

20 SEGURIDAD

- 1 EL PROBLEMA DE LA SEGURIDAD
- 2 AUTENTIFICACIÓN
- 3 AMENAZAS A PROGRAMAS
- 4 AMENAZAS AL SISTEMA
- 5 MONITOREO DE AMENAZAS
- 6 ENCRIPCIÓN

20.1 EL PROBLEMA DE LA SEGURIDAD

- La seguridad debe considerar el ambiente externo del Sistema y protegerlo de:
 - Accesos no autorizados
 - Modificación o destrucción maliciosa
 - Introducción accidental de inconsistencia
- Es más fácil proteger contra mal uso accidental que malicioso

20.1 EL PROBLEMA DE LA SEGURIDAD

- Para proteger el sistema se deben tomar medidas en dos niveles:
 - Físico
 - Humano
- MS-DOS y Macintosh añadieron seguridad cuando ya estaba diseñado
- Windows NT se diseñó con funciones de seguridad desde el principio
- ¿qué opción es mejor?

20.1 EL PROBLEMA DE LA SEGURIDAD

- Considerar tipo de Intrusos al diseñar un sistema:
 - Curioso casual por parte de usuarios no técnicos
 - Husmeo por parte de personal interno
 - Intentos decididos por hacer dinero
 - Espionaje comercial o militar

20.1 EL PROBLEMA DE LA SEGURIDAD

- Pérdida accidental de datos:
 - **Actos fortuitos:** incendios, inundaciones, terremotos, guerras, motines o ratas que roen cintas
 - **Errores de HW o SW:** fallas de CPU, discos o cintas ilegibles, errores de telecomunicaciones, errores de programación
 - **Errores humanos:** captura incorrecta de datos, montaje de una cinta o disco equivocado, extravío de un disco o cinta etc.

20.2 AUTENTIFICACIÓN

- La identidad del usuario se establece muy a menudo mediante palabras secretas (passwords), las que pueden ser consideradas un caso especial de llaves o capacidades
- Se basa en:
 - Posesión del usuario (llave o tarjeta)
 - Conocimiento del usuario (id de usuario y contraseña)
 - Atributo del usuario (huella dactilar, patrón de retina, firma digital)

20.2 AUTENTIFICACIÓN

- Las passwords deben ser mantenidas en secreto.
 - Cambio frecuente de passwords.
 - Uso de passwords "no adivinables"
 - Registrar todos los intentos de acceso no válidos
- Contraseña de 4 dígitos -> programa que pruebe una contraseña cada milisegundo -> se demora 5 segundos en descubrirla
- Contraseñas más complejas, los usuarios las anotan-> riesgoso

20.2 AUTENTIFICACIÓN

- Envejecer las contraseñas: cambiar contraseña cada cierto período y conservar un historial

20.2 AUTENTIFICACIÓN

- Para provocar problemas en algún sistema primero se tiene que iniciar sesión en el sistema, por lo cual se debe pasar por el procedimiento de validación.
 - Hackers: grandes programadores
 - Crackers: personas que se introducen de manera indebida en los sistemas de computación

20.2 AUTENTIFICACIÓN

LOGIN: ken
PASSWORD: FooBar
SUCCESSFUL LOGIN

(a)

LOGIN: carol
INVALID LOGIN NAME
LOGIN:

(b)

LOGIN: carol
PASSWORD: Idunno
INVALID LOGIN
LOGIN:

(c)

- a) Inicio de sesión logrado
- b) Inicio de sesión rechazado después de introducir el nombre: entrega mayor información pues permite al cracker seguir probando nombres de inicio
- c) Inicio de sesión rechazado después de ingresar nombre y contraseña: siempre se pide la contraseña, no se da mayor información, sólo que inicio de sesión + contraseña no es correcta

Cómo se introducen al sistema los crackers

```
LBL> telnet elxsi
ELXSI AT LBL
LOGIN: root
PASSWORD: root
INCORRECT PASSWORD, TRY AGAIN
LOGIN: guest
PASSWORD: guest
INCORRECT PASSWORD, TRY AGAIN
LOGIN: uucp
PASSWORD: uucp
WELCOME TO THE ELXSI COMPUTER AT LBL
```

Forma en que un cracker se introdujo en una computadora del Departamento de Energía de EEUU en LBL,

- uucp: programa de copiado de Unix a Unix

Cómo se introducen al sistema los crackers

Se puede compilar cierta información:

- nombres de pila y apellidos, calles, ciudades, palabras de un diccionario, matrículas de automóvil.
- 1979 se hizo un estudio con esta información y el 86% correspondían a contraseñas guardadas en Unix
- 1997, 82% se encontraron coincidencias. En Unix se puede capturar el archivo de contraseñas (situado en /etc/passwd)

20.2 AUTENTIFICACIÓN

Bobbie, 4238, e(Dog4238)
Tony, 2918, e(6%%TaeFF2918)
Laura, 6902, e(Shakespeare6902)
Mark, 1694, e(XaB@Bwcz1694)
Deborah, 1092, e(LordByron,1092)

Sal Password

Seguridad de contraseñas en Unix: Uso de sal para frustrar el cálculo previo de contraseñas cifradas

- sal: número aleatorio de n bits

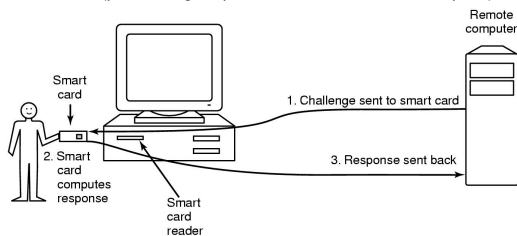
20.2 AUTENTIFICACIÓN

Cómo mejorar la seguridad de las contraseñas:

- deben tener como mínimo 7 caracteres
- deben contener letras tanto mayúsculas como minúsculas
- deben contener al menos un dígito o carácter especial
- no deben ser palabras de diccionarios, nombres de personas etc.

