Sistemas Operativos II

Clase 6

Seguridad Segunda Parte

Universidad Arturo Jauretche
Ingeniería Informática

Docentes:
Coordinador: Ing. Jorge Osio
Profesores: Ing. Eduardo Kunysz
Ing. Daniel Alonso

Errores de código:
BUGS

CÓMO EXPLOTAR LOS ERRORES (BUGS) EN EL CÓDIGO

- Como pueden atacar los usuarios externos y trastornar el sistema operativo desde el exterior?
- Por lo general a través de Internet. Casi todos los mecanismos de ataque aprovechan los errores en el sistema operativo, o en algún programa de aplicación popular como Internet Explorer o Microsoft Office.
- El escenario típico es que alguien descubre un error en el sistema operativo y después encuentra la manera de explotarlo para comprometer a las computadoras que ejecutan el código defectuoso.
- Hay varias formas en que se pueden explotar los errores. Una manera simple y directa consiste en los pasos siguientes:

CÓMO EXPLOTAR LOS ERRORES (BUGS) EN EL CÓDIGO

1. Ejecutar una exploración de puertos automatizada para encontrar máquinas que acepten conexiones telnet.
2. Tratar de iniciar sesión adivinando las combinaciones de nombre de inicio de sesión y contraseña.
3. Una vez adentro, ejecutar el programa defectuoso con datos de entrada que activen el error.
4. Si el programa contiene errores en SETUID root, crear un shell SETUID root.
5. Obtener e iniciar un programa zombie que escuche en un puerto IP en espera de comandos.
6. Hacer que el programa zombie se inicie siempre que el sistema reinicie.

CÓMO EXPLOTAR LOS ERRORES (BUGS) EN EL CÓDIGO

La secuencia de comandos **se puede ejecutar por mucho tiempo**, pero hay una buena probabilidad de que tenga éxito en un momento dado.

Al asegurar que **el programa zombie se inicie cada vez que se reinicie la computadora**, el atacante se asegura de que siempre sea una computadora zombie.

Otro escenario común es iniciar un virus que infecte a las máquinas por Internet y explotar el error después de que el virus se instale en una nueva máquina. Aquí se reemplazan los pasos 1 y 2, lo demás sigue se hace igual.

CÓMO EXPLOTAR LOS ERRORES (BUGS) EN EL CÓDIGO

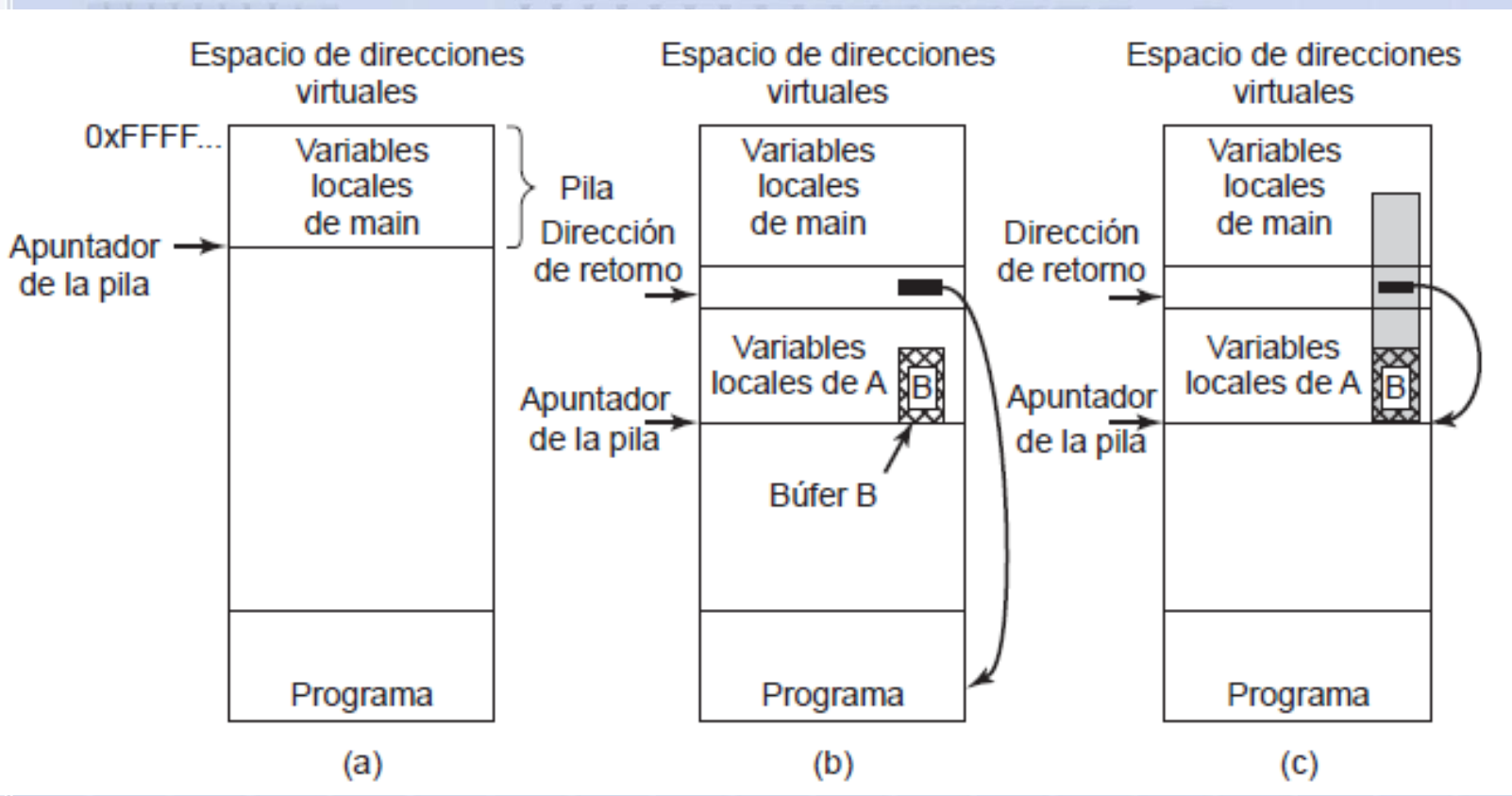
Ataques de desbordamiento del búfer

- Una de las fuentes más extensas de ataques, se deben al código C del SO
- Ningún compilador de C realiza la comprobación de los límites en los arreglos. En consecuencia, tampoco se revisa la siguiente secuencia de código, que no es válida:

```
int i;
char c[1024];
i = 12000;
c[i] = 0;
```

