VPN Lab: The Container Version

57118135 许皓钦

Task1

主机 U：10.9.0.5，主机 V：192.168.60.5，VPN 服务器：10.9.0.11

主机 U 和 VPN 服务器可以通：

```
root@87a10c76e584:/# ping 10.9.0.11
PING 10.9.0.11 (10.9.0.11) 56(84) bytes of data.
64 bytes from 10.9.0.11: icmp_seq=1 ttl=64 time=0.119 ms
64 bytes from 10.9.0.11: icmp_seq=2 ttl=64 time=0.045 ms
64 bytes from 10.9.0.11: icmp_seq=3 ttl=64 time=0.050 ms
64 bytes from 10.9.0.11: icmp_seq=4 ttl=64 time=0.061 ms
^C
--- 10.9.0.11 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3081ms
rtt min/avg/max/mdev = 0.045/0.068/0.119/0.029 ms
root@87a10c76e584:/#
```

VPN 服务器和主机 V 可以通：

```
root@3b6105386b7f:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=64 time=0.089 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=64 time=0.048 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=64 time=0.048 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=64 time=0.049 ms
^C
--- 192.168.60.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3054ms
rtt min/avg/max/mdev = 0.048/0.058/0.089/0.017 ms
root@3b6105386b7f:/#
```

主机 U 和主机 V 不能通：

```
root@87a10c76e584:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C
--- 192.168.60.5 ping statistics ---
20 packets transmitted, 0 received, 100% packet loss, time 19450
ms

root@87a10c76e584:/#
```

用 tcpdump 观察 eth0 接口，并在 U 上 ping VPN 服务器：

```
root@3b6105386b7f:/# tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
14:35:39.772084 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 1, length 64
14:35:39.772139 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 15, seq 1, length 64
14:35:40.784098 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 2, length 64
14:35:40.784117 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 15, seq 2, length 64
14:35:41.809821 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 3, length 64
14:35:41.809852 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 15, seq 3, length 64
14:35:42.831243 IP 10.9.0.5 > 10.9.0.11: ICMP echo request, id 15, seq 4, length 64
14:35:42.831275 IP 10.9.0.11 > 10.9.0.5: ICMP echo reply, id 15, seq 4, length 64
```

用 tcpdump 观察 eth1 接口，并在 V 上 ping 192.168.60.11：

```
root@3b6105386b7f:/# tcpdump -i eth1 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
14:36:56.130479 ARP, Request who-has 192.168.60.6 tell 192.168.60.5, length 28
14:37:11.078122 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 30, seq 1, length 64
14:37:11.078163 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, id 30, seq 1, length 64
14:37:12.080462 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 30, seq 2, length 64
14:37:12.080475 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, id 30, seq 2, length 64
14:37:13.104948 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 30, seq 3, length 64
14:37:13.104966 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, id 30, seq 3, length 64
14:37:14.127697 IP 192.168.60.5 > 192.168.60.11: ICMP echo request, id 30, seq 4, length 64
14:37:14.127714 IP 192.168.60.11 > 192.168.60.5: ICMP echo reply, id 30, seq 4, length 64
```

Task2

Task2.a

将第 16 行处代码改成自己的名字：

```
 1 #!/usr/bin/env python3
 2
 3 import fcntl
 4 import struct
 5 import os
 6 import time
 7 from scapy.all import *
 8
 9 TUNSETIFF = 0x400454ca
10 IFF_TUN   = 0x0001
11 IFF_TAP   = 0x0002
12 IFF_NO_PI = 0x1000
13
14 # Create the tun interface
15 tun = os.open("/dev/net/tun", os.O_RDWR)
16 ifr = struct.pack('16sH', b'hongj%d', IFF_TUN | IFF_NO_PI)
17 ifname_bytes  = fcntl.ioctl(tun, TUNSETIFF, ifr)
18
19 # Get the interface name
20 ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
21 print("Interface Name: {}".format(ifname))
22
23 while True:
24     time.sleep(10)
```

运行代码:

```
root@87a10c76e584:/volumes# chmod a+x tun.py
root@87a10c76e584:/volumes# tun.py
Interface Name: hongj0
```

