Universidad Carlos III

Ingeniería de la Ciberseguridad

Curso 2024-25

Práctica 1

Extracción de contraseñas para binarios y permisos en ACL's

Ingeniería Informática, Cuarto curso

Adrián Fernández Galán (NIA: 100472182, e-mail: 100472182@alumnos.uc3m.es)
César López Mantecón (NIA: 100472092, e-mail: 100472092@alumnos.uc3m.es)
Manuel Gómez-Plana Rodríguez (NIA: 100472310, e-mail: 100472310@alumnos.uc3m.es)

Prof. Antonio Nappa
Grupo: 81

1. Introducción

En este documento se recoge el proceso de desarrollo de la primera práctica de la asignatura *Ingeniería de la Ciberseguridad*. En esta práctica hemos logrado obtener 9 *flags* mediante el descubrimiento de contraseñas y ataque a archivos binarios.

2. Estrategias para el descubrimiento de contraseñas

Para extraer las contraseñas de los ejecutables se han empleado 2 estrategias distintas: uso de *John the Ripper* para la obtención de contraseñas y ataque sobre los binarios.

2.1. John The Ripper

`John` es una herramienta para la obtención de contraseñas débiles a partir de su *hash*. Dado el conocimiento que teníamos de las contraseñas hemos generado una *wordlist* con todas las contraseñas posibles con el alfabeto proporcionado. Esto se reduce a las permutaciones de 5, 6 y 8 elementos de los conjuntos de caracteres usados para generar cada contraseña. De esta forma hemos podido realizar un ataque de diccionario sobre las contraseñas de los 3 archivos.

Para generar la *wordlist* se ha empleado el siguiente código de *python*:

```python
import itertools

# charset level 1
charset_level1 = "abcdefg123456lab"

# Generate all permutations of length 5
permutations = itertools.permutations(charset_level1, 5)

with open("level1_wordlist.txt", "w+") as file:
    # Print the result
    for p in permutations:
        file.write(''.join(p) + "\n")

# charset level 2
charset_level2 = "abcdefg123456uc3m"

# Generate all permutations of length 6
permutations = itertools.permutations(charset_level2, 6)

with open("level2_wordlist.txt", "w+") as file:
    # Print the result
    for p in permutations:
        file.write(''.join(p) + "\n")

# charset level 3
charset_level3 = "abcdefg123456profe"

# Generate all permutations of length 8
permutations = itertools.permutations(charset_level3, 8)

with open("level3_wordlist.txt", "w+") as file:
    # Print the result
    for p in permutations:
        file.write(''.join(p) + "\n")
```

Listing 1: Script para la generación de *wordlists*

Sin embargo, no ha sido posible generar la *wordlist* para *level3* debido gran número de contraseñas posibles y, consecuentemente, al gran tamaño del archivo.

Usando este método, hemos podido extraer las contraseñas para los archivos *level1* y *level2*. En la siguiente tabla se recogen los tiempos que ha llevado obtener cada contraseña:

Archivo	Longitud de la *wordlist* en líneas	Tiempo
level1	524160	53s
level2	8910720	383s
level3	1764322560	Not Finished

Cuadro 1: Tiempos por contraseña usando *john*

Para el archivo *level3* no hemos empleado esta estrategia debido al elevado tiempo de computación que precisa en comparación con la siguiente estrategia.

2.2. Explotación de binarios

Hemos empleado 3 herramientas distintas para el análisis y la explotación de binarios: `objdump`, `Ghidra` y `gdb`. Mediante la primera, hemos analizado y comprendido el funcionamiento del ejecutable; logrando identificar dos subrutinas de gran importancia: `strcmp` y `decode_password`. Esto lo hemos hecho a través del siguiente comando:

```
objdump -D level1 level2 level3
```

Con esto, hemos podido analizar el código ensamblador y descubrir que la función `decode_password` recibe un *string* y, a través de una serie de operaciones, lo transforma in-situ.

Ejecutando los programas sobre un entorno controlado con `gdb` hemos detenido la ejecución al final de la subrutina `decode_password` y leído el valor de la cadena de texto en memoria:

```
gdb level1                    # ejecucion en entorno gdb
(gdb) break decode_password   # breakpoint
(gdb) run                     # ejecucion hasta la rutina
(gdb) x/s $rdi                # contrasena cifrada
(gdb) finish                  # salto a la instruccion ret
(gdb) x/s $rdi                # obtener contrasenia en claro
0x7fffffffe48a: "l23bf"
(gdb) continue                # continuar ejecucion
Enter Password: l23bf
Correct! The flag is: 379
    b9e4d3e1590165d5bdb1b1a4ea61ca998f25546d1f5820394e98971098d84
[Inferior 1 (process 1409) exited normally]
```

Listing 2: Obtención de contraseña para level1 en gdb

Dado que los tres binarios siguen la misma estrategia, ha sido posible obtener todas las contraseñas mediante este procedimiento.