CÓMO EXPLOTAR LOS ERRORES (BUGS) EN EL CÓDIGO

- El resultado es que se sobrescribe cierto byte de memoria que esté a una posición de **10,976 bytes** fuera del arreglo, posiblemente con consecuencias desastrosas.



CÓMO EXPLOTAR LOS ERRORES (BUGS) EN EL CÓDIGO

Ataques mediante cadenas de formato

- Algunos programadores detestan escribir en la computadora aunque sean excelentes mecanógrafos.
- ¿Por qué nombrar una variable como *cuenta_referencia cuando es obvio que cr significa lo mismo* y nos ahorra 15 pulsaciones de tecla en cada ocurrencia?
- Considere el siguiente fragmento de un programa de C que imprime la tradicional bienvenida al lenguaje C al inicio de un programa:

```
char *s = “Hola programador”;
printf(“%s”,s);
```

CÓMO EXPLOTAR LOS ERRORES (BUGS) EN EL CÓDIGO

La llamada a la función *printf* tiene dos argumentos: la cadena de formato "%s", que le indica que debe imprimir una cadena, y la dirección de la misma.

Al ejecutarse esta pieza de código, se imprime la cadena en la pantalla (o a donde vaya la salida estándar). El código es correcto.

Pero suponga que el programador se vuelve flojo y en vez de lo anterior escribe:

```
char *s = "Hola programador";
printf(s);
```

Esta llamada a *printf se permite, ya que printf tiene un número variable de argumentos, de los cuales* el primero debe ser una cadena de formato.

CÓMO EXPLOTAR LOS ERRORES (BUGS) EN EL CÓDIGO

Seis meses después se instruye a otro programador:

```
char s[100], g[100] = "Hola"; /* declara s y g; inicializa g */
gets(s) /* lee una cadena del teclado y la coloca en s */
strcat(g, s); /* concatena s al final de g */
printf(g); /* imprime g */
```

- Hasta ahora todo va bien (excepto por el uso de *gets, que está sujeto a los ataques de desbordamiento de búfer.*
- un usuario conocedor se daría cuenta con rapidez de que la entrada que se acepta del teclado es una cadena de formato y como tal, funcionarán todas las especificaciones de formato permitidas por *printf.*

CÓMO EXPLOTAR LOS ERRORES (BUGS) EN EL CÓDIGO

- hay unos cuantos formatos son especiales. Por ejemplo, "%n" no imprime nada, sino que calcula cuántos caracteres se debe haber enviado ya como salida en la posición en la que aparezca en la cadena, y almacena este valor en el siguiente argumento de *printf para procesarlo.*

```
int main(int argc, char *argv[])
{
int i=0;
printf("Hola %nprogramador\n", &i); /* %n se almacena en i
    */
printf("i=%d\n", i); /* ahora i es 6 */
}
```

Cuando este programa se compila y ejecuta, el resultado es:

Hola programador i=6

CÓMO EXPLOTAR LOS ERRORES (BUGS) EN EL CÓDIGO

- La variable *i se ha modificado mediante una llamada a printf*
- Significa que si se imprime una cadena de formato tal vez se almacene una palabra (o muchas) en la memoria.
- Como al imprimir una cadena de formato se puede llegar a sobrescribir la memoria, ahora tenemos las herramientas necesarias para sobrescribir la dirección de retorno de la función *printf en la pila y saltar hacia cualquier* otra parte.
- A este método se le conoce como **ataque mediante cadenas de formato.**.
- Si el programa tiene SETUID root, el atacante puede crear un shell con privilegios de usuario raíz (root)
- Una forma de combatir estos ataques es marcar las páginas de la pila como de lectura/escritura, pero no de ejecución.

