



## 网络安全线上实验

学 院： 信息科学与工程学院

专业班级： 信息安全 1401 班

指导老师： 王伟平

学 号： 0906140126

姓 名： 王珊珊

# 目录

目录.....	1
一、实验名称.....	1
二、实验目的.....	1
三、实验任务.....	1
3.1 任务一：写数据包监听程序.....	1
3.1.a 了解嗅探训练.....	1
3.1.b 写过滤器.....	5
3.2 任务二：欺骗.....	7
3.2.a 写一个欺骗程序.....	7
3.2.b ICMP 回送请求欺骗.....	8
四、实验总结.....	10

## 实验二 数据包嗅探和欺骗实验

### 一、实验名称

数据包嗅探和欺骗实验

### 二、实验目的

数据包嗅探和欺骗是网络安全最重要的两个概念；它们是网络通信中两个主要的威胁。对于了解网络的安全措施，能够理解这两个威胁是必不可少的。有许多数据包嗅探和欺骗工具，如 `wireshark`, `tcpdump`, `netwox` 等，其中一些工具被安全专家以及攻击者广泛地使用。能够使用这些工具对于学生来说是很重要的，但对于学网络安全课程的学生更重要的是了解这些工具是如何工作的，即包嗅探和欺骗是如何在软件中实现的。

本实验的目的是让学生掌握最基本的嗅探和欺骗工具的技术。学生要使用一些简单的嗅探和欺骗程序，阅读源代码，并进行修改，最终可以对这些项目的技术方面有更加深入的了解。这个实验结束，学生应该能够写出自己的嗅探和欺骗程序。

### 三、实验任务

#### 3.1 任务一：写数据包监听程序

嗅探程序通过使用 `pcap` 库可以很容易地进行编写。利用 `PCAP`，嗅探器的任务变得在 `pcap` 库调用一系列简单的程序。在序列结束时，数据包将被放置在缓冲区中，一旦它们被捕获，就进行进一步处理。所有的数据包捕获的细节由 `pcap` 库处理。Tim Carstens 写了一个如何使用 `pcap` 库编写嗅探程序的教程。本教程是在 <http://www.tcpdump.org/pcap.htm>。

##### 3.1.a 了解嗅探训练

请从上述教程中下载 `sniffex.c` 程序，编译并运行它。你应该提供截图证据表明你的程序成功运行并产生预期的结果。

运行结果如图 1、2 所示

```
Terminal
[11/18/2016 07:01] seed@ubuntu:~$ su
Password:
[11/18/2016 07:01] root@ubuntu:/home/seed# gcc -o sniffex sniffex.c -l pcap
[11/18/2016 07:03] root@ubuntu:/home/seed# ./sniffex
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth0
Number of packets: 10
Filter expression: ip

Packet number 1:
  From: 192.168.159.128
  To: 203.208.43.102
  Protocol: TCP
  Src port: 56089
  Dst port: 443
  Payload (46 bytes):
00000 17 03 03 00 29 00 00 00 00 00 00 00 25 1d 44 86 ....%D.
00016 0e d0 21 69 45 4e 9a c5 74 6c f6 49 b1 d1 b1 d8 ..!EN..tl.I...
00032 d0 e6 59 56 32 fe 49 fc 0a e6 40 23 f0 ca ..YV2.I...@#..

Packet number 2:
  From: 203.208.43.102
  To: 192.168.159.128
  Protocol: TCP
  Src port: 443
  Dst port: 56089

Packet number 3:
  From: 203.208.43.102
  To: 192.168.159.128
  Protocol: TCP
  Src port: 443
  Dst port: 56089
  Payload (60 bytes):
00000 17 03 03 00 37 00 00 00 00 00 00 00 06 b1 71 da 68 ....7.....q.h
00016 a8 ec 8d 4e 6d df 0b e6 22 c3 2d 1f a3 52 21 b2 ...Nm...R!..
00032 6a 51 6b 6f 69 6b 5e 40 92 9f 47 26 b2 43 00 c1 jQkotk^@..G&.C..
00048 26 55 17 0a 4b ae 86 85 be 21 9d bf &U..K...!..
```

