

Protección y seguridad

Fernando Schapachnik

Departamento de Computación, FCEyN,
Universidad de Buenos Aires, Buenos Aires, Argentina

Sistemas Operativos, primer cuatrimestre de 2018

(2) Créditos

Incluye aportes de Rodolfo Baader.

(3) No confundir gordura con hinchazón

- Protección:
 - Se trata de los mecanismos para asegurarse de que nadie pueda meter los garfios en los datos del otro.
 - Qué usuario puede hacer cada cosa.
- Seguridad:
 - Se trata de asegurarse que quien dice ser cierto usuario, lo sea.
 - También se trata de impedir la destrucción o adulteración de los datos.
- La distinción puede ser un poco tirada de los pelos, así que nosotros no la vamos a hacer.
- Pero en algunos ambientes académicos se considera válida.

(4) Seguridad de la información

- Definición más moderna: La seguridad de la información se entiende como la preservación de las siguientes características:
 - Confidencialidad
 - Integridad
 - Disponibilidad
- Seguridad de la Información \neq Seguridad Informática

(5) Los protagonistas

- Los sistemas de seguridad suelen tener:
 - Sujetos.
 - Objetos.
 - Acciones.
- La idea es decir qué sujetos pueden realizar qué acciones sobre qué objetos.
- Importante: los roles de sujeto y objeto no son excluyentes. Caso típico: los procesos.
- Veamos algunos ejemplos.

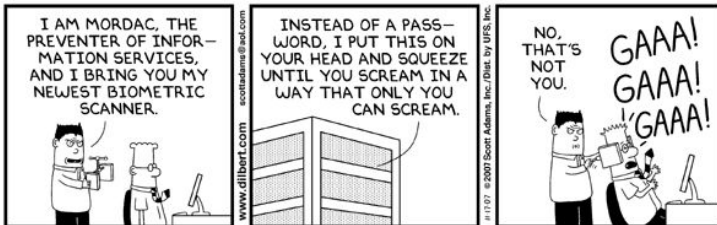
(6) Los protagonistas (cont.)

- La abstracción más común es la de usuario.
- Un usuario es un sujeto del SO, que pueden ejecutar acciones y que a veces es dueño de cosas.
- ¿De qué cosas? Los objetos: archivos, procesos, memoria, conexiones, puertos, etc.
- ¿Y qué puede hacer? Leer, escribir, copiar, abrir, borrar, imprimir, ejecutar, matar (un proceso), etc.
- Es muy común que los usuarios se agrupen en, justamente, grupos: colecciones de usuarios.
- Los grupos también son sujetos del sistema de permisos.
- También se puede usar otra abstracción: los roles. A un usuario se le asignan roles. Los roles son los que pueden o no hacer cosas. Ejemplo: operador, usuario común, último orejón del tarro, administrador.

(7) Empecemos por el principio

- Tristezas de un doble A+1:
 - Authentication.
 - Authorization.
 - Accounting.
- Autenticación: ¿sos quién decís ser? Algo que sé, algo que tengo, algo que soy. Múltiples factores. Contraseñas, medios biométricos. Fuerte uso de la criptografía.
- Autorización: qué podés hacer.
- Auditoría (a veces llamado contabilidad): dejo registrado qué hiciste.
- Las funciones de autenticación y autorización suelen estar claramente diferenciadas.

(8) Acceso biométrico



© Scott Adams, Inc./Dist. by UFS, Inc.

(9) Algo sobre Cripto

- Criptografía (escritura oculta): Rama de las matemáticas y de la informática que se ocupa de cifrar/descifrar información utilizando métodos y técnicas que permitan el intercambio de mensajes de manera que sólo puedan ser leídas por las personas a quienes van dirigidos.
- El criptoanálisis es el estudio de los métodos que se utilizan para quebrar textos cifrados con objeto de recuperar la información original en ausencia de la clave.
- No vamos a dar muchos detalles, pero sí algunos rudimentos.
- Algoritmos de encriptación *simétricos* son aquellos que utilizan la misma clave para encriptar y para desencriptar. Ejemplos: Caesar, DES, Blowfish, AES.
- Algoritmos *asimétricos* usan claves distintas, y constituyeron un gran avance científico. El más famoso: RSA.
- Funciones de hash one-way. MD5, SHA1, SHA-256, etc.

