Borland C++ Builder 6 as the Registry Detection System by coordinating Win API and TRegistry. It works no matter by modifying win.ini and system.ini under the structure of Window 9x or by modifying Run Value and RunService Value. Moreover, it can intercept the viruses from reading and writing of the ValueKey. As the users deny the modified ValueKey, the viruses cannot run after resetting. In other words, the viruses lose their effects.

The results present that Registry Detection System can intercept the modifying commands and prevent the automatic running from unknown viruses. By doing so, we can acquire the basic technology of intercepting the unknown viruses.

The application of this study can improve the functions of Anti-Virus. In this way, through protection from unknown viruses can be obtained.

壹、前言

一、研究動機

自國小三年級時進入電腦實驗班，那時對電腦特別有興趣，剛好也是網際網路蓬勃發展的時代，相對來說，網際網路也是 Windows 平台病毒擴散的溫床，故每逢遭受病毒之苦，腦海中就時常浮現許多疑惑。記得接觸電腦一段時間後，在網際網路遨遊，沒想到才看幾個網頁，網頁上就寫著電腦已經中毒，當時認爲防毒程式之病毒碼已更新，足以抵抗病毒，沒想到重新開機後，資料已被刪的一乾二淨。於是對電腦安全產生極大的興趣，至今已通過美國微軟舉行之認證考試，順利通過 MCP(Microsoft Certified Professional)與 MCSA(Microsoft Certified System Administrator)兩張國際證照。

根據觀察，自磁片傳播，到網際網路發展至今，已從一般的 PE 病毒演化至孺蟲，病毒利用各種方式與型態威脅著系統安全，但目前對於病毒的處理方式仍依舊不變。總是爲病毒開始傳染擴散後，防毒公司再以人工方式分析，並製作病毒碼，但在這一段分析時間，病毒已不知感染了多少 Windows 平台，又因其爲未知病毒之故，一般的防毒系統內含比對引擎對其根本束手無策，造成病毒不斷傳染，以致災情擴大，而等到病毒碼公佈後，卻爲時已晚。故本研究乃是對於病毒原始碼進行分析，比對其特徵，尋出其共同的運作模式，並以 Borland C++ Builder 6 作爲應用程式開發環境，規劃一系統以病毒之共同特徵作爲目標，利用反向運作監控並阻斷其對系統之影響，完成已知病毒與未知病毒的基礎攔截技術，對於電腦使用者也是一大保障。

二、實驗目的：

1、分析目前電腦病毒之運作模式與其特徵。

2、利用 Borland C++ Builder 6 開發出參數監控系統用以對未知病毒之攔截。

貳、研究過程與方法：

一、研究設備：

1、編號一電腦：(病毒實驗，網路封閉)

(1)硬體設備：Intel Pentium III 733Mhz 處理器、512MB 記憶體、10GB 硬碟。

(2)軟體設備：WindowsME 中文版作業系統、數種電腦病毒原執行檔與原始碼、IDA Pro 4.5 靜態反編譯軟體、Borland C++ Builder 6 英文版、RegistryMonitor 英文版。

2、編號二電腦：(程式開發環境)

(1)硬體設備：Intel Pentium M 1.5Ghz 處理器、256MB 記憶體、30GB 硬碟。

(2)軟體設備：WindowsXP 中文作業系統、Borland C++ Builder 6 英文版。

二、實驗一：分析目前電腦病毒之運作模式與其特徵

1、經由電腦日常使用觀察，可知病毒欲植入於電腦，且病毒若要在開機時重新運行，則必定完全依賴於 Windows 系統裡內建之啓動功能，也就是說，當病毒啓動時，將會修改 win.ini,system.ini,Registry,任一項或多項。

故：

條件：電腦病毒需爲單程式、屬 WIN32 常駐型，且不與系統內正常程式合併。

∵電腦病毒欲開機重新執行則 $\xrightarrow{\text{必定}}$ 修改系統裡內建之啓動功能

∴電腦病毒欲開機重新執行則 $\xrightarrow{\text{必定}}$ 修改 win.ini,system.ini,Registry,任一項或多項。

∵電腦病毒若不修改內建啓動功能則 $\xrightarrow{\text{必定}}$ 無法在開機時自動執行

∴電腦病毒若不修改 win.ini,system.ini,Registry,任一項或多項參數則 $\xrightarrow{\text{必定}}$ 無法在開機時自動執行。

∴若封鎖 win.ini,system.ini,Registry 參數則電腦病毒 $\xrightarrow{\text{必定}}$ 無法在開機時自動執行。

.參數整理如表一：

表一、啓動參數整理

Autoexec.bat	實體 DOS 底下驅動程序和應用程序之配置，在 Windows9x 開機使用之，但在 WindowsNT 架構下的系統已放棄此配置方式。
Win.ini	亦是 Windows9x 底下的驅動程序和應用程序之配置，其中 run=爲病毒時常爲依靠的參數項。而在 WindowsNT 架構下之系統以不復見 run= 參數命令。但其仍是系統的一大命脈。
System.ini	與 win.ini 一般，紀錄電腦執行時的重要資訊。其中有一項 shell=Explorer.exe，可在後面加上”,”開機時自動執行其他程式。但仍與 win.ini 一樣，已不復見 shell=之參數項。
Registry/Run	Registry 乃是 Windows 系統運作的一大命脈，若是任意刪除，則系統將無法運作。而 Run 機碼是 Windows 內建的啓動項目，多數程式、病毒最常依靠 Run 在開機時自動執行。
Registry/RunService	同 Run，是病毒常寄生的機碼。

