西南科技大学

网络攻击与防御

实验报告

实验题目：数据包嗅探及欺骗

学生姓名：

学生学号：

一、 实验作业题目

1. 基本命令
2. 数据包的嗅探
3. 数据包的欺骗

二、 实验思路

基本命令

之前设置过共享文件夹开机自动挂载，所以挂载就不赘述了。

在桌面上 Labsetup 文件夹里打开终端

使用 docker-compose build 命令构造镜像

```
[03/05/25]seed@VM:~/.../Labsetup3$ ls
Labsetup
[03/05/25]seed@VM:~/.../Labsetup3$ cd Labsetup/
[03/05/25]seed@VM:~/.../Labsetup$ ls
docker-compose.yml  volumes
[03/05/25]seed@VM:~/.../Labsetup$ docker-compose build
attacker uses an image, skipping
hostA uses an image, skipping
hostB uses an image, skipping
```

使用 docker-compose up 命令启动镜像

```
[03/05/25]seed@VM:~/.../Labsetup$ docker-compose up
Creating network "net-10.9.0.0" with the default driver
Creating hostA-10.9.0.5 ... done
Creating seed-attacker  ... done
Creating hostB-10.9.0.6 ... 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 ]
```

使用 docker-compose down 关闭容器

```
[03/11/25]seed@VM:~/.../Labsetup$ docker-compose down
Removing hostB-10.9.0.6 ... done
Removing hostA-10.9.0.5 ... done
Removing seed-attacker  ... done
Removing network net-10.9.0.0
```

使用 docker ps 命令查看当前正在运行的 docker 容器的基本信

息，CONTAINER ID 是容器 ID， IMAGE 是该容器所使用的 Docker 镜像名称，COMMAND 是容器启动时执行的命令，CREATED 是容器创建的时间，STATUS 是容器的当前状态，PORTS 是容器内部端口与宿主机端口的映射关系，NAMES 是容器的名称

```
[03/05/25]seed@VM:~/.../Labsetup$ docker ps
CONTAINER ID        IMAGE                                COMMAND
 CREATED             STATUS              PORTS               NAMES
7edd20993214        handsonsecurity/seed-ubuntu:large   "bash -c ' /etc/init…"
 38 seconds ago      Up 36 seconds                           hostB-10.9.0.6
ed5469c63fc6        handsonsecurity/seed-ubuntu:large   "/bin/sh -c /bin/bash"
 38 seconds ago      Up 36 seconds                           seed-attacker
d91e2951c66e        handsonsecurity/seed-ubuntu:large   "bash -c ' /etc/init…"
 38 seconds ago      Up 36 seconds                           hostA-10.9.0.5
```

也可以 dockps 查看简单信息

```
[03/05/25]seed@VM:~/.../Labsetup$ dockps
7edd20993214  hostB-10.9.0.6
ed5469c63fc6  seed-attacker
d91e2951c66e  hostA-10.9.0.5
```

进入容器 A，查看 hostA 的 IP：10.9.0.5

```
[03/05/25]seed@VM:~/.../Labsetup$ docker exec -it d91e2951c66e /bin/bash
root@d91e2951c66e:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.5  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:05  txqueuelen 0  (Ethernet)
        RX packets 69  bytes 8258 (8.2 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
```

使用 exit 命令退出容器 A

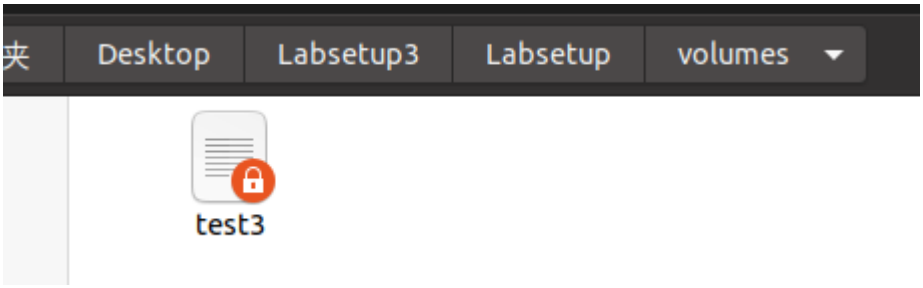
```
root@d91e2951c66e:/# exit
exit
```

用 docker 的 volumes 来实现文件夹共享

进入 attacker 容器中，在 volumes 文件夹中创建 test3

```
[03/05/25]seed@VM:~/Desktop$ dockps
7ac5a0ff895c  hostB-10.9.0.6
bea6c51affec  hostA-10.9.0.5
a83d02b5e4b3  seed-attacker
[03/05/25]seed@VM:~/Desktop$ docksh a
root@VM:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr  volumes
root@VM:/# cd volumes/
root@VM:/volumes# ls
root@VM:/volumes# touch test3
```