(10) Funciones de hash

- Ya saben qué es una función de hash.
- En cripto se utilizan hashes *de una vía*.
- Es decir, la idea es que sea prácticamente imposible obtener la preimagen.
- Se suele pedir que cumplan con:
 - Resistencia a la preimagen. Dado h debería ser difícil encontrar un m tal que $h = \text{hash}(m)$.
 - Resistencia a la segunda preimagen. Dado m_1 debería ser difícil encontrar un $m_2 \neq m_1$ tal que $\text{hash}(m_1) = \text{hash}(m_2)$.
 - Etc.
- Muy útiles para almacenar contraseñas (conviene que las contraseñas no se puedan leer). Importante: Debo usar SALT e iterar varias veces porque las funciones de hash son muy rápidas

(11) El método RSA

- Autores: Ronald Rivest, Adi Shamir, y Len Adleman.
- Se toman dos números de muchos, muchos dígitos.
- A uno se lo denomina clave pública, al otro clave privada. Cada persona necesita su clave privada (que protege) y su clave pública (que difunde).
- Para encriptar un mensaje, interpreto cada letra como si fuera un número, y hago una cuentita que involucra la clave pública del receptor.
- Para descifrarlo es necesaria la clave privada, y hacer otra cuentita.

(12) Algo de detalle sobre RSA

- Tomemos p y q primos (de 200 dígitos aprox).
- Multipliquémoslos: $n = pq$.
- Calculemos también: $n' = (p - 1)(q - 1)$.
- Elijámos un entero e que esté entre 2 y $n' - 1$ y que sea *coprimo* con n' .
- ¿Qué era ser coprimo? Significa no tener factores comunes.
- e y n van a ser nuestra clave de encriptación (pública).
- Computamos d para que cumpla que el resto de $d.e/n' = 1$ (es fácil de hacer).
- d y n van a ser nuestra clave de desencriptación (privada).

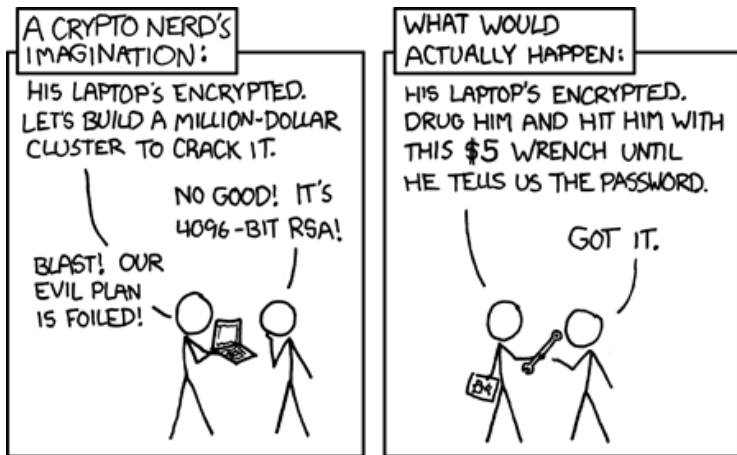
(13) Más detalles sobre RSA

- ¿Cómo encripto? Para cada letra m calculo el resto de dividir m^e por n .
- ¿Cómo desencripto? Para cada letra encriptada c calculo el resto de dividir c^d por n .
- Lo bueno es que la clave pública la puedo publicar en el diario. El método funciona porque factorizar es muy difícil (NP), aún para las computadoras más potentes.
- Si se pudiese factorizar fácilmente, el método no serviría.

