Sistemas Operativos

Protección y Seguridad

Sistemas Operativos

- ☑ Versión: Mayo 2025
- ☑ Palabras Claves: Seguridad, Protección, Permisos, Matriz de Acceso, ACL, Acceso, Dominios, Capacidades
- ☑ Bibliografía recomendada: Sistemas Operativos Modernos, Andrew S. Tanembaum

Introducción

- Los recursos informáticos deben protegerse frente a accesos no autorizados, destrucciones maliciosas o introducción accidental de incoherencias.
- Los recursos a proteger son (entre otros):
  - Datos/Información
  - CPU, Memoria, dispositivos
- El responsable de llevar a cabo la tarea es, entre otros, el Sistema Operativo, a través de un conjunto de mecanismos.
- En base a la correcta o incorrecta aplicación de los mecanismos de **protección**, podremos determinar el nivel de **Seguridad** con el que cuenta el sistema

Protección ≠Seguridad

- **Protección**
  - ✓Mecanismos específicos del SO para resguardar la información dentro de una computadora, para controlar el acceso de los procesos (o usuarios) a los recursos existentes.
- **Seguridad**:
  - ✓Medida de la confianza en que se puede preservar la integridad de un sistema y sus datos
  - ✓Es un concepto más general (lo vamos a ir desarrollando)

Seguridad

- La seguridad utiliza distintos mecanismos con el fin de proteger y garantizar ante:
  - Amenazas
    - Confidencialidad de los datos (Intercepción / Modificación)
    - Integridad de los Datos (Modificación)
    - Disponibilidad (Interrupción)
  - Intrusos
    - Acceso indebido al sistema o datos
  - Perdida Accidental de Datos
    - Accidentes Naturales
    - Errores de HW o SW
    - Errores humanos

Protección

- Acceso al sistema (1) vs.
  Acceso a recursos en un sistema (2)
- (1) se resuelve a través de:
  - Autenticación
  - Control sobre una base de datos de usuarios
- (2) se resuelve a través de:
  - Permisos
  - Control de acceso (obligatorio o no)

Políticas ≠ Mecanismos.

- ☑ Es importante definir las políticas antes que los mecanismos.
- ☑ Las políticas (**qué**) definen lo que se quiere hacer, en base a los objetivos. Las podemos asociar a los papeles. Rara vez incluyen configuraciones
- ☑ Los mecanismos (**cómo**) definen cómo se hace. En este punto aparecen las configuraciones e implementaciones reales.
- ☑ Hay diferentes mecanismos para cumplir una política.
- ☑ Esta separación garantiza la flexibilidad de un sistema.

Objetos y Dominios

- ☑ Un sistema informático es una colección de procesos y objetos
- ☑ Objetos: de HW (CPU, Memoria, etc.) o de SW (archivos, programas, semáforos)
- ☑ Cada objeto debe tener un identificador único que permita referenciarlo
- ☑ Los procesos pueden realizar un conjunto finito de operaciones sobre los objetos
- ☑ Un ***dominio*** es un conjunto de pares (objeto, derecho). Cada par especifica un objeto y un subconjunto de operaciones que se pueden realizar con él.