20.2 AUTENTIFICACIÓN empleando un objeto físico

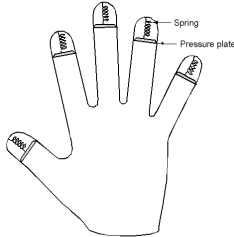
Tarjeta de plástico – franja magnética o chip - que se inserta en un lector enlazado con el computador. Además se utiliza una contraseña (pin de 4 dígitos para no colocar un teclado completo)



20.2 AUTENTIFICACIÓN por biometría

Mide características físicas del usuario que son difíciles de falsificar, esto se conoce como biometría, como un lector de huellas dactilares o de patrón de voz.

Dispositivo para medir la longitud de los dedos



20.2 AUTENTIFICACIÓN por biometría

Cualquier técnica que se basa en imágenes puede falsificarse.

- Huellas con moldes de cierto material
- fotografía retinal

Entonces se miden las pulsaciones de la retina

En el análisis de firmas

- Se compara con la original (falla)
- movimiento de la pluma y presión aplicada al firmar

20.3 AMENAZAS A PROGRAMAS

Goal	Threat
Data confidentiality	Exposure of data
Data integrity	Tampering with data
System availability	Denial of service

Metas de seguridad y sus amenazas

20.3 AMENAZAS A PROGRAMAS

- Caballo de Troya
 - Segmento de código que hace mal uso de su ambiente.
 - Explota mecanismos para permitirle a programas escritos por unos usuarios ser ejecutados por otros usuarios.
 - Ej: emular login -> ingresa contraseña primero muestra error
- > luego aparece la pantalla real
- Windows utiliza ctrl+alt+supr para cerrar sesión actual que pudiera ser no real

20.3 AMENAZAS A PROGRAMAS

- Bombas de lógica
 - Fragmento de código escrito por un programador de una empresa y colocado en el sistema operativo de producción
 - Chequea la contraseña en forma periódica de ese programador, si fuera despedido y ya no ingresa la pwd, entonces estalla la bomba (borrar disco, borrar archivos etc.)
 - En general se re-contrata al empleado como asesor

20.3 AMENAZAS A PROGRAMAS

- Puerta Trampa
 - Identificador de un usuario específico o password que burla los procedimientos normales de seguridad.
 - Podría estar inducido en un compilador.
 - Programadores arrestados por desfalco, haciendo que se abone en sus cuentas medio centavo a sus cuentas (gran cantidad de transacciones)
 - Difícil de detectar, revisar todo el código -> pruebas de caja blanca

20.3 AMENAZAS A PROGRAMAS

- Puerta Trampa: stramp: ver si nombre de sesión es "ZZZZZ". Se inicia sesión, sin importar la contraseña
- A) código normal de inicio de sesión
- B) código con una trampa insertada

```

while (TRUE) {
    printf("login: ");
    get_string(name);
    disable_echoing();
    printf("password: ");
    get_string(password);
    enable_echoing();
    v = check_validity(name, password);
    if (v) break;
}
execute_shell(name);
(a)

while (TRUE) {
    printf("login: ");
    get_string(name);
    disable_echoing();
    printf("password: ");
    get_string(password);
    enable_echoing();
    v = check_validity(name, password);
    if (v || strcmp(name, "zzzzz") == 0) break;
}
execute_shell(name);
(b)

```

Desbordamiento de Búfer

- La mayoría de los sistemas operativos están escritos en C.
- Falencia: ningún compilador de C verifica las cotas de los arreglos
- Ej:

```

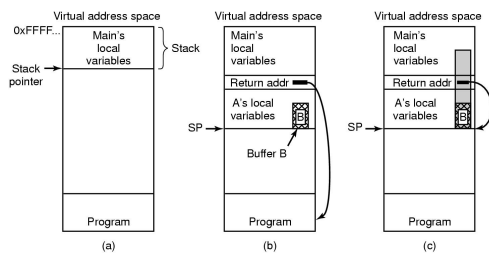
int i;
char c[1024];
i=12000;
C[i]=0;

```

10.976 bytes afuera del arreglo c -> se produce desbordamiento de buffer

Desbordamiento de Buffer

- A) ejecutando el programa principal
- B) después de invocar el procedimiento A
- C) el desbordamiento del buffer en gris



