Università degli Studi di Salerno

Facoltà di Scienze Matematiche Fisiche e Naturali

INFORMATICA

Sicurezza su Reti 2

Malware Analysis & Removal Tool

Gruppo Certification Authority

Anno Accademico 2007-2008

Abstract

- Analizziamo la macchina infetta in attesa di attività anomale.
- Quando queste si presentano, adoperiamo dei tools di monitoraggio per identificare i processi coinvolti e terminarli.
- Reso il sistema utilizzabile, individuiamo il meccanismo coi cui il malware si avvia automaticamente al boot, e lo neutralizziamo.
- Individuiamo i files maligni e ne conduciamo una sommaria analisi statica per avere riscontro delle attività osservate dall'analisi *live*, cercando di ricostruire la logica di esecuzione del programma ostile.
- Eliminiamo i files incriminati, invertiamo le operazioni da loro effettuate e riavviamo la macchina per assicurarci che meccanismi sfuggiti alla prima analisi non provvedano a reinstallare il malware al reboot.
- Scriviamo un semplice *Removal Tool* che automatizzi le operazioni effettuate.
- Reinstalliamo il malware e verifichiamo l'efficacia del tool.

Analisi Iniziale

Fatto il login sulla macchina infetta, dopo un certo intervallo di tempo in cui sembra esserci una certa attività su disco, si presenta un popup d'errore che avverte di problemi sul drive `C:`. Cliccando `OK` segue un secondo popup, `Attention: FORMAT C:` (vedi fig. 1)

Tali popup sono decisamente sospetti dato che, differentemente da reali errori di sistema, sono accompagnati dalla presenza di entries sulla taskbar. Inoltre il secondo popup appare decisamente fake (*Warning* è più adeguato di *Attention* in questi casi, e il *FORMAT C:* maiuscolo e lapidario non è molto professionale). L'icona del task, inoltre, è facilmente riconoscibile come quella standard delle applicazioni **Visual Basic**. A questo punto proviamo instintivamente a utilizzare il **Task Manager** per identificare l'applicazione e terminarla, ma sembra essere stato rimosso/alterato.

Facciamo il gioco del malware e dopo aver cliccato OK apriamo `C:\`, ottenendo la vista piuttosto inquietante in fig. 2.

Essendo il sistema in esecuzione, appare improbabile una corruzione drastica del file system, mentre è più plausibile un'alterazione a livello di visualizzazione. Digitando infatti `C:\Windows` nella barra degli indirizzi vi si

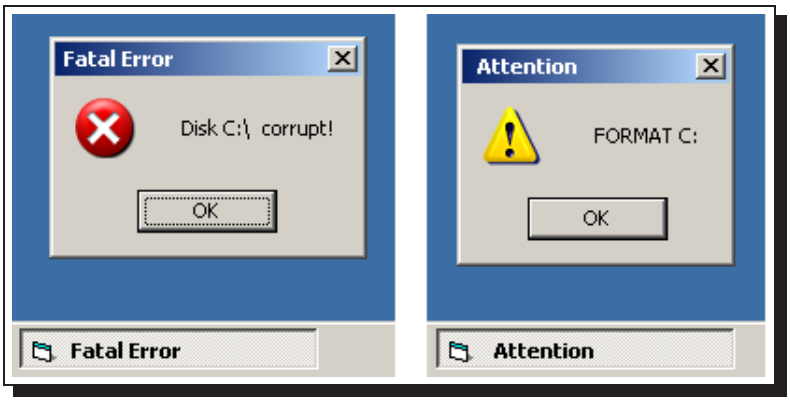

Figura 1: I popups... dovremmo preoccuparci?

accede tranquillamente, e un semplice `attrib -h c:\windows` in un prompt dei comandi conferma il sospetto che le subdirectory di `C:\` siano state semplicemente impostate come nascoste. Intanto, il sistema è diventato inutilizzabile: dopo l'OK al secondo popup, si verificano di continuo una serie di operazioni

- apertura della cartella **Documenti**
- avvio dell' **Utilità di deframmentazione dischi**
- avvio di **Internet Explorer**

e a un certo punto parte automaticamente il *reboot* del sistema.