但是這只能算是觀察上的推論。爲了求其真實性，追溯病毒執行模式，求證是否乃利用參數修改來自動開機執行實驗以下步驟：

(1) 分析多種蠕蟲之自動執行部分程式碼。
(2) 用 IDA Pro 靜態反編邊譯程式對病毒執行檔進行分析。
(3) 分析病毒 Registry 恢復方法

三、實驗二：參數監控系統之攔截

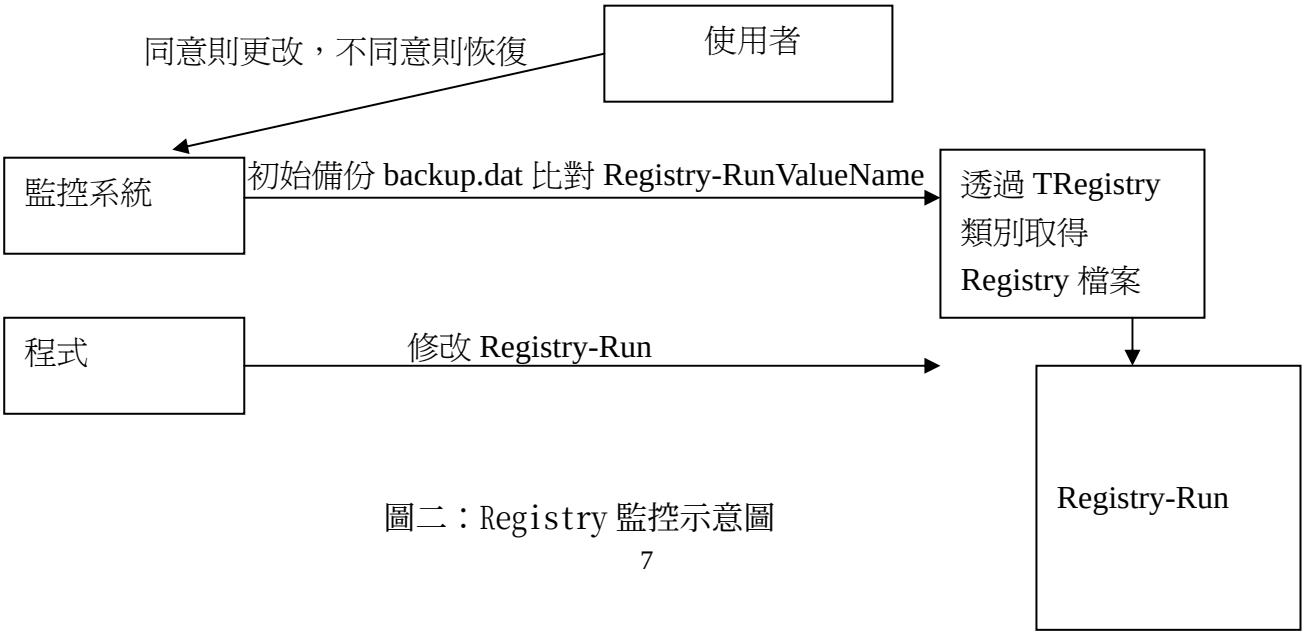
理論：經實驗一結果可知，病毒欲執行時，將對於自動執行之系統參數修改，故阻擋病毒開機自動執行之方式，乃是對參數進行監控。實驗便設計一偵測系統對於 Registry,win.ini 兩參數進行監視。若 Registry 或 win.ini 進行更動則跳出警訊視窗提醒使用者，並可獲得修改之參數資訊，能下達更改與否之指令。本程式開發環境使用 Borland C++ Builder 6(如圖一)。

1、Ini 監控：

圖一：ini 監控示意圖

本實驗利用 win.ini 作爲實驗，備份原始 win.ini，利用 C++Builder6 之 Timer 元件定時得取新的 win.ini 檔案，並且利用交叉比對法，進行文字上的比對，假若有任意一個字被修改，則立刻將整行截取，對使用者進行詢問是否恢復，若是則恢復之。要注意的是 run=爲自動啓動之參數項目，所以要修改啓動項目時，乃是更改 run=之參數項，而非新增，故不可用 TStringList 將數量匯出比較，TStringList 利用只能偵測新增參數項。至於 system.ini 與 Autoexec.bat 亦皆爲文字檔，是以也可用此方式監控。

2、Registry 監控系統



圖二：Registry 監控示意圖

Registry 監控系統類似於 Win.ini 監控系統，只是對象用於 Registry。因 WinApi 之 Registry 之控制函數較爲複雜麻煩，故本研究擬設利用 TRegistry 類別，透過其對 Registry 之讀寫。經由實驗一可知，多數病毒都是用類似於 RegCreateKey 函數，新增一 Value 值，使得病毒在電腦開機時自行啓動。是以此研究利用 TstringList 對 Registry-Run 進行數量控制，若是數量新增後，立刻跳出視窗提出警訊，顯示新增的 ValueName，並提供使用者移除之功能。雖然在實驗一之觀察顯示，病毒大多只有新增一項之 Value，但此系統爲了防止出錯，利用 ListBox 將新增修改之項目排序處理。

並在完成兩系統後，執行以下步驟：

(1)以手動修改，檢視監控系統是否成功攔截。

(2)以程式修改之，檢視監控系統是否成功攔截。

參、研究結果與討論：

一、實驗一分析目前電腦病毒之運作模式與其特徵結果：

1、病毒碼與 IDA Pro 對於自動執行功能分析

經由參數表可知，Windows 系統內建的自動啓動參數有數個，故研究將原始碼對於特殊字樣尋找。

特殊字樣：Reg,Registry,win.ini,system.ini, CurrentVersion

(1) Bagle 病毒原始碼

```
szRegAutoPath    "SOFTWARE\Microsoft\Windows\CurrentVersion\Run"//路徑位置

invoke    RegDeleteKey, HKEY_CURRENT_USER, offset szRegBasePath
invoke    RegCreateKey, HKEY_CURRENT_USER, offset szRegAutoPath, addr hkHandle //Win Api 函數控制
invoke    RegDeleteValue, hkHandle, offset szBglAutoKey
invoke    RegCloseKey, hkHandle
```

