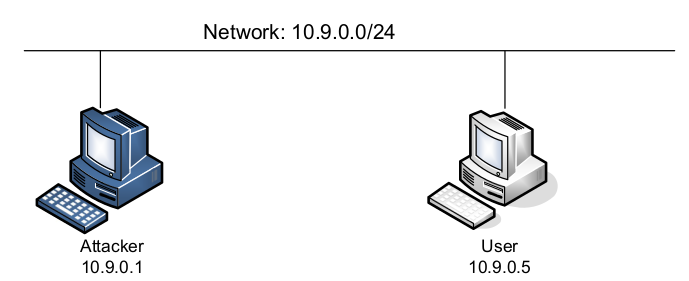
王嘉麟 57118130 lab1

网络结构：



**Task1.1A**

Attacker中，使用root执行#sniff.py；

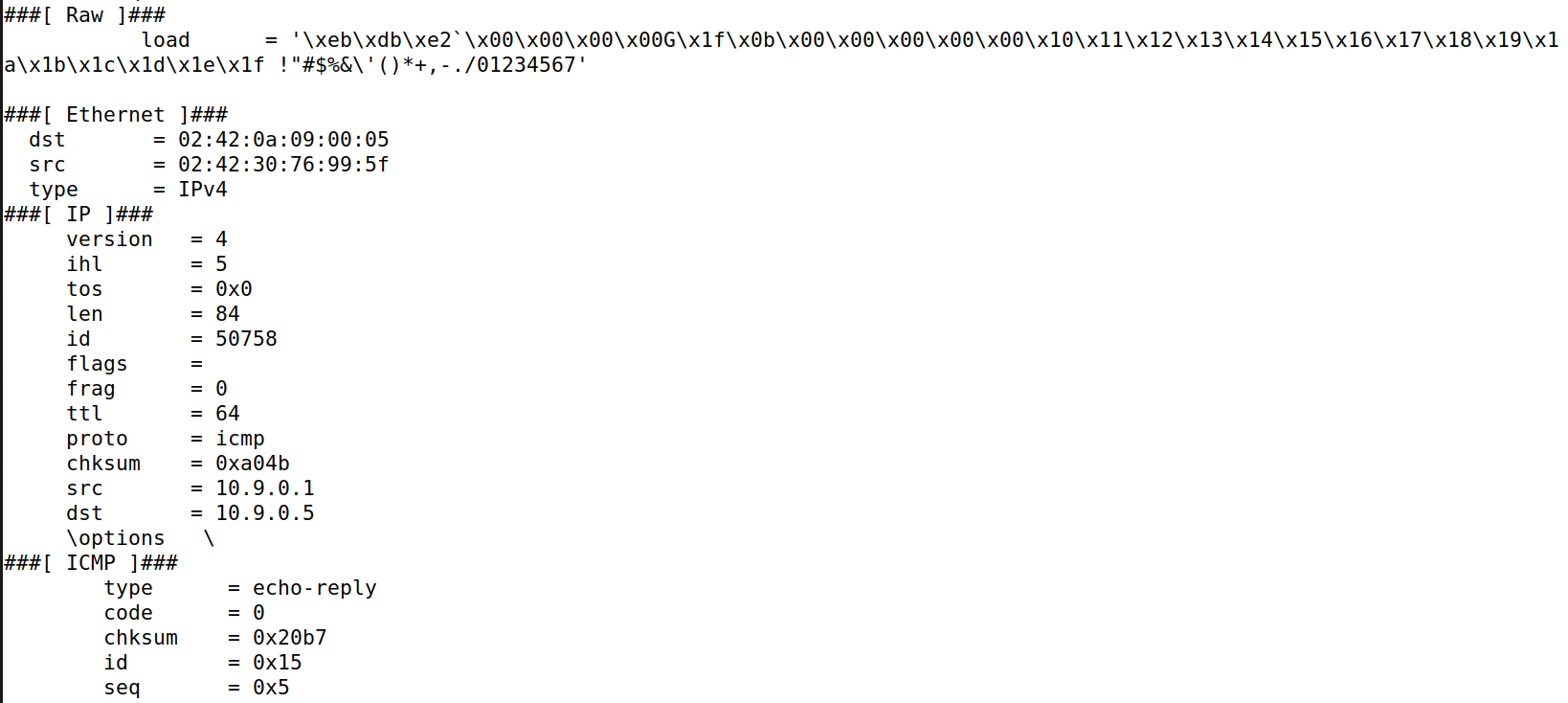
同时，用户机host-10.9.0.5 上使用ping命令发送icmp报文#ping 10.9.0.1