图 1 sniffex.c 运行结果图

```
Terminal
Protocol: TCP
Src port: 56089
Dst port: 443

Capture complete.
[11/18/2016 07:04] root@ubuntu:/home/seed# ./sniffex
sniffex - Sniffer example using libpcap
Copyright (c) 2005 The Tcpdump Group
THERE IS ABSOLUTELY NO WARRANTY FOR THIS PROGRAM.

Device: eth0
Number of packets: 10
Filter expression: ip

Packet number 1:
  From: 192.168.159.128
  To: 128.230.208.76
  Protocol: TCP
  Src port: 48671
  Dst port: 80
  Payload (419 bytes):
00000 47 45 54 20 2f 7e 77 65 64 75 2f 73 65 65 64 2f GET /-wedu/seed/
00016 4c 61 62 73 5f 31 32 2e 30 34 2f 53 79 73 74 65 Labs.12.04/Syste
00032 6d 2f 45 46 53 2f 20 48 54 50 2f 31 2e 31 0d n/EFS/ HTTP/1.1.
00048 0a 48 6f 73 74 3a 20 77 77 77 2e 63 69 73 2e 73 .Host: www.cis.s
00064 79 72 2e 65 64 75 0d 0a 55 73 65 72 2d 41 67 65 yr.edu..User-Age
00080 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 20 nt: Mozilla/5.0
00096 28 58 31 31 3b 20 55 62 75 6e 74 75 3b 20 4c 69 (X11; Ubuntu; Li
00112 6e 75 78 20 69 36 38 36 3b 20 72 76 3a 34 39 2e nux i686; rv:49.
00128 30 29 20 47 65 63 6b 6f 2f 32 30 31 30 30 31 30 0) Gecko/2010010
00144 31 20 46 69 72 65 66 6f 78 2f 34 39 2e 30 0d 0a 1 Firefox/49.0..
00160 41 63 63 65 70 74 3a 20 74 65 78 74 2f 68 74 6d Accept: text/htm
00176 6c 2c 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 68 l,application/xh
00192 74 6d 6c 2b 78 6d 6c 2c 61 70 70 6c 69 63 61 74 tml+xml,applicat
00208 69 6f 6e 2f 78 6d 6c 3b 71 3d 30 2e 39 2c 2a 2f ion/xml;q=0.9,*/
00224 2a 3b 71 3d 30 2e 38 0d 0a 41 63 63 65 70 74 2d *;q=0.8..Accept-
00240 4c 61 6e 67 75 61 67 65 3a 20 65 6e 2d 55 53 2c Language: en-US,
00256 65 6e 3b 71 3d 30 2e 35 0d 0a 41 63 63 65 70 74 en;q=0.5..Accept
00272 2d 45 6e 63 6f 64 69 6e 67 3a 20 67 7a 69 70 2c -Encoding: gzip,
00288 20 64 65 66 6c 61 74 65 0d 0a 52 65 66 65 72 65 deflate..Refere
00304 72 3a 20 68 74 70 3a 2f 2f 77 77 77 2e 63 69 r: http://www.ci
```

图 2 sniffex.c 运行结果图

- 1.请用自己的话来描述，嗅探程序是必要的库调用序列。这是一个总结，而不是像教程中的一个详细的解释。
- 2.为什么你需要 root 权限运行嗅探训练？如果没有根特权执行程序，程序错误的地方在哪？

普通的情况下，网卡只接收和自己的地址有关的信息包，即传输到本地主机的信息包。要使 Sniffer 能接收并处理这种方式的信息，系统需要支持 BPF，Linux 下需要支持 SOCKET-PACKET。但一般情况下，网络硬件和 TCP/IP 堆栈不支持接收或者发送与本地计算机无关的数据包，所以，为了绕过标准的 TCP/IP 堆栈，网卡就必须设置为混杂模式。一般情况下，要激活这种方式，内核必须支持这种伪设备 BPFfilter，而且需要 root 权限来运行这种程序，所以 Sniffer 需要 root 身份安装，如果只是以本地用户的身份进入了系统，那么不可能嗅探到 root 的密码，因为不能运行 Sniffer。