程式碼 1-1

較特別的是 Bagle 是註冊機碼至 HKEY_CURRENT_USER，但仍有 Run 此機碼，且可自動執行。一般來說 HKEY_LOCAL_MACHINE 是較爲常見，而 HKEY_CURRENT_USER 容易忽略掉。

(2)LoveLetter 原始碼(I Love You 病毒)

```
regcreate    //Win API  控制
"HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run\MSKernel32",dir
system&"\MSKernel32.vbs"    //路徑
regcreate
"HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices\Win32D
LL",dirwin&"\Win32DLL.vbs"
```

程式碼 1-2

LoveLetter 也就是所謂的 I Love you 病毒，在此可看出它是最常見的蠕蟲寫法。新增機碼至 HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run。讓開機的時候自動執行，從原始碼即可知爲 VBS 病毒。

(3)MyDoom 病毒原始碼

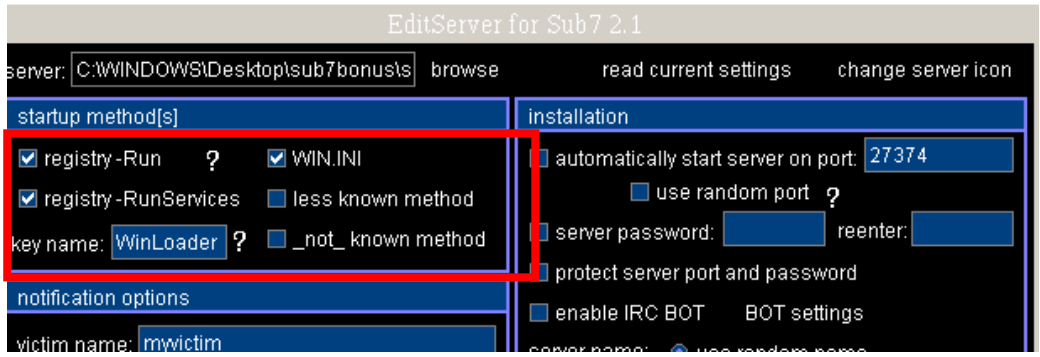
```
HKEY k;
    char regpath[128];
    char valname[32];
    /* "Software\\Microsoft\\Windows\\CurrentVersion\\Run" */
    rot13(regpath, "Fbsgjner\\Zvpebfbsg\\Jvaqbjf\\PheeragIrefvba\\Eha");
    rot13(valname, "GnfxZba");  /* "TaskMon" */
    if (RegOpenKeyEx(HKEY_LOCAL_MACHINE, regpath, 0, KEY_WRITE, &k) != 0)
        if (RegOpenKeyEx(HKEY_CURRENT_USER, regpath, 0, KEY_WRITE, &k) != 0)
            return;
    RegSetValueEx(k, valname, 0, REG_SZ, sync->sync_instpath, lstrlen(sync->sync_instpath)+1);
    RegCloseKey(k);
```

程式碼 1-3

MyDoom 病毒亦是利用 WinApi 函數自動將字串值註冊於 Registry，在開機之時，病毒將自動運行。

(4)SubSeven 木馬使用啓動

在 SubSeven 木馬裡，有一項完整的 Server 編輯器(如圖三)。SubSeven 木馬 Server 編輯程式可選取一個或數個自動開機的選項。例如：Registry-Run ,Win.ini,Registry-RunServices(Windows 系統內建的自動開機選項)。



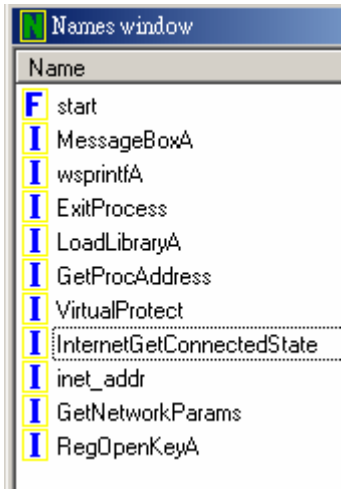
圖三：SubSeven EditServer 程式運作

(5)Worm NetSky.C 原執行檔 利用 IDA Pro 分析之結果

```
.idata:0041A2C0 ; Imports from ADVAPI32.dll
.idata:0041A2C0 ;
.idata:0041A2C0 ; LONG __stdcall RegOpenKeyA(
                  HKEY hKey,LPCSTR lpSubKey
                  ,PHKEY phkResult)

.idata:0041A2C0 ; extrn RegOpenKeyA:dword
.idata:0041A2C4
```

程式碼 1-4



圖四：IDA Pro 分析 Names

可看出 Worm NetSky.C 利用 WinApi 函數 RegOpenKey 讀取 Registry 檔案(如圖四)。代表著 NetSky 與 Registry 息息相關。雖然此處無法看出他對 Registry 進行讀寫，但經由執行病毒原執行檔後發現，會建立一字串值：

```
“Software\\Microsoft\\Windows\\CurrentVersion\\Run\ICQ Net”
Path="Windows\winlogon.exe -stealth"
```

此亦是利用 Windows 系統內建的啓動功能-Registry-Run，在開機時自動運行。

(6)Worm MSBlast(疾風病毒)