sniff.py 抓取到的报文

10.9.0.5 -> 10.9.0.1 响应请求如下

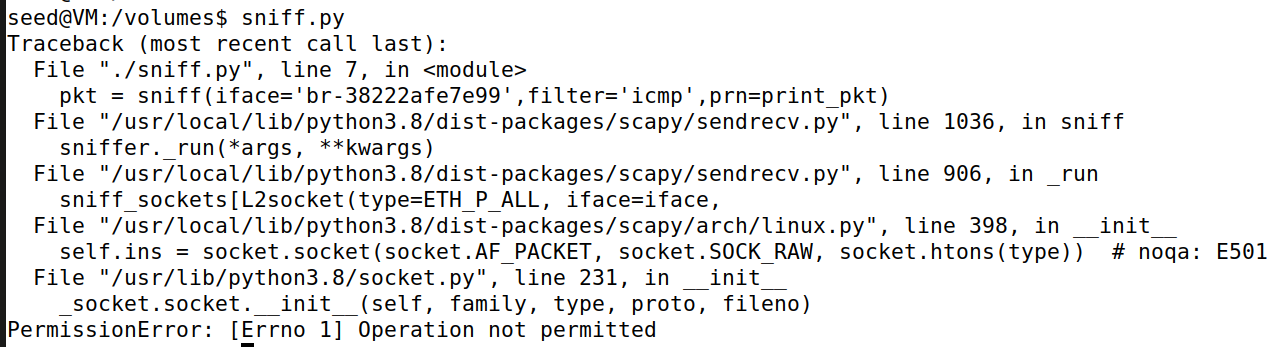


10.9.0.1 -> 10.9.0.5 应答如下



Attacker中切换到seed用户，再次执行#sniff.py

用户机host-10.9.0.5 还是使用ping命令发送icmp报文#ping 10.9.0.1



sniff.py直接因为权限不足被拦下，通过报错看出在创建socket的时候报错，scapy的很多模块涉及到内核态，需要root权限执行

**Task1.1B**

针对三种事件，定义三个不同filter以及响应函数

**sniffICMP.py抓取icmp报文**

#!/usr/bin/env python3

from scapy.all import \*

def print\_pkt1(pkt):

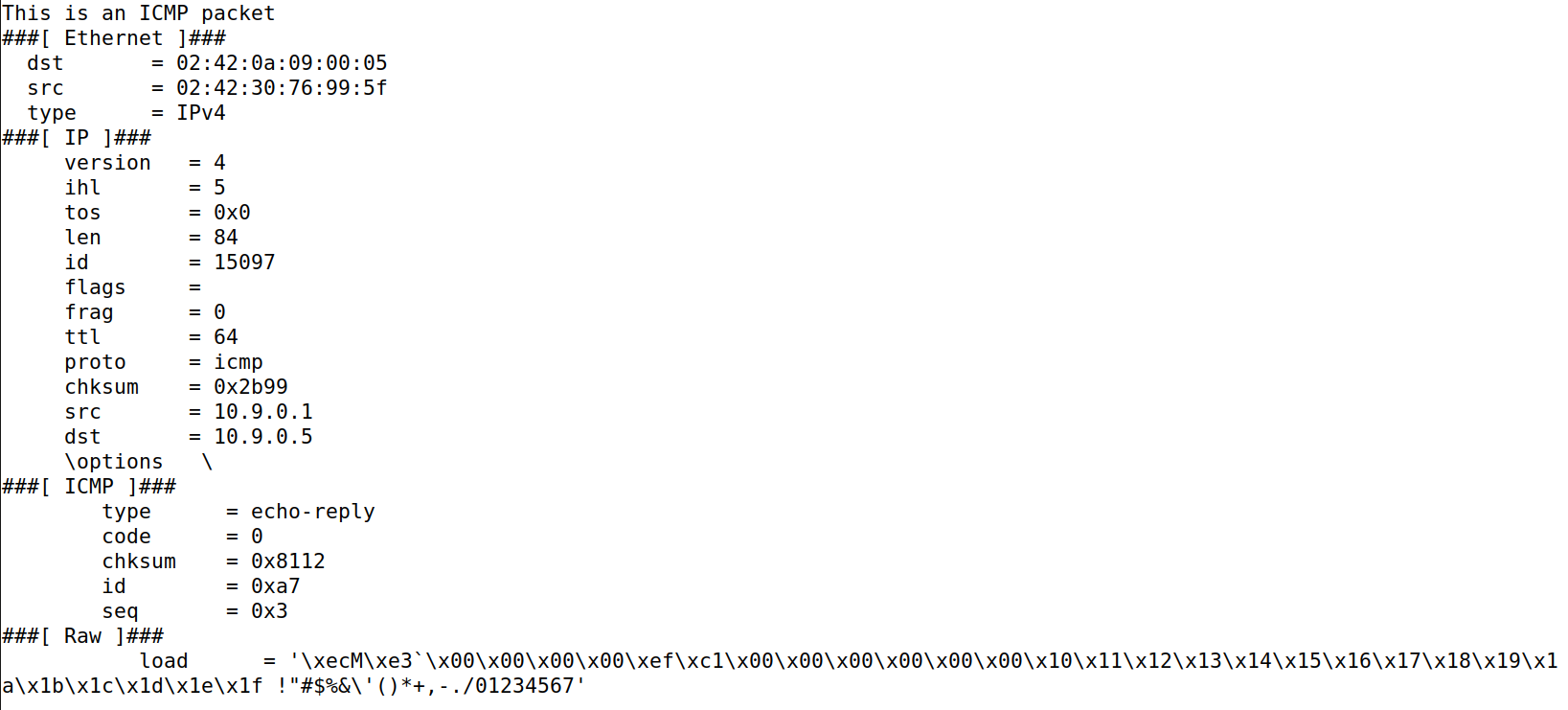
print("This is an ICMP packet")

