

SEGURIDAD EN EL KERNEL: COMO PROTEGER UN SISTEMA OPERATIVO CONTRA ROOTKITS

Sistemas operativos
Universidad Nacional Autónoma de México
Facultad de ingeniería
Meléndez Gómez Anuar
Zambrano Serrano Héctor

Start

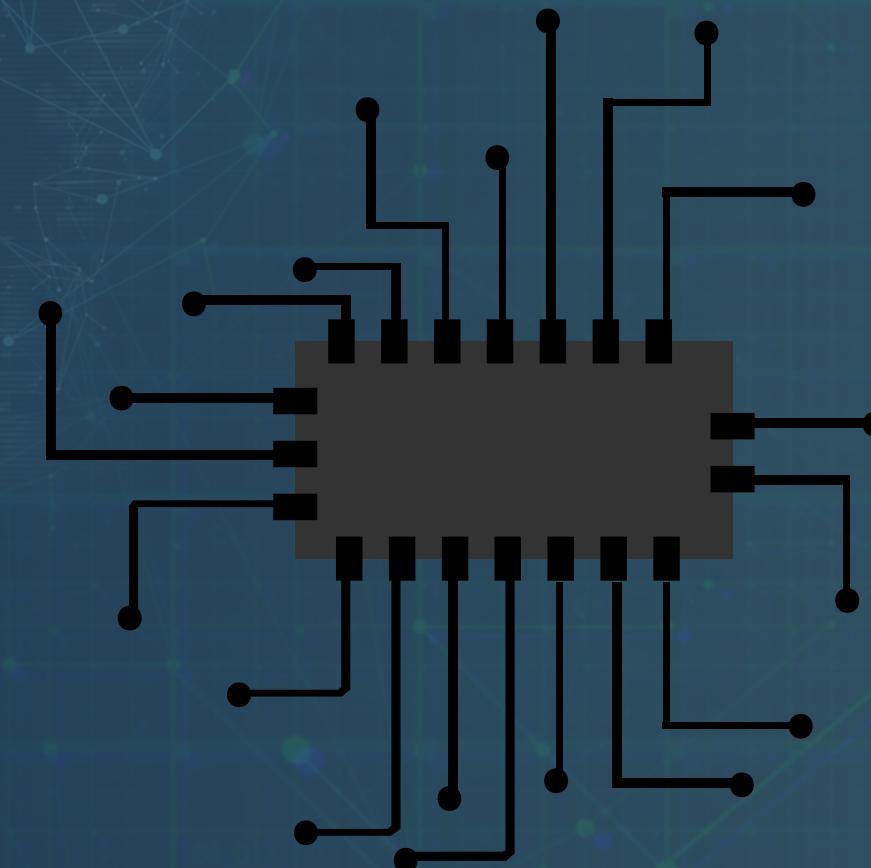
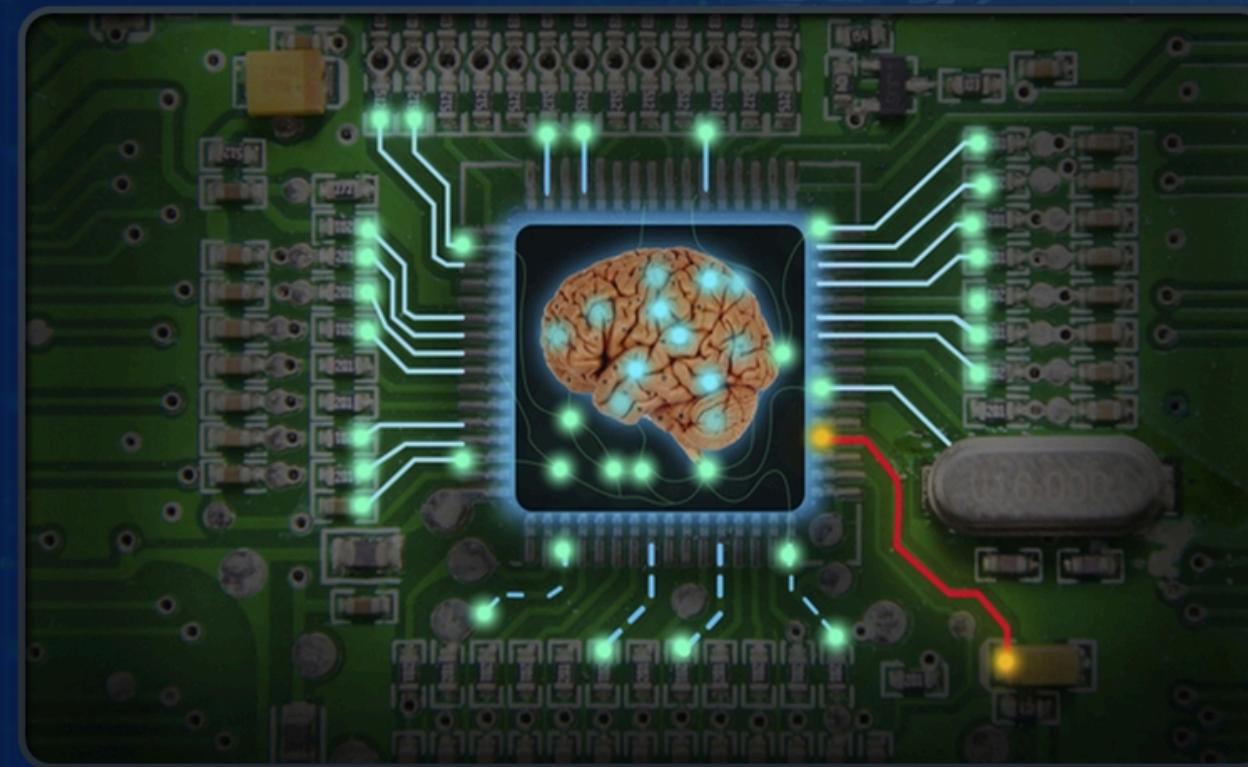


¿QUÉ ES UN KERNEL?

