网络安全技术第 2 次实验

--制作网络嗅探器与数据报分析

姓　　名：柳致远

学　　号：2113683

专　　业：密码科学技术

指导老师：贾岩

目录

一、 实验内容说明

1、实验背景

网络嗅探是一种利用计算机的网络接口截获其它计算机数据报文的工具，可以检测网络的流量，实现网络数据的检测的捕获，已经逐渐成为网络分析过程中的重要方法。嗅探器作为一种嗅探工具，是通过对网卡的编程来实现网络通讯的，对网卡的编程是使用通常的套接字（socket）方式来进行。

raw socket 即原始套接字，可以通过创建 socket 时，设置参数 SOCK_RAW 来创建。相比平时使用 SOCK_STREAM 创建的应用层的套接字，raw socket 可以直接处理 ip 首部和 tcp 首部，并且可以监听抓取经过本机的 ip 数据包和 mac 帧，可以处理 ICMP 和 IGMP 等网络控制报文。

下面展示 ip 首部结构信息，在代码中对 ip 数据报的分析函数中有重要体现：

IP 首部结构

IPv4 Header Format

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length															
4	32	Identification																Flags			Fragment Offset											
8	64	Time To Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															

TCP/UDP 套接字只能接收和操作传输层或者传输层之上的数据报,因为当 IP 层把数据报往上传给传输层的时候，下层的数据报头部信息(如 IP 数据报头部和 Ethernet 帧头部)都已经去除了。在本次实验中由于需要对接收到的 ip 数据头分析，故本次实验需要用到原始套接字。

2、实验要求

1) 利用原始套接字编写一个网络嗅探器捕获网络数据报
2) 分析基本的数据报信息
3) 实现简单的过滤器功能

3、实验准备

本次实验由于需要使用到原始套接字，故需要在实验开始前关闭自己电脑安全系统的防火墙，以及利用管理员身份运行我们的程序。

二、实验代码分析

1、数据报捕获

```
def snifflpData(myip, choose):
    host_ip = myip  # 获取 IP 即设置要捕获的网卡
    sniffer = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_IP)
    sniffer.bind((host_ip, 0))
    sniffer.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)
    if choose == 'a':
        sniffer.ioctl(socket.SIO_RCVALL, socket.RCVALL_ON)
    print("[*] 正在嗅探数据.....")
    count = int(input("请输入您想嗅探的数据包个数：  "))
    while count:
        recv_data, addr = sniffer.recvfrom(1500)
        if recv_data:
            try:
                decodeIpData(recv_data)
            except:
                continue
        else:
            continue
        count -= 1
    sniffer.ioctl(socket.SIO_RCVALL, socket.RCVALL_OFF)
    sniffer.close()
    return recv_data
```

snifflpData 函数接收两个参数，分别是用户输入的 ip 地址，另一个参数为用户想进入的捕获模式的选项（a 为混杂模式，b 为普通模式）。使用 setsockopt 设置套接字选项 IP_HDRINCL，以便告诉操作系统不要修改 IP 头部的内容，直接接收原始的 IP 数据包。使用原始套接字 sniffer 开始循环接收数据包，这里用户可以输入想接收的数据包个数（在本次实验中以 10 个数据包为例），将接收