打开新的终端，进入桌面上的 Labsetup/volumes 文件夹，能看到 test3 存在



数据包的嗅探

使用 chmod 775 给文件权限，所有者具有读写执行权

```
[03/05/25]seed@VM:~/Desktop$ chmod 775 sniffer.py
```

输入 ip -br a 查看所有的网络接口，复制 br-15f1f62793f8

```
[03/11/25]seed@VM:~/.../Labsetup$ ip -br a
lo               UNKNOWN        127.0.0.1/8 ::1/128
ens33            UP             61.139.2.137/24 fe80::106:d87b:30e4:81bf/64
docker0          DOWN
br-15f1f62793f8  UP             10.9.0.1/24 fe80::42:9ff:febb:b3ae/64
vethe1be4a4@if20 UP             fe80::7453:faff:fe18:77e/64
veth6fac1e8@if22 UP             fe80::4c32:11ff:fe14:fe6a/64
```

修改 sniffer.py，修改 br-15f1f62793f8，保存并退出

```
pkts = sniff(iface='br-15f1f62793f8', filter='ip', prn=process_packet)
wrpcap("sniffer.cap",pkts)
print("packets have been saved!")
:wq
```

查看容器信息

```
[03/11/25]seed@VM:~/.../Labsetup$ dockps
d08ae81a8ed0  seed-attacker
3e85149f065f  hostA-10.9.0.5
4a22c0cc5e4b  hostB-10.9.0.6
```

打开两个终端，分别进入 attacker 和 host-A

```
[03/11/25]seed@VM:~/.../Labsetup$ docksh d
root@VM:/#
```

```
[03/11/25]seed@VM:~/.../Labsetup$ docksh 3
root@3e85149f065f:/#
```

运行脚本后，在容器 attacker 中 ping host-A（10.9.0.5），

```
root@VM:/# 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=0.184 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=0.281 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=64 time=0.281 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=64 time=0.273 ms
^C
--- 10.9.0.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3079ms
rtt min/avg/max/mdev = 0.184/0.254/0.281/0.040 ms
```

在脚本监听处看到已经捕获到的 ICMP 协议包

```
[03/11/25]seed@VM:~/Desktop$ sudo ./sniffer.py
IP: 10.9.0.1 --> 10.9.0.5 ttl: 64
            ICMP type: 8
---
IP: 10.9.0.5 --> 10.9.0.1 ttl: 64
            ICMP type: 0
---
IP: 10.9.0.1 --> 10.9.0.5 ttl: 64
            ICMP type: 8
---
IP: 10.9.0.5 --> 10.9.0.1 ttl: 64
            ICMP type: 0
---
IP: 10.9.0.1 --> 10.9.0.5 ttl: 64
            ICMP type: 8
---
IP: 10.9.0.5 --> 10.9.0.1 ttl: 64
            ICMP type: 0
---
IP: 10.9.0.1 --> 10.9.0.5 ttl: 64
            ICMP type: 8
---
```

嗅探 TCP 协议包：在 host-A 容器内输入 nc -lnv 10.9.0.5 6901，

监听 6901 端口

```
root@3e85149f065f:/# nc -lnv 10.9.0.5 6901
Listening on 10.9.0.5 6901
Connection received on 10.9.0.1 34648
```

在 attackers 容器内输入 nc -v 10.9.0.5 6901，出现 succeeded，说明握手成功

```
root@VM:/# nc -v 10.9.0.5 6901
Connection to 10.9.0.5 6901 port [tcp/*] succeeded!
```

查看脚本捕获到的 tcp 三次握手包

```
---
IP: 10.9.0.1 --> 10.9.0.5 ttl: 64
           TCP  port: 34648 --> 6901
           TCP  seq: 2680003714
           TCP  ack: 0
---
IP: 10.9.0.5 --> 10.9.0.1 ttl: 64
           TCP  port: 6901 --> 34648
           TCP  seq: 3638772625
           TCP  ack: 2680003715
---
IP: 10.9.0.1 --> 10.9.0.5 ttl: 64
           TCP  port: 34648 --> 6901
           TCP  seq: 2680003715
           TCP  ack: 3638772626
```