CÓMO EXPLOTAR LOS ERRORES (BUGS) EN EL CÓDIGO

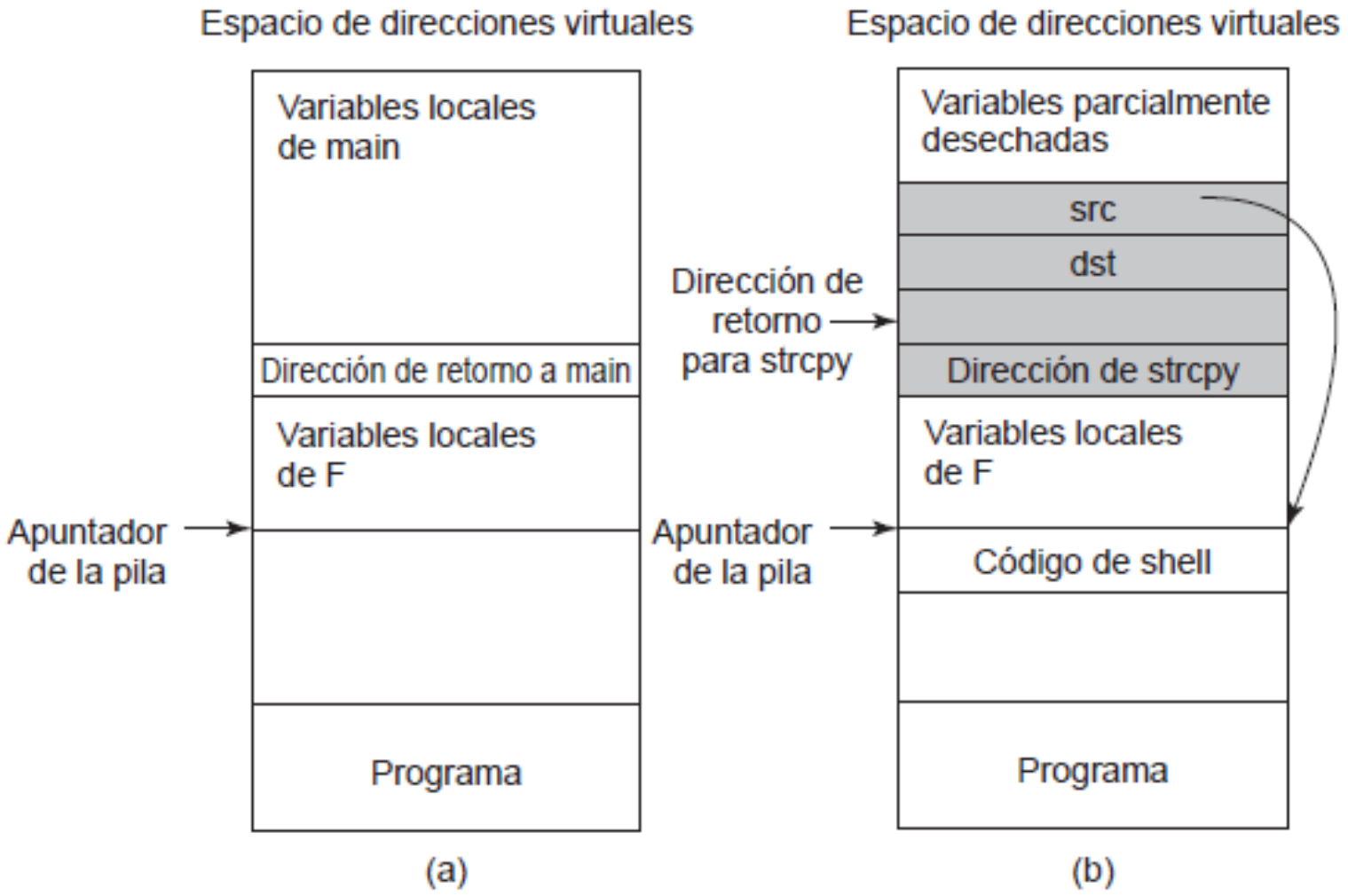
Ataques de retorno a libc

- Existe un ataque que funciona aún si los programas en la pila no se pueden ejecutar. A esto se le conoce como **ataque de retorno a libc.**
- Suponga que uno de los ataques anteriores sobreescribe la dirección de retorno de la función actual, pero no puede ejecutar el código del atacante.
- Casi todos los programas en C están vinculados con la biblioteca *libc que contiene* las funciones clave que la mayoría de los programas necesitan.
- Ej, *strcpy copia una cadena de bytes arbitraria de cualquier* dirección a cualquier otra dirección.
- Se puede *copiar* el programa del atacante "**shellcode**" al segmento de datos y ejecutarlo.

CÓMO EXPLOTAR LOS ERRORES (BUGS) EN EL CÓDIGO

- En la figura (a) siguiente podemos ver la pila justo después de que el programa ha llamado a una función *f.*
- *Este* programa se está ejecutando con privilegios de superusuario (es decir, SETUID root) y tiene un error que permite al atacante llevar su código de shell a la memoria, como se ilustra en (b).
- Además de llevar el código de shell a la pila, el ataque tiene que sobrescribir las cuatro posiciones sombreadas de retorno. En la dirección de retorno ahora está la dirección de *strcpy, por lo que f irá "de vuelta" a strcpy.*

CÓMO EXPLOTAR LOS ERRORES (BUGS) EN EL CÓDIGO



CÓMO EXPLOTAR LOS ERRORES (BUGS) EN EL CÓDIGO

Ataques por desborde de enteros

- Las computadoras realizan aritmética de enteros en números de longitud fija, por lo general de 8, 16, 32 o 64 bits.
- Si la suma de dos números que se van a sumar o multiplicar excede al máximo entero que se puede representar, se produce un desborde.
- Los programas de C almacenan y utilizan el valor incorrecto. Ej. dos enteros de 16 bits, con el valor 40,000. Si se multiplican el resultado aparente es 4096.
- Algunos programas de gráficos tienen parámetros de línea de comandos que proporcionan la altura y el ancho de un archivo de imagen.

CÓMO EXPLOTAR LOS ERRORES (BUGS) EN EL CÓDIGO

Ataques por inyección de código

- Un programa necesita duplicar un archivo y el programador podría utilizar la función *system, que crea una bifurcación del shell y ejecuta su* argumento como un comando del shell.
- **Suponga que el usuario escribe "abc" y "xyz; rm –rf /" en** vez de lo de la figura. Este comando primero copia el archivo y después **trata de eliminar de manera recursiva cada archivo** del sistema de archivos.

CÓMO EXPLOTAR LOS ERRORES (BUGS) EN EL CÓDIGO

```
int main(int argc, char *argv[])
{
   char org[100], dst[100], cmd[205] = "cp ";                  /* declara 3 cadenas */
   printf("Escriba el nombre del archivo de origen: ");        /* pide el archivo de origen */
   gets(org);                                                  /* obtiene la entrada del teclado */
   strcat(cmd, scr);                                           /* concatena src después de cp */
   strcat(cmd, " ");                                           /* agrega un espacio al final de cmd */
   printf("Escriba el nombre del archivo de destino: ");       /* pide el nombre del archivo de salida */
   gets(dst);                                                  /* obtiene la entrada del teclado */
   strcat(cmd, dst);                                           /* completa la cadena de comandos */
   system(cmd);                                                /* ejecuta el comando cp */
}
```

Si se escribe abc y xyz este programa hace: cp abc xyz

CÓMO EXPLOTAR LOS ERRORES (BUGS) EN EL CÓDIGO

Ataques por escalada de privilegios

- El **atacante engaña al sistema para que le proporcione más** permisos de acceso de los que tiene.
- Un ejemplo es de un programa que utilizó el **demonio cron**, el cual permite a los usuarios programar el trabajo a realizar.
- Este demonio se ejecuta como root y puede acceder a los archivos desde cualquier cuenta de usuario.
- El programa del atacante establecía su directorio en el demonio cron. Después fallaba de una manera que obligara a realizar un vaciado de núcleo.
- El programa atacante se estructuraba para que fuera un conjunto válido de comandos del demonio cron. El primer comando cambiaba la ejecución a SETUID root y el segundo ejecutaba este programa.