pkt.show()

pkt = sniff(iface='br-38222afe7e99',filter='icmp',prn=print\_pkt1)

在attacker上运行#sniffICMP.py，然后在主机10.9.0.5上执行#ping 10.9.0.1，发送对应报文

得到如下结果



**sniffTCP.py抓取源地址是10.9.0.5 ,目的端口是23的tcp报文，中间使用and连接**

#!/usr/bin/env python3

from scapy.all import \*

def print\_pkt2(pkt):

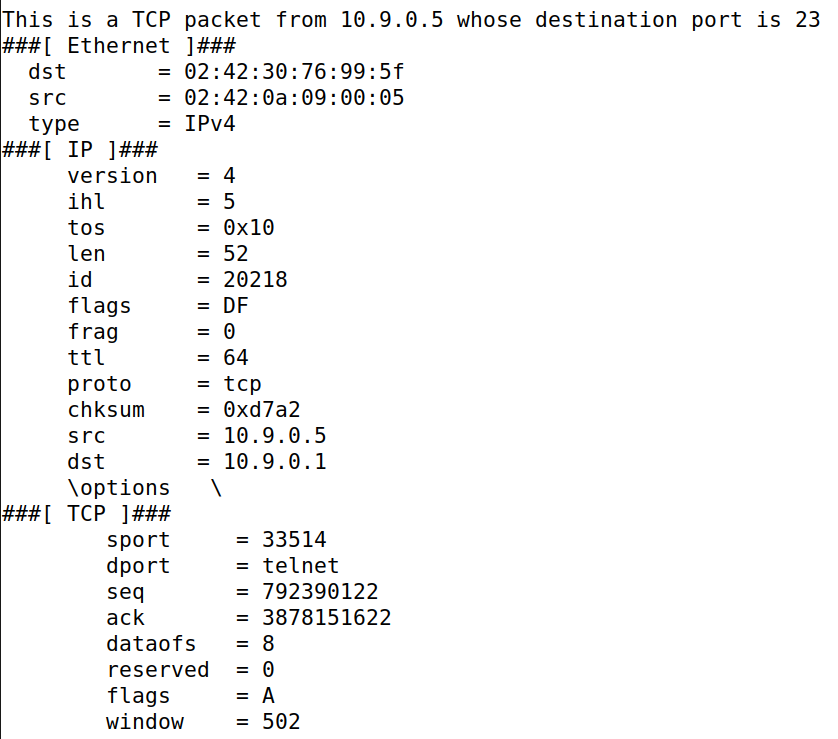
print("This is a TCP packet from 1.9.0.5 and destination port is 23")

pkt.show()

pkt = sniff(iface='br-38222afe7e99',filter='tcp and src host 1.1.1.1 and dst port 23',prn=print\_pkt2)