(14) Firma digital con RSA

- Firma digital: calculo un hash del documento.
- Encripto con mi clave privada el hash.
- Entrego el documento + el hash encriptado.
- El receptor lo desencripta con mi clave pública. Si lo puede desencriptar exitosamente se asegura de que yo sea el autor.
- Luego verifica que el hash así obtenido se corresponda con el documento.
- Además de la cuestión técnica, no hay que descuidar el marco legal que le da validez (Ley 25.506)

(15) Sin embargo...



"Security", Randall Munroe, <http://xkcd.com/538/>

(16) Algo sobre autenticación remota con hash

- Existe un ataque llamado “replay-attack”.
- Las funciones de hash no lo impiden.
- Para eso se utilizan métodos basados en *Challenge-Response*.
 - El servidor elige un número al azar, que comunica al cliente.
 - El cliente tiene que encriptar la contraseña utilizando ese número como semilla.
 - El servidor hace lo mismo y se fija si coinciden.
 - Hecha la ley, hecha la trampa. Esto también puede ser atacado.

(17) Representando permisos

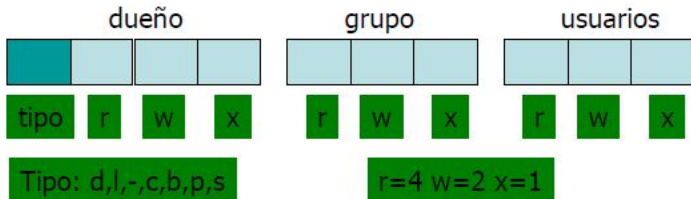
- La forma más sencilla de concebir a la autorización es como una matriz de control de accesos.
- Una matriz de Sujetos \times Objetos. En las celdas figuran las acciones permitidas.
- Detalle de implementación: se puede almacenar como una matriz centralizada, o separada por filas o columnas. Los archivos suelen guardar qué puede hacer cada usuario con ellos.
- Todo lo que no está dicho no se puede hacer.
- Principio muy común: *mínimo privilegio*.
- Sin embargo, ¿qué pasa cuando se crea un objeto nuevo?
- Se suelen definir unos permisos por defecto. En general están dados por el tipo de objeto.

(18) DAC vs. MAC

- Este esquema se llama: *Discretionary Access Control*.
- La idea es que los atributos de seguridad se tienen que definir explícitamente. El dueño decide los permisos.
- Otro esquema posible es *MAC: Mandatory Access Control*.
- Se lo utiliza para manejar información altamente sensible.
- Cada sujeto tiene un grado.
- Los objetos creados heredan el grado del último sujeto que los modificó.
- Un sujeto sólo puede acceder a objetos de grado menor o igual que el de él.
- Ejemplo real, más sofisticado: Modelo Bell-Lapadula.

(19) DAC en UNIX

- Permisos básicos en UNIX:



```
192.168.234.139 - PuTTY
-rw-r--r-- 1 root shadow  928 Feb 14 08:23 shadow
-rw-r--r-- 1 root root    928 Feb 14 08:23 shadow-
-rw-r--r-- 1 root root    165 Feb 13 22:05 shells
drwxr-xr-x 2 root root   4096 Feb 13 22:05 skel
drwxr-xr-x 2 root root   4096 Mar  1 15:30 snmp
drwxr-xr-x 3 root root   4096 Feb 14 08:14 snort
drwxr-xr-x 2 root root   4096 Feb 13 22:06 snb
drwxr-xr-x 4 root root   4096 Feb 15 14:44 ssl
-rw-r--r-- 1 root root   2088 Feb 24 2010 syslog.conf
drwxr-xr-x 2 root root   4096 Feb 13 22:06 syslogd
drwxr-xr-x 2 root root   4096 Feb 13 22:05 terminfo
drwxr-xr-x 3 root root   4096 Feb 13 22:05 texmf
-rw-r--r-- 1 root root    21 Feb 13 22:06 timezone
-rw-r--r-- 1 root root  1260 May 30 2008 ucf.conf
drwxr-xr-x 4 root root   4096 Feb 13 22:06 udev
drwxr-xr-x 3 root root   4096 Feb 13 22:05 ufw
-rw-r--r-- 1 root root    274 Nov  4 2009 updatedb.conf
drwxr-xr-x 2 root root   4096 Feb 13 22:06 vim
drwxr-xr-x 2 root root   4096 Feb 13 22:06 vsm
drwxr-xr-x 2 root root   4096 Feb 24 11:07 webalizer
-rw-r--r-- 1 root root   4496 Sep  5 2010 wgetrc
drwxr-xr-x 3 root root   4096 Feb 13 22:06 X11
drwxr-xr-x 2 root root   4096 Feb 13 22:06 xml
root@debian6: /etc#
```