Ataques genéricos contra la seguridad

- Al diseñar un sistema, se debe comprobar que:
 - Solicitar páginas de memoria, espacio de disco o cintas (ver si tienen información relevante)
 - Probar llamadas al sistema no permitidas o permitidas pero con parámetros no permitidos
 - Iniciar sesión y luego apretar DEL, BREAK a la mitad de la secuencia de inicio de sesión
 - Buscar manuales que digan "no hagan X", probar tantas variaciones de X como se pueda
 - Ver si detecta los trucos en el código
 - etc

Principios de diseño que proporcionan seguridad

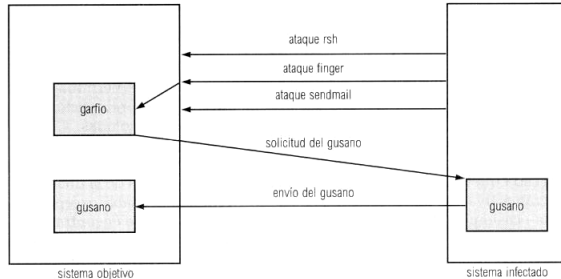
- El diseño del sistema debe ser público: significa que siempre se piense que puede ser conocido por los intrusos
- La política predeterminada debe ser no permitir el acceso
- Verificar la vigencia de la autorización
- Debe darse a cada proceso el mínimo de privilegios
- Mecanismo de protección debe ser sencillo, uniforme e incorporado a las capas más bajas del sistema
- El esquema debe ser aceptable y fácil de usar. Si cuesta mucho trabajo proteger información los usuarios no la resguardarán

20.4 AMENAZAS AL SISTEMA

- Worms (gusanos): usan como mecanismo de multiplicación programas standalone. Se copian a si mismos.
- Gusanos en la internet
 - Explotan las características de networking de UNIX (acceso remoto) y bugs en los programas finger y sendmail.
 - Programa "enganchado" subido a la red que arrastra al gusano.

20.4 AMENAZAS AL SISTEMA

Gusano (worm) de internet de Morris



20.4 AMENAZAS AL SISTEMA

- Virus : fragmento de código incrustado en un programa legítimo.
 - Principalmente afectan sistemas de microcomputadores.
 - Se "obtienen" bajando programas infectados desde sitios públicos o intercambiando disquetes infectados.
 - Computación "segura": comprar sw de fábrica, evitar copias.
 - Programa de una sola línea que solía paralizar cualquier sistema Unix:
 - `Main(){while(1) fork();}` // crea procesos hasta que se llena la tabla de procesos e impide que se inicie cualquier otro proceso

20.4 AMENAZAS AL SISTEMA

Tipos de virus:

1. Acompañantes
2. De programa ejecutable
3. De memoria
4. De sector de arranque
5. De controlador de dispositivo
6. De macro
7. De código fuente

20.4 AMENAZAS AL SISTEMA

1. Virus Acompañantes:

- No infecta en realidad un programa, pero se ejecuta cuando se supone que se debe ejecutar el programa.
- Ej: Usuario teclea prog en MS-DOS
 - ejecuta prog.com y luego prog.exe, por ende si se escribe un virus con nombre prog.com, ejecuta este y luego el verdadero prog.exe
- En windows ocurre con Inicio – Ejecutar. Un virus puede cambiar además el objetivo de un acceso directo y hacer que apunte al virus. Ejecuta el virus primero y luego el prog. Original
- ¿es buena esta característica para un virus?

20.4 AMENAZAS AL SISTEMA

2. Virus de programa ejecutable:

- Infectan programas ejecutables
- Los más sencillos sobrescriben el programa ejecutable con su propio código -> virus de sobre-escritura
- Ejemplo de lógica:

20.4 AMENAZAS AL SISTEMA

Procedimiento
recurrente que
encuentra
archivos
ejecutables en un
sistema Unix

```
#include <sys/types.h>          /* standard POSIX headers */
#include <sys/stat.h>
#include <dirent.h>
#include <fcntl.h>
#include <unistd.h>
struct stat sbuf;                /* for lstat call to see if file is sym link */

search(char *dir_name)
{
    DIR *dirp;
    struct dirent *dp;

    dirp = opendir(dir_name);
    if (dirp == NULL) return;
    while (TRUE) {
        dp = readdir(dirp);
        if (dp == NULL) {
            chdir(".");
            break;
        }
        if (dp->d_name[0] == '.') continue; /* skip the . and . directories */
        lstat(dp->d_name, &sbuf);
        if (S_ISLNK(sbuf.st_mode)) continue; /* is entry a symbolic link? */
        if (chdir(dp->d_name) == 0) { /* skip symbolic links */
            search("."); /* if chdir succeeds, it must be a dir */
            /* yes, enter and search it */
        } else { /* no (file), infect it */
            if (access(dp->d_name, X_OK) == 0) /* if executable, infect it */
                infect(dp->d_name);
        }
        closedir(dirp); /* dir processed; close and return */
    }
}
```

20.4 AMENAZAS AL SISTEMA

Virus de programa ejecutable:

- El programa principal de este virus primero copiaría su programa binario en un arreglo.
- Recorre todo el sistema comenzando por el directorio raíz, e invocando a *search* con el directorio raíz como parámetro
- Procedimiento recurrente *search* procesa un directorio abriéndolo y leyendo las entradas una por una con *readdir* hasta obtener un NULL.
- Si encuentra un archivo ejecutable, invoca a *infect* con el nombre del archivo a infectar como parámetro
- Infect es el procedimiento de infección real que obviamente no se muestra, abre el archivo nombrado en su parámetro, copia sobre el archivo el virus que se guardó en el arreglo y luego cierra el archivo.