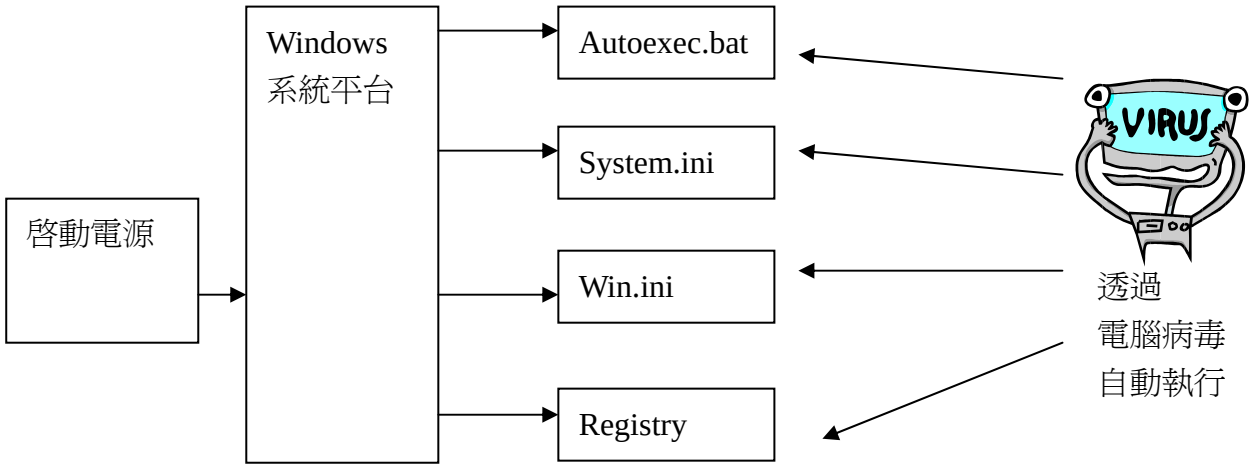
```
RegCreateKeyExA
(0x80000002, "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\\windows",
NULL, NULL, \ NULL, 0xF003F, NULL, &keystatus, NULL);
  RegSetValueExA(keystatus, "windows auto update", NULL, (ULONG)
1, "msblast.exe", (ULONG) 0x32);
  RegCloseKey(keystatus);
```

程式碼 1-5

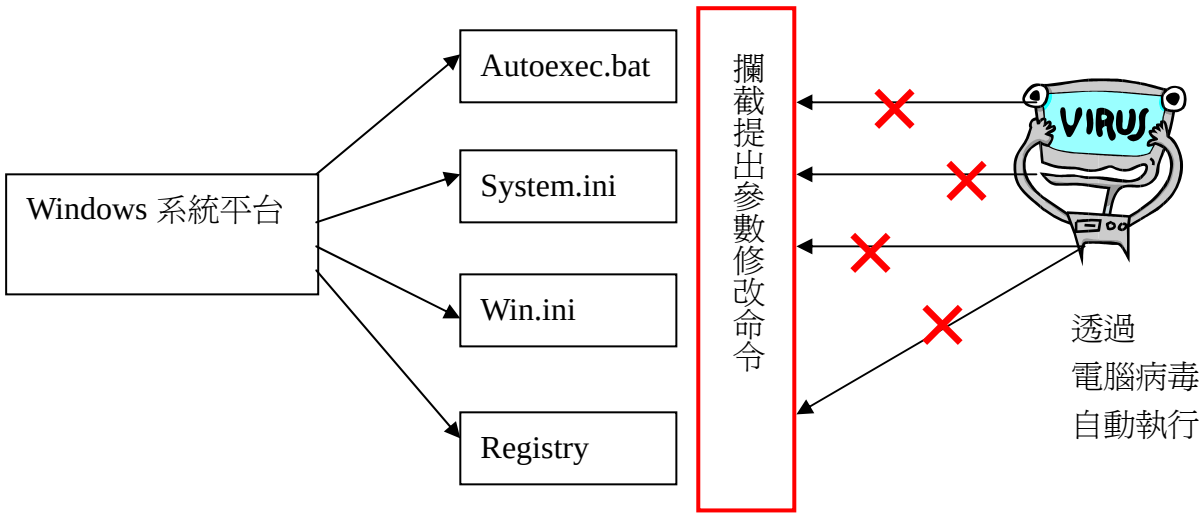
此病毒乃是利用 RegCreateKeyExA WinAPI 函數，對電腦 Registry 進行修改。較爲特別的是，他的傳染方式爲利用 Windows 平台之漏洞，故相較於其他者，其可在短時間內迅速傳播。

經由多數 Windows 平台下的病毒可看出，與傳統 PE 病毒的破壞方式已不復見。反而是修改 Registry 方式的蠕蟲病毒爲多數。當然，目前的病毒不勝枚舉，無法一一列出，但經由上述觀察研究發現，上處列出之病毒程式碼乃近代較爲有名或曾經引起重大災害之病毒，全都是利用 WinApi 函數對 Registry 進行修改，雖然指令有相異之處，但差別上只在程式開發環境或是封裝方式不同。

若能在 Windows 系統建立參數監控偵測系統，對於系統重要參數如：Autoexec.bat,System.ini,Win.ini,Registry 進行監控或是阻斷寫入，便可防止電腦病毒開機時自動運行，易言之，當電腦重新開機後，惡意程式將無法運行，也就是說記憶體內無病毒存在。有別於一般的檔案辨識掃描引擎。故不論是已知或是未知病毒，若能有效攔阻病毒修改參數，則可使病毒無法運行 (如圖五、圖六)。



圖五：參數啓動示意圖



圖六：封閉參數示意圖

2、殘餘系統影響

在實驗一機器裡，發現已移除參數項，但系統卻無法完善地正常運作，對使用者是一大困擾。經由文獻上發現，病毒的運行除了參數修改外，架構較爲複雜的電腦病毒，會殘留許多對電腦影響的指令或是 Registry 機碼字串。

(1)NetSky 蠕蟲對系統預設值之修改

```
path=%SystemRoot%\Drivers\etc\hosts

127.0.0.1 www.norton.com
127.0.0.1 norton.com
127.0.0.1 yahoo.com
…

127.0.0.1 google.com
```