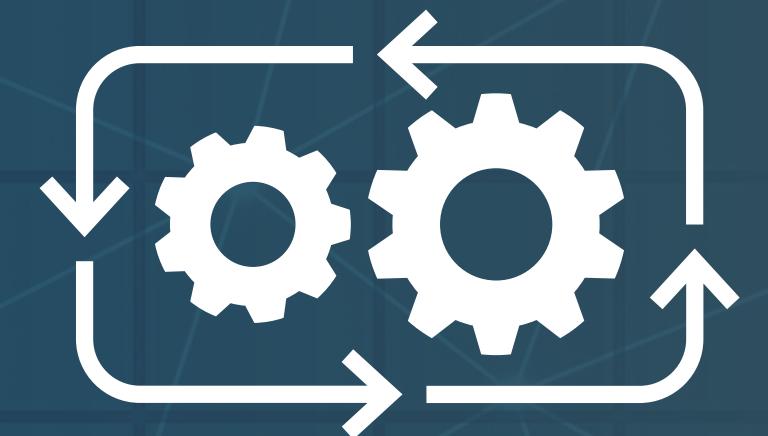


El kernel es el núcleo del sistema operativo (SO). Actúa como puente entre el hardware y las aplicaciones que usamos diariamente. Su función es gestionar la memoria, los procesos, el almacenamiento, la red y los dispositivos de entrada/salida. Dado su rol crítico, un ataque al kernel compromete todo el sistema operativo.

Uno de los ataques más sigilosos y peligrosos son los **rootkits**.



FUNCIONES PRINCIPALES DEL KERNEL

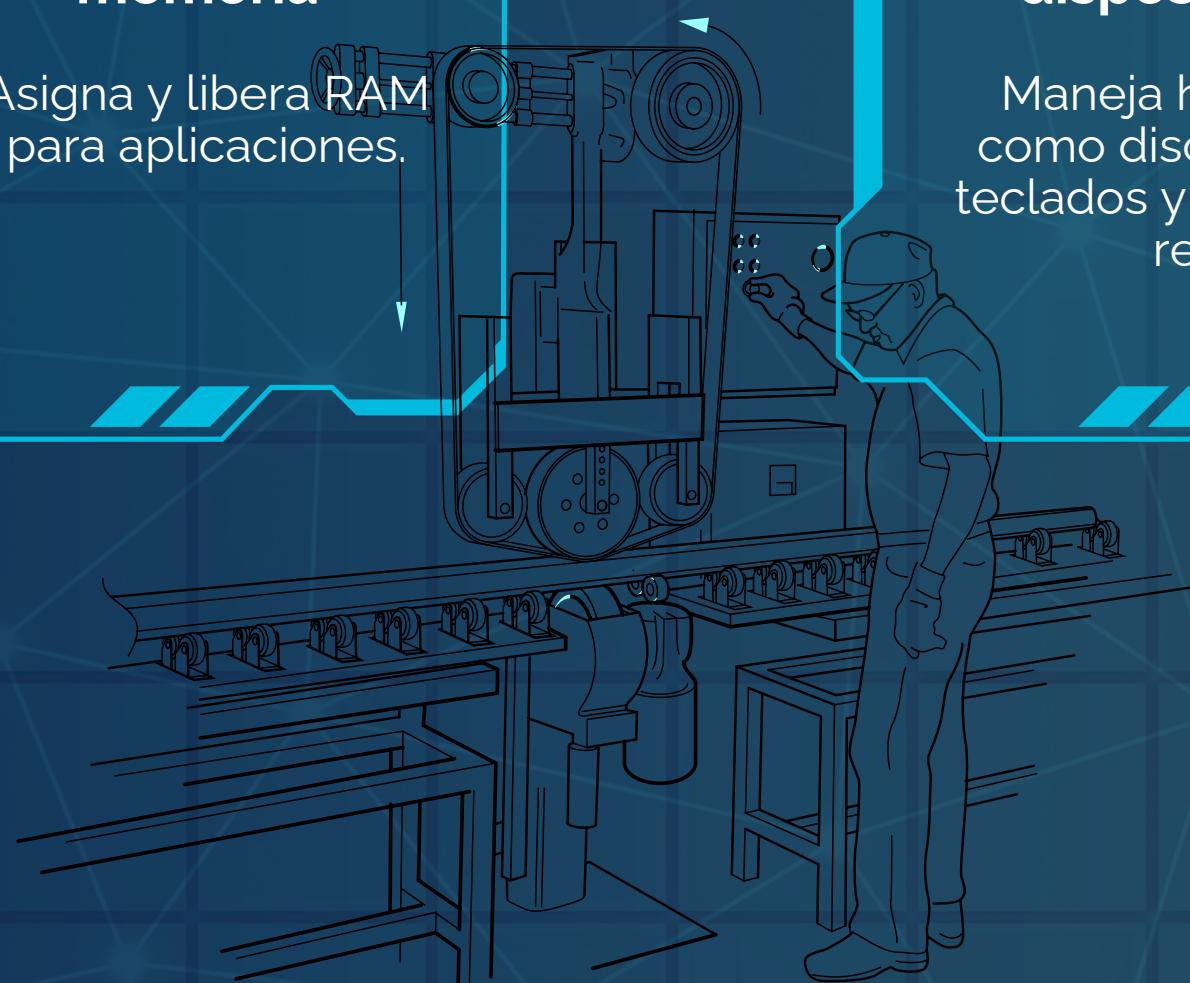


1. Gestión de procesos
Decide qué programas se ejecutan y cuándo.

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.