20.4 AMENAZAS AL SISTEMA

Virus de programa ejecutable:

¿Se puede mejorar?

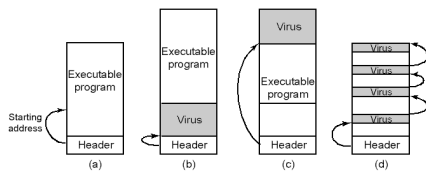
- Infect genere un número aleatorio -> en una invocación de 128 habría infección. Ventaja: reduce la posibilidad de una pronta detección, el objetivo es que tenga tiempo de extenderse. (propiedad de los virus reales biológicos)
- Verificar si el archivo ya está infectado.
- No cambiar la hora de la última modificación del archivo ni el tamaño del mismo (ocultar infección)
- El programa completo ocupa menos de una página en C y el segmento de texto ocupa menos de 2kb después de compilarlo
- ¿Para qué sirve saber esto?

20.4 AMENAZAS AL SISTEMA

Virus de programa ejecutable:

- Problema de virus de sobre.escritura -> fácil de detectar, pues se ejecuta el prog. Infectado, disemina el virus pero el programa no funciona correctamente.
- La solución es anexar el virus al programa y dejar que el programa funcione correctamente para permitir que se extienda. Esto se denominan **Virus Parásitos**.
- Se pueden anexar al principio, al final o en una parte intermedia del prog. ejecutable.

20.4 AMENAZAS AL SISTEMA



- a) Programa ejecutable
- b) Con un virus al principio (RAM)
- c) Con un virus al final (lo más común)
- d) Con un virus repartido en el espacio libre dentro del programa (complejo)

20.4 AMENAZAS AL SISTEMA

3. Virus residentes en la memoria:

- Se ha visto que se ejecuta prog. Infectado -> virus se ejecuta.
- En cambio un virus residente en memoria permanece en la memoria todo el tiempo
- Captura uno de los vectores de interrupción copiando su contenido en una variable temporal y colocando ahí su propia dirección, dirigiendo esa interrupción a su propio código
- Cuando ocurre una llamada al sistema entonces el virus se ejecuta en modo kernel

20.4 AMENAZAS AL SISTEMA

4. Virus de sector de arranque:

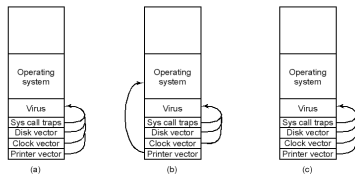
- Bios -> lee registro de arranque maestro del ppio. Del disco de arranque, lo coloca en RAM y lo ejecuta -> lee el primer sector de arranque de la partición activa y lo ejecuta -> se carga el S.O.
- Se sobre-escribe el registro de arranque maestro (o MBR , master boot record)
- Se copia el verdadero sector de arranque en un lugar seguro en el disco para poder arrancar el S.O.
- En windows el fdisk se salta la primera pista del sector de arranque (buen lugar para ocultarse)
- Otra opción es actualizar la lista de sectores defectuosos y esconderlo entonces como sector defectuoso

20.4 AMENAZAS AL SISTEMA

4. Virus de sector de arranque:

- Windows carga los controladores uno por uno y cada uno captura el vector de interrupción que necesita. Esto tarda unos segundos.
- Esto proporciona el tiempo necesario al virus para capturar todos los vectores de interrupción

20.4 AMENAZAS AL SISTEMA



- a) Después de que el virus ha capturado todos los vectores de interrupción
- b) Después de que el S.O. ha recuperado el vector de interrupción de la impresora
- c) Después de que el virus se ha percatado de la pérdida del vector de interrupción de impresora y lo ha recapturado
- ¿queda residente en memoria?

20.4 AMENAZAS AL SISTEMA

5. Virus de controlador de dispositivo:

- En Windows y algunos sistemas Unix los controladores de dispositivos son programas ejecutables que viven en el disco y se cargan en el momento del arranque
- Los controladores se ejecutan en modo kernel.

20.4 AMENAZAS AL SISTEMA

6. Virus de macros:

- Word y Excel permiten a los usuarios escribir macros para agrupar varias comandos que después pueden ejecutarse con una sola tecla.
- Las macros son programas completos en Visual Basic, Office almacena las macros para cada documentos junto con el doc.
- Se crea una macro y se anexa a la fn Abrir_Archivo. La macro puede infectar otros doc., borrar archivos etc.
- Son fáciles de escribir, pues no se requieren tantas habilidades técnicas

20.4 AMENAZAS AL SISTEMA

6. Virus de código fuente:

- Son los virus más portables
- Buscar programas en C , cuando se ejecuta el programa, se invoca el virus.
- Al procedimiento *infect* se inserta la línea:
 - `#include <virus.h>`al principio de cada programa fuente en C, se agrega:
 - `Run_virus( );`Que activa al virus

20.4 AMENAZAS AL SISTEMA

Cómo se diseminan los virus:

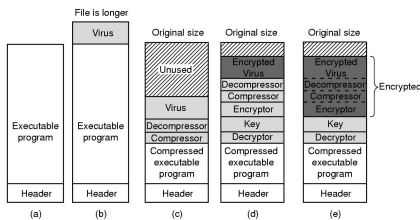
- Colocando el Virus donde probablemente puede ser copiado (página web: shareware)
- Infectando programas sobre el disco duro o el disquete
- Puede tratar de extenderse sobre una LAN (una org. En particular)
- Atachado al correo electrónico: cuando se abre, la lista de direcciones se utiliza para reproducir el virus

20.4 AMENAZAS AL SISTEMA

Técnicas antivirus y anti-antivirus:

- SW antivirus
 - Se hace que el virus infecte un programa que no hace nada -> archivo señuelo, para obtener una copia del virus en su forma más pura
 - Se produce un listado exacto del código del virus para introducirlo en la BD de virus conocidos
 - Cuando el antivirus ya está instalado, examina todos los archivos ejecutables del disco en busca de virus que estén en la BD, también verifica las longitudes de los archivos desde su última revisión
 - Como los virus tienen variantes las búsquedas no pueden ser exactas y se necesitan búsquedas difusas

20.4 AMENAZAS AL SISTEMA



- Un programa
- Un programa infectado
- Un programa infectado comprimido (mantener la longitud)
- Un virus cifrado o encriptado (código no es igual al de la BD del antivirus)
- Un virus comprimido con código de compresión cifrado

20.4 AMENAZAS AL SISTEMA

MOV A,R1	MOV A,R1	MOV A,R1	MOV A,R1	MOV A,R1
ADD B,R1	NOP	ADD #0,R1	OR R1,R1	TST R1
ADD C,R1	ADD B,R1	ADD B,R1	ADD B,R1	ADD C,R1
SUB #4,R1	NOP	OR R1,R1	MOV R1,R5	MOV R1,R5
MOV R1,X	ADD C,R1	ADD C,R1	ADD C,R1	ADD B,R1
	NOP	SHL #0,R1	SHL R1,0	CMP R2,R5
	SUB #4,R1	SUB #4,R1	SUB #4,R1	SUB #4,R1
	NOP	JMP .+1	ADD R5,R5	JMP .+1
	MOV R1,X	MOV R1,X	MOV R1,X	MOV R1,X
			MOV R5,Y	MOV R5,Y

(a) (b) (c) (d) (e)

Ejemplo de **virus polimórfico** (virus que sufre una mutación cada vez que se copia), se quiere hacer ; $x=(A+B+C-4)$, que es el proc. De descifrado:

- Código original
- Agrega NOP
- Agrega ADD, OR, SHL, JMP
- Agrega registro R5 que no se necesita en este código
- Intercambiar instrucciones sin alterar lo que hace el programa

20.4 AMENAZAS AL SISTEMA

Verificadores de integridad:

- Busca virus en el disco duro
- Si está limpio, calcula una suma de verificación para cada uno de los programas ejecutables y escribe la lista de sumas para todos los archivos de un directorio en un archivo -> checksum
- La siguiente vez calculará todas las sumas de verificación y comprobará que coincidan
- ¿Qué pasa si el virus borra el archivo con las sumas de verificación?
- Se encripta a través de una tarjeta inteligente con una clave

20.4 AMENAZAS AL SISTEMA

Verificadores de comportamiento:

- El programa antivirus reside en la memoria mientras el computador está operando y captura él mismo todas las llamadas al sistema

20.4 AMENAZAS AL SISTEMA

Cómo evitar virus:

- Usuarios:
 - 1) Escoger un S.O. que diferencie sesiones para cada usuario y para el administrador del sistema
 - 2) Instalar SW comprado con caja sellada
 - 3) Comprar un buen antivirus y obtener sus actualizaciones del sitio web del fabricante
 - 4) Tener cuidado con los datos adjuntos en los email (en especial con el tipo de extensión)
 - 5) Respalidar con cierta frecuencia en medios externos

20.4 AMENAZAS AL SISTEMA

Cómo evitar virus:

- Industria:
 - 1) Hacer S.O. Sencillos. Más funciones -> más fallas
 - 2) Proteger en forma selectiva contra escritura ciertos cilindros del disco para evitar que se infecten
 - 3) ROM tipo flash (que se usa para BIOS y la memoria CMOS), no permitir su modificación.

20.4 AMENAZAS AL SISTEMA

- Recuperación después de un ataque viral
 - 1) Apagar equipo
 - 2) Ejecutar S.O. desde un dispositivo protegido contra escritura
 - 3) Ejecutar antivirus de CD original, pues también puede estar infectado
 - 4) Archivos ASCII y JPEG no contienen virus
 - 5) Archivos que podrían contener virus, deben convertirse a otro formato como texto ASCII plano, guardarse en un medio externo
 - 6) Reformatar el disco
 - 7) Reinstalar S.O.

20.5 MONITOREO DE AMENAZAS

- Chequear patrones sospechosos de actividad: i.e., varios intentos incorrectos de ingreso de password pueden implicar un intento de adivinación de password.
- Registro de auditoría: registra la hora, usuario y tipo de todos los accesos a un objeto; es útil para recuperarse de una violación de seguridad y para desarrollar mejores medidas de seguridad.
- Revisar el sistema periódicamente, buscando baches en la seguridad; se hace cuando el computador está siendo poco usado.