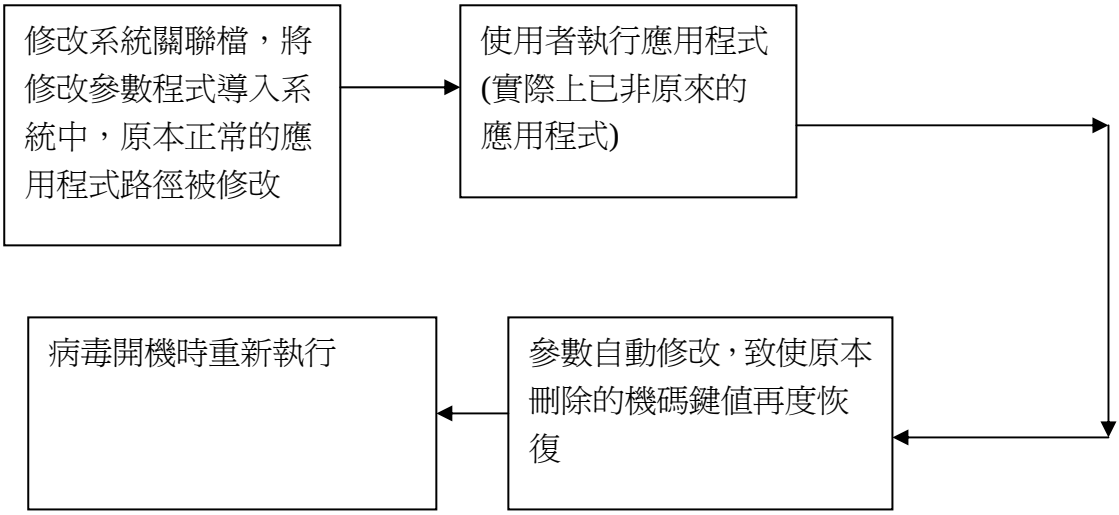
造成連向以上位置者，目的地將全部轉向至 localhost，則無法開其網頁。對於使用者亦是一大困擾。不過，此不至於影響到整個系統的穩定性。

(2)關聯檔恢復 Registry

某些惡意程式利用殘留方式，處心積慮欲使修改參數之命令百無一失，假若 Registry 等參數原設定在開機自動執行，但使用者移除參數後，仍會發現參數猶在。其原因乃是病毒已留下另一命令致使參數復存。

由循環可知，不論使用者如何刪除機碼鍵值，其參數將有機會恢復，除非使用者從不使用應用程式。最常見的就是替換 notepad，原因乃是 txt 檔非常普遍。以下便是關聯檔恢復 Registry

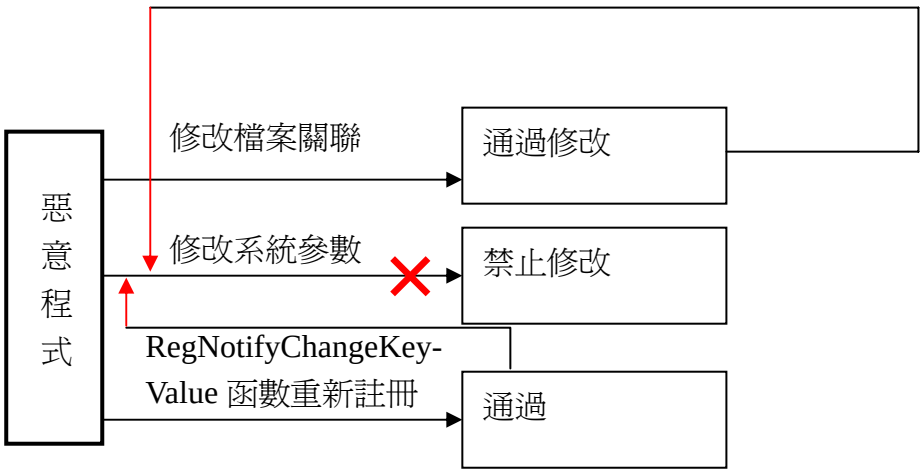
的示意圖(圖七)：



圖七：關聯檔恢復示意圖

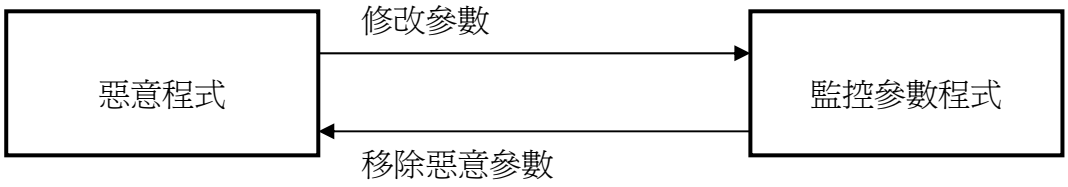
(3)WinAPI RegNotifyChangeKeyValue 函數恢復 Registry

另一種恢復 Registry 方法，是在編寫木馬程式文獻中發現，雖少有實例，但編寫方式相當簡單。運行便是利用 WinAPI RegNotifyChangeKeyValue 函數。當病毒偵測到系統參數改變時，經由 bool 值將參數是否修改傳入，若值爲需修改，則強制加入 Value 於欲修改之參數。如此只要使用者一移除惡意程式自動運行參數，則立刻又被修改(圖八)。



圖八：RegNotifyChangeKeyValue 函數關聯檔恢復示意圖

若是監控參數，且惡意程式使用 RegNotifyChangeKey 會造成無限迴圈(圖九)。



圖九： RegNotifyChangeKeyValue
函數關聯檔恢復示意圖

(4)VBS 檔案覆蓋恢復運作

LoveLetter 部份程式碼：

```
elseif (ext="mp3") or (ext="mp2") then
set mp3=fso.CreateTextFile(f1.path&".vbs")
mp3.write vbscopy
mp3.close
```