- SETUID y SETGID son permisos de acceso que pueden asignarse a archivos o directorios en un sistema operativo basado en Unix.
- Se utilizan principalmente para permitir a los usuarios del sistema ejecutar binarios con privilegios elevados temporalmente para realizar una tarea específica.
- Si un archivo tiene activado el bit SETUID se identifica con una “s” en un listado de la siguiente forma:
`-rwsr-xr-x 1 root shadow 27920 ago 15 22:45 /usr/bin/passwd`

- Existen otros permisos: Sticky Bit
- *Chattr*: Utiliza otros atributos (append only, immutable, etc).
- *Posix ACLs* (*getfacl*, *setfacl*) y *NFSv4 ACLs*: flexibilizan las ACLs standard, posibilitando dar permisos a usuarios específicos, a más de un grupo, etc.

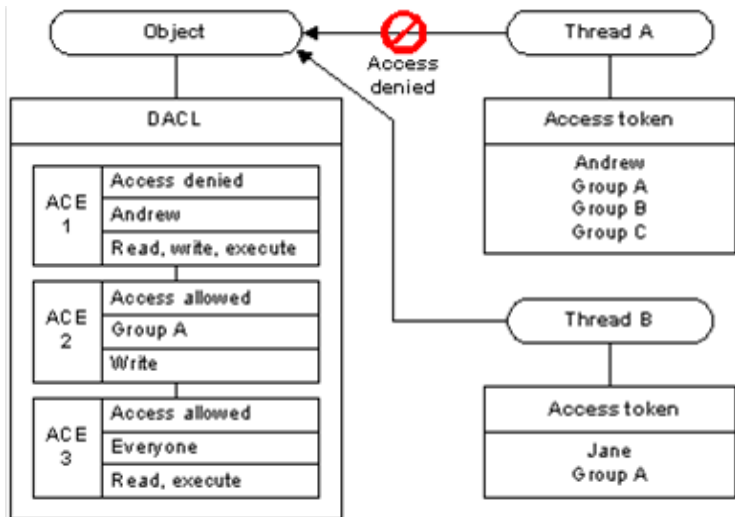
(22) Algunos puntos sensibles

- Permiso para propagar permisos.
- Revocación: ¿inmediata o diferida?
- Permisos en los archivos: si puedo modificar el archivo, no tengo que poder modificar el permiso.

(23) Los procesos

- Los procesos en tanto sujetos, ¿a qué pueden acceder?
- En general, heredan los permisos del usuario que los está corriendo.
- Problema: ¿cómo implementar de manera segura el cambio de contraseña?
- En Unix: setuid bit.
- Los permisos del proceso no son los del usuario que lo corre si no los del propietario del programa.
- CUIDADO: si bien los esquemas que permiten que un proceso corra con mayores privilegios que el usuario a veces son necesarios, son peligrosos.

(24) ACLs en Windows



(25) Windows Integrity Control

- Es un ejemplo de MAC.
- Se basa en el modelo Biba de control de integridad.
- Define cuatro niveles de integridad: System, High, Medium, Low.
- Archivos, carpetas, usuarios, procesos, todos tienen niveles de integridad.
- El nivel medio es el nivel por defecto para usuarios estándar y objetos sin etiquetas.
- El usuario no puede darle a un objeto un nivel de integridad más alto que el suyo.