3.请打开和关闭的嗅探程序混杂模式。你能证明这种模式在打开和关闭时的区别吗？请描述你如何证明。打开和关闭混杂模式的运行结果如下图所示。

混杂模式 (Promiscuous Mode) 是指一台机器能够接收所有经过它的数据流，而不论其目的地址是否是他。是相对于通常模式（又称“非混杂模式”）而言的。这两种方式区别很大。一般来说，非混杂模式的嗅探器中，主机仅嗅探那些跟它直接有关的通信，如发向它的，从它发出的，或经它路由的等都会被嗅探器捕捉。而在混杂模式中则嗅探传输线路上的所有通信。在非交换式网络中，这将是整个网络的通信。这样做最明显的优点就是使更多的包被嗅探到，它们因你嗅探网络的原因或者对你有帮助，或者没有。但是，混杂模式是可被探测到的。一个主机可以通过高强度的测试判定另一台主机是否正在进行混杂模式的嗅探。其次，它仅在非交换式的网络环境中有效工作（如集线器，或者交换中的 ARP 层面）。再次，在高负荷的网络中，主机的系统资源将消耗的非常严重。

Linux 下通过 `ifconfig eth0 promisc` 将 eth0 设置成混杂模式，通过 `ifconfig eth0 -promisc` 取消混杂。