EJEMPLO DE GESTIÓN EN UN KERNEL

Supongamos que en Python requerimos leer un archivo

```
1 ✓ with open("datos.txt", "r") as archivo:  
2     |     contenido = archivo.read()  
3
```

¿Qué pasa detrás de escenas?

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.



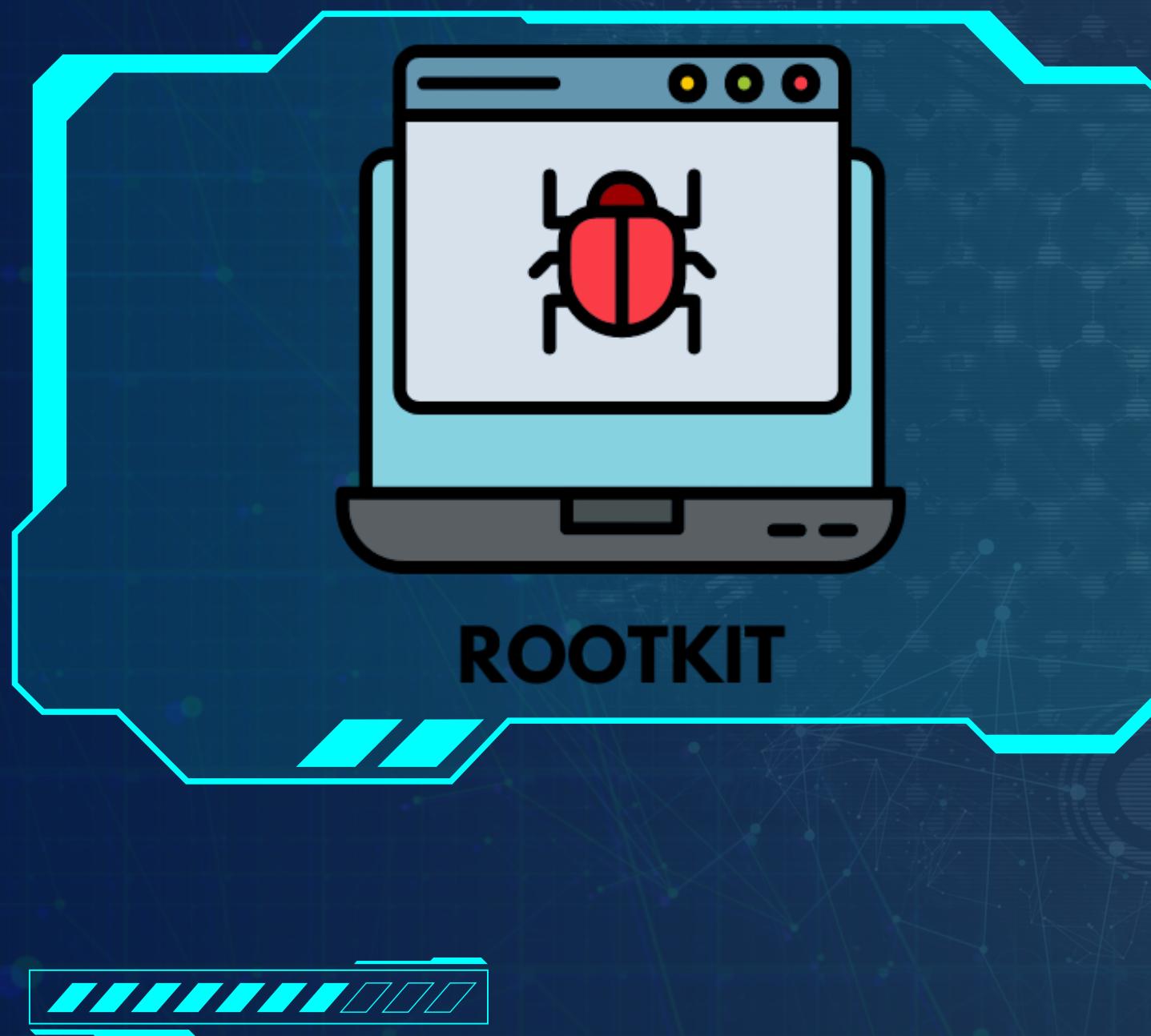
¿QUÉ SON LOS ROOTKITS?

Un rootkit es un software malicioso que permite el acceso no autorizado a un sistema mientras oculta su presencia. El término combina "root" (usuario con privilegios máximos en sistemas Unix/Linux) y "kit" (conjunto de herramientas).