MALWARE

MALWARE

- Cuando se infecta una máquina, se instala software que reporta la dirección de la máquina capturada de vuelta a ciertas máquinas.
- También se instala una **puerta trasera** en la máquina, que permite a los criminales controlar con facilidad la máquina para que haga lo que le indiquen.
- Una máquina que se controla de esta forma se denomina **zombie, y una colección de estas máquinas se conoce como botnet, una contracción de "robot** network" (red de robots).

MALWARE (usos de botnet)

- Un criminal que controla una botnet la puede rentar para varios fines nefastos (y siempre comerciales).
- *Normalmente los criminales detrás de* la operación utilizan las máquinas de otras personas para que realicen el trabajo sucio, es difícil rastrearlos.
- También se puede utilizar para el chantaje, el cual es una posibilidad.
- Para instalar un ***keylogger en la máquina infectada.***
- Otra forma es permanecer invisible hasta que el

usuario inicie sesión de manera correcta en su cuenta bancaria de Internet.

MALWARE

Caballos de Troya (troyanos)

- Una práctica muy común es escribir cierto programa con una utilidad genuina e incrustar el malware en su interior. Como ejemplo los juegos, los reproductores de música, etc.
- A este método se le conoce como ataque de **caballo de Troya o *troyano en honor al caballo*** de madera lleno de soldados griegos que se describe en la *Odisea de Homero.*
- Cuando se inicia el programa llama a una función que escribe el malware en el disco como un programa ejecutable y lo inicia.

MALWARE

Virus

- ¿qué es un virus? Para resumir, un **virus es un programa que se puede reproducir** a sí mismo al adjuntar su código a otro programa.

El virus también puede hacer otras cosas además de reproducirse a sí mismo.

Los **gusanos** son como los virus, pero se duplican en forma automática.

MALWARE

Cómo funcionan los virus

- El escritor del virus trabaja en ensamblador (o tal vez en C) para obtener un producto pequeño y eficiente.
- Después de escribir su virus, lo inserta en un programa utilizando un **dropper.**
- **Luego, ese programa infectado se distribuye, tal vez** publicándolo en una colección de software gratuito como "nuevo juego emocionante".
- Ya iniciado, por lo general empieza por infectar a otros programas en la máquina y después ejecuta su **carga útil.**

- **VAMOS A ANALIZAR ALGUNOS VIRUS:**

MALWARE

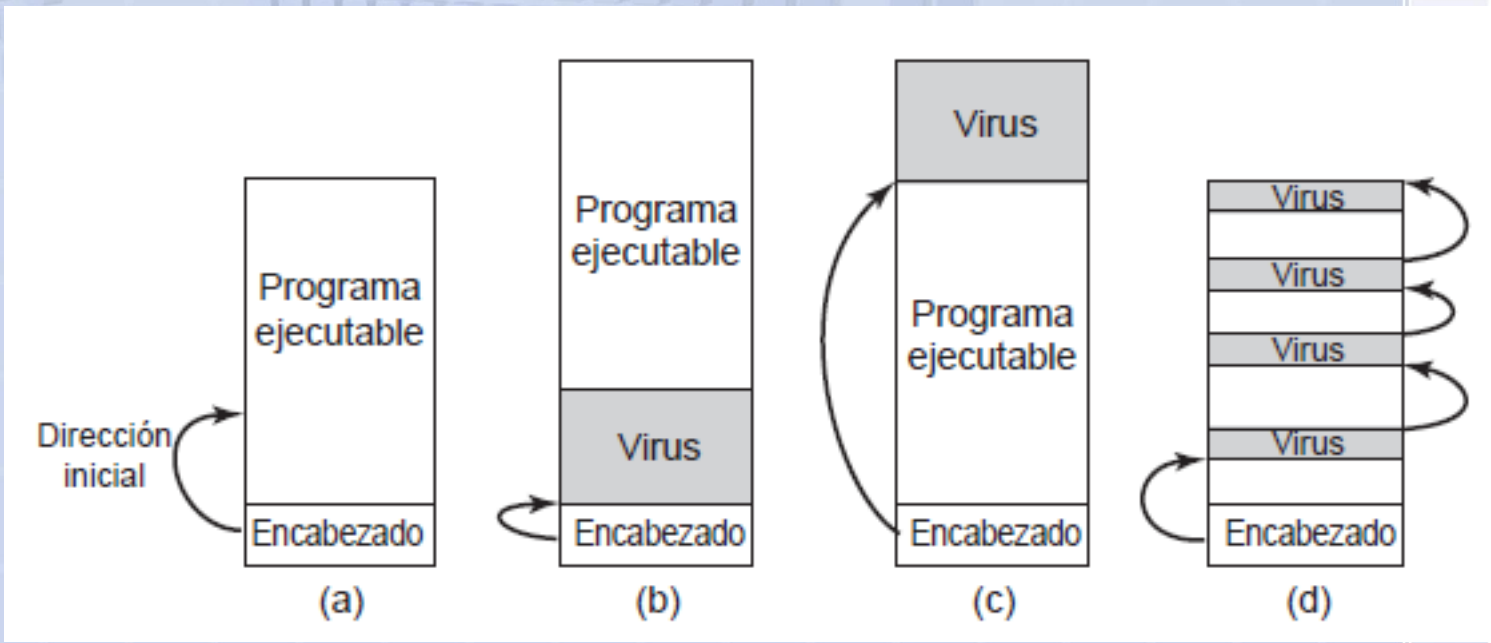
Virus de compañía

En realidad no infecta a u programa, sino que se ejecuta cada vez que lo hace el programa