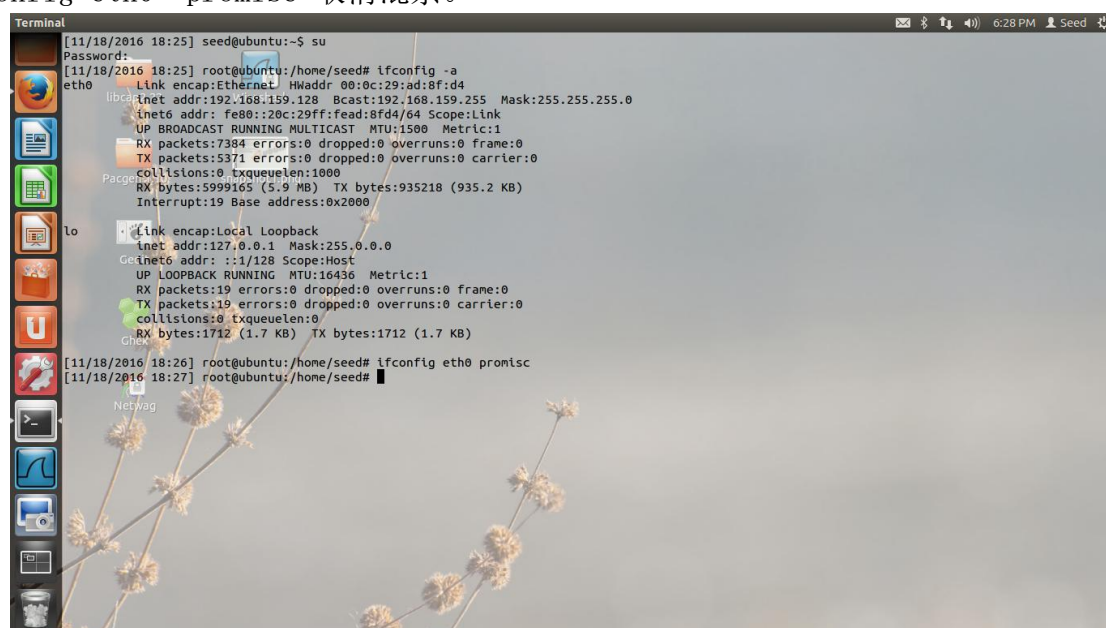


图 3 混杂模式设置图

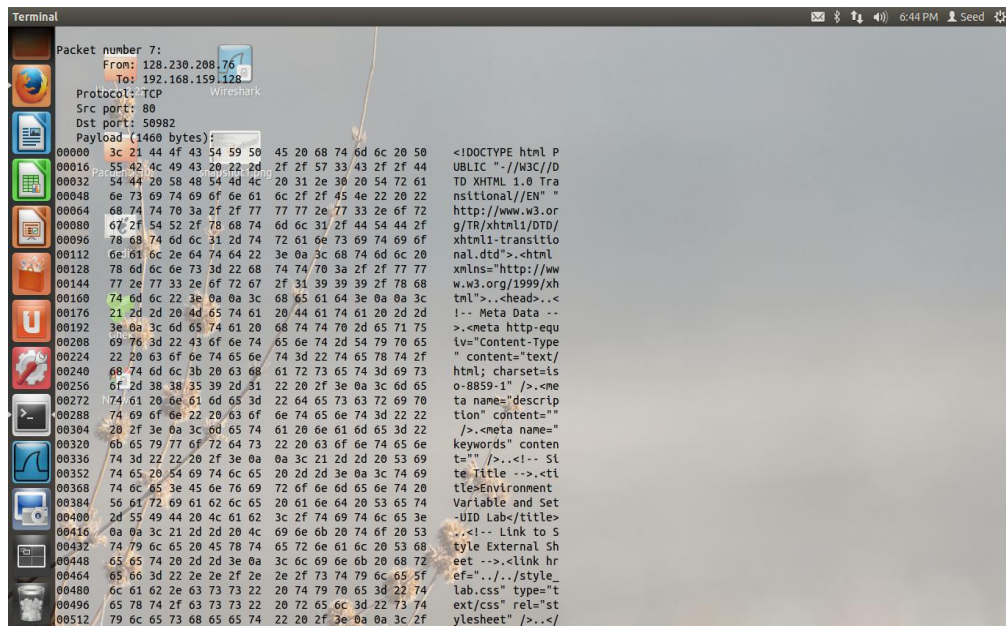


图 4 混杂模式下嗅探结果图

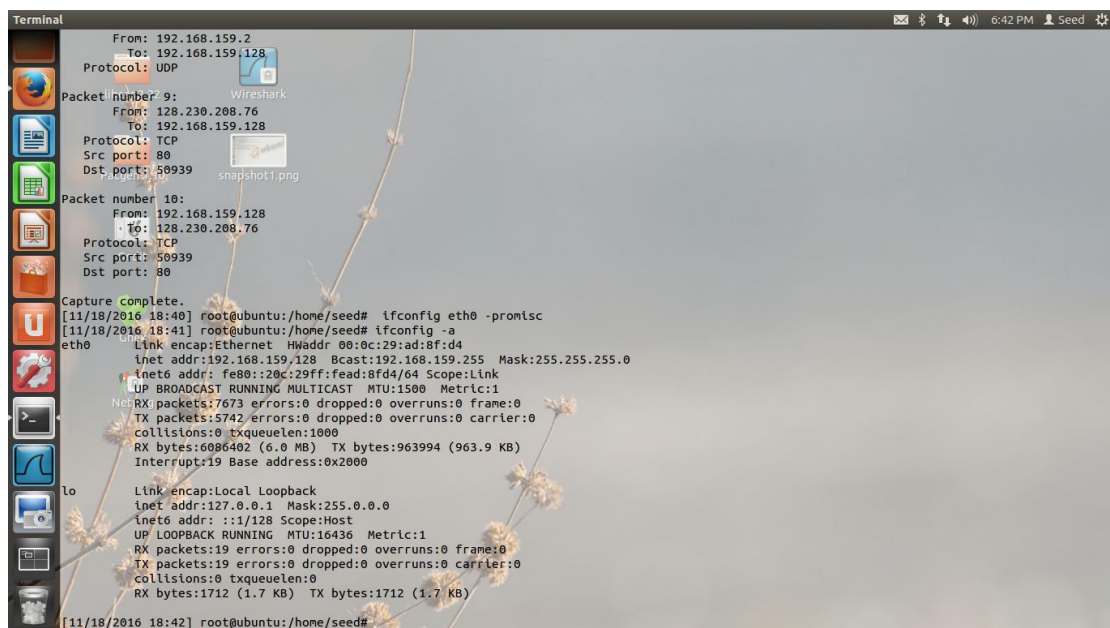