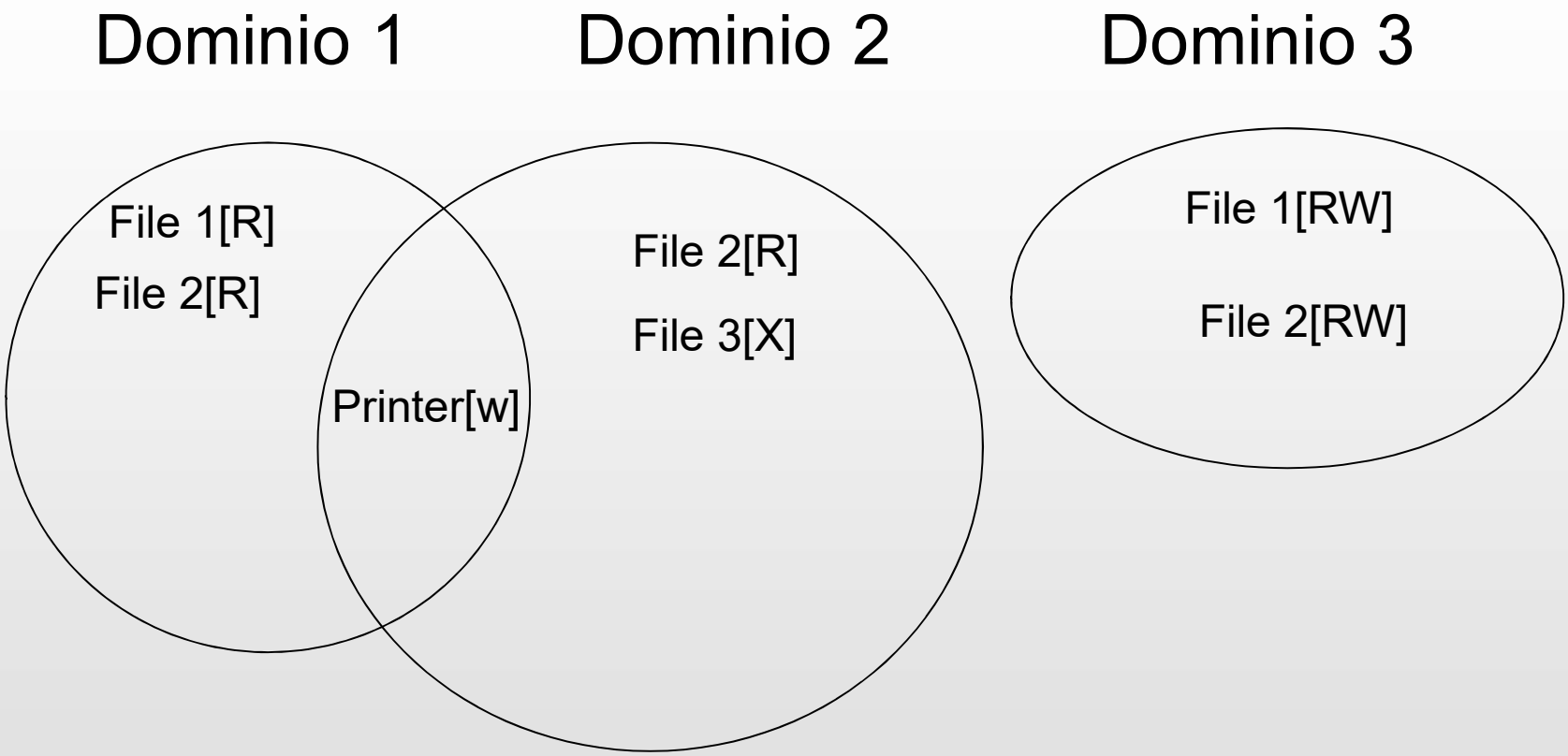
Objetos y Dominios

- Un ***derecho*** (right) significa autorización para efectuar esas operaciones.
- Por ejemplo:
  - el dominio D contiene la pareja (archivo A, {read,write}).
  - Un proceso que se ejecuta dentro del dominio D puede leer y grabar el archivo A.
- ¿Quiénes pueden ejecutarse en un dominio?:
  - Puede ser un usuario y define qué puede hacer ese usuario (por defecto lo que no se permite se deniega)
  - Puede ser un proceso y el conjunto de objetos a los que podrá acceder dependerá de la identidad del proceso
  - Puede ser un procedimiento y definirá el conjunto de variables a las que puede acceder (variables locales, globales, etc…)

Dominios y procesos

- ☑ **Principio de need-to-know o POLA (Principle of least authority):** define que los procesos accedan sólo a los objetos que necesitan (con los derechos que necesiten) para completar su tarea.
- ☑ La relación entre un proceso y un dominio puede ser *estática* o *dinámica*
- ☑ Relación estática: si el conjunto de objetos a los que el proceso accede durante su ciclo de vida es fijo.
  - Siempre mismo dominio
  - Puede generar que los procesos tengan más privilegios que los que necesitan en sus fases de ejecución
- ☑ Relación dinámica: si el conjunto de objetos puede variar.
  - Puede cambiar de dominio. Por ejemplo usando los bits SETUID y SETGID en UNIX sobre los archivos