(26) Buffer overflows

Repaso de Orga I (simplificado):

- Cuando se invoca a una función en C, primero se hace un push de los parámetros y luego del IP.
- Las variables locales reservan espacio en la pila.

```
void f(char *origen) {  
    char buffer[16];  
    strcpy(buffer, origen);  
}  
void main(void) {  
    char grande[18];  
    f(grande);  
}
```

(27) Buffer overflows (cont.)

Antes del strcpy la pila se ve así:

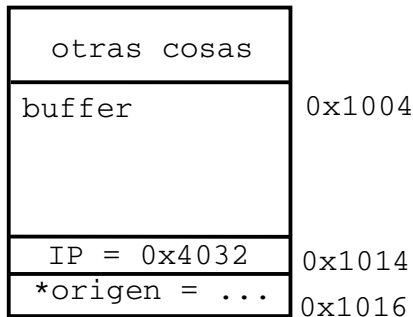


Figura: La pila antes del strcpy.

(28) Buffer overflows (cont.)

Después del strcpy la pila se ve así:

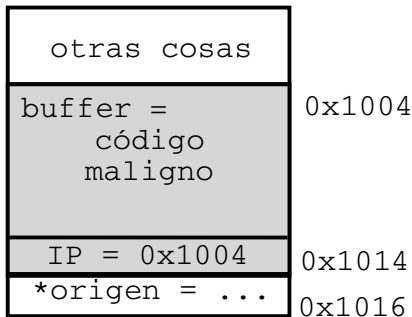


Figura: La pila después del strcpy.

(29) Buffer overflows (cont.)

Misceláneos:

- La variación presentada es “stack based”, pero también hay “heap based”: misma idea.
- Hay distintas formas de detectarlo, e intentar prevenirlo.

(30) Control de parámetros

- Ejemplo: CGI que felicita por el cumpleaños.
- Corre con privilegios elevados (pe, root).
- Invocación:
 - `http://www.example.com/cgi-bin/felicitar.cgi?fulano@dc.uba.ar`

Programa:

```
$direccion=argv[1]
```

```
echo Feliz cumple | mail -subject='Felicitación' $direccion
```

(31) Control de parámetros (cont.)

Ataque:

`http://www.example.com/cgi-bin/felicitar.cgi?fulano@dc.uba.ar; rm -rf /`

Se ejecuta:

```
echo Feliz cumple | mail -subject='Felicitación'
fulano@dc.uba.ar; rm -rf /
```

(32) Control de parámetros (cont.)

Dos soluciones:

- Utilizar el mínimo privilegio posible.
- Validar los parámetros.

Otro enfoque: “*tainted* data”.

(33) También sucede en bases de datos

- Cuando se da en bases de datos se llama *SQL injection*.
- Formulario pide que se ingrese un número.
- El programa no sanitiza la entrada y lo usa directamente.
- Ejemplo: formulario pide ingreso de LU para aprobar alumnos.
- Programa ejecuta:

```
db.execute("UPDATE alumnos SET aprobado=true WHERE lu='"+input.getString("lu")+"'")
```
- Usuario malicioso ingresa:

```
307/08'; DROP alumnos; SELECT '
```

(34) También sucede en bases de datos (cont.)



“Exploits of a Mom”, Randall Munroe, <http://xkcd.com/327/>

(35) Condiciones de carrera

Definición: “Comportamiento anómalo debido a una dependencia crítica inesperada en el timing de los eventos”.

Ejemplo (típico): Crear el archivo si no existe.

Problema: combinado con links, sobrescribir archivos importantes.

(36) Condiciones de carrera (cont.)

Incorrecto:

Existe(in a: archivo, in d: disco) \leftarrow res: bool
 $\{true\}$
 $\{res = a \in d\}$

Crear(in a: archivo, inout d: disco)
 $\{d_0 = d \wedge (a \notin d)\}$
 $\{d = d_0 \cup \{a\}\}$

Código:

if (not(Existe(a, d))) then Crear(a, d) fi

(37) Condiciones de carrera (cont.)