Dopo un reboot, confermiamo un nostro sospetto: non dando l'OK ai popup il sistema resta perfettamente utilizzabile, segno che si tratta di operazioni bloccanti che, se non portate a termine, evitano l'esecuzione delle fastidiose operazioni appena elencate.

Essendoci fatta un'idea delle operazioni effettuate dal malware, proseguiamo con un'analisi più accurata.

Monitoraggio e identificazione

Installiamo dei preziosi strumenti freeware della **SysInternals** (società poi acquisita dalla **Microsoft** stessa): **Process Explorer**, **Process Monitor** e **Autoruns**. Attendiamo il popup d'errore e con **Process Explorer** identifichiamo immediatamente il processo `csrss.exe`, omonimo di un processo

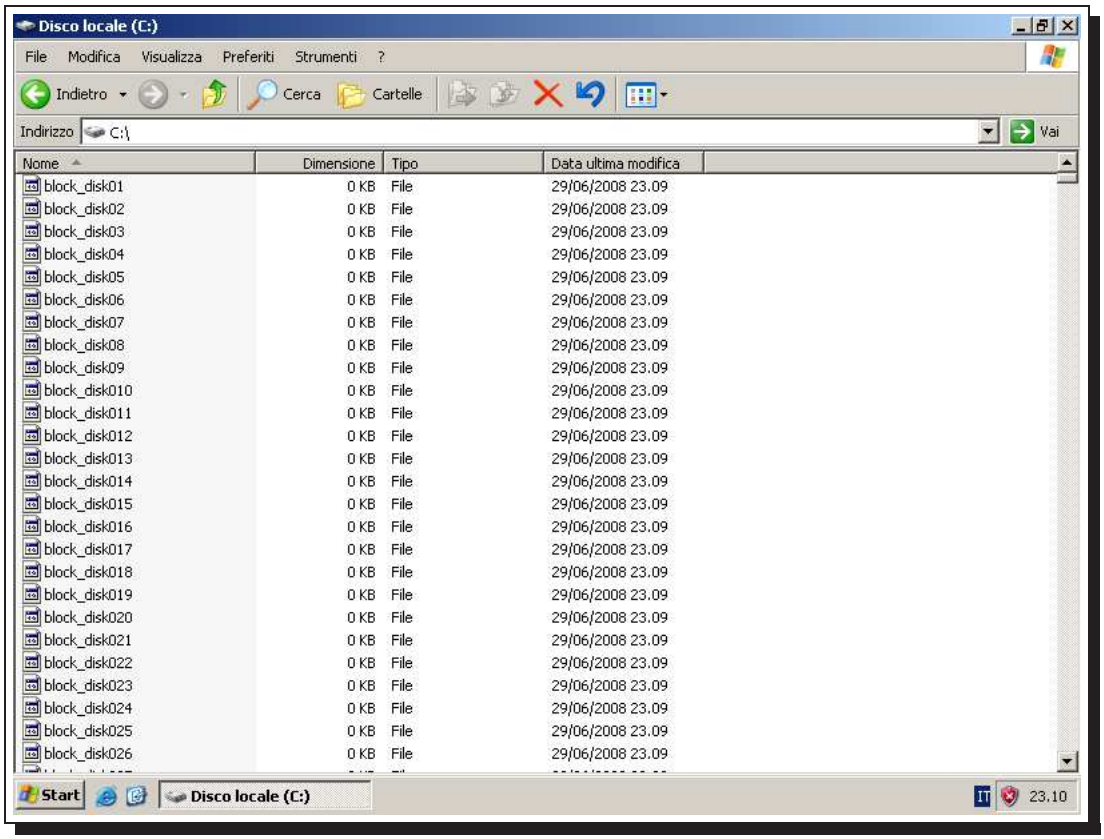

Figura 2: C: dopo l'OK al secondo popup

di sistema legittimo ma facilmente riconoscibile dall'icona e dalla struttura dell'albero dei processi.