在attacker上运行#sniffTCP.py，然后在主机10.9.0.5上执行#telnet 10.9.0.1，发送对应报文

得到如下结果



**sniffIP.py抓取源或目的地址是128.230.0.0/16子网内的报文**

#!/usr/bin/env python3

from scapy.all import \*

def print\_pkt3(pkt):

print("This is a packet in net 128.230.0.0/16")

pkt.show()

pkt = sniff(iface='br-38222afe7e99',filter='net 128.230.0.0/16',prn=print\_pkt3)

使用scapy构造来自和发向128.230.0.0/16子网的报文如下

#!/usr/bin/python3

from scapy.all import \*

pkt = IP(src="128.230.1.1",dst="10.9.0.1")

pkt2 = IP(src="10.9.0.5",dst="128.230.1.1")

pkt.show()

send(pkt,verbose=0)

print("Send an IP packet whose src IP is 128.230.1.1")

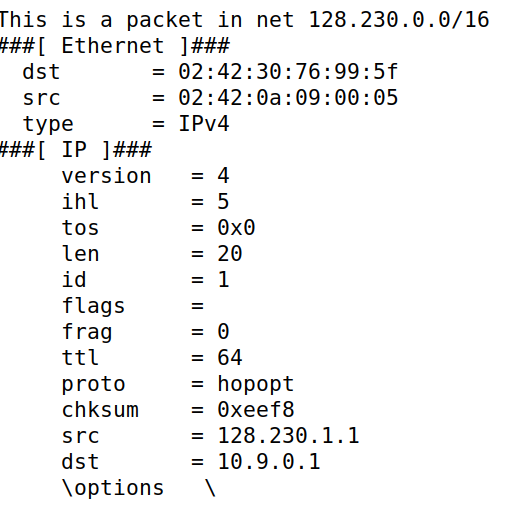
pkt2.show()

send(pkt2,verbose=0)

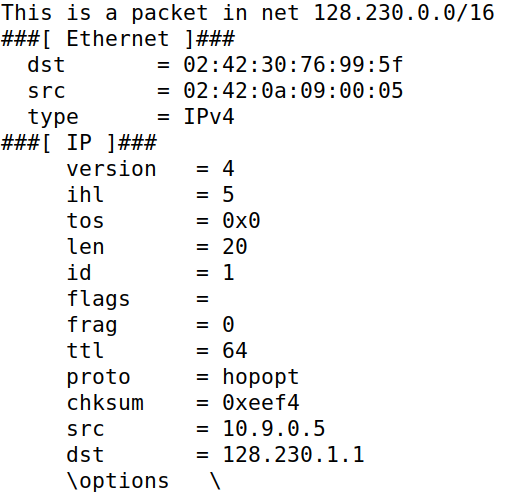
print("Send an IP packet whose dst IP is 128.230.1.1")

然后在10.9.0.5上运行上面程序，在10.9.0.1上运行#sniffIP.py进行嗅探，得到下面两张图

源地址为128.230.1.1：



目的地址为128.230.1.1：



**Task1.2**

如下程序实现构造一个ICMP echo-request包，可以指定任意IP地址，以下代码指定10.1.0.1为源地址，10.1.0.5为目的地址

#!/usr/bin/python3

from scapy.all import \*

ip = IP(src="10.1.0.1",dst="10.1.0.5")

icmp = ICMP()

pkt = ip/icmp

pkt.show()

send(pkt,verbose=0)

print("Send an ICMP packet")

**Task1.3**

使用scapy实现traceroute命令，跟踪每一条路由器ip地址和花费时间

使用sr1函数发送报文并接收一个回复报文，查阅ICMP相关文档，得知type=11，code=0时，该ICMP回复报文表示ttl超时，实现代码如下

testTTL.py

#!/usr/bin/python3

from scapy.all import \*

ip = IP()

ip.dst = sys.argv[1]

ip.ttl = 1

protocol = ICMP()

print("Traceroute " + str(ip.dst) + " for no more than 30 jumps")

while ip.ttl <= 30:

send\_time = time.time()

pkt = ip / protocol

re = sr1(pkt, timeout=1, verbose=False)

re\_time = time.time()

if not re:

print("No." + str(ip.ttl) + " router IP missing, ICMP ignored")

else:

print("No." + str(ip.ttl) + " router IP :" + re.getlayer(IP).src + ", costs " + str(1000\*(re\_time - send\_time)) + "ms")

if re.getlayer(ICMP).type != 11 or re.getlayer(ICMP).code != 0:

print(str(ip.ttl) + " jumps in all")

