Universidad Nacional Autónoma de México

Facultad de Ciencias

Organización y Arquitectura de Computadoras
2025-2

Práctica 09

Docentes:
José Galaviz Ricardo Pérez Ximena Lezama

Autores:
Fernanda Ramírez Juárez Ianluck Rojo Peña

Fecha de entrega: Jueves 9 de abril de 2026

Preguntas.

1. En las llamadas a sistema, las bibliotecas y APIs funcionan como intermediario entre el usuario y las llamadas a sistema. ¿Qué bibliotecas contienen esas llamadas a sistema en Unix y en Windows? ¿Que llamadas al sistema están contenidas en esos archivos?

   En sistemas operativos, las bibliotecas y APIs actúan como intermediarios entre las aplicaciones de usuario; las llamadas al sistema *(system calls)* funciones básicas, de bajo nivel, proporcionadas por el kernel que permiten operaciones como el manejo de archivos, procesos, memoria y dispositivos.

   1) ***Unix/Linux***:

   Las llamadas al sistema en Unix/Linux son accesibles a través de bibliotecas estándar que actúan como **wrappers** *(envoltorios)* para interactuar con el kernel.

   Bibliotecas principales.

   - *GNU C Library(glibc)*: La implementación estándar más común en Linux. Proporciona funciones como `open()`, `fork()` o `write()`, que finalmente invocan las llamadas al sistema.
   - *musl libc*: Alternativa ligera a `glibc`, común en sistemas embebidos o distribuciones minimalistas.
   - *libSystem (macOS/BSD)*: En sistemas BSD y macOS, esta biblioteca cumple un rol similar a glibc.

   Ejemplos de llamadas al sistema en Unix/Linux.

   - Archivos: `open()`, `read()`, `write()`, `close()`.
   - Procesos: `fork()`, `execve()`, `wait()`, `exit()`.

- Memoria: `brk()`, `sbrk()`, `mmap()`, `munmap()`.
- Redes: `socket()`, `bind()`, `listen()`, `accept()`.
- Dispositivos: `ioctl()`.

2) ***Windows***:

Windows no expone directamente las llamadas al sistema, sino que ofrece capas de abstracción como la **WinAPI** y la **NT API** *(nativa del kernel)*.

Bibliotecas principales.

- *kernel32.dll*: Proporciona funciones de alto nivel para manejo de procesos, memoria y archivos.
- *ntdll.dll*: Interfaz entre las llamadas de WinAPI y las llamadas nativas del kernel *(NT API)*. Contiene wrappers como NtCreateFile.
- Otras DLLs: *user32.dll* (interfaz gráfica), *ws2_32.dll* (redes).

Ejemplos de llamadas equivalentes en Windows.

- Archivos: `CreateFile()`, `ReadFile()`, `WriteFile()`, `CloseHandle()`.
- Procesos: `CreateProcess()`, `ExitProcess()`, `WaitForSingleObject()`.
- Memoria: `VirtualAlloc()`, `VirtualFree()`.
- Redes: `WSASocket()`, `Connect()`, `Send()`.

Las llamadas nativas del kernel *(NT API)* tienen prefijo Nt ejemplo: **NtCreateFile**, pero no están documentadas oficialmente por Microsoft. Se accede a ellas indirectamente a través de **ntdll.dll**.

2. Describe que causa, que llamada a sistema y que es o que hacen las siguientes vulnerabilidades a Sistema causadas por la llamadas a Sistema en Linux (registradas con los sufijos CVE):

- Dirty Cow
- Dirty Pipe
- Baron Samedit Sudo
- Residual Risk Flaw

Vulnerabilidades en Linux relacionadas con llamadas al sistema (CVE).

1) ***Dirty COW (CVE-2016-5195)***.

- Su causa de debe a la condición de carrera *(race condition)* en el mecanismo **Copy-On-Write** *(COW)* del kernel al manejar mapeos de memoria privada.
- Permite a un usuario sin privilegios modificar archivos solo lectura (incluyendo binarios SUID o archivos del sistema).
- Llamadas al sistema involucradas:
  - `mmap()`: Para mapear memoria.
  - `write()`: Intento de escritura en memoria mapeada.

  Fallo clave: El kernel no sincronizaba correctamente hilos al actualizar páginas COW.

- Impacto:
  - Elevación de privilegios local *(Local Privilege Escalation, LPE)* que es un tipo de ataque donde un usuario sin permisos administrativos logra aumentar sus privilegios dentro del sistema.
  - Ejemplo: modificar /etc/passwd o ganar acceso root.

  Parcheado en **Linux 4.8.3** (2016).

2) ***Dirty Pipe (CVE-2022-0847)***.

- Su causa es el error en el manejo de pipes y la caché de páginas del kernel. Aprovecha una condición de carrera en splice() para escribir en archivos de solo lectura.
- Llamadas al sistema involucradas:
  - `splice()`: Transfiere datos entre pipes y archivos sin copiar a user-space.

  Fallo clave: El kernel no validaba permisos al escribir en caché de páginas compartidas.
- Impacto:
  - Sobrescritura de archivos críticos *(ejecutables SUID, configuraciones del sistema)*.

  Afectó kernels **5.8** a **5.16.11** (parcheado en 2022).

3) ***Baron Samedit (CVE-2021-3156, Sudo)***.

- Su causa es el desbordamiento de búfer *(heap-based overflow)* en `sudoedit` al procesar argumentos malformados en modo no-interactivo.
- Llamadas al sistema involucradas:
  - `execve()`: Para ejecutar `sudoedit`.

  No es un fallo del kernel, sino del manejo de memoria en espacio de usuario (sudo).
- Impacto:
  - Elevación a root sin contraseña.

  Afectó a Sudo antes de la versión **1.9.5p2** (2021).

4) ***Residual Risk***.

- No es una vulnerabilidad específica, sino un concepto de seguridad que refiere a riesgos remanentes tras aplicar controles. Puede involucrar llamadas al sistema si hay permisos mal configurados, ejemplos servicios que no dropean privilegios.
- Relación con llamadas al sistema:
  - Cualquier llamada mal gestionada.
  - Ejemplo: `open()`, `chmod()` puede dejar riesgos residuales.
- Impacto:
  - Depende de la implementación.
  - Ejemplo: ejecución de código con privilegios no revocados.

5) ***Flaw (Término genérico)***.

- Su causa es el error genérico en el kernel o programas (no específico). Como bugs en manejo de hilos (`clone()`), drivers (`ioctl()`), etc.
- Llamadas al sistema involucradas:

  - Depende del fallo.
  - Ejemplo: `ioctl()` para drivers, `clone()` para hilos.
- Impacto: Desde denegación de servicio (DoS) esto significa que un atacante puede crashar o bloquear un proceso, servicio o incluso todo el sistema, volviéndolo inaccesible o inestable, hasta ejecución de código arbitrario es decir un atacante puede ejecutar comandos o programas con privilegios elevados.