Lo terminiamo con **Process Explorer** e abbiamo conferma della corretta individuazione dallo sparire del popup e dal corretto funzionamento del sistema dopo il kill. Annotiamo il path dell'eseguibile (`C:\windows\system\csrss.exe`) e passiamo a identificare il meccanismo con cui questo viene avviato al boot. Con **Autoruns** troviamo subito il riferimento al path incriminato: per l'esecuzione al boot è stato adoperato uno dei metodi più comuni/semplici, ovvero l'aggiunta di un valore nella chiave di registro

`HKLM\Software\Microsoft\Windows\CurrentVersion\Run`

Rimuoviamo l'entry con **Autoruns** stesso e riavviamo, verificando che il processo non abbia ulteriori metodi di attivazione: i popup non si ripresentano. Eliminiamo i files di spazzatura in C: con un `del c:\block*`.

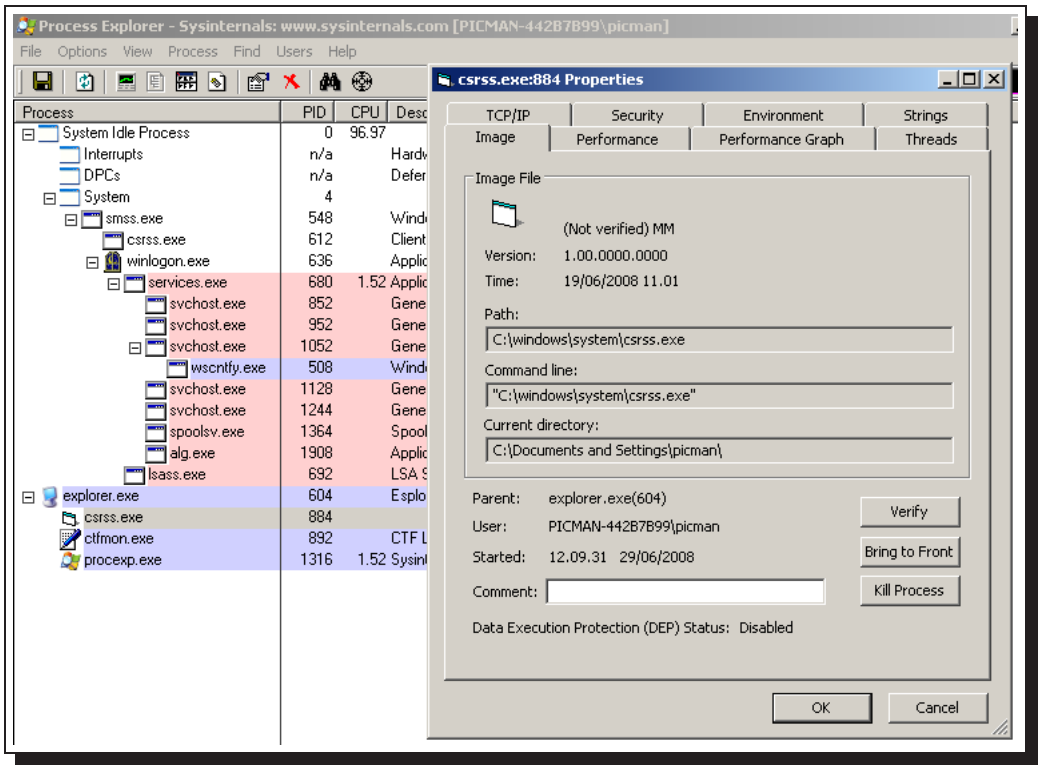

Figura 3: Individuazione del processo con Process Explorer

Osserviamo che il Task Manager è ancora inutilizzabile: evidentemente non era il processo maligno a bloccarne l'apertura, ma è stato corrotto l'eseguibile. Abbiamo riscontro dell'ipotesi andando a controllare `C:\windows\system32\taskmgr.exe`, che si rivela essere un file di testo contenente

```
questo non è il task manager

ahhhaaaahaa


:)

Ciao amici!

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa[....]a
```