程式碼 1-6

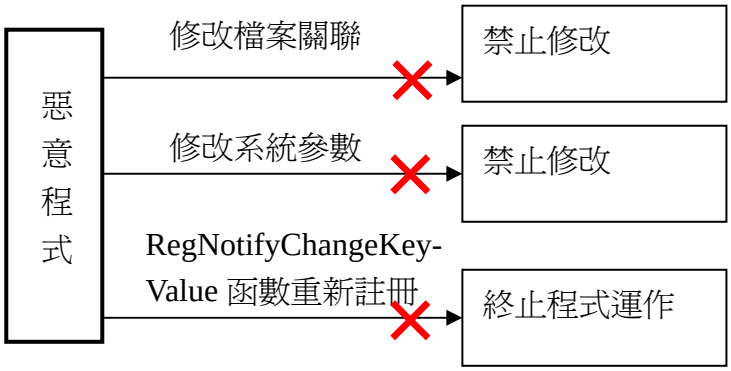
LoveLetter 是屬於 VBS 病毒，上面指令乃是將自身病毒寫入 MP3 或 MP2 檔案，並將副檔名改成 VBS。只是副檔名設爲隱性，以致使用者難以分辨，而造成病毒快速傳染。檔案傳染方式在以前的 PE 病毒時常利用。不過較爲特別的是，LoveLetter 是直接覆蓋文件，且利用系統漏洞致使副檔名看起來似*.mp3，這也是 LoveLetter 迅速傳播的原因。

3、避免強制寫入 Registry 與 VBS 檔案覆蓋

而對於惡意程式檔案關聯與 WinAPI RegNotifyChangeKeyValue 函數的參數修改運作模式，提出分別針對兩種解決方式(如圖十)：

(1) 檔案關聯乃是內存於 Windows Registry 機碼內，故只需監控 Registry 之運行，若病毒修改檔案關聯時，使之無效，則可完全阻絕因檔案關聯修改之故，使惡意程式新增參數恢復。

(2) 中斷惡意程式運行，其關閉後，使得惡意程在未關機之前無法運行，並移除惡意程式之新增參數。



圖十： RegNotifyChangeKeyValue
函數恢復示意圖

(3) 至於 VBS 類型之傳染方式，假若使用者開啓有毒檔案，若利用參數監控，則會發現病毒將會註冊參數，亦能杜絕病毒開機自動執行，但是所有的文件將無一倖免，並非能完全抵檔病毒破壞。但要防止破壞也極爲簡單，只需停用 VBS 功能與 Windows Scripting Host，設定預設開啓程式爲文字編輯器，則能完全防止 VBS 程式運作，意即病毒無法執行。

二、實驗二參數監控系統之攔截結果：

1、Ini 監控系統(如圖十一)。

圖十一： WIN.INI 監控系統

Ini 監控系統使用 TIniFile 類別來操控 ini 檔案，故在檔頭加上#include "IniFiles.hpp"。並使用 Timer 元件，以方便程式碼重覆運作執行。
主要版面爲：

```
TListBox *ListBox1; // ListBox1 爲顯性，儲存未授權函數
TButton *Button2;
TListBox *ListBox2; // ListBox2 設爲隱性，備份原有參數值。
TButton *Button3;
TTimer *Timer1; //重複執行程式碼之作用
```

圖十一爲 win.ini 監控。測試下列者：
(1)手動修改，新增參數值。
(2)執行以下程式碼(2-1)檢視，監控系統是否能偵測程式對 Win.ini 未經授權之修改。

```
//修改WIN.INI程式
TIniFile* ini = new TIniFile("D:\\ winini監控\\win.ini");
TStringList* SectionList = new TStringList();
__try
  {
    ini->ReadSections(SectionList);
    for (int i = 0; i < (SectionList->Count - 1); i++)
    {
      if (ini->ValueExists(SectionList->Strings[i],"Run"))
      {
        ini->WriteString(SectionList->Strings[i],
"Run","loader.exe");
      }
    }
  }
__finally
  {
    SectionList->Free();
```

程式碼2-1

而實驗結果發現，TIniFile與檔案比對法兩者相輔相成，能成功完成對於win.ini監控。由於System.ini運作模式，只須更改檔案路徑即可達到對於System.ini之監控。