Objetos y Dominios

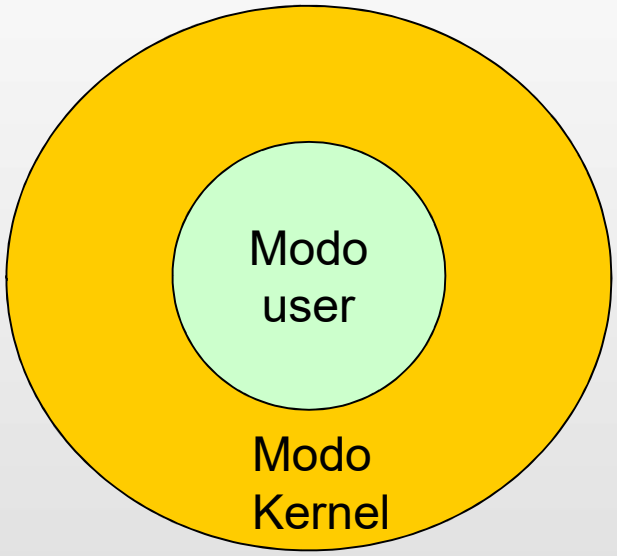


Ejemplos (I)

- ☑ Un conjunto de programadores pueden pertenecer al mismo dominio
- ☑ En Unix el dominio lo definen el UID y el GID
- ☑ Dado (UID, GID) hay un conjunto de objetos a los que se puede acceder con determinados permisos
- ☑ Dos procesos con igual (UID, GID) pueden acceder al mismo conjunto de archivos.
  - Mismo Dominio

Ejemplos (II)

- ☑ Modo usuario y modo supervisor



- ☑ Dominios de protección organizados jerárquicamente en una estructura de anillos concéntricos
- ☑ Los privilegios se asignan por anillo y desde ese punto hacia adentro.

Ejemplos (III)

- ☑ setuid y setgid

```
jpablop@jpp:~$ ls -la /usr/bin/passwd
-rwsr-xr-x 1 root root 51224 jul 21  2015 /usr/bin/passwd

jpablop@jpp:~$ ls -la /usr/bin/sudo
-rwsr-xr-x 1 root root 128616 sep 22  2015 /usr/bin/sudo
```

Matriz de acceso

- ☑ Controla la pertenencia de objetos a dominios y sus derechos.
- ☑ Las filas representan dominios
- ☑ Las columnas representan objetos.
- ☑ Cada elemento access(i,j) representa el conjunto de operaciones (derechos) que un proceso puede invocar en un objeto O_j dentro del dominio D_i.
- ☑ Implementa las políticas de protección/seguridad de un sistema

Ejemplo de Matriz de acceso

Objeto / Dominio	File1	File2	File3	Printer
D1	Read	Read		Print
D2		Read	execute	print
D3	Read/write	Read/write		

Operación: switch

- Un proceso puede cambiar de un dominio a otro.
- Para poder cambiar de un dominio a otro se debe habilitar la operación *switch* sobre un objeto (dominio).
- La matriz en sí es un objeto que posee atributos. De esta manera se define si sobre un dominio se realiza asignación estática o no:
  - La conmutación del dominio D_i al dominio D_j estará permitida si se encuentra definida en el switch access(i,j)
- Dominio y matriz son objetos sobre los cuales se definen accesos

Operación: switch

Objeto / Dominio	File1	File2	File3	Printer	D1	D2	D3
D1	Read	Read		Print		**switch**	
D2		Read	execute	print			
D3	Read/write	Read/write			**switch**		**switch**

Operación copy

- Es un derecho que se asocia a un elemento access(i,j) de la matriz.
- Indica que un proceso ejecutándose en ese dominio puede copiar los derechos de acceso de ese objeto dentro de su columna
- Se denota con *.

Aplicación de operación copy

Objeto / Dominio	File1	File2	File3	Printer
D1	Read	Read		Print
D2		**Read ***	execute	print
D3	Read/write			
Objeto / Dominio	**File1**	**File2**	**File3**	**Printer**
D1	Read	Read		Print
D2		**Read ***	execute	print
D3	Read/write	**Read**		

Variantes de copy

- ***Transferencia:*** si un derecho se copia desde matriz(i,j) a matriz(k,j), el derecho desaparece para matriz(i,j), o sea, el derecho fue transferido.
- ***Propagación o copia limitada:*** Se copia el derecho pero no el *derecho a copia* en el nuevo (R* es copiado como R, no como R*).