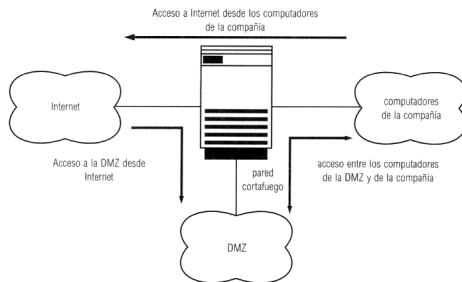
20.5 MONITOREO DE AMENAZAS

•Revisar:

- Passwords breves o fáciles de adivinar
- Programas no autorizados
- Programas no autorizados en direcciones del sistema
- Procesos de inesperada larga duración.
- Protecciones impropias de directorio.
- Protecciones impropias en archivos de datos del sistema
- Entradas peligrosas en la ruta de búsqueda de programas (Caballo de Troya).
- Cambios a programas del sistema; monitorear valores de checksum (sumas de verificación, sector de booteo)

20.5 MONITOREO DE AMENAZAS

¿cómo pueden conectarse computadores confiables sin peligro a una red no confiable? Seguridad en redes a través de separación de dominios vía cortafuegos (firewall)



20.6 ENCRIPCIÓN

- Encriptar texto legible en texto cifrado.
- Propiedades de una buena técnica de encriptación:
 - Relativamente simple para usuarios autorizados encriptar y desencriptar datos.
 - El esquema de encriptación no depende de lo secreto que sea el algoritmo si no de los parámetros, denominados clave de encriptación.
 - Es extremadamente difícil para un intruso determinar la clave de encriptación.

20.6 ENCRIPCIÓN

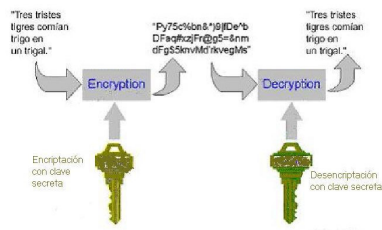
- Los estándares de encriptación de datos (DES) sustituyen caracteres y los reordenan sobre la base de una clave de encriptación proporcionada a usuarios autorizados vía mecanismos seguros. El esquema es tan seguro como el mecanismo.
- Encriptación con clave pública, basada en que cada usuario tiene dos claves:
 - Clave pública – usada para encriptar datos.
 - Clave privada – usada para desencriptar datos, conocida sólo por usuarios específicos.
 - Ambos se comunican conociendo la clave pública del otro

20.6 ENCRIPCIÓN

- Debe ser un esquema de encriptación que pueda ser hecho público sin revelar cómo es el esquema de desencriptación.
 - Algoritmo eficiente para probar si un número es primo o no.
 - No se conoce un algoritmo eficiente para encontrar los factores primos de un número.
- En términos generales, existen tres tipos de algoritmos de criptografía:
 - Clave secreta (simétricas)
 - Clave pública (asimétricas)
 - hashing

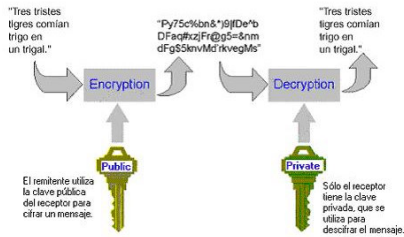
20.6 ENCRIPCIÓN

- Criptografía con claves secretas: encriptación simétrica



20.6 ENCRIPCIÓN

- Criptografía con claves públicas: encriptación asimétrica



20.6 ENCRIPCIÓN

- El tercer tipo se denomina Hash o Message Digest y a diferencia de los anteriores no utiliza técnicas de llave
- Se mapean mensajes grandes en números pequeños de largo fijo. La función hash genera checksum criptográfico sobre el mensaje para proteger su integridad.
- Es virtualmente imposible saber que mensaje lo generó, es imposible computacionalmente encontrar dos mensajes que generen el mismo checksum.

20.6 ENCRIPCIÓN

- Sea:
 - E: algoritmo general de encriptación
 - D: Algoritmo general de desencriptación
 - k: clave de encriptación
 - m: mensaje
- Se debe cumplir que:
 - $Dk(Ek(m)) = m$
 - Ek y Dk se calculen de forma eficiente
 - Si se publica E, no hay forma de calcular D
 - La seguridad debe depender de k y no de los algoritmos E y D

20.6 ENCRIPCIÓN

Algoritmo RSA (Rivest, Shamir y Adleman)

- Clave pública: $\langle E, n \rangle$
- Clave privada: $\langle D, n \rangle$
- E, D, n : enteros positivos
- Cada mensaje se representa como un entero entre 0 y $n-1$ (mensajes largos se descomponen en mensajes cortos. Cada uno se representa por un entero)
- Se definen E y D como:
 - $E(m) = m^E \bmod n = C$
 - $D(C) = C^D \bmod n = m$

20.6 ENCRIPCIÓN

Propiedades del algoritmo:

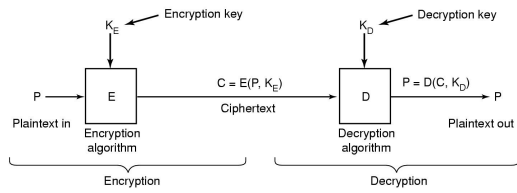
- E : llave pública para codificar
- D : llave privada para decodificar
- m : mensaje
- Se debe cumplir que:
 1. $D(E(m)) = m$
 2. E y D se calculan de forma fácil
 3. Si se publica E , no hay forma de calcular D
 4. $E(D(m)) = m$

20.6 ENCRIPCIÓN

¿cómo se obtiene D y E ?