重开一个终端，输入 ip address 命令，看到已经有自定义的接口:

```
3: hongj0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state
 DOWN group default qlen 500
    link/none
```

Task2.b

在原代码中 22 和 23 行加入手册所给代码:

```
 1 #!/usr/bin/env python3
 2
 3 import fcntl
 4 import struct
 5 import os
 6 import time
 7 from scapy.all import *
 8
 9 TUNSETIFF = 0x400454ca
10 IFF_TUN   = 0x0001
11 IFF_TAP   = 0x0002
12 IFF_NO_PI = 0x1000
13
14 # Create the tun interface
15 tun = os.open("/dev/net/tun", os.O_RDWR)
16 ifr = struct.pack('16sH', b'hongj%d', IFF_TUN | IFF_NO_PI)
17 ifname_bytes  = fcntl.ioctl(tun, TUNSETIFF, ifr)
18
19 # Get the interface name
20 ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
21 print("Interface Name: {}".format(ifname))
22 os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
23 os.system("ip link set dev {} up".format(ifname))
24
25 while True:
26    time.sleep(10)
```

再次运行代码并查看，发现自定义的接口已经分配了地址。

```
5: hongj0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global hongj0
       valid_lft forever preferred_lft forever
```

Task2.c

用手册所给 while 循环替换原代码：

```python
#!/usr/bin/env python3

import fcntl
import struct
import os
import time
from scapy.all import *

TUNSETIFF = 0x400454ca
IFF_TUN   = 0x0001
IFF_TAP   = 0x0002
IFF_NO_PI = 0x1000

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'hongj%d', IFF_TUN | IFF_NO_PI)
ifname_bytes  = fcntl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

while True:
# Get a packet from the tun interface
        packet = os.read(tun, 2048)
        if packet:
                ip = IP(packet)
                print(ip.summary())
```

运行代码并在 U 上 ping192.168.53.0/24 下的地址，代码出现响应，

但是 ping 不通，因为实际主机并不存在。

```
root@87a10c76e584:/volumes# tun.py
Interface Name: hongj0
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
```

```
seed@VM: ~/Desktop
root@87a10c76e584:/# ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
^C
--- 192.168.53.1 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6149ms

root@87a10c76e584:/#
```

运行代码并在 U 上 ping192.168.60.5，代码未响应，且 ping 不通，因为未添加对应路由。

```
root@87a10c76e584:/volumes# tun.py
Interface Name: hongj0
^CTraceback (most recent call last):
  File "./tun.py", line 27, in <module>
    packet = os.read(tun, 2048)
KeyboardInterrupt

root@87a10c76e584:/volumes#
```

```
root@87a10c76e584:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C
--- 192.168.60.5 ping statistics ---
8 packets transmitted, 0 received, 100% packet loss, time 7178ms
```

Task2.d

修改 while 循环如下：

```
while True:
# Get a packet from the tun interface
        packet = os.read(tun, 2048)
        if packet:
                pkt = IP(packet)
                print(pkt.summary())
                if ICMP in pkt:
                        newip=IP(src=pkt[IP].dst,dst=pkt[IP].src,ihl=pkt[IP].ihl)
                        newip.ttl=99
                        newicmp=ICMP(type=0,id=pkt[ICMP].id,seq=pkt[ICMP].seq)
                        if pkt.haslayer(Raw):
                                data=pkt[Raw].load
                                newpkt=newip/newicmp/data
                        else:
                                newpkt=newip/newicmp
                os.write(tun,bytes(newpkt))
```

运行代码并在 U 上 ping 192.168.53.0/24 下面的地址，观察到返回的是构造的报文，且有 IP/ICMP/Raw 三层。

```
root@87a10c76e584:/volumes# tun.py
Interface Name: hongj0
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
```

seed@VM: ~/Desktop

```
root@87a10c76e584:/# ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
64 bytes from 192.168.53.1: icmp_seq=1 ttl=99 time=1.58 ms
64 bytes from 192.168.53.1: icmp_seq=2 ttl=99 time=1.29 ms
64 bytes from 192.168.53.1: icmp_seq=3 ttl=99 time=1.47 ms
64 bytes from 192.168.53.1: icmp_seq=4 ttl=99 time=1.48 ms
^C
--- 192.168.53.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3013ms
rtt min/avg/max/mdev = 1.287/1.454/1.582/0.106 ms
root@87a10c76e584:/#
```

