SEGURIDAD EN EL KERNEL

CÓMO PROTEGER UN SISTEMA OPERATIVO CONTRA ROOTKITS

Sistemas operativos

Grupo 06

Meléndez Gómez Anuar
Zambrano Serrano Héctor
Facultad de ingeniería

1. Introducción

1.1 ¿Qué es un kernel?

El kernel es un componente considerado central dentro de un sistema operativo, este funge como un enlace entre el hardware de una computadora y los programas que se ejecutan en ella. El kernel gestiona recursos como la memoria, CPU, dispositivos externos e internos y los permisos de seguridad

El kernel lleva a cabo cuatro responsabilidades consideradas clave:

1. **Gestión de procesos** → Decide qué programas se ejecutan y cuándo.
2. **Gestión de memoria** → Asigna y libera RAM para aplicaciones.
3. **Control de dispositivos** → Maneja hardware como discos duros, teclados y tarjetas de red.
4. **Llamadas al sistema (syscalls)** → Permite a los programas pedir servicios al sistema operativo.

1.2 Ejemplo práctico del funcionamiento de la kernel

Supongamos que en Python, queremos leer un archivo con el siguiente código:

```
with open("datos.txt", "r") as archivo:
contenido = archivo.read()
```

¿Qué pasa detrás de este proceso?

1. **El programa hace una llamada al sistema (**open**)** → El kernel recibe la petición.
2. **El kernel verifica permisos** → ¿Tiene el programa acceso al archivo?
3. **El kernel interactúa con el disco duro** → Lee los datos del archivo.
4. **El kernel copia los datos a la memoria del programa** → El código Python recibe el contenido.

Sin el kernel, el programa no podría acceder directamente al disco duro por razones de seguridad.

1.3 ¿Qué son los Rootkits?

Un rootkit es un tipo de software malicioso diseñado para darle a un hacker la capacidad de introducirse en un dispositivo y hacerse con el control del mismo. Por lo general, los rootkits afectan el software o el sistema operativo del dispositivo que infectan, pero algunos pueden actuar sobre su hardware o firmware. Los rootkits operan en segundo plano, sin dar muestras de que están activos.

Tras introducirse en un equipo, el rootkit permite al ciberdelincuente robar datos personales o financieros, instalar otras aplicaciones maliciosas o unir el equipo a una botnet para propagar spam o para sumarse a un ataque distribuido de denegación de servicio (DDoS).

1.4 Tipos de Rootkits

- **Kernel-mode**: Operan en el núcleo del sistema, altamente peligrosos.
- **User-mode**: Se ejecutan en espacios de usuario, menos privilegios pero más comunes.
- **Bootkits**: Infectan el arranque del sistema.
- **Firmware**: Se alojan en hardware (ej: BIOS).

1.5 Riesgos y casos reales

Persistencia: Un rootkit puede permanecer años en un sistema sin ser detectado.

Robo de información: Credenciales, datos bancarios, espionaje corporativo.

Ataques a infraestructuras críticas: Ej. Stuxnet (no era un rootkit puro, pero usaba técnicas similares para sabotear centrifugadoras nucleares en Irán).

---

2. Desarrollo

2.1 ¿Cómo afectan los Rootkits a los sistemas operativos?

Un **rootkit** es un tipo de malware diseñado para ocultar su presencia y la de otros programas maliciosos en un sistema operativo (SO). A diferencia de virus o troyanos tradicionales, los rootkits operan con altos privilegios (generalmente a nivel **kernel**) y pueden manipular el comportamiento del SO para evadir detección.

Los rootkits pueden ingresar al sistema mediante:

a. **Exploits de vulnerabilidades** (ej: fallos en drivers o servicios con privilegios).
b. **Inyección de código en procesos legítimos** (ej: secuestro de DLLs en Windows).
c. **Modificación del arranque (Bootkit)** → Infecta el gestor de arranque (GRUB, Windows Boot Manager).