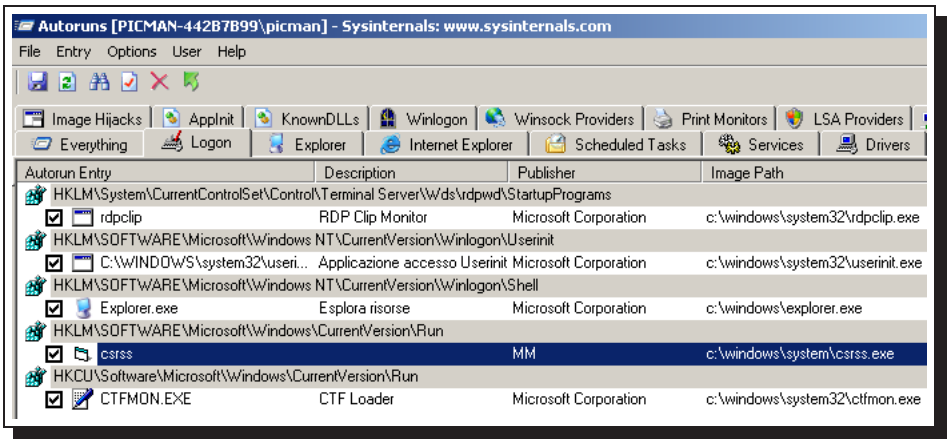

Figura 4: Individuazione del meccanismo di autostart con Autoruns

(con varie centinaia di a).

Ripristiniamo l'exe del **Task Manager** da un'installazione pulita di Windows XP ed eseguiamo una seconda live analisys con **Process Monitor**, questa volta sapendo già dove guardare e cercando di seguire man mano le operazioni effettuate dal programma ostile. Impostiamo **Process Monitor** perché visualizzi solo le operazioni effettuate dal processo che ha come *image path* `C:\windows\system\csrss.exe`.

A questo punto eseguiamo manualmente il malware e attendiamo il popup bloccante. **Process Monitor**, come previsto, mostra in tempo reale la creazione dei files di 0 Kb in `C:\`. Notiamo inoltre la creazione di altrettanti files in `C:\windows\temp`, passata inosservata in precedenza.

C'è traccia anche della manipolazione del registro per l'autostart, e, cosa più interessante, notiamo che poco prima dell'apparizione del popup il processo `csrss` che stiamo osservando accede a una dll nella sua stessa directory, dal nome eloquente: `C:\windows\system\PopupProject.dll`.

Andiamo a controllare in `C:\windows\system` e osservando la data di creazione dei files notiamo un insieme di dll sospette, oltre alla gia' citata `PopupProject.dll`: `createDLL.dll`, `createDLL2Project.dll`, `runProject.dll`, `RebootProject.dll`. Copiamo `csrss.exe` e le cinque dll in una directory e la teniamo da parte per le analisi successive.

A questo punto, diamo l'OK ai popup per far sì che l'esecuzione del malware continui. Notiamo delle operazioni `SetBasicInformationFile` sulle directory `C:\windows`, `C:\programmi` e `C:\Documents and Settings` subito dopo la chiusura dei popup, che impostano l'attributo hidden sulle

directories. In seguito l'esecuzione diventa difficile da seguire, a causa delle fastidiose operazioni avviate dal malware (riavvii compresi).