Task3

修改 tun，重命名为 client，修改代码如下：

```
19 # Get the interface name
20 ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
21 print("Interface Name: {}".format(ifname))
22 os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
23 os.system("ip link set dev {} up".format(ifname))
24 os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))
25
26 sock=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
27 SERVER_IP="10.9.0.11"
28 SERVER_PORT=9090
29 while True:
30 # Get a packet from the tun interface
31         packet = os.read(tun, 2048)
32         if packet:
33                 ip = IP(packet)
34                 print(ip.summary())
35                 sock.sendto(packet,(SERVER_IP,SERVER_PORT))
36
```

修改 tun，重命名为 server，修改代码如下：

```
# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))

server=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
SERVER_IP="0.0.0.0"
SERVER_PORT=9090
server.bind((SERVER_IP,SERVER_PORT))
while True:
        data,(ip,port)=server.recvfrom(2048)
        print("{}:{}-->{}:{}".format(ip,port,SERVER_IP,SERVER_PORT))
        pkt=IP(data)
        print("Inside:{}-->{}".format(pkt.src,pkt.dst))
```

在 U 上运行 client，并 ping 192.168.60.5，看到代码响应未发生变化，依然 ping 不通。

```
root@87a10c76e584:/volumes# client.py
Interface Name: hongj0
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
```

```
seed@VM: ~/Desktop
root@87a10c76e584:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C
--- 192.168.60.5 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4083ms

root@87a10c76e584:/#
```

在上一步的基础上，在 VPN 服务器运行 server，看到管道外部是 10.9.0.5->0.0.0.0，管道内部是 192.168.53.99->192.168.60.5。

```
root@3b6105386b7f:/volumes# server.py
Interface Name: hongj0
RTNETLINK answers: File exists
10.9.0.5:54843-->0.0.0.0:9090
Inside:192.168.53.99-->192.168.60.5
10.9.0.5:54843-->0.0.0.0:9090
Inside:192.168.53.99-->192.168.60.5
10.9.0.5:54843-->0.0.0.0:9090
Inside:192.168.53.99-->192.168.60.5
10.9.0.5:54843-->0.0.0.0:9090
Inside:192.168.53.99-->192.168.60.5
10.9.0.5:54843-->0.0.0.0:9090
Inside:192.168.53.99-->192.168.60.5
10.9.0.5:54843-->0.0.0.0:9090
Inside:192.168.53.99-->192.168.60.5
10.9.0.5:54843-->0.0.0.0:9090
```

Task4

首先打开 docker-compose.yml，确保 IP 转发处于打开状态。

```
sysctls:
        - net.ipv4.ip_forward=1
```

修改 server 代码如下：

```
19 # Get the interface name
20 ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
21 print("Interface Name: {}".format(ifname))
22 os.system("ip addr add 192.168.53.11/24 dev {}".format(ifname))
23 os.system("ip link set dev {} up".format(ifname))
24 #os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))
25
26 server=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
27 SERVER_IP="0.0.0.0"
28 SERVER_PORT=9090
29 server.bind((SERVER_IP,SERVER_PORT))
30 while True:
31         data,(ip,port)=server.recvfrom(2048)
32         print("{}:{}-->{}:{}".format(ip,port,SERVER_IP,SERVER_PORT))
33         pkt=IP(data)
34         print("Inside:{}-->{}".format(pkt.src,pkt.dst))
35         os.write(tun,data)
36         print("write")
```

运行 server，client，在 U 上 ping192.168.60.5，此时在 VPN 服务器上用 tcpdump -nni eth1 命令看到 eth1 接口收到返回。

```
root@3b6105386b7f:/# tcpdump -nni eth1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
16:09:31.088594 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 190, seq 6, length 64
16:09:31.088691 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 190, seq 6, length 64
16:09:32.111154 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 190, seq 7, length 64
16:09:32.111191 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 190, seq 7, length 64
16:09:33.137894 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 190, seq 8, length 64
16:09:33.137919 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 190, seq 8, length 64
16:09:34.162250 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 190, seq 9, length 64
16:09:34.162275 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 190, seq 9, length 64
16:09:35.183891 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 190, seq 10, length 64
16:09:35.183912 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 190, seq 10, length 64
```