Operación owner

- ☑ Permite agregar nuevos derechos y borrar ya existentes.
- ☑ Si matriz(i,j) incluye el derecho de *owner* entonces un proceso ejecutándose en el dominio Di puede agregar y borrar cualquier entrada en la columna j.

Aplicación de operación owner

Objeto / Dominio	File1	File2	File3	Printer
D1	Read	Read		Print
D2		Read * **owner**	execute	print
D3	Read/write			
Objeto / Dominio	**File1**	**File2**	**File3**	**Printer**
D1	Read	Read**/write**		Print
D2		Read * owner	execute	print
D3	Read/write	**Read**		

Operación control

- ☑ Indica que pueden modificarse y borrarse derechos dentro de una fila.
- ☑ La operación control es aplicable sólo a dominios.
- ☑ Si matriz (i,j) incluye el derecho de control, entonces un proceso ejecutándose en el dominio Di puede remover cualquier derecho de acceso dentro de la fila j.

Ejemplo de operación control

Objeto / Dominio	File1	File2	File3	Printer	D1	D2	D3
D1	Read	Read		Print		switch	**control**
D2		Read	execute	print			
D3	Read/ Write	Read/ write			switch		switch

☑Cualquier proceso que se ejecute en el dominio D1, puede controlar los Switch Access de cualquier columna del dominio D3

Resumiendo:

- ☑ Copy, owner y control son operaciones que se utilizan para controlar cambios al contenido de la matriz de acceso.
- ☑ Switch y Control: son aplicables sólo a dominios.
- ☑ Copy y owner: puede modificar derechos dentro de una columna.
- ☑ Control: puede modificar derechos dentro de una fila

Implementación de la matriz de acceso

- La forma de representar y almacenar la matriz de acceso, generalmente no es bajo el formato de matriz, ya que se desperdiciaría mucho espacio (es ineficiente)
- El principal problema es que puede tener muchos elementos vacíos
- Generalmente se almacenan solo los elementos ocupados utilizando 2 métodos:
  - Almacenarla por filas
  - Almacenarla por columnas
- Se debe optimizar el acceso para que sea rápido.

Implementación de la matriz de acceso

- Alternativas de implementación:
  - Tabla Global: la más simple.
  - Lista de Control de Acceso por objetos
  - Lista de Capacidades por Dominio

Tabla Global

- ☑ Es la mas sencilla de implementar
- ☑ Consiste en conjunto de tuplas <dominio,objeto, derechos-acceso>
- ☑ Cada vez que se ejecuta una operación **M** sobre un objeto $\mathbf{O_j}$ sobre el dominio $\mathbf{D_i}$, se analiza la tabla y se verifica si se encuentra una terna $< \mathbf{D_i}, \mathbf{O_j}, \mathbf{M} >$
  - ☑ Si se encuentra se permite la operación
  - ☑ Si no se encuentra se deniega
- ☑ Su principal desventaja es que el tamaño de la tabla hace que no se pueda almacenar toda en memoria