图 5 混杂模式关闭图



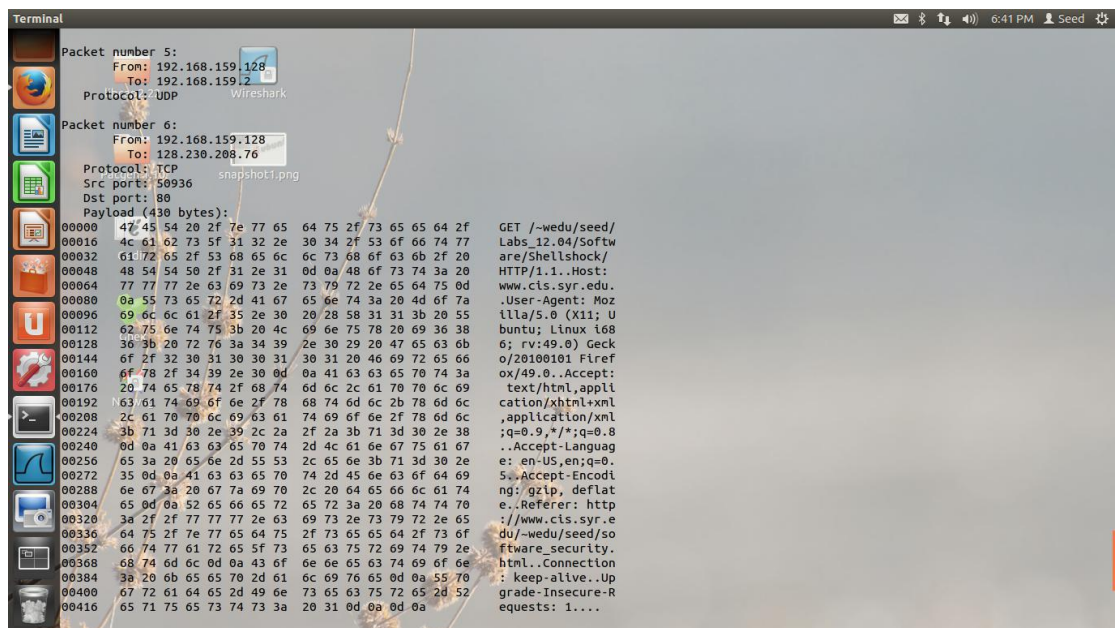


图 6 非混杂模式下嗅探结果图

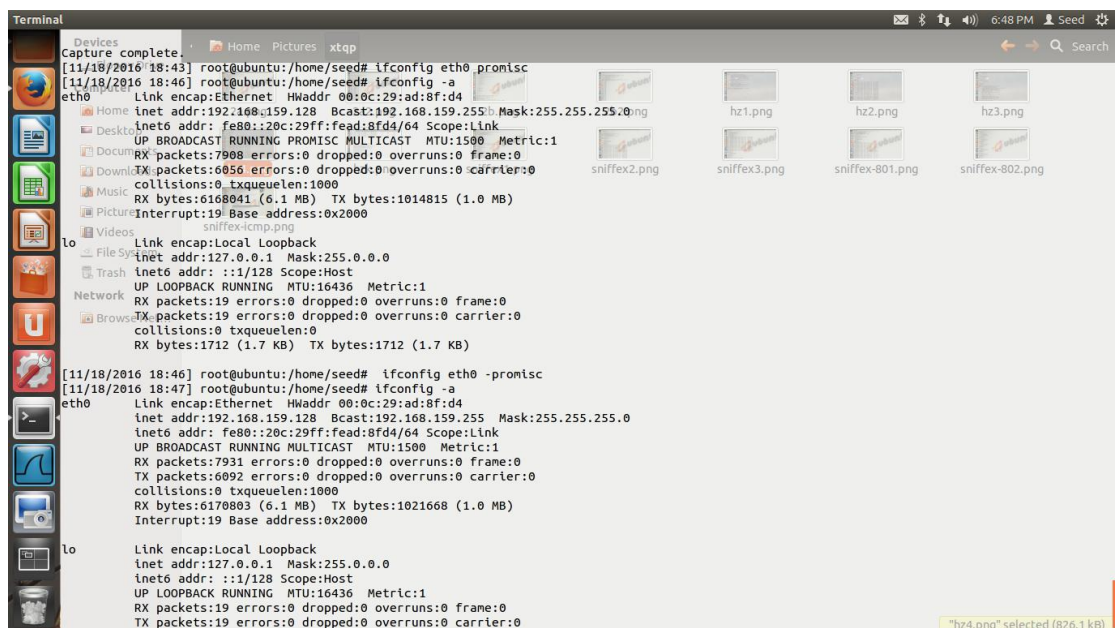


图 7 两种模式对比图

### 3.1.b 写过滤器

为嗅探程序写过滤表达式用来捕捉以下规定的数据包，在报告中，需要运行结果截图来展示每一个过滤表达式的运行结果。运行结果如下图所示。

1. 捕捉两个特定主机间的 ICMP 包：如图 8 所示，捕捉的是端口 192.168.159.128 和端口 128.230.208.76
2. 捕捉目的端口范围为 10-100 的 TCP 包：如图 9、10 所示，捕捉的是目的