2.2 Mecanismos de ocultación de los Rootkits

a. Hooking de Llamadas al Sistema
    i. Qué hace: Intercepta y modifica las llamadas entre aplicaciones y el kernel.
    ii. Objetivo: Ocultar procesos, archivos o conexiones maliciosas.
    iii. Detección: Herramientas como Process Hacker (Windows) o strace (Linux) monitorizan llamadas anómalas.
    iv. Ejemplo: Un rootkit cambia la función readdir() para ocultar archivos.

b. DKOM (Direct Kernel Object Manipulation)
    i. Qué hace: Manipula estructuras de datos del kernel en memoria RAM (sin modificar disco).
    ii. Objetivo: Ocultar procesos, drivers o puertos abiertos.
    iii. Deteccion: Analisis de memoria con volatility
    iv. Cómo funciona:
        1. **En Windows**: Modifica la lista EPROCESS (procesos activos) o HANDLE_TABLE.
        2. **En Linux:** Altera la lista task_struct
    v. Ejemplo: El rootkit Shadow Walker borraba procesos de la lista.

c. Enmascaramiento de Procesos y Archivos
    i. Qué hace: Falsifica nombres o atributos de procesos/archivos para parecer legítimos.
    ii. Objetivo: Evitar sospechas en herramientas como Task Manager o ls.
    iii. Técnicas comunes:
        1. Process Injection: Inyecta código malicioso en procesos legítimos (ej: explorer.exe).
        2. Timestomping: Cambia fechas de creación/modificación de archivos.
        3. Nombres engañosos: Usa nombres como svch0st.exe (en lugar de svchost.exe).

iv. Detección: Herramientas como Process Explorer (Sysinternals) verifican firmas digitales y rutas.

2.3 Caso en la vida real: Incidente del rootkit de Sony BMG

2.3.1 Contextualización

Un caso famoso relacionado con el uso de rootkits es el de Sony BMG, en su tiempo una de las mayores discográficas del mundo. Esta discográfica lanzó CDs de música con un rootkit oculto, con el objetivo de reducir la piratería. Esto generó un escándalo de seguridad y privacidad.

Sony incluyó un rootkit en millones de CDs. Cuando el CD se insertaba en una PC con Windows, se instalaba **automáticamente** un software DRM llamado **XCP (Extended Copy Protection)** o **MediaMax**.
Cuando el rootkit era instalado, este **ocultaba archivos y procesos** (comportamiento típico de un rootkit), **limitaba las copias del CD a solo 3**, **monitorear los hábitos de escucha** del usuario y **no se podía desinstalar fácilmente** (y dejaba vulnerabilidades críticas).

2.3.2 Técnicas usadas por el rootkit

1. Ocultamiento en el Sistema

- El rootkit modifica el kernel de Windows para **esconder cualquier archivo o proceso que empezará con "**
- sys
- *sys***"**.
- Esto permitía que el DRM de Sony **no fuera detectado** por el usuario o antivirus.

Ejemplo en Windows:

C:\> dir /a
No mostraba los archivos ocultos por el rootkit (sysdrm.exe)

2. Vulnerabilidades Introducidas

- El rootkit **desactivaba protecciones del sistema** y permitía que:
    - **Otros malware** explotaran la misma técnica para ocultarse.
    - **Backdoors** fueran instalados por atacantes externos.

3. Persistencia y Dificultad para Desinstalar

- Si un usuario intentaba eliminar el software manualmente, el CD **podía dañar el sistema operativo**.

2.3.3 Técnicas para proteger el kernel

A. Habilitar Secure Boot (Arranque Seguro)

¿Qué es?
Secure Boot es una característica de UEFI que impide que se cargue código malicioso durante el arranque. Solo permite que se ejecuten binarios firmados con claves autorizadas por el firmware.