在 attacker 容器界面输入“nihao”（相当于发送 tcp 数据包）

```
root@3e85149f065f:/# nc -lnv 10.9.0.5 6901
Listening on 10.9.0.5 6901
Connection received on 10.9.0.1 34648
nihao!
```

在脚本监听处捕获到了 tcp 数据包

```
---
IP: 10.9.0.1 --> 10.9.0.5 ttl: 64
           TCP  port: 34648 --> 6901
           TCP  seq: 2680003715
           TCP  ack: 3638772626
           TCP  load: b'nihao!\n'
---
IP: 10.9.0.5 --> 10.9.0.1 ttl: 64
           TCP  port: 6901 --> 34648
           TCP  seq: 3638772626
           TCP  ack: 2680003722
```

在 attacker 容器中关闭 tcp 通道（ctrl+c），脚本监听到 tcp 包

```
IP: 10.9.0.1 --> 10.9.0.5 ttl: 64
           TCP  port: 34648 --> 6901
           TCP  seq: 2680003722
           TCP  ack: 3638772626
---
IP: 10.9.0.5 --> 10.9.0.1 ttl: 64
           TCP  port: 6901 --> 34648
           TCP  seq: 3638772626
           TCP  ack: 2680003723
---
IP: 10.9.0.1 --> 10.9.0.5 ttl: 64
           TCP  port: 34648 --> 6901
           TCP  seq: 2680003723
           TCP  ack: 3638772627
```

嗅探 UDP 协议包：在 host-A 容器内输入 nc -lnvu 10.9.0.5 6901，在 attackers 容器内输入 nc -vu 10.9.0.5 6901，出现 succeeded，说明握手成功

```
root@3e85149f065f:/# nc -lnvu 10.9.0.5 6901
Bound on 10.9.0.5 6901
Connection received on 10.9.0.1 37469
XXXXX
```

```
root@VM:/# nc -vu 10.9.0.5 6901
Connection to 10.9.0.5 6901 port [udp/*] succeeded!
```

查看脚本捕获到的 udp 包

```
IP: 10.9.0.1 --> 10.9.0.5 ttl: 64
           UDP  port: 37469 --> 6901
           UDP  load: b'X'
---
IP: 10.9.0.1 --> 10.9.0.5 ttl: 64
           UDP  port: 37469 --> 6901
           UDP  load: b'X'
---
IP: 10.9.0.1 --> 10.9.0.5 ttl: 64
           UDP  port: 37469 --> 6901
           UDP  load: b'X'
---
IP: 10.9.0.1 --> 10.9.0.5 ttl: 64
           UDP  port: 37469 --> 6901
           UDP  load: b'X'
---
IP: 10.9.0.1 --> 10.9.0.5 ttl: 64
           UDP  port: 37469 --> 6901
           UDP  load: b'X'
```

在 attacker 容器界面输入“nihao”（相当于发送 udp 数据包）

```
root@3e85149f065f:/# nc -lnvu 10.9.0.5 6901
Bound on 10.9.0.5 6901
Connection received on 10.9.0.1 37469
XXXXXnihao
```

脚本监听处捕捉到 udp 数据包

```
IP: 10.9.0.1 --> 10.9.0.5 ttl: 64
           UDP  port: 37469 --> 6901
           UDP  load: b'nihao\n'
```

数据包的欺骗

chmod 775 spoof.py

vi spoof.py，修改网络接口，只捕获源 host-A（10.9.0.5）的 ICMP 数据包

```
pkts = sniff(iface = 'br-15f1f62793f8', filter = 'icmp and src host 10.9.0.5', p
rn = spoof_pkt)
wrpcap("spoof.cap",pkts)
print("packets have been saved!")
:wq
```

sudo ./spoof.py 开启嗅探

在 host-A 中尝试 ping 1.1.1.1

```
root@bea6c51affec:/# ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=99 time=8.30 ms
64 bytes from 1.1.1.1: icmp_seq=1 ttl=127 time=364 ms (DUP!)
64 bytes from 1.1.1.1: icmp_seq=2 ttl=99 time=5.40 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=127 time=205 ms (DUP!)
^C
--- 1.1.1.1 ping statistics ---
2 packets transmitted, 2 received, +2 duplicates, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 5.400/145.616/363.556/149.638 ms
```