MALWARE

Virus de programa ejecutable

Los virus que infectan programas ejecutables son más complejos que los de compañía. El tipo más simple de virus de programa ejecutable sólo sobrescribe al programa. **La figura proporciona la lógica de infección de dicho virus.**



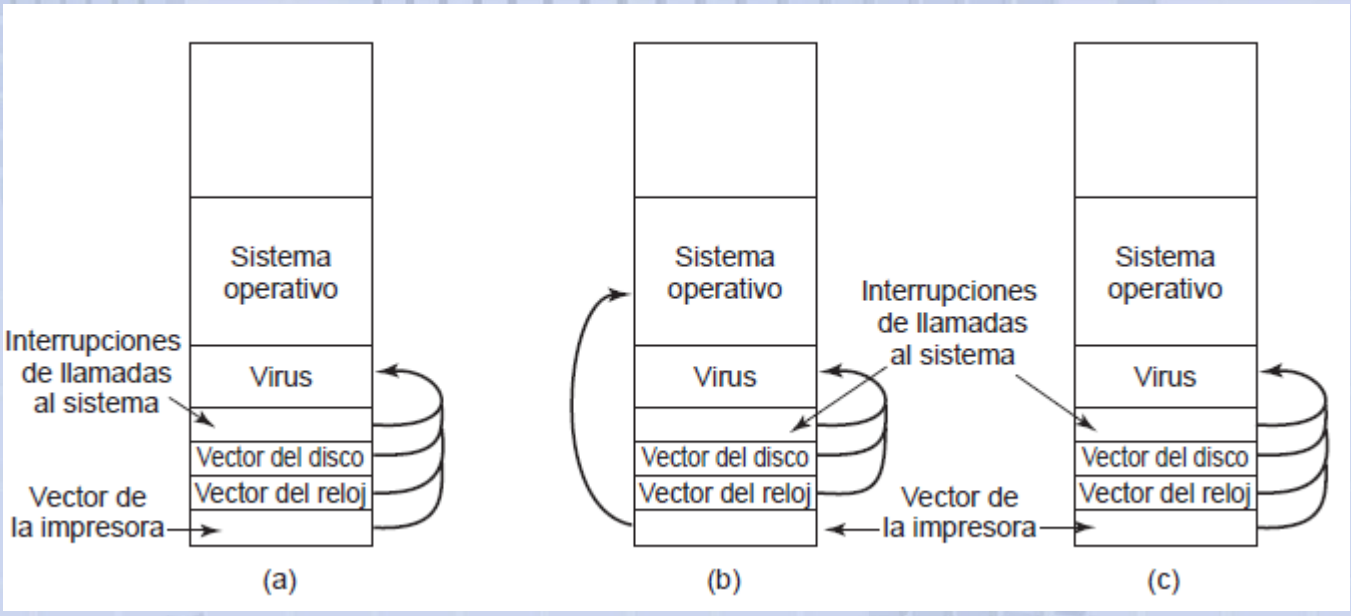
MALWARE

Virus del sector de arranque

- Cuando una PC inicia el BIOS lee el registro maestro de arranque de la parte inicial del disco de arranque y lo coloca en la RAM para ejecutarlo. Después, ese programa carga el sistema operativo o utiliza un cargador para cargarlo.
- Se creo un virus que pudiera sobrescribir el registro maestro de arranque o el sector de arranque.
- Este virus copia primero el verdadero sector de arranque en un lugar seguro en el disco, para que pueda iniciar el sistema operativo cuando termine. El virus se disfraza como sectores defectuosos.

Cuando se inicia la computadora, el virus se copia a sí mismo en la RAM. Cuando está listo, inicia el sistema operativo y por lo general permanece residente en la memoria, para poder observar la acción.

MALWARE


Sistema operativo
Interrupciones de llamadas al sistema
Virus
Vector del disco
Vector del reloj
Vector de la impresora
(a)
Sistema operativo
Virus
Vector del disco
Vector del reloj
(b)
Interrupciones de llamadas al sistema
Sistema operativo
Virus
Vector del disco
Vector del reloj
Vector de la impresora
(c)

MALWARE

Cómo se esparcen los virus

- Se inserta el virus en un programa que ha escrito (o robado) y empieza a distribuirlo en un sitio web.
- Para empezar, tal vez el virus infecte más archivos en el disco duro. También puede tratar de infectar el sector de arranque del disco duro.
- El virus se puede escribir de manera que compruebe si la máquina infectada está en una LAN. Después el virus puede empezar a infectar archivos desprotegidos en todos los servidores conectados a la LAN.
- Si el administrador ejecuta el archivo estando conectado como superusuario, el virus puede infectar ahora los archivos binarios del sistema, los drivers de dispositivos, el sistema operativo y los sectores de arranque.
- Las máquinas en una LAN pueden tener acceso remoto.
-

MALWARE

- Para evitar este escenario, todas las empresas deben tener una directiva general que indique a los administradores no cometer errores.
- Otra forma de esparcir un virus es publicar un programa infectado en un grupo de noticias de USENET.
- También es posible crear una página Web que requiera un complemento de navegador especial para verla, y después asegurarse que los complementos estén infectados.
- Un ataque distinto es infectar un documento y después enviarlo por correo electrónico a muchas personas.
- Para empeorar más las cosas, el virus puede entonces buscar la libreta de direcciones del usuario y después enviarse por correo a todos los remitentes de esa lista.

MALWARE

Gusanos

Morris descubrió dos errores en Berkeley UNIX que permitían obtener acceso no autorizado a las máquinas a través de Internet.