Win.ini監控系統程式之原始碼如下：

```
//---------------------------------------------------------------------
-----------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------
-----------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
char* backuppath = "d:\\winini°Ê±±\\back.dat" ; //備份位置
char* cpath = "d:\\winini°Ê±±\\win.ini" ;  //win.位置
```

```
//---------------------------------------------------------------------
-----------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
-----------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
CopyFile(cpath,backuppath,true); //初始值，若備份檔不存在，則自動存檔。
//反之則否，故bool初始值設true。
}
}
//---------------------------------------------------------------------
-----------
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
bool find = false;              //設bool初始值爲false
  TStringList *s1,*s2,*s3;
  s1 = new TStringList;
  s2 = new TStringList;
  s3 = new TStringList;
  s1->LoadFromFile(backuppath);         //讀入備份檔
  s2->LoadFromFile(cpath);             //讀入win.ini檔
    for (int i=0 ; i<s1->Count ; i++)       //進行s1與s2之比較
    {
      for (int j=0 ; j<s2->Count ; j++)
      {
        if (s1->Strings[i] == s2->Strings[j])
        {
          find = true;                 //相等者，跳出迴圈
          break;
         }
     }
      if (!find)          //不等者，輸入備份相異於ListBox2
      ListBox2->Items->Add(s1->Strings[i]);
      find = false;
    }
    for (int i=0 ; i<s2->Count ; i++)
    {
      for (int j=0 ; j<s1->Count ; j++)
      {
        if (s2->Strings[i] == s1->Strings[j])
        {
          find = true;
          break;     }
      }
      if (!find)
    s3->Add(s2->Strings[i]); //不等者，輸入win.ini相異之處於
                             //TSringList ->s3
```

```
    find = false;
  }
//迴圈結束

for (int i=0;i<s3->Count;i++)
     {
       ListBox1->Items->Add(s3->Strings[i]);
//將s3資料輸入於ListBox1
     }
  if (s3->Count>0) //假若s3輸出，意即參數遭修改
{
   If (IsIconic(Application->Handle))
   {
   ShowWindow(Application->Handle, SW_RESTORE);
   }
 else
   {
    SystemParametersInfo(SPI_SETFOREGROUNDLOCKTIMEOUT, 0, 0,
    SPIF_SENDCHANGE);
    SetForegroundWindow(Application->MainForm->Handle);
    }
 }  //強制跳出視窗

CopyFile(cpath,backuppath,false);
//此處需注意，爲何要將還沒有授權的先做爲標準?原因乃是若不爲此，則由於此//處使用
Timer
//定時器，下次比對資料時，仍會得到相異之處，因次循環而造成//ListBox全爲修改的參
數項。故先以未經授權者作爲基準。
}

//下一段是恢復按鈕，其乃是利用TIniFile功能修改win.ini，將未經授權的參數移除之。
//-------------------------------------------------------------------
-----------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
int index=ListBox1->ItemIndex;
if(index>-1)
  {
   Timer1->Enabled =false; //此處必須先停止系統，否則當程式在修改時//亦會造
成雖修改，但是系統還在運行，造成系統誤認系統刪除之功能亦爲未經授//權。但由於備份
資料是存於back.dat，故此段時間系統不會失效。

   String newstr = ListBox1->Items->Strings[index];
   String oldstr = ListBox2->Items->Strings[index];

  oldstr = StringReplace(oldstr, "=", ",",
TReplaceFlags()<<rfReplaceAll);
//尋找I/關鍵字"="作爲判別，將"="代換爲","，以便CommaText辨別。
   TStringList *Value=new TStringList;
   Value->CommaText=oldstr;
//分解字串，使得一爲"="之左(參數名稱)，一爲參//數之右(參數內容)
```

```
   String ValueName = Value->Strings[0];
   oldstr = Value->Strings[1];
   delete Value;
   TIniFile* ini = new TIniFile(cpath);
   TStringList* SectionList = new TStringList();
  __try
  {
    ini->ReadSections(SectionList); // 取得 Win.ini所有參數

    for (int i = 0; i < (SectionList->Count - 1); i++)

    {
      if (ini->ValueExists(SectionList->Strings[i],ValueName))
      {
        ini->WriteString(SectionList->Strings[i],ValueName,oldstr);
      }
    }
  // 檢查每一個參處是否有指定之參數名稱, 若有則覆蓋之.
  }
  __finally

  {
    SectionList->Free();
    ini->Free();//釋放資源   };
   CopyFile(cpath,backuppath,false);
//將修改後的參數作爲新的備份標準
   ListBox1->Items->Delete(index);  //移除ListBox的參數項
   ListBox2->Items->Delete(index);
   ShowMessage("恢復成功");
   Timer1->Enabled =true;
//Enable Timer1，使得參數監控系統繼續運
   }
}
//---------------------------------------------------------------------
-----------
//此Button乃是授權參數通過，由於已將修改值作爲基準，故只需移除ListBox上之
Items。
void __fastcall TForm1::Button3Click(TObject *Sender)
{
int index=ListBox1->ItemIndex;
ListBox1->Items->Delete(index);
ListBox2->Items->Delete(index);
}
//---------------程式結束//---------------------------------- 程式碼
2-2
```

2、Registry 監控系統：

本系統乃是用 TRegistry 類別對 Registry 進行操控，故需在檔頭加入#include <Registry.hpp> 並配合 TStringList 對 ValueNames 數量進行監控。

```
TListBox *ListBox1;
        TButton *Button2;
        TTimer *Timer1;
        TButton *Button1;
        TEdit *Edit1; //讀取字串值
```

\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run
爲其監控路徑。並進行以下步驟：
(1)手動至 Registry 新增一 Value。
(2)以一程式測試之，如下程式碼。

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
TRegistry *Reg = new TRegistry;
Reg->RootKey=HKEY_LOCAL_MACHINE;
Reg->OpenKey("\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\R
un",true);
Reg->WriteString("loader","c:\\Windows\\loader.exe");
Reg->Free();
```

程式碼 2-3

實驗結果發現，Regedit 當新增 Value 時(手動)，能正確攔截到參數。
執行上述之程式碼，結果亦然，故可成功利用 TRegistry 類別對 Registry 進行 Value 新增的監控。(如圖十二)

圖十二：Registry 監控

Registry 監控原始碼

```
//---------------------------------------------------------------------
-----------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------
-----------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
-----------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{

}
//---------------------------------------------------------------------
-----------
void __fastcall TForm1::FormCreate(TObject *Sender)
//Form開啓時的判斷，將備份檔存入資料夾中，若存在則略過。
{
        TStringList *bac = new TStringList;
        TStringList *backf = new TStringList;
        TRegistry *Registry = new TRegistry;

        Registry->RootKey=HKEY_LOCAL_MACHINE;

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

Registry->GetValueNames(bac);
        int b=bac->Count;
        for (int i=0;i<b;i++)
          {
           backf->Add(Registry->ReadString(bac->Strings[i]));
          }
//將Registry之key值下的Value讀入bac，亦將字串值讀入bacf
for(int j = 0;;j++)
        {

            if(!FileExists("backup" + IntToStr(j) + ".dat"))
                  {
                  bac->SaveToFile("D:\\backup.dat");
                  break;
                  }
             else
                {
```

```
                    break;
                  }
        }
//判斷檔案back.dat是否存在，是則跳脫迴圈，反之則寫入檔案。

for(int s = 0;;s++)
        {
             if(!FileExists("backstring" + IntToStr(s) +
                ".dat"))
                  {
                  backf->SaveToFile("D:\\backstring.dat");
                  break;
                  }
             else
                {

                  break;
                 }
        }
//判斷檔案backstring.dat是否存在，是則跳脫迴圈，反之則寫入檔案。

    delete bac,backf; //清除資源
}
//---------------------------------------------------------------------
-----------
//此處爲移除未經授權的Value
void __fastcall TForm1::Button2Click(TObject *Sender)
{
   int index=ListBox1->ItemIndex;
   if(index>-1) //判斷ListBox是否有Item項目
 {
   String dels = ListBox1->Items->Strings[index];
   Reg->RootKey=HKEY_LOCAL_MACHINE;
   Reg->OpenKey("\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Ru
   n",false);
    Timer1->Enabled =false; //停止Timer1運作
    Reg->DeleteValue(dels);
    ShowMessage("刪除成功");
    TStringList *bac = new TStringList;
    TStringList *backf = new TStringList;
    Reg->GetValueNames(bac);
         int b=bac->Count;
         for (int i=0;i<b;i++)
           {
            backf->Add(Reg->ReadString(bac->Strings[i]));
           }
     bac->SaveToFile("D:\\backup.dat");
     backf->SaveToFile("D:\\backstring.dat");
     delete bac,backf;
    ListBox1->Items->Delete(index);
    Timer1->Enabled =true;//啓動Timer1
  }
```