(DUP!)表示收到了重复的响应包，说明网络与 1.1.1.1 之间连通，但出现了重复包，因为该脚本会队源 ip 为 10.9.0.5 的 ICMP 包进行一个 ip 反转，所以导致请求包与响应包重复

```
[03/05/25]seed@VM:~/Desktop$ sudo ./spoof.py
Original Packet.........
Source IP:  10.9.0.5
Destination IP:  1.1.1.1
Spoofed Packet........
Source IP:  1.1.1.1
Destination IP:  10.9.0.5
Original Packet.........
Source IP:  10.9.0.5
Destination IP:  1.1.1.1
Spoofed Packet........
Source IP:  1.1.1.1
Destination IP:  10.9.0.5
```

Ping 8.8.8.8

```
root@3e85149f065f:/# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=99 time=32.5 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=127 time=51.8 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=99 time=15.1 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=127 time=51.9 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=3 ttl=99 time=17.0 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=127 time=52.5 ms (DUP!)
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, +3 duplicates, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 15.108/36.829/52.544/16.248 ms
```

```
Spoofed Packet........
Source IP:  8.8.8.8
Destination IP:  10.9.0.5
Original Packet.........
Source IP:  10.9.0.5
Destination IP:  8.8.8.8
Spoofed Packet........
Source IP:  8.8.8.8
Destination IP:  10.9.0.5
Original Packet.........
Source IP:  10.9.0.5
Destination IP:  8.8.8.8
Spoofed Packet........
Source IP:  8.8.8.8
Destination IP:  10.9.0.5
```

与 1.1.1.1 一致

Ping 10.9.0.6

```
root@3e85149f065f:/# 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=0.450 ms
64 bytes from 10.9.0.6: icmp_seq=1 ttl=99 time=23.6 ms (DUP!)
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.264 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=99 time=11.6 ms (DUP!)
^C
--- 10.9.0.6 ping statistics ---
2 packets transmitted, 2 received, +2 duplicates, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 0.264/8.998/23.646/9.628 ms
```

```
Source IP:  10.9.0.5
Destination IP:  10.9.0.6
Spoofed Packet........
Source IP:  10.9.0.6
Destination IP:  10.9.0.5
Original Packet.........
Source IP:  10.9.0.5
Destination IP:  10.9.0.6
Spoofed Packet........
Source IP:  10.9.0.6
Destination IP:  10.9.0.5
Original Packet.........
Source IP:  10.9.0.5
Destination IP:  10.9.0.6
Spoofed Packet........
Source IP:  10.9.0.6
Destination IP:  10.9.0.5
```

与 1.1.1.1 一致

Ping 10.9.0.1

```
Source IP:  10.9.0.1
Destination IP:  10.9.0.5
Original Packet.........
Source IP:  10.9.0.5
Destination IP:  10.9.0.1
Spoofed Packet........
Source IP:  10.9.0.1
Destination IP:  10.9.0.5
Original Packet.........
Source IP:  10.9.0.5
Destination IP:  10.9.0.1
Spoofed Packet........
Source IP:  10.9.0.1
Destination IP:  10.9.0.5
```

与 1.1.1.1 一致

Ping 10.9.0.9

```
root@3e85149f065f:/# ping 10.9.0.9
PING 10.9.0.9 (10.9.0.9) 56(84) bytes of data.
From 10.9.0.5 icmp_seq=1 Destination Host Unreachable
From 10.9.0.5 icmp_seq=2 Destination Host Unreachable
From 10.9.0.5 icmp_seq=3 Destination Host Unreachable
^C
--- 10.9.0.9 ping statistics ---
4 packets transmitted, 0 received, +3 errors, 100% packet loss, time 3049ms
pipe 4
```

网络不可达，监听脚本处无响应

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）

问题：使用 spoof.py 开启嗅探时，在 attacker 容器内 ping 指定 IP 地址无法监听到 ICMP 包

解决方法：查看 spoof.py 脚本，发现他存在“filter”过滤，只对源 ip 地址为 10.9.0.5 的 ICMP 包进行 IP 反转，所以我们在 host-A 内进行 ping 指定 IP 即可解决问题

体会及收获：

此次实验收获颇丰，深入理解了 Docker 相关命令的使用，熟练掌握了数据包的嗅探与欺骗技术的原理，还了解到了 TCP 三次握手的原理和过程，也意识到网络安全知识的复杂性和重要性，实践过程中需要保持严谨态度，不断调试和优化，才能实现预期效果。