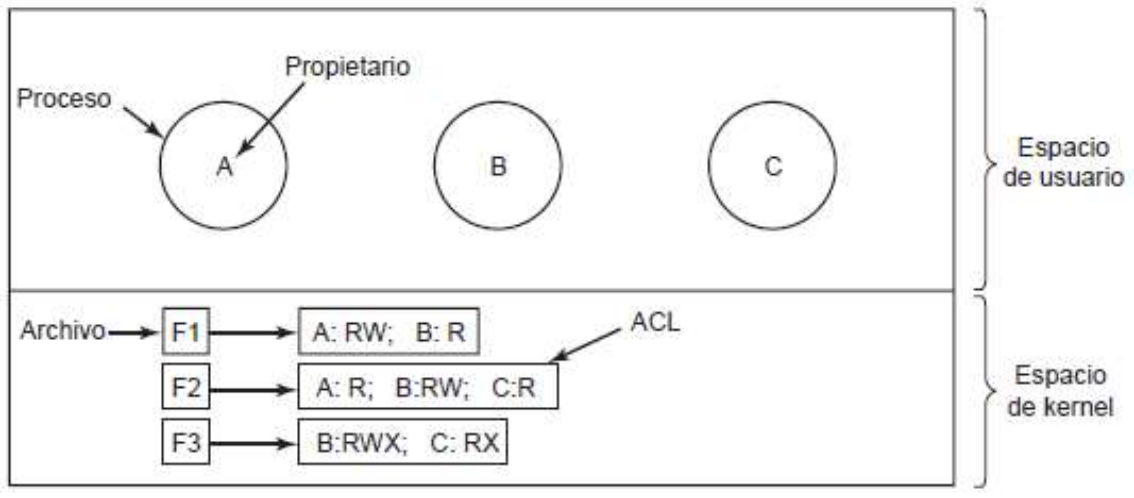
	Objeto										
Dominio	Archivo1	Archivo2	Archivo3	Archivo4	Archivo5	Archivo6	Impresora1	Plotter2	Dominio1	Dominio2	Dominio3
1	Lectura	Lectura Escritura								Enter	
2			Lectura	Lectura Escritura Ejecución	Lectura Escritura		Escritura				
3						Lectura Escritura Ejecución	Escritura	Escritura			

Lista de control de Acceso (ACL)

- ☑ Consiste en asociar con cada objeto una lista (ordenada) que contenga todos los dominios que pueden acceder al objeto, y la forma de hacerlo
- ☑ Cada **columna** de la matriz se puede ver como una lista de acceso a un objeto, descartándose elementos vacíos
- ☑ Para cada objeto, hay una lista de pares ordenados <dominio, derechos>
- ☑ Cuando se intenta realizar una operación **M** sobre un objeto $O_{j\ (F1, F2, F3)}$ en el dominio $\mathbf{D_{i\ (A, B, C)}}$, se busca en la lista en el objeto $\mathbf{O_j}$ una entrada $\mathbf{<D_i, R_k>}$, donde **M** pertenece al conjunto $\mathbf{R_k}$

Lista de control de Acceso (ACL)

- Cada archivo tiene una ACL asociada. El archivo F1 tiene dos entradas en su ACL
  - La primera indica que cualquier proceso que sea propiedad del usuario A puede leer y escribir en el archivo.
  - La segunda indica que cualquier proceso que sea propiedad del usuario B puede leer el archivo.
  - Todos los demás accesos están prohibidos.