Correcto: debilitar la precondition y hacer la operación atómica (en el sentido de indivisible).

CrearSiNoExiste(in a: archivo, inout d: disco)

$\{d = d_0\}$

$\{((a \notin d_0) \implies d = d_0 \cup \{a\}) \wedge (a \in d_0 \implies d = d_0)\}$

(38) Código Malicioso

- Malware: Malicious Software
- Se denomina malware al software malicioso, diseñado para llevar cabo acciones no deseadas y sin el consentimiento explícito del usuario.
- Existen muchos tipos de malware, entre otros: virus, troyanos, gusanos, bots, adware, keyloggers, dialers, rootkits, ransomware, rogueware, etc.

(39) Principales métodos de infección

- Descarga desde páginas webs (a veces involuntariamente).
- Adjuntos por email.
- Vulnerabilidades en software.
- Compartir dispositivos de almacenamiento.
- Otros protocolos y aplicaciones en Internet: mensajería instantánea, P2P, redes sociales

(40) Evolución del malware


- 1987-1999: virus clásicos, los creadores no tenían ánimo de lucro, motivación intelectual y protagonismo.
- 2000-2004: explosión de los gusanos en Internet, propagación por correo electrónico, aparición de las botnets.
- 2005-2009: claro ánimo de lucro, profesionalización del malware, explosión de troyanos bancarios y programas espías.
- 2010-2011: casos avanzados de ataques dirigidos, espionaje industrial y gubernamental, ataque a infraestructuras críticas, proliferación de infecciones en dispositivos móviles.

(41) Medidas para prevenir infecciones

- Usar antivirus actualizado (esto sólo no alcanza!).
- Actualización del sistema operativo, navegador y resto de aplicaciones.
- Uso de usuario restringido vs administrador.
- Sentido común y uso responsable.

- Los sistemas operativos modernos suelen proveer distintas formas de aislar a los usuarios y procesos. Reciben el nombre colectivo de *sandboxes*:
 - chroot()
 - jail()
- Llevado al extremo, es uno de los usos de la virtualización.

(43) Otros tipos de ataque

- Algunos ataques no sirven (de manera directa) para tomar control.
- Por ejemplo:
 - Negación de servicio.
 - Escalado de privilegios.
- Muchas veces se combinan.
- Recuerden: esto es sólo la punta del iceberg... 
- Si les gusta, cursen la materia optativa “Seguridad de la Información”.

(44) Principios generales

- Mínimo privilegio.
- Simplicidad.
- Validar todos los accesos a datos.
- Separación de privilegios.
- Minimizar la cantidad de mecanismos compartidos.
- Seguridad multicapa.
- Facilidad de uso de las medidas de seguridad.

(45) El rol de la confianza

- Supongamos que ha aparecido una vulnerabilidad en el sistema operativo que usamos en nuestra PC.
- Obtenemos el parche de seguridad correspondiente.
- Lo instalamos.
- Elevamos el nivel de seguridad de nuestra PC.
- Confiamos en que ya no es vulnerable.

(46) El rol de la confianza

- Pero además implícitamente confiamos en:
- Que el parche viene del vendedor del sistema operativo y que no fue modificado.
- Que el vendedor probó correctamente el parche antes de liberarlo.
- Que el ambiente de prueba del vendedor se corresponde con nuestro ambiente.
- Que el parche se instaló correctamente.

(47) El rol de la confianza

- Cualquier política, mecanismo, o procedimiento de seguridad está basado en asumir hechos que, de ser incorrectos, destruyen todo lo construido.
- Hay que tener esto en mente, porque si no entendemos en que se basa la política, el mecanismo, o el procedimiento de seguridad, se pueden asumir cosas inválidas y llegar a conclusiones erróneas.

- *Introduction to Computer Security*, Matt Bishop
- *Smashing the Stack for Fun and Profit*, Aleph One,
<http://www.insecure.org/stf/smashstack.txt>