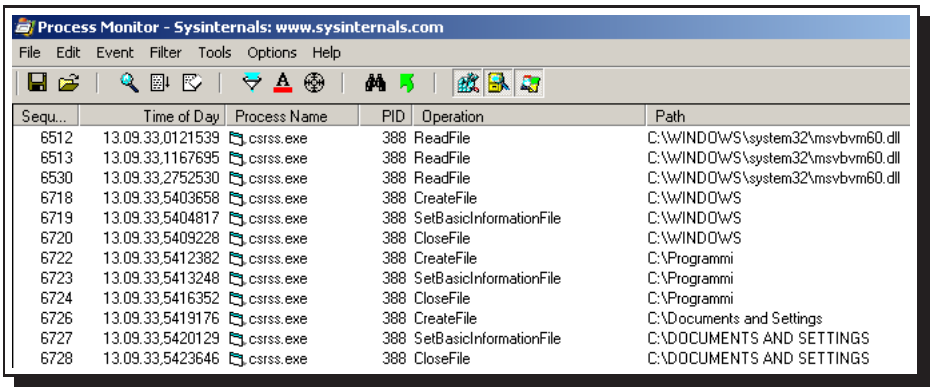

Figura 5: Live Analysis con Process Monitor

Interrompiamo l'analisi terminando il processo e, prima di passare ad analizzare i files incriminati notiamo che rieseguire il malware non ha effettuato operazioni sull'eseguibile del Task Manager: evidentemente l'alterazione rilevata in precedenza era stata effettuata da chi aveva installato il malware nella macchina, e non dal malware stesso.

Passiamo ora ad analizzare `csrss.exe` e le DLL correlate per ottenere riscontro su quanto osservato ed eventuali ulteriori informazioni sulle operazioni effettuate, in modo da poterle invertire nel removal tool.

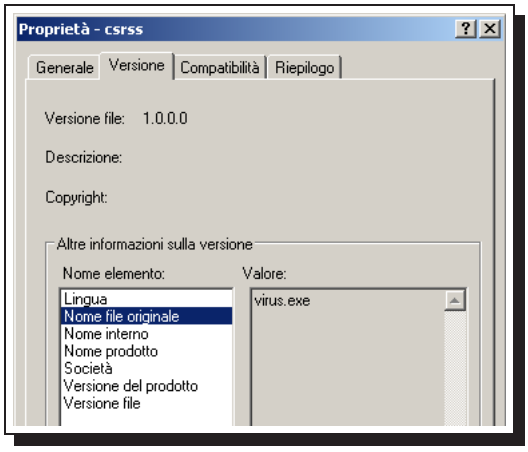

Figura 6: Nome originale: virus.exe!

Binary Analysis

Iniziamo con esaminare le proprietà dei files, e notiamo subito il *Nome originale* di `csrss.exe`: `virus.exe`. Evidentemente gli artefici dell'installazione del malware non hanno tra le loro priorità il non lasciare tracce, come testimonia anche il file di testo spacciato come `taskmgr.exe`. Come *Società* del file e delle DLL notiamo `MM`: una firma degli autori :) ? Ormai certi che si tratti di un'applicazione *Visual Basic*, dalle DLL caricate caratteristiche di VB come `MSVBVM60.DLL` oltre che dall'icona, cerchiamo un tool ad hoc per l'analisi di applicazioni *Visual Basic*. Installiamo **VB Decompiler Lite** (freeware) e gli diamo in pasto prima l'eseguibile e poi le dll: su 3 dll si è rifiutato di funzionare adeguatamente, e abbiamo deciso di provare a ottenere qualche informazione col rinomato ma complesso **IDA Pro** (in versione 4.9, freeware).

