西南科技大学

网络攻击与防御

实验报告

实验题目：linux 系统基础学习

学生姓名：

学生学号：

一、 实验作业题目

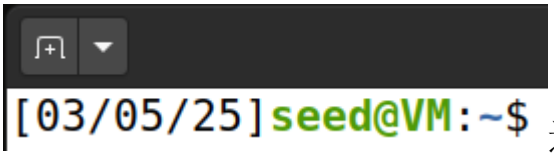
1. Linux 文件操作命令练习

2. Docker 操作和 Linux 网络操作命令练习

二、 实验思路

Linux 文件操作命令练习

1. 打开终端进入用户目录



进入了当前用户的主目录

2. 列出该目录下所有的文件和文件夹，如有 Desktop 文件夹则进入

```
[03/05/25]seed@VM:~$ ls -a
.               .fontconfig       Share
..              .gdbinit          .ssh
.bash_history   .gnupg            .sudo_as_admin_successful
.bash_logout    .local            Templates
.bashrc         .mozilla          .vboxclient-clipboard.pid
.cache          Music             .vboxclient-display-svga-x11.pid
.config         .pam_environment  .vboxclient-draganddrop.pid
Desktop         Pictures          .vboxclient-hostversion.pid
Documents       .profile          .vboxclient-seamless.pid
Downloads       Public            Videos
[03/05/25]seed@VM:~$ cd Desktop/
[03/05/25]seed@VM:~/Desktop$
```

输入 ls -a 显示当前目录下的所有文件和文件夹，输入 cd Desktop/ 进入 Desktop 文件夹

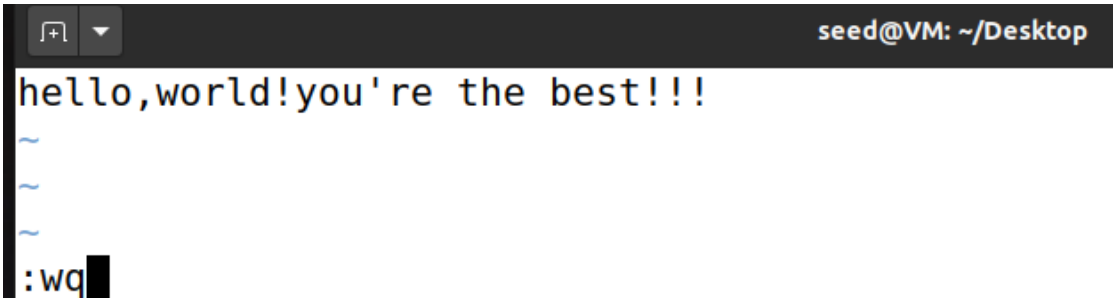
3. 创建一个名称为“test”的空文件

```
[03/05/25]seed@VM:~/Desktop$ touch test
```

创建名为 test 的文件

4. 使用 vim 或 gedit 编辑器进行编辑，加入任意内容

输入命令 vim test 进入编辑 test 文件，按“i”输入内容



输入完成后，按“Esc”后输入“:wq”保存并退出

5. 保存后，用 cat 命令输出该文件的内容

```
[03/05/25]seed@VM:~/Desktop$ cat test
hello,world!you're the best!!!
```

6. 创建一个空文件夹“d1”

```
[03/05/25]seed@VM:~/Desktop$ mkdir d1
```

输入 mkdir d1 创建空文件夹

7. 将 test 移入 d1 中，并重命名为 test2

输入“mv test d1/test2”，即 test 移动到路径为 d1/test2

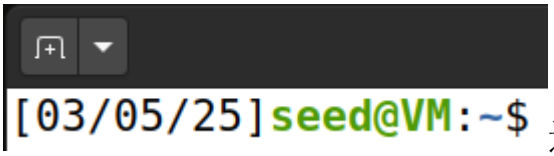
```
[03/05/25]seed@VM:~/Desktop$ mv test d1/test2
```

8. 使用 ls 命令查看操作是否成功

```
[03/05/25]seed@VM:~/Desktop$ ls
d1
[03/05/25]seed@VM:~/Desktop$ ls ./d1
test2
```

Docker 操作和 Linux 网络操作命令练习

1. 打开终端进入用户目录



进入了当前用户的主目录

2. 使用 cp 命令将共享文件夹 Share 中的 Labsetup 文件夹（请提前解压好并放在自定义指定的共享文件夹目录下）拷贝至 Desktop 上

查看共享文件夹是否设置成功：

```
[03/05/25]seed@VM:~/Desktop$ cd /mnt/hgfs
[03/05/25]seed@VM:.../hgfs$ ls
seed-share
```