- Hay que obtener p y q : números enteros, primos y muy grandes
- Se calculan $n = p \cdot q$
- Se elige $E < n$ tal que E y $((p-1) \cdot (q-1)) = 1$
- Se calcula d tal que $E \cdot D \bmod ((p-1) \cdot (q-1)) = 1$
- La llave pública es $\langle E, n \rangle$
- La llave privada es $\langle D, n \rangle$

20.6 ENCRIPCIÓN



Relación entre el texto simple y el texto cifrado

20.6 ENCRIPCIÓN

Codificación RSA

- Dado un texto, se separa en grupo de caracteres de tamaño fijo y se transforman a números enteros
- Por ejemplo:
Alra cadabra ABRA CADABRA Alra Cadabra
x1 x2 x3
- x_i es entero tal que $x_i < n-1$. Desde x_i debe ser posible volver al string

20.6 ENCRIPCIÓN

- Codificar:
 $(x_i^E) \bmod n = y_i$
- Decodificar:
 $(y_i^D) \bmod n = x_i$

Ejemplo:

- Sea $p = 2$ y $q = 5$ (números primos)
- Sea $n = 10$
- $(p-1) \cdot (q-1) = 4$
- $E < n$, $E = 3$
- Cálculo de D : $E \cdot D \bmod ((p-1) \cdot (q-1)) = 1$

20.6 ENCRIPCIÓN

- Cálculo de D: $E \cdot D \bmod ((p-1) \cdot (q-1)) = 1$
- O sea, $E \cdot D \bmod 4 = 1$, $E \cdot D = 4k + 1$
- $E \cdot D = 4k + 1$ ¿Existe k?
- $3 \cdot D = 4k + 1$, $D = (4k + 1)/3$. Con $k=5$, $D=7$
- $21 \bmod 4 = 1$ (funciona)
- Sea $x=2$ el número a encriptar $2^E = 8 \bmod 10 = 8$, $y=8$
- Ahora desencriptaremos. $8^D = 2097152 \bmod 10 = 2$
- (funciona)

20.6 ENCRIPCIÓN

RSA-129 and Its Factors

1143816257578888676692357799761466120102182967212423625625618429357\
06935245733897830597123563958705058989075147599290026879543541
=
3490529510847650949147849619903898133417764638493387843990820577
x
32769132993266709549961988190834461413177642967992942539798288533

20.6 ENCRIPCIÓN

¿cómo se eligen las claves?

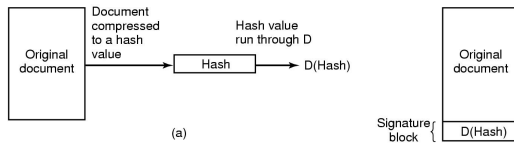
- La seguridad depende del número de bits:
 - Netscape y explorer usan RSA con 48 bits
 - Mejor seguridad se obtiene con 128 y 1024 bits, pero el cálculo se vuelve complejo
- ¿Se puede quebrar RSA?
 - En 1977 se lanzó el reto de quebrar un string de 129 dígitos (430 bits), se pensaba que era inviolable pues los algoritmos de factorización de grandes números estimaban 40 cuatrillones de años de cómputo
 - En abril de 1994, 4 científicos (en realidad fueron app.600 personas) reportaron que habían quebrado el código, el mensaje era: THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE.
 - Se usaron 1600 computadores

20.6 ENCRIPCIÓN

- La premisa es que factorizar números grandes es una tarea computacionalmente compleja
- RSA se usa para encriptar mensajes pequeños: números de tarjetas de crédito, password etc.

Firmas digitales

Firmar mensajes de email y otros doc. digitales de modo tal que no puedan ser negados después por quien los envió



a) Cálculo de un bloque de firma b) Lo que recibe el destinatario

Clasificación de seguridad

- Seguridad de libro naranja – Defensa de EEUU 1985
- X: indica que hay requisitos nuevos en ese pto.
- ->: indica que los requisitos de la categoría inmediata inferior también se aplican en ese pto.

