

Table of Contents

Lab Setup Instructions	2
Executing Commands in the Containers	3
Important	4
Finding the network interface of the attacker machine	4

Lab Setup Instructions

Along with each experiment there is a **Labsetup.zip** file, which contains the docker container configuration file required to set up the environment for each lab. Please use tis link to download the zip file for sniffing and spoofing lab.

https://seedsecuritylabs.org/Labs_20.04/Files/Sniffing_Spoofing/Labsetup.zip

After downloading and extracting the lab setup file move into that directory and run the following command:

docker-compose build

```
# docker-compose up
```

```
seed@VM: ~/Labsetup seed@VM: ~/Labsetup seed@VM: ~/Labsetup
[07/27/22]seed@VM:~/.../Labsetup$ docker-compose up
Creating network "net-10.9.0.0" with the default driver
Pulling attacker (handsonsecurity/seed-ubuntu:large)...
large: Pulling from handsonsecurity/seed-ubuntu
da7391352a9b: Pulling fs layer
14428a6d4bcd: Pulling fs layer
da7391352a9b: Downloading [>
]da7391352a9b: Downloading [=====>
]da7391352a9b: Downloading [=====>
]da7391352a9b: Downloading [=====>
]da7391352a9b: Downloading [=====>
]da7391352a9b: Download complete
da7391352a9b: Pull complete
14428a6d4bcd: Pull complete
2c2d948710f2: Pull complete
b5e99359ad22: Pull complete
3d2251ac1552: Pull complete
1059cf087055: Pull complete
b2afee800091: Pull complete
c2ff2446bab7: Pull complete
4c584b5784bd: Pull complete
Digest: sha256:41efab02008f016a7936d9cadfbe8238146d07c1c12b39cd63c3e73a0297c07a
Status: Downloaded newer image for handsonsecurity/seed-ubuntu:large
Creating seed-attacker ... done
Creating hostB-10.9.0.6 ... done
Creating hostA-10.9.0.5 ... done
Attaching to seed-attacker, hostB-10.9.0.6, hostA-10.9.0.5
hostA-10.9.0.5 | * Starting internet superserver inetd      [ OK ]
hostB-10.9.0.6 | * Starting internet superserver inetd      [ OK ]
```

After running the commands open a new terminal window using the shortcut **Ctrl + Shift + T** or using the GUI.

In the newly opened terminal check if all the required containers have been deployed successfully using the following command :

```
# docker ps
```

```
seed@VM: ~/.../Labsetup x seed@VM: ~/.../Labsetup x seed@VM: ~/.../Labsetup x seed@VM: ~/.../Labsetup x seed@VM: ~/.../Labsetup x
[07/27/22]seed@VM:~/.../Labsetup$ docker ps
CONTAINER ID        IMAGE                                     COMMAND                  CREATED
STATUS             PORTS                               NAMES                   UP
ef1990b7d089        handsonsecurity/seed-ubuntu:large       "bash -c ' /etc/init..." 8 minutes ago
Up 8 minutes                               hostA-10.9.0.5
3cdb62380e58        handsonsecurity/seed-ubuntu:large       "bash -c ' /etc/init..." 8 minutes ago
Up 8 minutes                               hostB-10.9.0.6
81030b329a90        handsonsecurity/seed-ubuntu:large       "/bin/sh -c /bin/bash"    8 minutes ago
Up 8 minutes                               seed-attacker
[07/27/22]seed@VM:~/.../Labsetup$ █
```

Executing Commands in the Containers

All docker containers will run in the background. To access the various containers we just created and execute the attacks in, we must find the container's id. This is done using the previous “**docker ps**” command.

The first column shows the container IDs for each of the running containers and the names column gives you a description of that container.

To access and execute the commands inside the container we will use the `docker exec` command or the `docksh` command.

We will use the container ID of the docker container that we would like to login to in the command given below ie. replace [**container ID**] in the commands below with the required container ID.

Commands :

```
# docker exec -it [ containerID ] /bin/bash
```

OR USE

```
# docksh [ container ID ]
```

```
[07/27/22]seed@VM:~/.../Labsetup$ docker exec -it 3c /bin/bash
root@3cdb62380e58:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.9.0.6 netmask 255.255.255.0 broadcast 10.9.0.255
    ether 02:42:0a:09:00:06 txqueuelen 0 (Ethernet)
    RX packets 64 bytes 9498 (9.4 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@3cdb62380e58:/#
```

Important

Within the container for the **attacker machine**, move to the **volumes** directory from the root directory before performing any of the experiments inside the containers. This stores the exploit files in the “**volumes**” folder inside the host VM. If not **all files that were created will be lost** when the containers are shut down.

Command:

```
# cd volumes/
```

Finding the Network Interface of the Attacker Machine

The interface of the attacker machine's network adapter is required for most tasks in all labs and is the one displayed along with the attacker machine's IP address. In most cases the attacker machine's IP address is **10.9.0.1** and its interface is generated dynamically.

To get this we run the command :

ifconfig

```
root@VM:/volumes# ifconfig
br-67588b77d6c0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.8.0.1 netmask 255.255.255.0 broadcast 10.8.0.255
    inet6 fe80::42:3ff:fe5d:193d prefixlen 64 scopeid 0x20<link>
    ether 02:42:03:5d:19:3d txqueuelen 0 (Ethernet)
    RX packets 354 bytes 23498 (23.4 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 353 bytes 105465 (105.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

br-7cb72c7cc646: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.9.0.1 netmask 255.255.255.0 broadcast 10.9.0.255
    inet6 fe80::42:46ff:feae:46f9 prefixlen 64 scopeid 0x20<link>
    ether 02:42:46:ae:46:f9 txqueuelen 0 (Ethernet)
    RX packets 696 bytes 120886 (120.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 86 bytes 11867 (11.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```