Escribió un programa llamado **gusano que se duplicaba a sí mismo, el cual** podía explotar estos errores y duplicarse en segundos en cada máquina a la que podía obtener acceso.

El gusano consistía en dos programas: el bootstrap y el gusano en sí. El bootstrap se llamaba *l1.c. Se compilaba y ejecutaba en el sistema al que estaba atacando.*

Una vez en ejecución, se conectaba a la máquina de la que había llegado, enviaba el programa del gusano y lo ejecutaba

El gusano analizaba las tablas de enrutamiento de su nuevo host para ver a qué máquinas estaba conectado y propagarse

MALWARE

Spyware

En general, el spyware es software que se carga de manera clandestina en una PC sin que su propietario se entere, y se ejecuta en segundo plano para hacer cosas a espaldas del propietario.

Hay una cantidad considerable de software gratuito que contiene spyware. muchos sitios Web muestran anuncios de pancarta que llevan a los navegantes a páginas Web infestadas de spyware

Es posible adquirir spyware (de hecho, cualquier tipo de malware) con sólo visitar una página Web infectada. , la página Web puede redirigir el navegador a un archivo ejecutable (*.exe).*

MALWARE

Acciones que realiza el spyware

Todos los puntos en la siguiente lista son comunes.

1. Modifica la página de inicio del navegador.
2. Modifica la lista de páginas favoritas (sitios favoritos) del navegador.
3. Agrega nuevas barras de herramientas al navegador.
4. Cambia el reproductor de medios predeterminado del usuario.
5. Cambia el motor de búsqueda predeterminado del usuario.
6. Agrega nuevos iconos al escritorio de Windows.
7. Reemplaza los anuncios (*banners) en las páginas Web con los que elije el mismo* spyware.
8. Coloca anuncios en los cuadros de diálogo estándar de Windows y Genera un flujo continuo e imparable de anuncios emergentes (*pop-up).*

MALWARE

Rootkits

Un **rootkit es un programa o conjunto de programas y archivos que intenta ocultar su existencia**.

Los rootkits se pueden instalar mediante cualquiera de los métodos analizados hasta ahora, incluyendo los virus, gusanos y spyware.

Tipos de rootkits

1. **Rootkits de firmware. Por lo menos en teoría, para ocultar un rootkit en el BIOS, éste se**

reprograma y se incluye una copia del rootkit. Dicho rootkit obtendría el control cada vez que se iniciara la máquina

MALWARE

2. **Rootkits de hipervisor. Un tipo de rootkit muy sigiloso podría ejecutar todo el sistema operativo**

y las aplicaciones en una máquina virtual bajo su control. Este tipo de rootkit modifica la secuencia de arranque, de tal forma que al encender la máquina se ejecute el hipervisor

directamente en el hardware, que a su vez inicia el sistema operativo y sus aplicaciones en una máquina virtual.

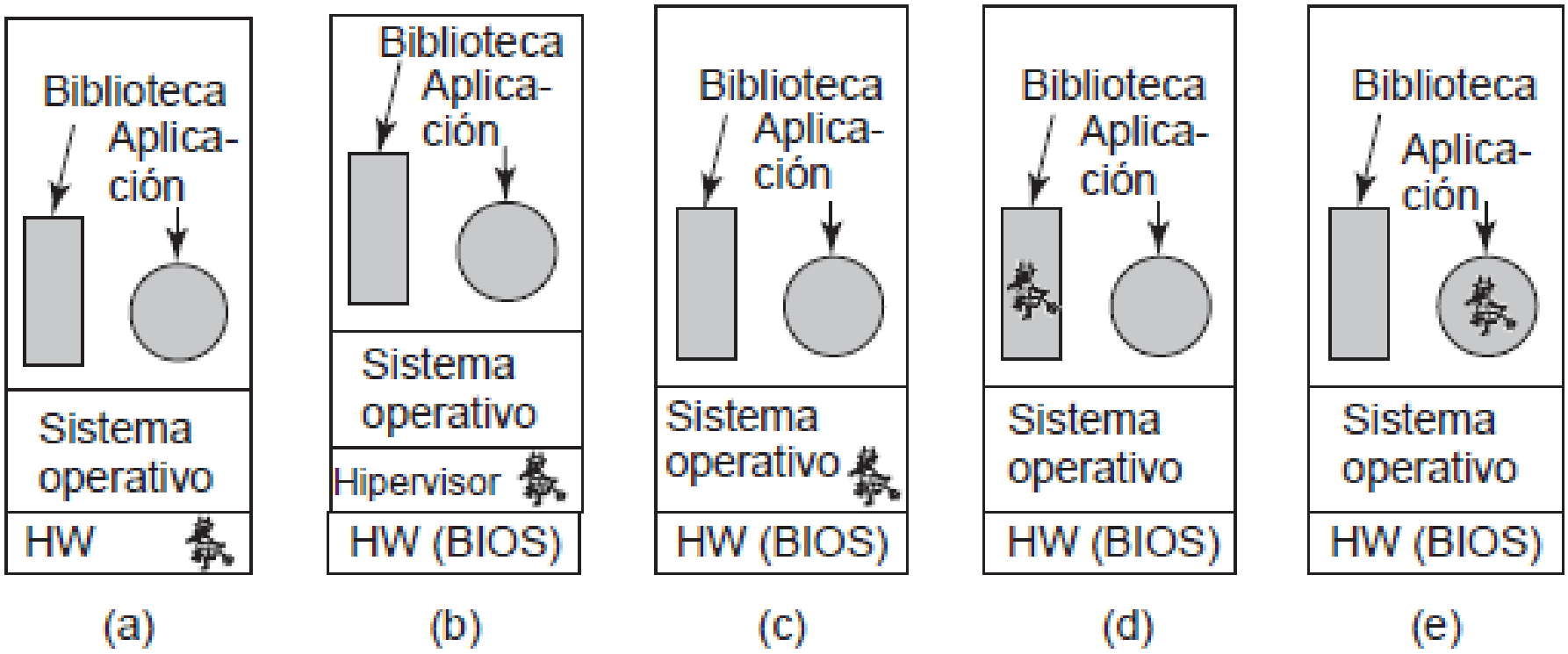
3. **Rootkits de kernel. El tipo más común de rootkit en la actualidad es uno que infecta al sistema** operativo y se oculta en él como un driver de dispositivo, o como un módulo de kernel que se puede cargar de manera opcional.