```
 else
 {}
}
//-------------------------------------------------------------------
-----------
//此爲爲程式核心，兩文件在此做數量比對。
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
 AnsiString sCurrentPath;
       sCurrentPath =
"\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run";
       Reg = new TRegistry;
       TStringList *str = new TStringList;
       TStringList *str2 = new TStringList;
       Reg->RootKey=HKEY_LOCAL_MACHINE;
       Reg->OpenKey(sCurrentPath,false);
       str->LoadFromFile("D:\\backup.dat");//將備份資料讀入
       Reg->GetValueNames(str2);


       int k=str->Count;   //讀取行數
       int q=str2->Count;
      if (k==q)
        {}

      else if (k<q)
      {
         for (int i=k;i<q;i++)
          {
          ListBox1->Items->Add(str2->Strings[i]);
          } //將差異新增項寫入ListBox1中

              if(IsIconic(Application->Handle))
              {
                ShowWindow(Application->Handle, SW_RESTORE);
              }
              else
              {
                SystemParametersInfo(SPI_SETFOREGROUNDLOCKTIMEOUT, 0,
0, SPIF_SENDCHANGE);
                SetForegroundWindow(Application->MainForm->Handle);
              }
//將視窗強制跳出，ShowWindow(Application->Handle, //SW_RESTORE);如此寫
法可避免程式最小化後無法強制跳出。

       TStringList *bac = new TStringList;
       TStringList *backf = new TStringList;
       Reg->GetValueNames(bac);
        int b=bac->Count;
        for (int i=0;i<b;i++)
          {
           backf->Add(Reg->ReadString(bac->Strings[i]));
```

```
          }

     bac->SaveToFile("D:\\backup.dat");
     backf->SaveToFile("D:\\backstring.dat");
     delete bac,backf;
//此處乃是將未經授權的Value作爲基準，以防止無限加入ListBox，造成不可預//期之錯誤。但可經由Button2移除未經授權基準備份。
        }
       else
       {}

        Reg->CloseKey();
        delete str,str2;
}
//---------------------------------------------------------------------------
//此處爲"授權Button"，由於已將其作爲基準，故不需更改檔案。只需移除//ListBox上的Item

void __fastcall TForm1::Button1Click(TObject *Sender)
{
int index=ListBox1->ItemIndex;
 if(index>-1)
 {
  ListBox1->Items->Delete(index);
 }
}



//---------------------------------------------------------------------------
//此處爲當使用者按下ListBox上的任意item，將會把Value之字串值顯現出來
void __fastcall TForm1::ListBox1Click(TObject *Sender)
{
   int index=ListBox1->ItemIndex;
   if(index>-1)
   {
    String Sv = ListBox1->Items->Strings[index];
    Reg->RootKey=HKEY_LOCAL_MACHINE;
    Reg->OpenKey("\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run",false);
    Sv = Reg->ReadString(Sv);
    Edit1->Text = Sv;
   }

 }

//---------------------------------------------------------------------------
----------程式結束//--程式碼2-4
```

本實驗可知，其實配合簡單的比對功能，就可完成對於參數的監控。其實由於檔案對於系統的操控乃是利用 WinAPI，故以上亦可利用 API 攔截對系統資訊之修改進行監控。而本實驗是純粹的利用檔案比對法來完成研究，缺點乃是不可知修改之程序爲何，但卻能實在的達到監控目的。

肆、結論與應用：

一、本研究發現，當病毒、尤其是蠕蟲與特洛依木馬程式執行時，將會在 Registry 註冊，少數對於 win.ini 透過開機時自動執行。

二、由於病毒在開機時須藉由電腦啓動參數重新執行，故若阻斷病毒新增參數項，則可避免病毒在開機時自動執行。

三、病毒可藉由 WinAPI 之 RegNotifyChangeKeyValue 函數與修改檔案關聯，當其存於記憶體時，強制新增 Value 至 Registry，但由於檔案關聯資料乃是存於 Registry，可利用監控 Registry，阻止之，而利用 RegNotifyChangeKeyValue 函數者，關閉程序則可避免。

四、今未知病毒極爲猖獗，但特徵與運作模式仍不離架構，故本研究設計的參數監控系統，包括 Win.ini 與 Registry 監控皆可攔阻未知病毒對系統之啓動參數修改，意即病毒無法在開機時自動執行。

五、參數監控技術若能與一般檔案比對引擎之防毒系統相輔相成，可提供電腦使用者安全的使用環境，避免未知病毒對電腦進行破壞，將傷害減到最低，亦是電腦使用者一大福音。

伍、參考文獻

一、參考書籍

1、李勁。精通C++Builder 6，台灣，文魁資訊股份有限公司，西元 2002 年 8 月。

2、洪維恩。C++教學手冊，台灣，博碩文化股份有限公司， 西元 2003 年 5 月。

3、陳惠貞、陳玄玲。電腦防毒 DIY，台灣，松崗電腦圖書資料股份有限公司，西元 1998 年。

4、蕭富貴。Borland C++ Builder 6 實用教學寶典，台灣，台科大圖書股份有限公司，C-22，西元 2003 年 10 月。