Criterion	D	C1	C2	B1	B2	B3	A1
Security policy							
Discretionary access control		X	X	→	→	X	→
Object reuse			X	→	→	→	→
Labels				X	X	→	→
Label integrity				X	→	→	→
Exportation of labeled information				X	→	→	→
Labeling human readable output				X	→	→	→
Mandatory access control				X	X	→	→
Subject sensitivity labels					X	→	→
Device labels					X	→	→
Accountability							
Identification and authentication		X	X	X	→	→	→
Audit			X	X	X	X	→
Trusted path					X	X	→

Clasificación de seguridad

- Seguridad de libro naranja – Defensa de EEUU 1985

Assurance						
System architecture	X	X	X	X	X	→
System integrity	X	→	→	→	→	→
Security testing	X	X	X	X	X	X
Design specification and verification			X	X	X	X
Covert channel analysis				X	X	X
Trusted facility management				X	X	→
Configuration management				X	→	X
Trusted recovery					X	→
Trusted distribution						X
Documentation						
Security features user's guide	X	→	→	→	→	→
Trusted facility manual	X	X	X	X	X	→
Test documentation	X	→	→	X	→	X
Design documentation	X	→	X	X	X	X

Clasificación de Seguridad

- Existen 4 divisiones para clasificar sistemas acorde a su seguridad:
- A (protección máxima)
- B,C,D (protección mínima)
- MS-DCS, Windows 95/98/Me están en la clasificación D

Clasificación de Seguridad

- Nivel C tiene 2 niveles:
 - C1: protege información privada y resguarda a los demás usuarios de lectura o destrucción accidental de información. Ej: implementaciones de Unix
 - C2: agrega a C1 de un control de acceso a nivel individual. Existen versiones seguras de Unix certificadas como C2. W/NT se puede configurar a C2.

Clasificación de Seguridad

- Nivel B tiene todo lo de C2 y agrega rótulos de sensibilidad a cada objeto. Se clasifican en B1, B2, B3.
- El nivel más alto es A. Al nivel B3 agrega especificaciones de diseño y verificación formal de seguridad
- En la actualidad no hay más de dos sistemas clase B y ninguno clase A

Sistemas de Confianza

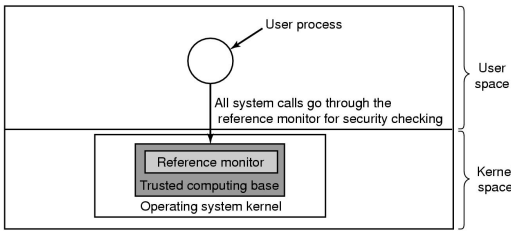
- ¿es posible construir un sistema de computación seguro?
- ¿por qué no se hace?
- Los sistemas actuales no son seguros pero los usuarios no están dispuestos a deshacerse de ellos
- La única forma de construir sistemas seguros es que sean sencillos, es decir que contengan pocas funciones. Más funciones -> más complejidad -> más código -> más errores de prog. -> más fallas de seguridad

Sistemas de Confianza

Base de Cómputo de confianza (TCB)

- Consiste en el HW y SW necesarios para cumplir todas las reglas en materia de seguridad
- Consta de:
 - HW
 - Partición del kernel y prog. De usuario que tienen facultades de superusuario: creación de procesos, conmutación de procesos, administración del mapa de memoria, administración de archivos y de E/S.

Sistemas de Confianza



Una parte importante de la TCB es el monitor de referencias

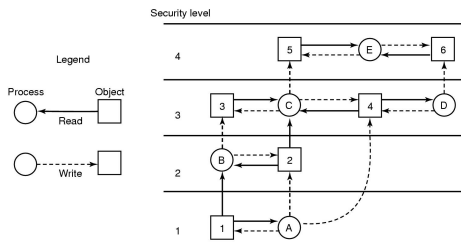
En general los S.O. no incluyen TCB, por eso son poco seguros

Sistemas de Confianza

Seguridad multinivel

- Modelo Bell-La Padula (1973)
- Propiedad de seguridad simple: un proceso que se ejecuta en el nivel de seguridad K sólo puede leer objetos de su nivel o de niveles inferiores (leer hacia abajo)
- Propiedad *: un proceso que se ejecuta en el nivel de seguridad K puede escribir sólo objetos en su nivel o en niveles superiores (escribir hacia arriba)

Sistemas de Confianza



Modelo de seguridad Bell-La Padula

¿se garantiza la integridad de los datos al escribir hacia arriba?

Sistemas de Confianza

- Modelo Biba
- Para garantizar la integridad de los datos se necesita lo siguiente:
 - Principio de integridad simple: un proceso que se ejecuta en el nivel de seguridad K sólo puede escribir objetos de su nivel o de un nivel inferior (escribir hacia abajo)
 - Propiedad de integridad *: un proceso que se ejecuta en el nivel de seguridad K sólo puede leer objetos de su nivel o de un nivel superior (leer hacia arriba)

Sistemas de Confianza



- Primera imagen 1024 por 768 píxeles
- Cada píxel consta de 8 bits para indicar rojo, verde y azul de ese píxel - RGB
- Cada píxel tiene 3 bits libres (1 por cada color)
- Entonces se tienen $1024 * 768 * 3 = 2,359,296$ bytes de información secreta que podría guardarse
- Segunda imagen contiene texto completo de: "Hamlet", "El rey Lear", "Macbeth", "El mercader de Venecia" y "Julio César" de Shakespeare. (274 kb) – Estenografía (se aplica un algoritmo especial de descompresión)

Sistemas de Confianza

- La demostración se puede ver:
 - www.cs.vu.nl/~ast/ (covered writing) al final de la página
 - Stenography demo: Windows 98/Me/NT/2000