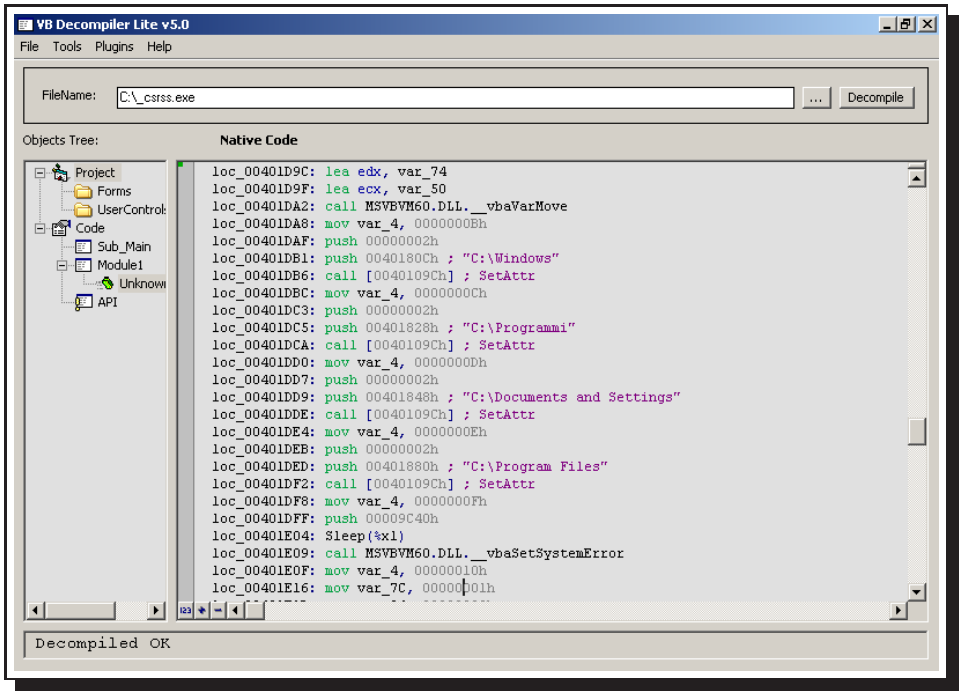

Figura 7: una schermata di VB Decompiler che mostra le chiamate a SetAttr che nascondono le directory.

Senza pretendere di effettuare un completo *reverse engineering* di tutto, andiamo a verificare in base alle stringhe contenute e agli indirizzi di funzioni

note (risolte dai tools usati nei rispettivi nomi) quale componente effettua quali operazioni. Seguono alcuni stralci significativi, file per file, e poi una valutazione complessiva.

csrss.exe

```
...
  loc_00401AB1: mov var_7C, 00401684h
        ; "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\csrss"
  loc_00401AB8: mov var_84, 00000008h
  loc_00401AC2: mov var_9C, 00401718h ; "C:\windows\system\csrss.exe"
  loc_00401ACC: mov var_A4, 00000008h
  loc_00401AD6: mov var_BC, 00401754h ; "REG_SZ"
...
  loc_00401B70: push 00401764h ; "regwrite"
...
  loc_00401DB1: push 0040180Ch ; "C:\Windows"
  loc_00401DB6: call [0040109Ch] ; SetAttr
  loc_00401DBC: mov var_4, 0000000Ch
  loc_00401DC3: push 00000002h
  loc_00401DC5: push 00401828h ; "C:\Programmi"
  loc_00401DCA: call [0040109Ch] ; SetAttr
  loc_00401DD0: mov var_4, 0000000Dh
  loc_00401DD7: push 00000002h
  loc_00401DD9: push 00401848h ; "C:\Documents and Settings"
  loc_00401DDE: call [0040109Ch] ; SetAttr
  loc_00401DE4: mov var_4, 0000000Eh
  loc_00401DEB: push 00000002h
  loc_00401DED: push 00401880h ; "C:\Program Files"
  loc_00401DF2: call [0040109Ch] ; SetAttr
...
```

Evidenti le chiamate per inserire il valore nel registro e per rendere invisibili le quattro directory indicate.

PopupProject.dll

```
...
  loc_11001A61: mov var_7C, 11001820h ; "Fatal Error"
  loc_11001A68: mov var_84, edi
  loc_11001A6E: call MSVBVM60.DLL.__vbaVarDup
  loc_11001A70: lea edx, var_74
  loc_11001A73: lea ecx, var_34
```

```
  loc_11001A76: mov var_6C, 110017ECh ; "Disk C:\  corrupt!"
  loc_11001A7D: mov var_74, edi
  loc_11001A80: call MSVBVM60.DLL.__vbaVarDup
  loc_11001A82: mov edi, [11001020h] ; MsgBox(arg_1, arg_2, arg_3, arg_4, arg_5)
...
  loc_11001AE8: mov var_7C, 11001864h ; "Attention"
  loc_11001AEF: mov var_84, 00000008h
  loc_11001AF9: call MSVBVM60.DLL.__vbaVarDup
  loc_11001AFB: lea edx, var_74
  loc_11001AFE: lea ecx, var_34
  loc_11001B01: mov var_6C, 11001840h ; "FORMAT C:"
...
```

Come indicato anche dal nome, la dll fa apparire i popup, di cui riconosciamo le stringhe.