拷贝文件：使用 cp -r 复制该目录和子目录到目标目录（~/Desktop）

```
[03/05/25]seed@VM:.../seed-share$ cp -r Labsetup ~/Desktop/
```

3. 进入 Desktop/Labsetup，查看其下的文件

```
[03/05/25]seed@VM:.../seed-share$ cd ~/Desktop/Labsetup/
[03/05/25]seed@VM:~/.../Labsetup$ ls -a
.  ..  docker-compose.yml  volumes
```

发现有文件 docker-compose.yml 和 volumes

4. 输出 docker-compse.yml 文件内容，尝试分析其结构

```
[03/05/25]seed@VM:~/.../Labsetup$ cat docker-compose.yml
version: "3"

services:
    attacker:
        image: handsonsecurity/seed-ubuntu:large
        container_name: seed-attacker
        tty: true
        cap_add:
                - ALL
        privileged: true
        volumes:
                - ./volumes:/volumes
        network_mode: host

    hostA:
        image: handsonsecurity/seed-ubuntu:large
        container_name: hostA-10.9.0.5
        tty: true
        cap_add:
                - ALL
        networks:
            net-10.9.0.0:
                ipv4_address: 10.9.0.5
        command: bash -c "
                      /etc/init.d/openbsd-inetd start  &&
                      tail -f /dev/null
                 "

    hostB:
        image: handsonsecurity/seed-ubuntu:large
        container_name: hostB-10.9.0.6
        tty: true
        cap_add:
                - ALL
        networks:
            net-10.9.0.0:
                ipv4_address: 10.9.0.6
        command: bash -c "
                      /etc/init.d/openbsd-inetd start  &&
                      tail -f /dev/null
                 "

networks:
    net-10.9.0.0:
        name: net-10.9.0.0
        ipam:
            config:
                - subnet: 10.9.0.0/24
```

version：Compose 文件的版本为 3

services：定义应用中的所有服务，该文件有 attacker、hostA、

hostB 三个服务

image：服务使用的镜像，所有容器都使用相同的基础镜像

Container_name：容器名称

tty：分配一个伪终端

cap_add：添加所有权限到容器

volumes：挂载的本地目录

networks：定义服务使用的网络。attacker 容器使用主机网络模式，而两个 host 容器使用自定义网络并分配了固定的 IP 地址，自定义网络的子网掩码为 10.9.0.0/24。

5. 输入 docker-compose build（或 dcbuild 命令）查看运行结果

```
[03/05/25]seed@VM:~/.../Labsetup$ docker-compose build
attacker uses an image, skipping
hostA uses an image, skipping
hostB uses an image, skipping
```

6. 输入 docker-compose up（或 dcup 命令）

```
[03/05/25]seed@VM:~/.../Labsetup$ docker-compose up
Creating network "net-10.9.0.0" with the default driver
Pulling attacker (handsonsecurity/seed-ubuntu:large)...
ERROR: Get https://registry-1.docker.io/v2/: net/http: request canceled whi
le waiting for connection (Client.Timeout exceeded while awaiting headers)
```

网络超时问题，参考课程资料解决

再次输入 dcup 启动容器

```
[03/05/25]seed@VM:~/.../Labsetup$ dcup
Pulling attacker (handsonsecurity/seed-ubuntu:large)...
large: Pulling from handsonsecurity/seed-ubuntu
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: sha256:41efab02008f016a7936d9cadfbe8238146d07c1c12b39cd63c3e73a0297
c07a
Status: Downloaded newer image for handsonsecurity/seed-ubuntu:large
Creating hostB-10.9.0.6 ... done
Creating hostA-10.9.0.5 ... done
Creating seed-attacker  ... done
Attaching to seed-attacker, hostA-10.9.0.5, hostB-10.9.0.6
hostA-10.9.0.5 |  * Starting internet superserver inetd                 [
OK ]
hostB-10.9.0.6 |  * Starting internet superserver inetd                 [
OK ]
```

创建和启动 seed-attacker、hostA-10.9.0.5 和 hostB-10.9.0.6

7. 新建一个终端，用 docker ps（或 dockps）查看容器开启状态

```
[03/05/25]seed@VM:~$ docker ps
CONTAINER ID        IMAGE                                COMMAND
 CREATED              STATUS              PORTS               NAMES
6113ea35285a        handsonsecurity/seed-ubuntu:large    "/bin/sh -c /bin/bash"
 About a minute ago   Up About a minute                       seed-attacker
34d39c4223c3        handsonsecurity/seed-ubuntu:large    "bash -c ' /etc/init…"
 About a minute ago   Up About a minute                       hostA-10.9.0.5
3aff49716360        handsonsecurity/seed-ubuntu:large    "bash -c ' /etc/init…"
 About a minute ago   Up About a minute                       hostB-10.9.0.6
```