端口为 80 端口，或初始端口为 80 端口的 TCP 包。

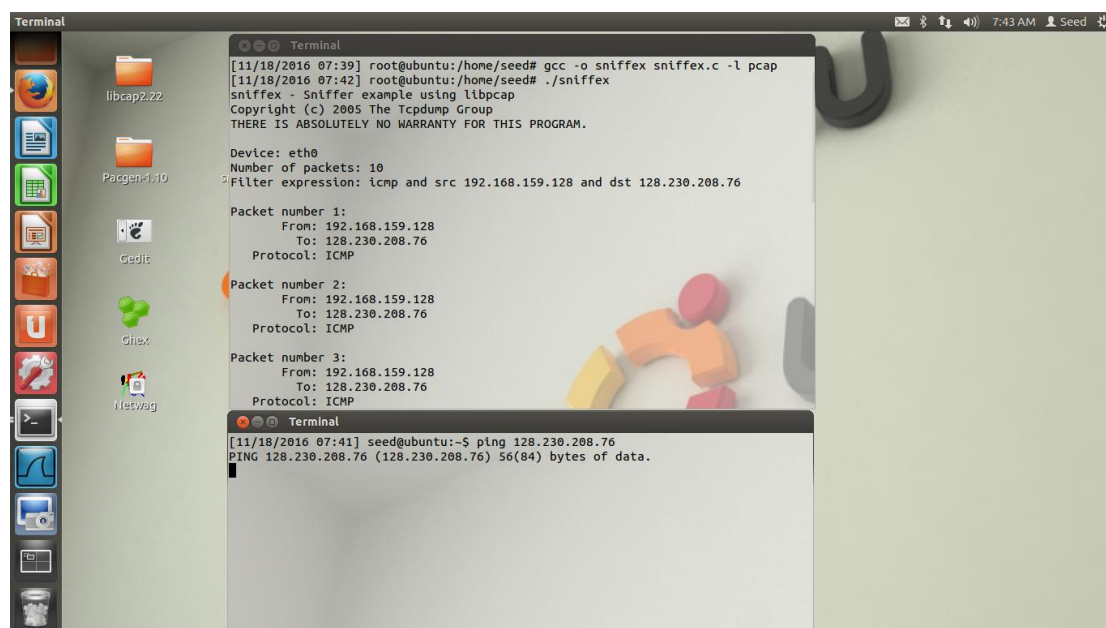


图 8 ICMP 包捕捉示意图

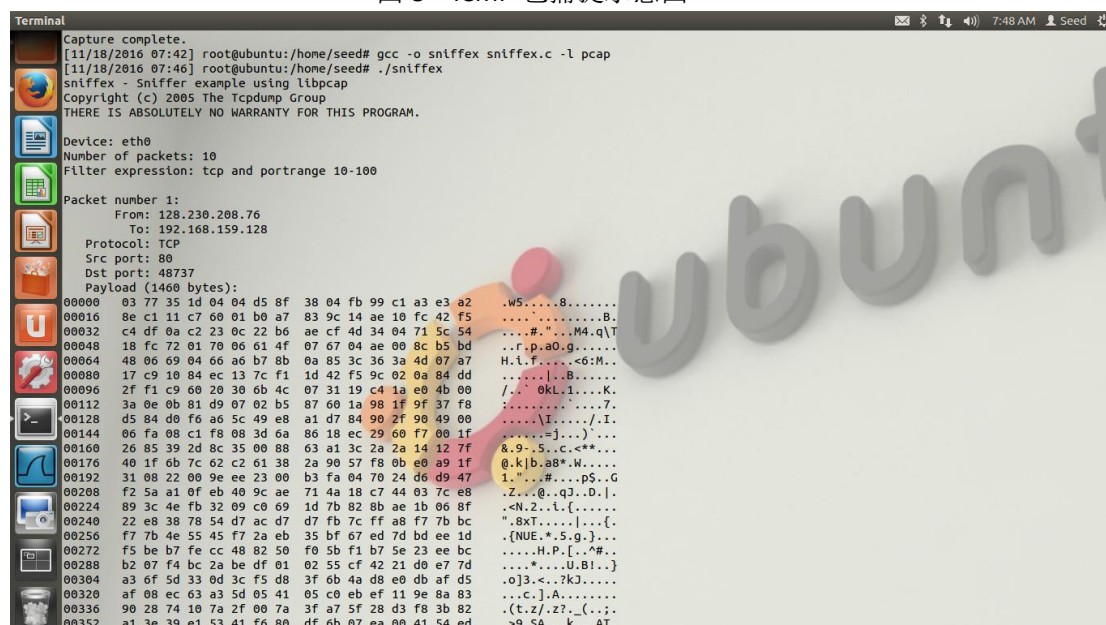


图 9 TCP 捕捉示意图





图 10 TCP 捕捉示意图

## 3.2 任务二：欺骗

当一个普通用户发送数据包时，操作系统通常不允许用户设置协议头组的所有领域（如 TCP，UDP，和 IP 报头）。操作系统将设置大部分的领域，而只允许用户设置几个字段，如目标 IP 地址、目标端口号等。

然而，如果用户有根权限，他们可以设置分组报头的任意字段，这就是包欺骗，可以采用原始套接字来实现。

原始套接字给程序员构建数据包的绝对控制权，允许程序员构建任何任意的数据包，包括设置头字段和有效载荷。使用原始套接字是很简单的，它包括四个步骤：（1）创建一个原始套接字，（2）设置套接字选项，（3）构造数据包，（4）通过原始套接字发送数据包。有许多在线教程，可以教你如何使用原始套接字。我们已经把一些教程与实验室的网页联系起来了。

请阅读它们，并学习如何写一个 **spoonfing** 程序。我们展示了一个简单的骨架如下所示。

### 3.2.a 写一个欺骗程序

你可以自己写一个的包程序或者下载一个程序。你需要提供的证据（例如，Wireshark 数据包跟踪）向我们展示你的程序成功地发送了伪造的 IP 数据包，运行结果如图 11.12 所示，伪造了源 ip 为 192.168.10.10 的数据包。

```
Terminal
[11/18/2016 08:30] seed@ubuntu:~$ su
Password:
[11/18/2016 08:30] root@ubuntu:/home/seed# cat rawudp.c
// ----rawudp.c-----
// Must be run by root lol! Just datagram, no payload/data
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/ip.h>
#include <netinet/udp.h>

// The packet length
#define PKT_LEN 8192

// Can create separate header file (.h) for all headers' structure
// The IP header's structure
struct ipheader {
    unsigned char    iph_ihl:5, iph_ver:4;
    unsigned char    iph_tos;
    unsigned short int iph_len;
    unsigned short int iph_ident;
    unsigned char    iph_flag;
    unsigned short int iph_offset;
    unsigned char    iph_ttl;
    unsigned char    iph_protocol;
    unsigned short int iph_checksum;
    unsigned int      iph_sourceip;
    unsigned int      iph_destip;
};

// UDP header's structure
struct udphheader {
    unsigned short int udph_srcport;
    unsigned short int udph_destport;
    unsigned short int udph_len;
    unsigned short int udph_checksum;
};

// total udp header length: 8 bytes (=64 bits)
```

