VULNERABILIDADES EN PROGRAMAS Y APLICACIONES: PARTE 1

Autor: Brais López Yáñez

Objetivo de la práctica:

El objetivo de esta sesión es estudiar y explotar ciertas vulnerabilidades comunes presentes en códigos nativos. En concreto, se analizarán debilidades relacionadas con:

- Riesgos asociados al uso de ejecutables con SetUID activado.
- Inyección de comandos.
- Debilidades en el uso de variables de entorno y enlaces simbólicos.
- Condiciones de carrera.
- Uso de LD_PRELOAD.
- Desbordamiento de pila.

Los ejercicios están extraídos de la web: https://exploit-exercises.com/, donde están disponibles distintos programas y máquinas virtuales vulnerables.

Configuración e inicio de las máquinas virtuales:

Para los ejercicios 1.1-1.6 utilizaremos la máquina Nebula,

(https://exploit-exercises.com/nebula/), mientras que el ejercicio 1.7 se realizará en la máquina Protostar (https://exploit-exercises.com/protostar/). Estas máquinas pueden ejecutarse en modo Live-CD, y todos los ejercicios son realizables de este modo. Basta con crear una máquina virtual Linux sin unidad de almacenamiento, y añadir a continuación la imagen del Live-CD.

Ejercicio 1.1: https://exploit-exercises.com/nebula/level01/

Este ejercicio consiste en aprovechar una vulnerabilidad en el ejecutable "/home/flag01/flag01",cuyo código fuente está disponible, para ejecutar código arbitrario como el usuario 'flag01'. El alumno deberá hacer login con usuario 'level01' y password 'level01'. El objetivo de este ejercicio es ejecutar el comando 'getflag' como usuario 'flag01'.

Observando el código proporcionado por nebula, comprobamos que existe una vulnerabilidad en la llamada al comando echo. Aprovechamos este punto débil, y lo que hacemos es creamos un archivo que almacene el comando getflag. Una vez hacemos esto, cambialos el PATH por defecto, para que cuando ejecutemos la ruta específica se ejecute el fichero echo, y por lo tanto el comando getflat.

```
level01@nebula:~$ cat echo
getflag
level01@nebula:~$ export PATH=.:$PATH
level01@nebula:~$ chmod +x echo
level01@nebula:~$ /home/flag01/flag01
You have successfully executed getflag on a target account
level01@nebula:~$ _
```

Se podría evitar no haciendo la llamada al comando echo:

system("/usr/bin/env echo and now what?");

Ejercicio 1.2: https://exploit-exercises.com/nebula/level02/

Este ejercicio consiste en aprovechar una vulnerabilidad en el ejecutable "/home/flag02/flag02", cuyo código fuente está disponible, para ejecutar código arbitrario como el usuario 'flag02'. El objetivo es el mismo que en el ejercicio anterior, ejecutar el comando 'getflag' como usuario 'flag02'. Los datos de login para este ejercicio son level02/level02.Notas sobre el ejercicio 1.2:

Para este ejercicio queremos obtener el mismo resultado que el anterior, ejecutar el comando getflag. Para ello observaremos el código propuesto del nivel 2, estudiándolo minuciosamente nos damos cuenta que tiene un problema con la variable de entorno, ya que no está controlada.

Al saber esto, lo que vamos a hacer va a ser cambiar la variable del entorno, por el comando deseado, una vez tenemos esto, ya tenemos el ejercicio realizado y procedemos a ejecutar la ruta del ejecutable.

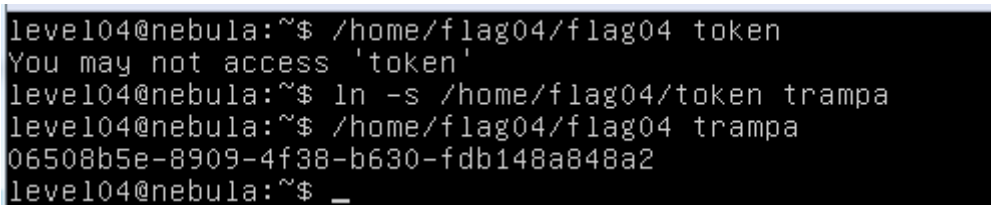
```
level02@nebula:~$ export USER="; getflag"
level02@nebula:~$ /home/flag02/flag02
about to call system("/bin/echo ; getflag is cool")

You have successfully executed getflag on a target account
level02@nebula:~$ _
```

Se podría evitar, controlando la variable de entorno USER.

Ejercicio 1.3: https://exploit-exercises.com/nebula/level04/

El objetivo de este ejercicio es leer el contenido del fichero "/home/flag04/token", para el que en principio no disponemos de permisos de lectura. Para ello deberemos explotar una vulnerabilidad en el ejecutable "/home/flag04/flag04". Los datos de login para este ejercicio son level04/level04.

Como podemos ver en el archivo presentado por Nebula, se impide el acceso a cualquier ruta que contenga la palabra token. Tras esto, descartamos cualquier posibilidad de hacerlo de forma directa. Para conseguir acceder al contenido del fichero token, lo que vamos a a hacer es crear un enlace simbólico, que nos permita acceder a token, engañando así, al sistema.



Se podría evitar controlando los accesos desde enlaces simbólicos, un ejemplo de mejora sería el siguiente ejercicio, en el que la vulnerabilidad es la condición de carrera que se produce.