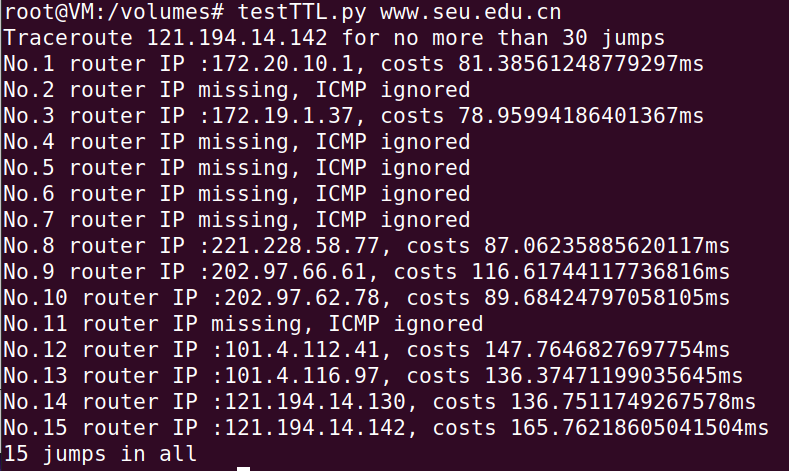
break

ip.ttl += 1

虚拟机在桥接模式下(校园网不支持桥接，需要使用热点)打开以太网#sudo dhclient ens33

在attacker（或vm）上执行#testTTL.py [www.seu.edu.cn](http://www.seu.edu.cn) ，得出共15跳路由，

路由顺序->vm->主机->运营商(中国电信)->校园网对外接口



Task1.4

在user即host-10.9.0.5上执行#ping x.x.x.x ，发送ICMP echo-request请求，使用scapy构造报文，实现不管对方ip是否在线，发送ICMP echo-reply。

检测到ICMP echo-request请求，就构造并发送响应echo-reply，以太头没必要构造，代码如下

snasp.py

#!/usr/bin/python3

from scapy.all import \*

def spoof\_reply(pkt):

#eth2 = pkt[Ether]

#eth = Ether()

#eth.dst = eth2.dst

#eth.src = eth2.src

#eth.type = eth2.type

ip2 = pkt[IP]

ip = IP()

ip.chksum = 0

ip.ihl = ip2.ihl

ip.dst = ip2.src

ip.src = ip2.dst

icmp = ICMP()

icmp.type = 0

icmp.seq = pkt[ICMP].seq

icmp.id = pkt[ICMP].id

icmp.chksum = 0

raw = pkt[Raw].load

pkt2 = ip/icmp/raw

del pkt2[IP].chksum

del pkt2[ICMP].chksum

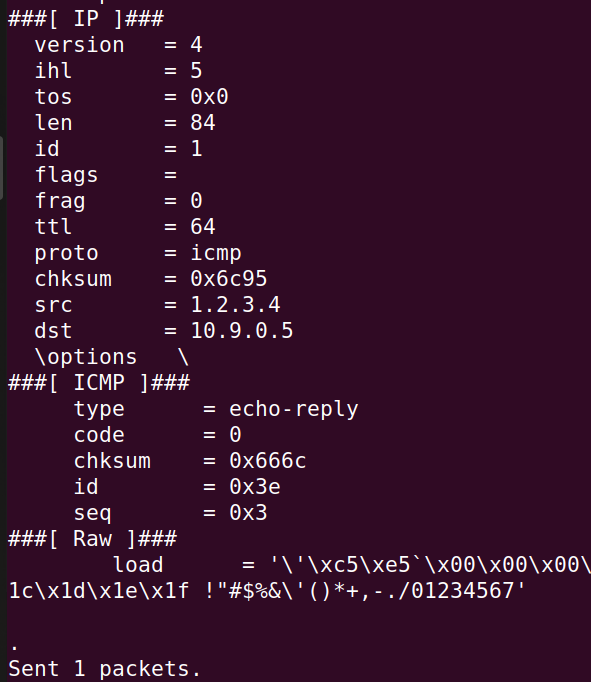
pkt2.show2()

send(pkt2)