图 11 伪造 ip 数据包结果图

```
Terminal
}
else
{
    printf("Count %u - sendto() is OK.\n", count);
    sleep(2);
}
close(sd);
return 0;
}

[11/18/2016 08:30] root@ubuntu:/home/seed# gcc rawudp.c -o rawudp
[11/18/2016 08:33] root@ubuntu:/home/seed# ./rawudp
- Invalid parameters!!!
- Usage ./rawudp <source hostname/IP> <source port> <target hostname/IP> <target port>
[11/18/2016 08:34] root@ubuntu:/home/seed# ./rawudp 192.168.10.10 21 203.106.93.91 8080
socket() - Using SOCK_RAW socket and UDP protocol is OK.
setsockopt() is OK.
Trying...
Using raw socket and UDP protocol
Using Source IP: 192.168.10.10 port: 21, Target IP: 203.106.93.91 port: 8080.
Count #1 - sendto() is OK.
Count #2 - sendto() is OK.
Count #3 - sendto() is OK.
Count #4 - sendto() is OK.
Count #5 - sendto() is OK.
Count #6 - sendto() is OK.
Count #7 - sendto() is OK.
Count #8 - sendto() is OK.
Count #9 - sendto() is OK.
Count #10 - sendto() is OK.
Count #11 - sendto() is OK.
Count #12 - sendto() is OK.
Count #13 - sendto() is OK.
Count #14 - sendto() is OK.
Count #15 - sendto() is OK.
Count #16 - sendto() is OK.
Count #17 - sendto() is OK.
Count #18 - sendto() is OK.
Count #19 - sendto() is OK.
Count #20 - sendto() is OK.
[11/18/2016 08:35] root@ubuntu:/home/seed#
```

图 12 伪造 ip 数据包结果图

### 3.2.b ICMP 回送请求欺骗

欺骗 ICMP 回送请求包代表另一台机器（使用另一台机器上的 IP 地址作为其源 IP 地址），这个包应该被发送到网络上一台远程的机器，你应该打开 Wireshark，因此如果你的欺骗是成功的，你可以看到来自远程机的返回应答。如图 13.14 所示。



## 四、实验总结

通过此次实验，我对包嗅探和欺骗实验的原理和过程有了进一步的理解和认识。嗅探 sniff。嗅探器可以窃听网络上流经的数据包。用集线器 hub 组建的网络是基于共享的原理的，局域网内所有的计算机都接收相同的数据包，而网卡构造了硬件的“过滤器”通过识别 MAC 地址过滤掉和自己无关的信息，嗅探程序只需关闭这个过滤器，将网卡设置为“混杂模式”就可以进行嗅探。用交换机 switch 组建的网络是基于“交换”原理的，交换机不是把数据包发到所有的端口上，而是发到目的网卡所在的端口，这样嗅探起来会麻烦一些，嗅探程序一般利用“ARP 欺骗”的方法，通过改变 MAC 地址等手段，欺骗交换机将数据包发给自己，嗅探分析完毕再转发出去。而包欺骗则是利用套接字来伪造数据，例如 ip 地址等。在这次实验中，拓展了我的知识面，学习到了许多新的知识和理论，对我的实践操作能力也是一次锻炼和提升，在日常的学习中，应该多多尝试体验这样的线上学习来提升自己。通过此次实验，将实践和理论知识相结合，进一步巩固了我们在课堂上学到的知识，印象更加深刻。