TIPOS DE ROOTKITS



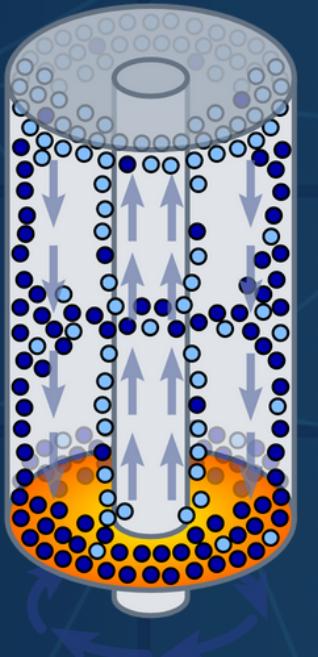
Tipos de rootkits:

- **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. (ej: Rovnix: Bootkit que robaba credenciales bancarias).
- **Firmware:** Se alojan en hardware (ej: BIOS).

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).



EJEMPLOS CONOCIDOS

- **Zeus (2007):** Botnet con rootkit para robo bancario.
- **DarkComet (2012):** RAT (Remote Access Trojan) con capacidades de rootkit para espionaje.





ESTRATEGIAS DE DEFENSA

1.

- Uso de herramientas antimalware especializadas. (en G MER Windows)

2.

- Monitoreo de comportamiento anómalo.

3.

- Actualizaciones y parches de seguridad.





CÓMO AFECTA UN ROOTKIT A UN SISTEMA OPERATIVO



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:



Exploits de vulnerabilidades

Fallos en drivers o servicios con privilegios



Inyección de código en procesos legítimos

Secuestro de DLLs en Windows



Modificación del arranque (Bootkit)

Infecta el gestor de arranque (GRUB, Windows Boot Manager).

MECANISMOS DE OCULTACIÓN EN ROOTKITS



1. HOOKING DE LLAMADAS AL SISTEMA

- **QUÉ HACE:** INTERCEPTA Y MODIFICA LAS LLAMADAS ENTRE APLICACIONES Y EL KERNEL.
- **OBJETIVO:** OCULTAR PROCESOS, ARCHIVOS O CONEXIONES MALICIOSAS.
- **DETECCIÓN:** HERRAMIENTAS COMO PROCESS Hacker (WINDOWS) O STRACE (LINUX) MONITORIZAN LLAMADAS ANÓMALAS.

```
1 #define _GNU_SOURCE
2 #include <dlfcn.h>
3 #include <dirent.h>
4 #include <string.h>
5
6 // Función original (se cargará dinámicamente)
7 typedef struct dirent* (original_readdir_t)(DIR);
8 original_readdir_t original_readdir;
9
10 // Función hookeadas
11 struct dirent* readdir(DIR* dirp) {
12     original_readdir = (original_readdir_t)dlsym(RTLD_NEXT, "readdir");
13     struct dirent* entry;
14
15     while ((entry = original_readdir(dirp)) != NULL) {
16         // Ocultar archivo llamado "malware.txt"
17         if (strstr(entry->d_name, "malware.txt") == NULL) {
18             return entry; // Mostrar todos los archivos EXCEPTO "malware.txt"
19         }
20     }
21     return NULL;
22 }
23
24 }
```

OBTENDREMOS UNA SALIDA COMO LA SIGUIENTE

```
gcc -shared -fPIC -o hook_readdir.so hook_readdir.c -ldl
LD_PRELOAD=./hook_readdir.so ls /ruta/del/directorio # ;"malware.txt" no aparecerá!
```



OCULTANDO EL ARCHIVO MALWERE.TXT DE CUALQUIER LISTADO DE LS

MECANISMOS DE OCULTACIÓN EN ROOTKITS



DKOM (DIRECT KERNEL OBJECT MANIPULATION)

- **QUÉ HACE:** MANIPULA ESTRUCTURAS DE DATOS DEL KERNEL EN MEMORIA RAM (SIN MODIFICAR DISCO).
- **OBJETIVO:** OCULTAR PROCESOS, DRIVERS O PUERTOS ABIERTOS.

```
PROCESS fffffaa8d`3b4e9080
SessionId: 1 Cid: 0b50 Peb: 00abf000 ParentCid: 0a88
DirBase: 1f4b7000 ObjectTable: fffffc987`1a456d40 HandleCount: 1032.
Image: explorer.exe
```

```
PROCESS fffffaa8d`3c1a2300
SessionId: 1 Cid: 0d20 Peb: 00dff000 ParentCid: 0b50
DirBase: 2a3c2000 ObjectTable: fffffc987`1a12a880 HandleCount: 215.
Image: chrome.exe
```

```
PROCESS fffffaa8d`3d5b7080
SessionId: 0 Cid: 04d8 Peb: 00000000 ParentCid: 03a0
DirBase: 0b1e5000 ObjectTable: fffffc987`0e7f9080 HandleCount: 0.
Image: svchost.exe
```

¿CÓMO FUNCIONA?

EN WINDOWS: LOS PROCESOS ACTIVOS SE ALMACENAN EN UNA LISTA ENLAZADA LLAMADA EPROCESS. UN ROOTKIT PUEDE MODIFICAR ESTA LISTA EN MEMORIA PARA "DESENLAZAR" UN PROCESO Y OCULTARLO.

EN LINUX: ALTERA LA LISTA TASK_STRUCT.
DETECCIÓN:
ANÁLISIS DE MEMORIA CON VOLATILITY



MECANISMOS DE OCULTACIÓN EN ROOTKITS



- ENMASCARAMIENTO DE PROCESOS Y ARCHIVOS
 - **OBJETIVO:** EVITAR SOSPECHAS EN HERRAMIENTAS COMO TASK MANAGER O LS.