Adicionalmente se ha decompilado el programa usando `Ghidra` con el fin de comprender mejor su funcionamiento. De esta forma hemos podido obtener el pseudocódigo en `C` de la función `decode_password` y descubrir que la contraseña incluida en el binario está cifrada mediante el desplazamiento de 4 posiciones en la tabla ascii.

2.3. Comparación de métodos

Ambos métodos, *John* y la explotación de binarios, son estrategias completamente distintas con sus ventajas y desventajas.

El uso de *John* requiere el conocimiento del hash de la contraseña e, idealmente, algo de contexto sobre la misma para poder limitar el espacio de búsqueda. Sin embargo, dada la naturaleza del ataque, tiene un coste de computación que crece exponencialmente; llegando a tiempos de ejecución extremenadamente largos para contraseñas de media longitud (como es el caso de de la tercera contraseña).

La explotación de binarios, en contraste, tiene un coste de ejecución bajo. No obstante, require de conocimiento de la arquitectura para la que se ha compilado el archivo y, evidentemente, de acceso al propio archivo. Si un atacante tiene acceso a un binario suele significar que ha conseguido corromper una parte crítica del sistema.

Con todo lo anterior concluímos que un ataque basado en fuerza bruta, como es el caso de *John*, requiere un menor coste en conocimiento sobre la contraseña y objetivo del ataque; a cambio de un mayor coste computacional. Por otro lado, la explotación de binarios necesita del acceso al archivo en un entorno controlado y un mayor conocimiento sobre programación a bajo nivel y la arquitectura del sistema atacado.

3. ACL's

Como se especifica en la práctica, la máquina virtual cuenta con 3 ficheros que el usuario *rover* no tiene permiso para leer. Es por ello que, analizando la lista de ACL's de los siguientes ficheros, podemos ver quién tiene los permisos adecuados.

```
$ getfacl /challenge/flag.txt
# file: challenge/flag.txt
# owner: root
# group: root
user::rw-
user:rover:---
user:admin:r--
user:student1:---
group::r--
mask::r--
other::r--
```

```
$ getfacl /projects/subproject/flag.txt
# file: projects/subproject/flag.txt
# owner: root
# group: root
user::rw-
user:1004:r--
user:rover:---
user:student1:r--
group::r--
mask::r--
other::r--
```

```
$ getfacl /secret/flag.txt
# file: secret/flag.txt
# owner: root
# group: root
user::rw-
user:rover:---
user:admin:r--
group::r--
mask::r--
other::r--
```

Se puede apreciar como el usuario `rover` no cuenta con permisos suficientes para poder leer las *flags*. De hecho, al usuario `rover` le es imposible acceder a las *ACL's* de las flag contenidas en *subproject* y *secret*. Sin embargo, el usuario `admin` puede acceder a todos los archivos necesarios.

Con esto en mente, hemos iniciado sesión como usuario `admin`, adivinando su contraseña (`password`). Desde ese momento hemos podido acceder a todos los archivos necesarios y recuperar las 6 *flags*.

4. Conclusión

Esta práctica nos ha permitido explorar distintas formas de obtención de contraseñas según la información y las herramientas con las que contemos. También nos ha permitido profundizar en el control de acceso basado en *ACL's*, un sistema que permite administrar los permisos en un sistema con mayor granularidad. Otros conceptos interesantes que presenta la práctica son la vulnerabilidad de las contraseñas sencillas a ataques de fuerza bruta o *guessing*.

Cabe destacar que el desarrollo de esta práctica no sólo nos ha permitido afianzar los conocimientos de la asigntura. También nos ha invitado a reflexionar sobre los tipos de ataques que se pueden hacer a un sistema y cómo podríamos proteger nuestros programas y aplicaciones ante los mismos.

El mundo tiende a la digitalización, por lo que la ciberseguridad es uno de los pilares del desarrollo tecnológico en la actualidad. Conocer los tipos de ataque frente a contraseñas y la administración del control de acceso en los sistemas son piezas clave en nuestro desarrollo como ingenieros informáticos.