Task5

修改 client 代码如下：

```
# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))

sock=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
SERVER_IP="10.9.0.11"
SERVER_PORT=9090
fds=[sock,tun]
while True:
        ready,_,_=select.select(fds,[],[])
        for fd in ready:
                if fd is sock:
                        data,(ip,port)=sock.recvfrom(2048)
                        pkt=IP(data)
                        print("From socket:{}-->{}".format(pkt.src,pkt.dst))
                        os.write(tun,data)
                if fd is tun:
                        packet=os.read(tun,2048)
                        if packet:
                                pkt=IP(packet)
                                print(pkt.summary())
                                sock.sendto(packet,(SERVER_IP,SERVER_PORT))
```

修改 server 代码如下：

```
# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))
os.system("ip addr add 192.168.53.11/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))
os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))

sock=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
SERVER_IP="0.0.0.0"
SERVER_PORT=9090
ip='10.9.0.5'
port=10000
sock.bind((SERVER_IP,SERVER_PORT))
fds=[sock,tun]
while True:
        ready,_,_=select.select(fds,[],[])
        for fd in ready:
                if fd is sock:
                        print("sock...")
                        data,(ip,port)=sock.recvfrom(2048)
                        print("{}:{}-->{}:{}".format(ip,port,SERVER_IP,SERVER_PORT))
                        pkt=IP(data)
                        print("Inside:{}-->{}".format(pkt.src,pkt.dst))
                        os.write(tun,data)
                if fd is tun:
                        print("tun...")
                        packet=os.read(tun,2048)
                        pkt=IP(packet)
                        print("Return:{}--{}".format(pkt.src,pkt.dst))
                        sock.sendto(packet,(ip,port))
```

运行 client，server，在 U 上 ping192.168.60.5，此时能够 ping 通，代码响应如下。

```
root@87a10c76e584:/volumes# client3.py
Interface Name: hongj0
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
From socket:192.168.60.5-->192.168.53.99
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
From socket:192.168.60.5-->192.168.53.99
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
From socket:192.168.60.5-->192.168.53.99
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
From socket:192.168.60.5-->192.168.53.99
IP / ICMP 192.168.53.99 > 192.168.60.5 echo-request 0 / Raw
From socket:192.168.60.5-->192.168.53.99
```

```
seed@VM: ~/Desktop
root@87a10c76e584:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=2.61 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=2.10 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=1.58 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=1.78 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=1.69 ms
^C
--- 192.168.60.5 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4016ms
rtt min/avg/max/mdev = 1.575/1.951/2.613/0.374 ms
root@87a10c76e584:/#
```

VPN 服务器上的响应如下：

```
root@3b6105386b7f:/volumes# server3.py
Interface Name: hongj0
RTNETLINK answers: File exists
sock...
10.9.0.5:59218-->0.0.0.0:9090
Inside:192.168.53.99-->192.168.60.5
tun...
Return:192.168.60.5--192.168.53.99
sock...
10.9.0.5:59218-->0.0.0.0:9090
Inside:192.168.53.99-->192.168.60.5
tun...
Return:192.168.60.5--192.168.53.99
sock...
10.9.0.5:59218-->0.0.0.0:9090
Inside:192.168.53.99-->192.168.60.5
```

Task6

保持 client 和 server 运行，在 U 上 telnet192.168.60.5，成功进入，观察到每输入一个字符，client 和 server 都有一个响应。断开 client 或 server 之一，继续在 telnet 输入，此时输入无响应，且 client 和 server 中只有未被断开的有响应，重新连上断开的 client 或 server，telnet 中之前输入的字符又再次出现了，因为之前输入的字符在缓冲区不断重发。

```
root@87a10c76e584:/# telnet 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
df189ce8ae4d login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

seed@df189ce8ae4d:~$ sdadadasdadadasdasadsad
```