¿CÓMO FUNCIONA?

EL ENMASCARAMIENTO ES UN CONJUNTO DE TÉCNICAS QUE PERMITEN A MALWARE OCULTARSE O DISFRAZARSE COMO ELEMENTOS LEGÍTIMOS DEL SISTEMA. AQUÍ TE EXPLICO CÓMO FUNCIONA CADA MÉTODO A NIVEL TÉCNICO:

1. PROCESS INJECTION (INYECCIÓN DE PROCESOS).
2. Timestomping (MANIPULACIÓN DE FECHAS).
3. NOMBRES ENGAÑOSOS.



EL INCIDENTE DEL ROOTKIT DE SONY BMG (2005)



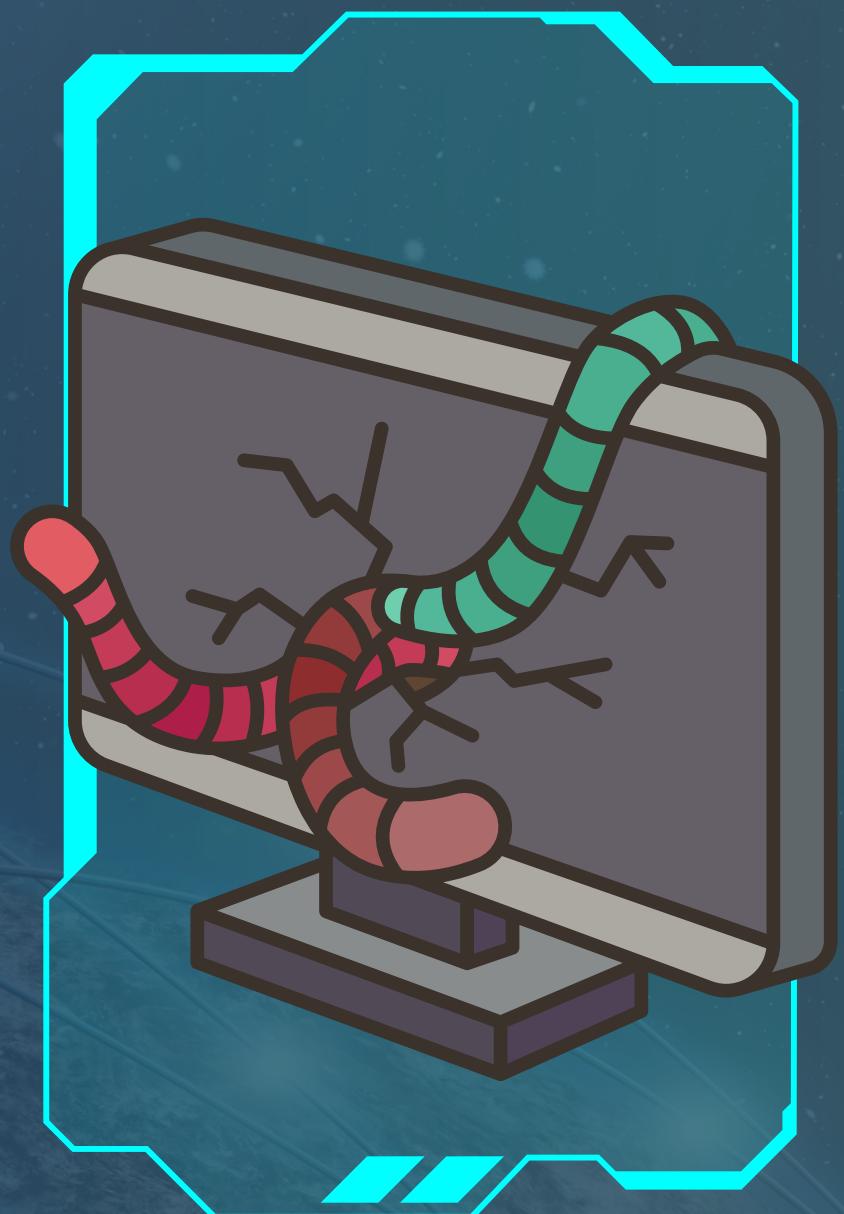
En 2005, Sony BMG (una de las mayores discográficas del mundo) lanzó CDs de música con un rootkit oculto, lo que generó uno de los mayores escándalos de seguridad y privacidad en la historia de la tecnología. Este caso es un ejemplo clásico de DRM (Gestión Digital de Restricciones) mal implementado que derivó en un grave problema de ciberseguridad.

FUNCIONAMIENTO

Cuando el CD se insertaba en una PC con Windows, instalaba automáticamente un software DRM llamado XCP (Extended Copy Protection) o MediaMax.

Este software:

- Ocultaba archivos y procesos (comportamiento típico de un rootkit).
- Limitaba las copias del CD a solo 3.
- Monitoreaba los hábitos de escucha del usuario.
- No se podía desinstalar fácilmente (y dejaba vulnerabilidades críticas).



TÉCNICAS USADAS POR EL ROOTKIT DE SONY

1.

Ocultamiento en el Sistema

El rootkit modificaba el kernel de Windows para esconder cualquier archivo o proceso que empezara con "syssys". Esto permitía que el DRM de Sony no fuera detectado por el usuario o antivirus.