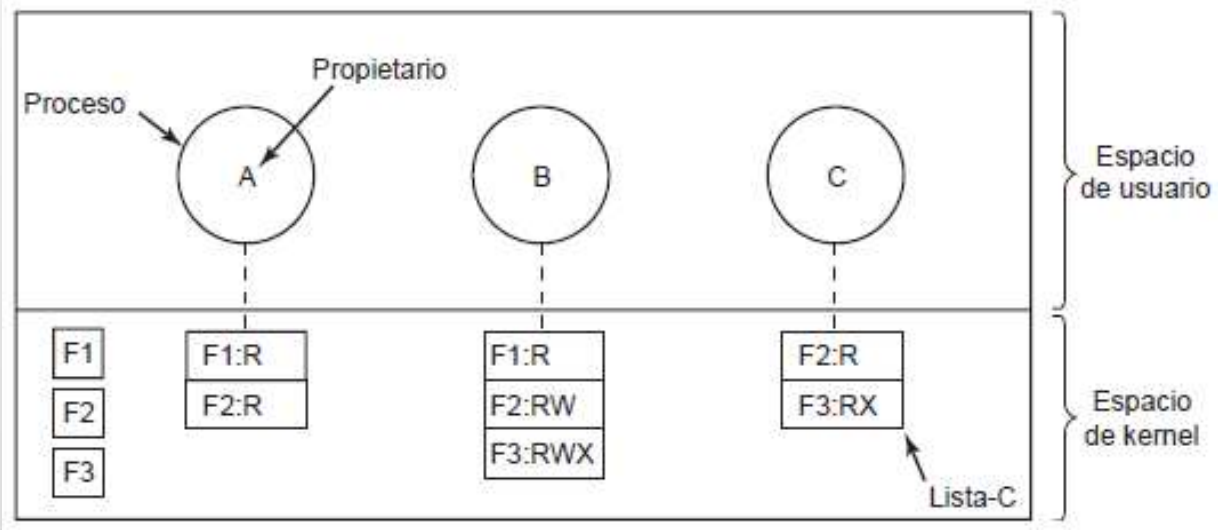


Lista de capacidades

- ☑ Es una lista de objetos del dominio con sus derechos (**división x filas**)
- ☑ A esta lista se le conoce como lista de capacidades y a los elementos individuales que contiene se les conoce como capacidades
- ☑ El proceso no la accede directamente
- ☑ Esta lista es un objeto protegido, a la que sólo accede el SO
- ☑ Cada proceso tiene una lista con los objetos que puede utilizar, junto con que operaciones (el dominio)
- ☑ Presenta dificultades al momento de revocar o modificar un permiso sobre un objeto
  - Se deben recorrer todas las listas de capacidades

Lista de capacidades

- ☑ Cada capacidad otorga al propietario ciertos derechos sobre un objeto
- ☑ Cada proceso tiene una lista con los objetos que puede utilizar, junto con que operaciones (el dominio)
- ☑ Por ejemplo, el proceso que pertenece al usuario A puede leer los archivos F1 y F2



Sistemas confiables

- Otro factor que afecta el nivel de seguridad de un sistema, es el código:
  - Código mal intencionado: Virus, gusanos, troyanos, etc.
  - Código con errores de programación: Backdoors
- ¿Es posible construir sistemas seguros? La respuesta es si, pero el gran enemigo de la seguridad es el agregado de funcionalidad.
  - Un sistema minimalista será probablemente muy seguro, pero pobremente usable
  - Un sistema con muchas características, probablemente tenga mas errores y abra más la posibilidad a que sea vulnerado

Sistemas confiables - Ejemplos

- MULTICS desarrollado en 1960 tenía como objetivo principal la seguridad
- El protocolo SMTP para el envío de mail se definió en el año 1982:
  - En aquel entonces solo permitía el envío de mensajes en formato de texto ASCII (seguro, pero poco funcional)
  - Con el tiempo se agregó la posibilidad de enviar otros tipos de documentos (ofimática, multimedia, etc.). Estos archivos pueden incluir macros o código malicioso.
- HTTP es otro claro ejemplo. Inicialmente solo permitía transferir datos desde un servidor a los clientes. Hoy en día permite ejecutar aplicaciones desde el lado del cliente y servir contenido dinámico, lo cual podría contener vulnerabilidades

Explotación de errores en código

- ☑ Los procesos, junto con el Kernel son una potencial amenaza a la seguridad de un sistema.
- ☑ Los atacantes aprovechan errores en la codificación del SO, o algún proceso con alto nivel de privilegios con el fin de que los mismos cambien su funcionamiento normal:
  - ☑ Desbordamiento de buffer
  - ☑ Cadenas de formato
  - ☑ Retorno a libc
  - ☑ Desbordamiento de enteros
  - ☑ Inyección de código

https://www.exploit-db.com/

Desbordamiento de buffer (buffer overflow)

- ☑ Tiene como objetivo sobreescribir datos de ciertas zonas de memoria intencionalmente
- ☑ Se trata de un error del software. Se quiere copiar una cantidad de datos más grande que el área definida
- ☑ Ejemplo simple:

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

- ☑ El resultado es que se sobrescribe cierto byte de memoria que esté a una posición de 10,976 bytes fuera del arreglo
- ☑ Los compiladores de C no realizan esta comprobación, con lo cual este error debe ser manejado en tiempo de ejecución

Desbordamiento de buffer (buffer overflow)

- ☑ Si el error del ejemplo anterior no pudiera ser detectado en runtime, podría alterarse el flujo del programa
- ☑ Si el dato que se sobreescribe no tiene un sentido, entonces probablemente lo que se verá es un error en la ejecución
- ☑ Si la posición de memoria se sobreescribe con un valor sensible, la ejecución podría encadenar una nueva ejecución de un programa malicioso
- ☑ Shellcode: código copiado especialmente preparado para obtener los privilegios del programa atacado.