4. **Rootkits de biblioteca. Otro lugar en donde se puede ocultar el rootkit es en la biblioteca**

del sistema; por ejemplo, *libc en Linux.* inspeccionar los argumentos de las llamadas al sistema.

MALWARE

5. **Rootkits de aplicación. Otro lugar para ocultar un rootkit es dentro de un programa de aplicación** extenso, en especial uno que cree muchos archivos nuevos mientras se ejecuta (perfiles de usuario, vistas previas de imágenes, etc.



DEFENSAS

La idea básica aquí es que debemos tener varios

niveles de seguridad, de manera que si se viola uno de ellos, aún quedan otros niveles de defensa.

Firewalls

En consecuencia, se requieren mecanismos para mantener los bits "buenos" adentro y los bits

"malos" afuera.

Una empresa puede tener muchas LANs conectadas en formas arbitrarias, pero todo el tráfico entrante y saliente de la empresa pasa de manera obligatoria a través de un firewall.

Hay dos variedades básicas de firewall: de hardware y de software. Las empresas optan por los firewalls de hardware; los individuos en su hogar eligen con frecuencia los firewalls de software.

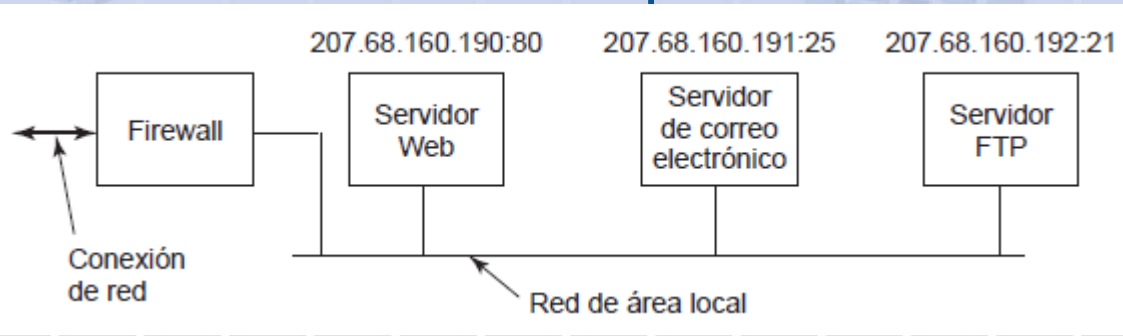
DEFENSAS

- Primero analizaremos los firewalls de hardware. En

la figura se ilustra un firewall de hardware genérico. Aquí, la conexión (cable o fibra óptica) del proveedor de red se conecta al firewall y no pueden entrar paquetes a la LAN ni salir de ella sin que el firewall lo apruebe.

- El propietario del firewall puede cambiar las reglas, por lo común a través de una interfaz Web
- El firewall más simple, el **firewall sin estado, se inspecciona el encabezado de cada paquete que pasa**

por él y se toma la decisión de aceptar o rechazar el paquete.



DEFENSAS

Los antivirus y las técnicas anti-antivirus

- En ese caso, la siguiente línea de defensa está compuesta por los programas antimalware, que se conocen comúnmente como **programas antivirus**.
- Las empresas de software antivirus tienen laboratorios de investigación. El primer paso es hacer que el virus infecte un programa, ***para obtener una copia del virus en su forma más pura. El siguiente paso es*** hacer un listado exacto del código del virus e introducirlo en la base de datos de virus conocidos. Las empresas compiten por el tamaño de sus bases de datos.
- El hecho de inventar nuevos virus sólo para aumentar el tamaño de la base de datos no se considera una buena conducta deportiva.

DEFENSAS

Comprobadores de integridad

La **comprobación de integridad es un método completamente distinto para detectar virus.**

Un programa antivirus que funciona de esta manera explora primero el disco duro en busca de virus.

Una vez que está convencido de que el disco está limpio, calcula una suma de comprobación para cada archivo ejecutable.

El algoritmo de sumas de comprobación podría ser algo tan simple como tratar a todas las palabras en el texto del programa como enteros de 32 o 64 bits y sumarlas,

Después escribe en un archivo llamado *sumacomp la lista de sumas de comprobación para todos los archivos relevantes en un directorio,* en ese directorio

DEFENSAS

Comprobadores del comportamiento

- La **comprobación del comportamiento es la tercera estrategia que utiliza el software antivirus. Con**
- Este método, el programa antivirus vive en memoria mientras que la computadora está funcionando, y atrapa por sí solo todas las llamadas al sistema. La idea es que pueda entonces monitorear toda la actividad y trate de atrapar algo que se vea sospechoso.
- Por ejemplo, ningún programa normal debería tratar de sobrescribir el sector de arranque, por lo que es casi un hecho que un intento por hacer esto se debe a un virus.