2.

Vulnerabilidades Introducidas

El rootkit desactivaba protecciones del sistema y permitía que otros malware explotaran la misma técnica para ocultarse, ademas de que 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. Sony finalmente lanzó un parche oficial, pero este también tenía agujeros de seguridad.

```
C:\> dir /a  
Volume in drive C has no label.  
Volume Serial Number is C407-3215
```

```
Directory of C:\
```

05/18/2023 06:37 AM	<DIR>	\$Recycle.Bin
02/25/2025 07:56 PM	112	bootTel.dat
05/17/2023 06:06 PM	<JUNCTION>	Documents and Settings [C:\Users]
04/30/2025 08:54 PM	12,288	DumpStack.log.tmp
05/02/2025 09:08 AM	6,802,497,536	hiberfil.sys
08/18/2023 08:31 PM	36	id.dat

No mostraba los archivos ocultos por el rootkit (\$sys\$drm.exe)

DESCUBRIMIENTO DEL ROOTKIT



El investigador Mark Russinovich (creador de Sysinternals y futuro CTO de Microsoft Azure) descubre el Rootkit usando Process Monitor, usada para analizar comportamiento oculto. Algunas de sus conclusiones fueron:

1. Persistencia:

- No había forma fácil de desinstalarlo.
- Intentar borrarlo manualmente podía dañar Windows.

2. Vulnerabilidades Críticas:

- Cualquier malware podía usar el prefijo "syssys" para esconderse.

Ademas, se llega a la conclusión de que el rootkit facilita ataques de malware. Ejemplo de ello fue un troyano llamado Backdoor.Ryknos usó el rootkit para ocultarse:

```
# Creaba un archivo "$sys$hacker.exe" que era invisible
Copy-Item "malware.exe" "$sys$hacker.exe"
```



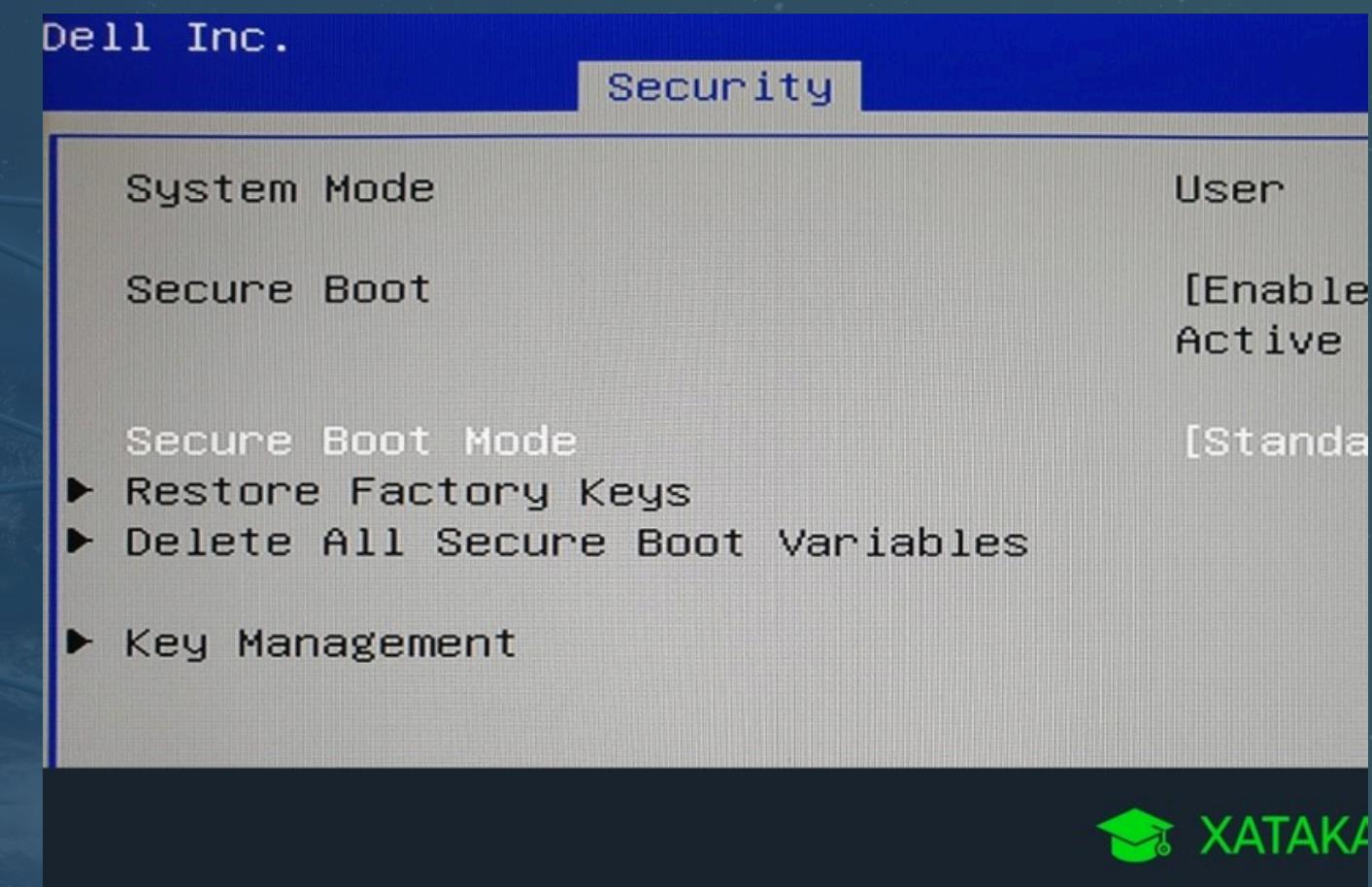
TÉCNICAS PARA PROTEGER EL KERNEL CONTRA ROOTKITS

A. HABILITAR SECURE BOOT (ARRANQUE SEGURO)



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

- Ingresar al BIOS/UEFI (normalmente con F2, DEL o F10 al arrancar).
- Activar la opción Secure Boot.
- Asegurarse de que el sistema esté instalado con UEFI, no con BIOS heredado.