¿Cómo protege contra rootkits?
Evita que rootkits de arranque (bootkits) o módulos del kernel no firmados se inyecten antes de que el sistema operativo arranque, donde podrían modificar el kernel y ocultar su presencia.

Cómo habilitarlo:

1. Ingresar al **BIOS/UEFI** (normalmente con F2, DEL o F10 al arrancar).

2. Activar la opción **Secure Boot**.

3. Asegurarse de que el sistema esté instalado con UEFI, no con BIOS heredado

B. Uso de SELinux / AppArmor

¿Qué es?

- **SELinux** (Security-Enhanced Linux) y **AppArmor** son mecanismos de control de acceso obligatorio (MAC) que refuerzan la seguridad del sistema.

- Limitan lo que los procesos pueden hacer, incluso si tienen permisos de root.

¿Cómo protege contra rootkits?
Un rootkit intenta acceder o modificar archivos críticos del sistema o cargar módulos maliciosos. SELinux puede impedir esto mediante **políticas** estrictas.

C. Verificación de integridad del kernel

¿Qué es?
Es el proceso de comparar archivos críticos del sistema (binarios, módulos, configuración) con versiones conocidas y confiables para detectar modificaciones maliciosas.

D. Kernel Module Signing (Firma de módulos)

¿Qué es?
Es una característica del kernel de Linux que permite firmar digitalmente los módulos del kernel para verificar su autenticidad antes de ser cargados.

¿Cómo protege?
Solo permite cargar módulos que estén **firmados** con una clave reconocida por el sistema. Rootkits que intenten cargarse como módulos sin firmar fallarán.

E. Sistemas de detección de rootkits (IDS)

¿Qué son?
Sistemas que escanean el sistema buscando patrones comunes de rootkits.

F. Deshabilitar carga dinámica de módulos

¿Qué es?
Permite bloquear la capacidad de cargar nuevos módulos al kernel después de arrancar el sistema.

¿Por qué es útil?
Después del arranque, un rootkit podría intentar cargarse como módulo (insmod, modprobe). Esta técnica lo impide por completo.

---

3. Desenlace

3.1 Conclusión

En un mundo donde las amenazas cambian constantemente, proteger el núcleo del sistema operativo no es opcional: es esencial. Si un atacante logra comprometer el kernel, tiene el control total del sistema, puede ocultar su presencia y desactivar cualquier medida de defensa. Para evitar este tipo de ataques, no basta con una sola solución; necesitamos un enfoque en capas.

Esto incluye arrancar solo software verificado con Secure Boot, exigir la firma digital de los módulos del kernel, usar tecnologías como IMA y TPM para verificar la integridad del sistema, y activar el modo de bloqueo del kernel para limitar acciones peligrosas.

REFERENCIAS.

- Geeknetic. (s.f.). *¿Qué es el Kernel y para qué sirve?* Geeknetic. https://www.geeknetic.es/Kernel/que-es-y-para-que-sirve
- ARMO. (s.f.). *Linux Kernel*. Armosec. https://www.armosec.io/glossary/linux-kernel/
- Fortinet. (s.f.). *Rootkit*. Fortinet. https://www.fortinet.com/lat/resources/cyberglossary/rootkit
- Free Software Foundation Europe. (s.f.). *El fiasco del rootkit de Sony*. FSFE. https://fsfe.org/activities/drm/sony-rootkit-fiasco.es.html
- Avast. (s.f.). *¿Qué es un rootkit y cómo eliminarlo?* Avast. https://www.avast.com/es-es/c-rootkit
- CrowdStrike. (s.f.). *Rootkits*. CrowdStrike. https://www.crowdstrike.com/en-us/cybersecurity-101/malware/rootkits/
- AVG. (s.f.). *¿Qué es un rootkit?* AVG. https://www.avg.com/es/signal/what-is-rootkit
- TechTarget. (s.f.). *Rootkit*. TechTarget. https://www.techtarget.com/searchsecurity/definition/rootkit