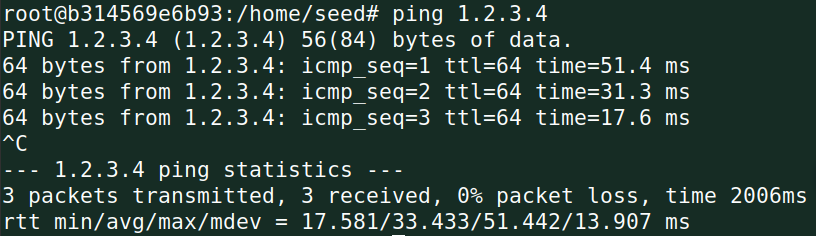
pkt = sniff(iface='br-38222afe7e99',filter='icmp[icmptype] == icmp-echo',prn=spoof\_reply)

**测试ping 1.2.3.4结果（互联网上的不存在主机）**

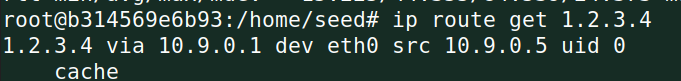
Attacker执行#snasp.py ，ICMP响应报文地址无误



同时host-10.9.0.5 执行#ping 1.2.3.4 ,可以看到收到ICMP响应报文

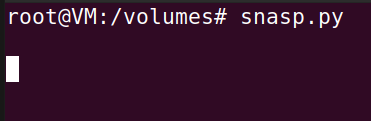


根据arp协议先验知识，首先user发送arp广播查看该ip是否在局域网内，发现不在局域网内，然后将报文发送给网关，实现路由。通过查看路由#ip route get 1.2.3.4 ，发现user将10.9.0.1当做网关，所以将该报文传递给10.9.0.1，而报文到了10.9.0.1后，只有snasp.py会发回报文给10.9.0.5，所以命令看起来正常

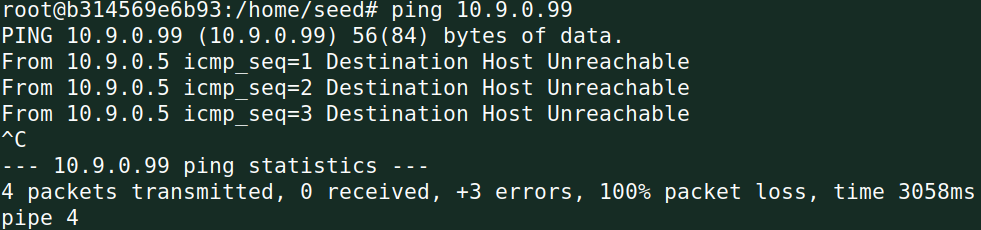


**测试ping 10.9.0.99结果（局域网内不存在主机）**

Attacker执行#snasp.py ，没有反应

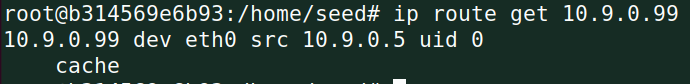


同时host-10.9.0.5 执行#ping 10.9.0.99 , 从本机10.9.0.5返回目的主机不可达



该地址在10.9.0.0/24局域网内，arp广播后没有响应，说明该主机不存在

通过查看路由#ip route get 10.9.0.99，并没有把报文交给网关

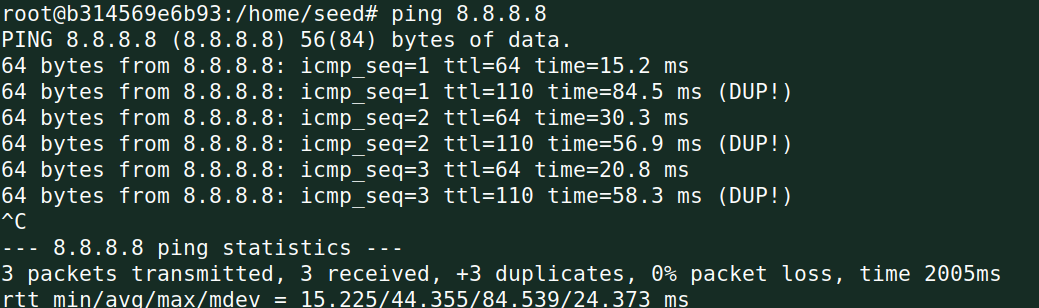


**测试ping 8.8.8.8结果（互联网上某主机）**

Attacker执行#snasp.py ，ICMP响应报文地址无误



同时host-10.9.0.5 执行#ping 8.8.8.8 ,可以看到收到ICMP响应报文重复，attacker的返回和8.8.8.8的返回重复



通过查看路由#ip route get 8.8.8.8，user将报文发送给网关后，ping命令的报文会被10.9.0.1发送到互联网上，经由互联网交给8.8.8.8，所以最后会有两个echo-reply