B. USO DE SELINUX / APPARMOR

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

```
PS C:\Users\Lenovo> semanage fcontext -a -t boot_t "/boot(/.*)?"
```

```
PS C:\Users\Lenovo> restorecon -R /boot
```

Esto asegura que solo procesos autorizados puedan interactuar con /boot, donde un rootkit podría intentar persistir.



C. VERIFICACIÓN DE INTEGRIDAD DEL KERNEL



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

Herramientas comunes:

- AIDE (Advanced Intrusion Detection Environment)
- Tripwire

¿Qué detecta?

- Cambios en archivos de sistema, como /bin/login, /sbin/init, etc.
- Archivos o módulos añadidos como /lib/modules/x/xmalware.ko.



D. KERNEL MODULE SIGNING (FIRMA DE MÓDULOS)



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.

CREACION DE UN MODULO

1. Crear claves (solo una vez):

```
PS C:\Users\Lenovo> cd /usr/src/linux  
PS C:\Users\Lenovo> make keys
```

3. Verificar firma

```
PS C:\Users\Lenovo> modinfo mymodule.ko | grep -i signature
```

2. Compilar el módulo con firma:

```
PS C:\Users\Lenovo> make CONFIG_MODULE_SIG=y
```

E. SISTEMAS DE DETECCIÓN DE ROOTKITS (IDS)

¿Qué son?

Sistemas que escanean el sistema buscando patrones comunes de rootkits.

Herramientas populares:

- rkhunter
- chkrootkit

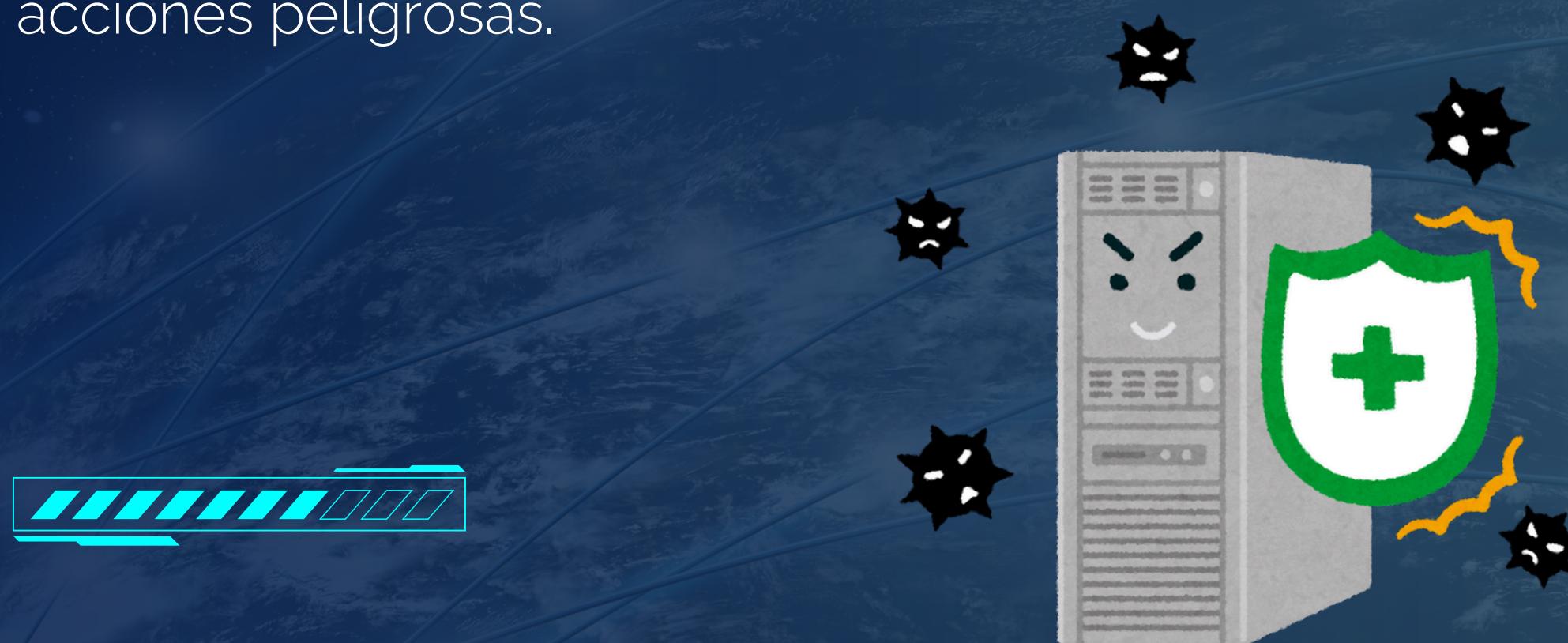
```
1 # Instalar rkhunter en Linux
2 sudo apt install rkhunter
3
4 # Actualizar base de datos
5 sudo rkhunter --update
6
7 # Ejecutar escaneo
8 sudo rkhunter --check
9
10 # Ver informe
11 cat /var/log/rkhunter.log
```

```
1 sudo apt install chkrootkit
2 sudo chkrootkit
```

CONCLUSION

En un entorno donde las amenazas evolucionan constantemente, proteger el kernel frente a rootkits se convierte en una prioridad: una vez que el núcleo del sistema operativo es comprometido, el atacante obtiene privilegios máximos, puede ocultar procesos, archivos y conexiones, y desactivar cualquier defensa tradicional.