RebootProject.dll

```
...
.text:110017A4                 unicode 0, <select * from Win32_OperatingSystem where Pr>
.text:110017A4                 unicode 0, <imary=true>,0
.text:11001812                 align 4
.text:11001814 aF:
.text:11001814                 unicode 0, <F>,0
.text:11001818 aWinmgmtsShutdo:                          ; DATA XREF: .text:11001ADC
.text:11001818                 unicode 0, <winmgmts:{(Shutdown)}//./root/cimv2>,0
.text:11001860 aExecquery:                               ; DATA XREF: .text:11001B1A
.text:11001860                 unicode 0, <ExecQuery>,0
.text:11001874 aReboot:                                  ; DATA XREF: .text:11001BA0
.text:11001874                 unicode 0, <Reboot>,0
...
```

Ovviamente, il comando di reboot (anche qui anticipato dal nome della dll).

runProject.dll

```
...
  loc_11001A0E: push 11001804h ; "Wscript.Shell"
...
  loc_11001A1F: call [11001054h] ; arg_1 = CreateObject(arg_2, arg_3)
...
```

```
  loc_11001A4B: mov eax, 11001824h ; "iexplore"
  loc_11001A50: push 00000001h
  loc_11001A52: push 11001838h ; "run"
...
  loc_11001ACB: mov eax, 1100185Ch ; "DFRG.MSC"
  loc_11001AD0: push 00000001h
  loc_11001AD2: push 11001838h ; "run"
...
```

Questa dll apre il folder Documenti, Internet Explorer e l'utility di deframmentazione.

createDLL.dll

```
...
.text:110017C0                 unicode 0, <Scripting.filesystemobject>,0
.text:110017F6                 align 4
.text:110017F8                 dd 1Ch
.text:110017FC aCBlock_disk0:                          ; DATA XREF: .text:11001A6E
.text:110017FC                 unicode 0, <C:\block_disk0>,0
.text:1100181A                 align 4
.text:1100181C a__vbastrmove   db '__vbaStrMove',0
.text:11001829                 align 10h
.text:11001830 aCreatetextfile:                        ; DATA XREF: .text:11001ABC
.text:11001830                 unicode 0, <CreateTextFile>,0
...
```

createDLL si occupa della creazione dei files vuoti `block_disk0[1-10000]` in C:.

createDLL2Project.dll

```
...
.text:110017A0                 unicode 0, <Scripting.filesystemobject>,0
.text:110017D6                 align 4
.text:110017D8                 unicode 0, <(>,0
.text:110017DC aCWindowsTempFi:                        ; DATA XREF: .text:11001A4E
.text:110017DC                 unicode 0, <C:\Windows\Temp\file>,0
.text:11001806                 align 4
.text:11001808 a__vbastrmove   db '__vbaStrMove',0
.text:11001815                 align 4
.text:11001818 aCreatetextfile:                        ; DATA XREF: .text:11001A9C
```

```
.text:11001818                unicode 0, <CreateTextFile>,0
...
```

createDLL2Project si occupa della creazione dei files vuoti `file[1-10000]` in `C:\Windows\Temp`.