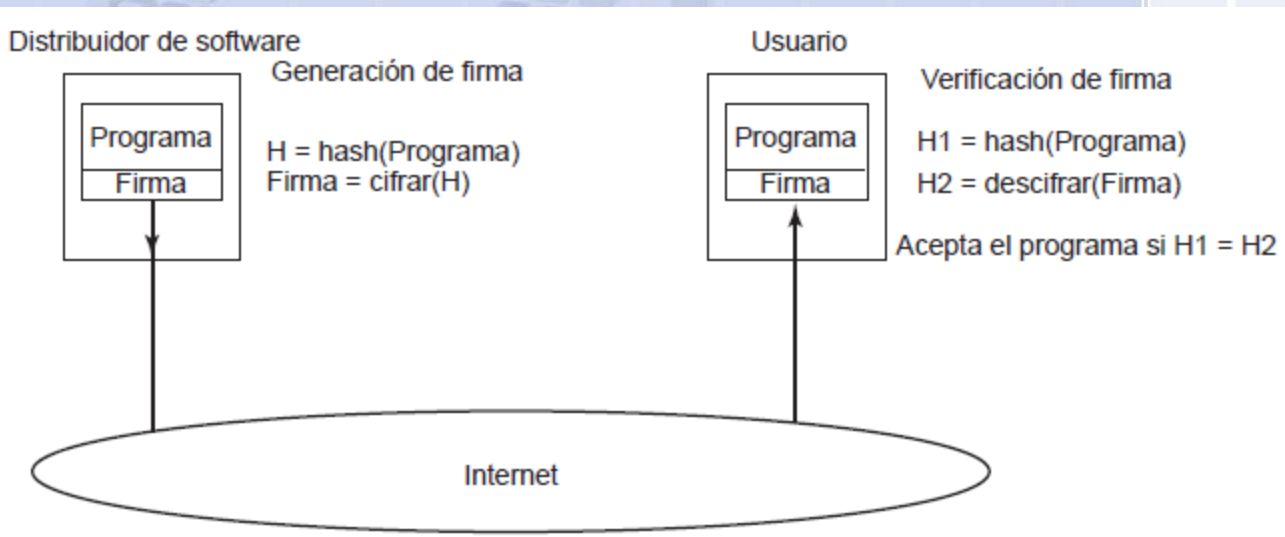
DEFENSAS

Firma de código

Un método completamente distinto de impedir que el malware entre en la computadora (recuerde:

defensa en profundidad) es ejecutar sólo el software de distribuidores confiables que no tenga modificaciones.

Un método que se utiliza con mucha frecuencia es la firma digital. La firma de código se basa en la criptografía de clave pública. Un distribuidor de software genera un par (clave pública, clave privada), y pone la primera clave a disposición del público



DEFENSAS

Detección de intrusos basada en modelos

Otro método para defender una máquina es instalar un **IDS *Intrusion Detection System (Sistema de detección de intrusos).***

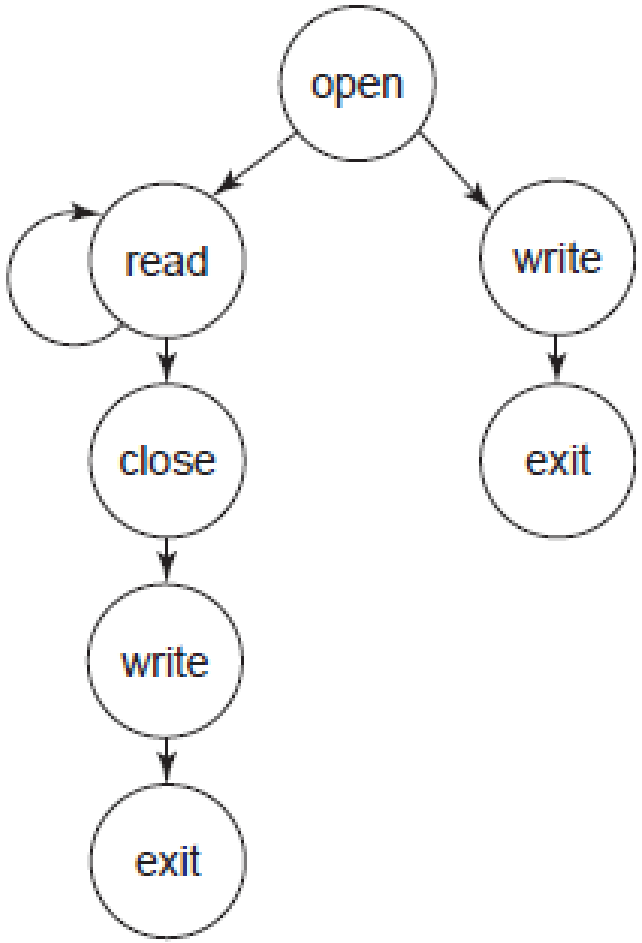
Hay dos tipos básicos de IDS: uno se concentra en inspeccionar los **paquetes de red entrantes** y el otro se enfoca **en buscar anomalías en la CPU**.

IDS: **detección de intrusos basada en modelos estáticos que se puede implementar mediante el uso de la técnica de encarcelamiento, como se muestra en la fig.**

DEFENSAS

```
int main(int argc *char argv[])
{
  int fd, n = 0;
  char buf[1];

  fd = open("datos", 0);
  if (fd < 0) {
      printf("Archivo de datos defectuoso\n");
      exit(1);
  } else {
      while (1) {
          read(fd, buf, 1);
          if (buf[0] == 0) {
            close(fd);
            printf("n = %d\n", n);
            exit(0);
          }
          n = n + 1;
      }
  }
}
```



DEFENSAS

Encapsulamiento de código móvil

En la actualidad cada vez hay más páginas Web que contienen pequeños programas conocidos como **applets.**

Cuando se descarga una página Web que contiene applets, el sistema obtiene estos applets y los ejecuta.

Los **agentes son otro ejemplo en el que los programas se envían de una máquina a otra para** ejecutarlos en la máquina de destino.

DEFENSAS

Seguridad de Java

Java es un lenguaje con seguridad de tipos, lo cual significa que el compilador rechazará cualquier

intento por utilizar una variable de una forma que no sea compatible con su tipo.

En Java, las construcciones que mezclan los tipos son prohibidos por la gramática.

Además, Java no tiene variables apuntadores, conversiones, asignación de almacenamiento controlada por el usuario (como *malloc y free) y todas las referencias a los arreglos se comprueban en tiempo* de ejecución.

Un applet compilado en forma apropiada las obedecerá de manera automática. Si el applet pasa todas las pruebas, se puede ejecutar de manera segura

bibliografía

Cap 9, tanembaum “Sistemas Operativos modernos