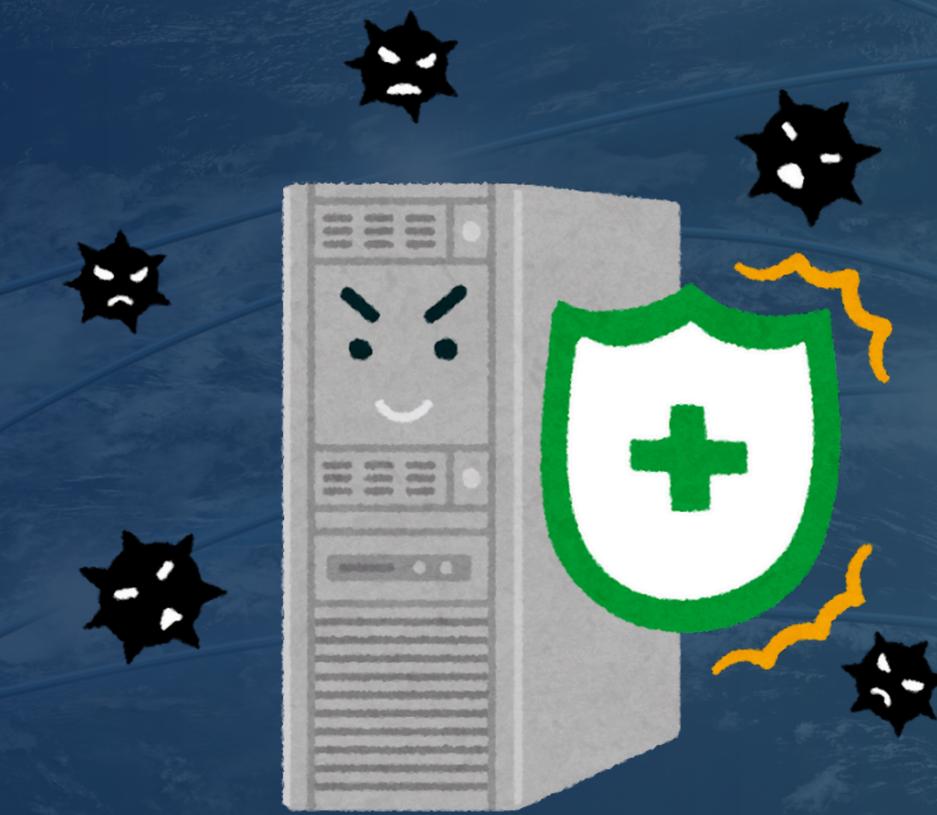
Para impedir este escenario catastrófico, es fundamental adoptar un enfoque multicapa que combine Secure Boot (garantizando que solo software firmado arranque), la firma digital obligatoria de módulos del kernel, mecanismos de medición y verificación de integridad como IMA junto con un TPM, y el modo de bloqueo del kernel para restringir acciones peligrosas.



CONCLUSION

A ello debe sumarse el hardening del kernel compilándolo con lo mínimo indispensable y desactivando interfaces de depuración, la monitorización activa con herramientas especializadas (rkhunter, chkrootkit, soluciones EDR) y un riguroso programa de actualizaciones para corregir vulnerabilidades tan pronto como se descubran.

Solo así podremos asegurar la confidencialidad, integridad y disponibilidad de nuestros sistemas, haciendo que incluso los ataques más sigilosos queden improbables y detectables antes de que causen un daño irreversible.





REFERENCIAS

CÓMO DETECTAR Y EVITAR LOS ROOTKITS. (2021, 30 MARZO). / [EL TEXTO DEL PÁRRAFO](#)

WHAT IS ROOTKIT? ATTACK DEFINITION & EXAMPLES | CROWDSTRIKE. (S. F.).
[EL TEXTO DEL PÁRRAFO](#)

FREDA, A. (2024, 30 JULIO). TODO LO QUE DEBE SABER SOBRE LOS ROOTKITS Y CÓMO PROTEGERSE. TODO LO QUE DEBE SABER SOBRE LOS ROOTKITS Y CÓMO PROTEGERSE. [EL TEXTO DEL PÁRRAFO](#)

ROBINSON, S., SHACKLETT, M. E., & ROSENCRANCE, L. (2025, 11 MARZO). WHAT IS A ROOTKIT? SEARCH SECURITY. [EL TEXTO DEL PÁRRAFO](#)

BURDOVA, C. (2022, 21 MARZO). ¿QUÉ ES UN ROOTKIT Y CÓMO SE ELIMINA? ¿QUÉ ES UN ROOTKIT Y CÓMO SE ELIMINA? [EL TEXTO DEL PÁRRAFO](#)