列出了当前正在运行的 docker 容器的详细信息

8. 使用 docker exec -it <ID>（或 docksh <ID>）命令进入容器 A

由上图可知容器 A 的 ID 为 34d39c4223c3

```
[03/05/25]seed@VM:~$ docksh 34d39c4223c3
root@34d39c4223c3:/#
```

9. 使用 ifconfig 或 ip a 命令查看主机 eth0 网卡下的 IP 地址和 MAC 地址

```
[03/05/25]seed@VM:~$ ifconfig -a
br-b4bc9df1ecfd: 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::42:90ff:fe98:c826  prefixlen 64  scopeid 0x20<link>
        ether 02:42:90:98:c8:26  txqueuelen 0  (以太网)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 39  bytes 4816 (4.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:75:96:32:d4  txqueuelen 0  (以太网)
        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

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 61.139.2.137  netmask 255.255.255.0  broadcast 61.139.2.255
        inet6 fe80::106:d87b:30e4:81bf  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:e6:41:dc  txqueuelen 1000  (以太网)
        RX packets 76808  bytes 104836839 (104.8 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 24612  bytes 2029757 (2.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

eth0 网卡 IP 为 61.139.2.137，MAC 地址为 00:0c:29:e6:41:dc

10. 使用 ping 命令测试容器 A 和容器 B 之间的连通性

在容器 A 内 ping 容器 B 的 IP：10.9.0.6，能 ping 通

```
root@34d39c4223c3:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
64 bytes from 10.9.0.6: icmp_seq=1 ttl=64 time=0.153 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.076 ms
64 bytes from 10.9.0.6: icmp_seq=3 ttl=64 time=0.082 ms
64 bytes from 10.9.0.6: icmp_seq=4 ttl=64 time=0.101 ms
^C
--- 10.9.0.6 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3064ms
rtt min/avg/max/mdev = 0.076/0.103/0.153/0.030 ms
```

在容器 B 内 ping 容器 A 的 IP:10.9.0.5，能 ping 通

```
[03/05/25]seed@VM:~$ docksh 3aff49716360
root@3aff49716360:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=64 time=0.098 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=0.071 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=64 time=0.083 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=64 time=0.064 ms
^C
--- 10.9.0.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3057ms
rtt min/avg/max/mdev = 0.064/0.079/0.098/0.012 ms
```

11. 再新建一个终端，将桌面上的 d1 文件夹移到 Labsetup/volumes 文件夹中

```
[03/05/25]seed@VM:~$ mv ~/Desktop/d1 ~/Desktop/Labsetup/volumes/
```

12. 进入容器 M，并查看/volumes 下是否有 d1 和 test2

```
[03/05/25]seed@VM:~$ docksh 6113ea35285a
root@VM:/# ls /volumes/
d1
root@VM:/# ls /volumes/d1
test2
```

13. 删除 d1 及其目录下的 test2

```
root@VM:/# rm -rf /volumes/d1
root@VM:/# ls /volumes/
```

-rf 强制删除文件或目录

14. 使用 ctrl+C 分别退出容器 A 和容器 M 的终端

输入 exit 退出容器

```
root@VM:/# exit
exit
[03/05/25]seed@VM:~$
```

15. 使用 docker-compose down（或 dcdown）命令关闭所有容器。

在启动容器界面按 Ctrl+C 退出

```
^CGracefully stopping... (press Ctrl+C again to force)
Stopping seed-attacker  ... done
Stopping hostA-10.9.0.5 ... done
Stopping hostB-10.9.0.6 ... done
```

输入 dcdown 关闭所有容器

```
[03/05/25]seed@VM:~/.../Labsetup$ dcdown
Removing seed-attacker  ... done
Removing hostA-10.9.0.5 ... done
Removing hostB-10.9.0.6 ... done
Removing network net-10.9.0.0
```

三、 实验分析及总结（写自己在实践过程中遇到的问题及解决方法；本次实验体会、收获）

体会及收获：

本次实验通过 Linux 文件操作和 Docker 网络操作的实践，让我对命令行工具的使用有了更深入的理解和掌握。

在文件操作部分，我学会了如何使用`ls`、`cd`、`touch`、`vim`、`cat`、`mkdir`、`mv`等命令来创建、编辑、移动和查看文件及文件夹。

在 Docker 操作部分，我了解了如何使用`docker-compose`来管理容器化应用的生命周期，包括构建、启动、停止和删除容器。此外，我还学习了如何查看容器的网络配置，以及如何在容器之间测试网络连通性。

通过这次实验，我不仅提升了自己的技术能力，还增强了解决实际问题的能力。在未来的学习和工作中，我将继续深入探索这些工具的高级功能，以适应不断变化的技术需求。