Conclusioni

L'analisi effettuata fa concludere che:

- il malware è fastidioso ma non causa danni: tutte le operazioni effettuate sono reversibili (solo il task manager è stato ripristinato prendendolo da un'altra installazione di Windows, ma abbiamo verificato che il programma non era stato responsabile della sua alterazione)
- non ci sono procedure di replicazione/infezione (per questo parliamo genericamente di malware e non di virus vero e proprio)

Uno pseudocodice approssimativo della logica d'esecuzione del programma potrebbe essere:

```
csrss {
    regWritePerAutostart();
    createDLL2Project.start();      //  files spazzatura in c:\windows\temp
    createDLL.start();              //  files spazzatura in c:\
    PopupProject.showFirstBlocking();   // popup C: corrupt
    PopupProject.showSecondBlocking();  // popup FORMAT C:
    setDirectoriesHidden();
    do {
        runProject.startRandom();   // iexplore, Documenti, defrag ...
        sleep(random(5000));
    } while (executionTime() < totSecondi);
    RebootProject.rebootNow();
}
```

Removal Tool

Abbiamo scelto di scrivere un semplice script VBS per invertire le operazioni effettuate dal malware e ripulire il sistema. Il codice, autoesplicativo, nell'ordine

- termina il processo maligno
- rimuove dal registro il valore per l'autostart
- rimuove i files del malware (exe e dll)
- rimuove i files di 0 kb creati in `C:\` e in `C:\windows\temp`
- ripristina la visibilità delle directories nascoste dal malware

Codice killMalware.vbs

```
''' SICUREZZA SU RETI 2 '''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' script di rimozione malware
'   by Dario Scarpa - gruppo Certification Authority
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
On Error Resume Next

''' termina processo del malware ''''''''''''''''''''''''''''''''''''''''''''''''
Set objWMIService = GetObject("winmgmts:" _
& "{impersonationLevel=impersonate}!\\.\root\cimv2")

Set colProcess = objWMIService.ExecQuery _
("Select * from Win32_Process Where Name = 'csrss.exe' and CommandLine != null" )
    ' Nota:
    '   "CommandLine != null" esclude il "crsss.exe" legittimo,
    '   di sistema, che ha CommandLine == null

For Each objProcess in colProcess
    objProcess.Terminate()
    WSCript.Echo "Processo maligno terminato! Segue pulizia del sistema..."
Next
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

''' rimuove dal registro l'autostart del malware ''''''''''''''''''''''''''''''''
const HKEY_LOCAL_MACHINE = &H80000002

Set oReg=GetObject("winmgmts" _
& ":{impersonationLevel=impersonate}!\\.\root\default:StdRegProv")

strKeyPath = "SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
strStringValueName = "csrss"
oReg.DeleteValue HKEY_LOCAL_MACHINE,strKeyPath,strStringValueName
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
```

```
''' rimuove files del malware ''''''''''''''''''''''''''''''''''''''''''''''''''''
Set fso=CreateObject("Scripting.FileSystemObject")
WinDir = fso.GetSpecialFolder(0)    ' %systemroot%
Drive = Left(WinDir,3)          ' "c:\"
MalwarePath = WinDir & "\system\"

Dim mwFiles(6)
mwFiles(0) = "csrss.exe"
mwFiles(1) = "PopupProject.dll"
mwFiles(2) = "createDLL.dll"
mwFiles(3) = "createDLL2Project.dll"
mwFiles(4) = "runProject.dll"
mwFiles(5) = "RebootProject.dll"

For Each mwFile In mwFiles
    fso.DeleteFile(MalwarePath & mwFile)
Next
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

'''rimuove files spazzatura in c:\windows\temp '''''''''''''''''''''''''''''''''''
fso.DeleteFile(windir & "\temp\file*")
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

'''rimuove files spazzatura in c:\ ''''''''''''''''''''''''''''''''''''''''''''''
fso.DeleteFile(Drive & "block_disk*")
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

'''ripristina visibilità  directories settate "hidden" ''''''''''''''''''''''''''''
Dim dirs(4)
dirs(0) = WinDir
dirs(1) = Drive & "Documents And Settings"
dirs(2) = Drive & "Program Files"
dirs(3) = Drive & "Programmi"

For Each dir In dirs
    fso.GetFolder(dir).Attributes = 0
Next
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

''' Ok! ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
WScript.Echo "Malware rimosso!"
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

WScript.Quit
```