到的数据包调用 decodeIpData 函数进行解析。

2、数据报解析

```
def decodeIpData(package):
    ip_data = {}
    # RFC791
    ip_data['version'] = package[0] >> 4  # 版本
    ip_data['headLength'] = package[0] & 0x0f  # 头长度  #& 按位与操作
    ip_data['DSField'] = package[1]  # Tos 服务字段
    ip_data['totalLength'] = (package[2] << 8) + package[3]  # 总长度
    ip_data['identification'] = (package[4] << 8) + package[5]  # 标识分片
    ip_data['flag'] = package[6] >> 5  # 标志位
    ip_data['moreFragment'] = ip_data['flag'] & 1  # 标识后面可否还有分片
    ip_data['dontFragment'] = (ip_data['flag'] >> 1) & 1  # 能否分片 0 可分片
    ip_data['fragmentOffset'] = ((package[6] & 0x1f) << 8) + package[7]  # 片偏移
    ip_data['TTL'] = package[8]  # 生存周期
    ip_data['protocol'] = check_protocol(package[9])  # 协议
    ip_data['headerCheckSum'] = (package[10] << 8) + package[11]  # 校验和
    # 以 IP 地址形式存储
    ip_data['sourceAddress'] = "%d.%d.%d.%d" % (package[12], package[13], package[14]
package[15])
    ip_data['destinationAddress']  =  "%d.%d.%d.%d"  %  (package[16],  package[17],
package[18], package[19])
    ip_data['options'] = []  # 可选项
    # 根据 headerLength 求出 options
    if ip_data['headLength'] > 5:
        temp = 5
        while temp < ip_data['headLength']:
            ip_data['options'].append(package[temp * 4] + 0)
            ip_data['options'].append(package[temp * 4] + 1)
            ip_data['options'].append(package[temp * 4] + 2)
            ip_data['options'].append(package[temp * 4] + 3)
            temp += 1
    # 根据 totalLength 求出 data
    ip_data['data'] = ""
    temp = ip_data['headLength'] * 4
    while temp < ip_data['totalLength']:
        ip_data['data'] += (str(hex(package[temp])).upper().replace("0X", ""))
        temp += 1
    pcaps.append(ip_data)
    return ip_data
```

函数首先创建一个空字典 ip_data 用于存储解析后的 IP 数据包信息，存储 RFC791 协议上各字段对应的含义以及对应的值。RFC791 协议的下，ip 数据各

字段的含义即大小已在实验背景部分截图体现，在此不做赘述。封装好这个数据报的 ip_data 字典后，将 ip_data 压入全局列表 pcaps 内，待 snifflpData 函数中的对数据报的循环接收结束后，pcaps 内存储着所有接收的数据报经过解析后封装成的字典。

3、过滤数据报

```
def filter(options, target):
    if options == 1:
        print('''
        --------a. 以混杂模式开始嗅探--------------------------
        --------b. 以普通模式开始嗅探-----------------------
        --------c. 返回上个选项---------------------------
        ''')
        choose = input("输入选项：")
        if choose == 'c':
            return
        snifflpData(myip=target, choose=choose)
        for pcap in pcaps:
            for i in pcap.keys():
                print("{}:{}".format(i, pcap[i]))
            print("\n")
    elif options == 2:
        while True:
            print('''
        --------a. 按源 ip 地址筛选--------------------------
        --------b. 按目的 IP 地址筛选-----------------------
        --------c. 同时按源 ip 地址和目的 ip 地址筛选---------
        --------d. 返回上个选项---------------------------
        ''')
            choice = input("输入选项：")
            ip_filter(choice)
            if choice == 'd':
                return
    elif options == 3:
        protocol = input("请输入协议名：")
        index = 0
        for pcap in pcaps:
            if pcap['protocol'] == protocol:
                index += 1
                print("第{}条记录".format(index))
                for i in pcap.keys():
                    print("{}:{}".format(i, pcap[i]))
                print("\n")
        if index == 0:
            print("协议名错误或无该记录！")
```

过滤函数 filter 接收两个参数，target 为用户输入的 ip 地址和用户输入的选项 options，用于在程序最开始进行以下选项：

```
def sniffer(target):
    while True:
        print('''
    --------1. 开始嗅探并输出嗅探结果（开始前请先选此选项）--------------
    --------2. 根据IP筛选-----------------------------------------------------
    --------3. 根据协议筛选-------------------------------------------------
    --------4. 退出程序-----------------------------------------------------
    ''')
        options = int(input("输入选项："))
        if options == 4:
            break
        filter(options, target)
```

在过滤函数中用户可以实现在第 1、2 步中接收、解析并封装的数据报进行对 ip 地址的筛选以及对协议类型的筛选。此时若重新嗅探则筛选结果也将更新，若无特殊情况请用户无需重复嗅探数据报防止之前嗅探的数据报被覆盖。

这里实现了对协议类型的筛选，若用户输入的 option 为 3，即为对协议类型的筛选，对于用户输入的协议类型，程序将遍历全局列表 pcaps 中的所有字典 pcap 内 key 为 protocol 字段的 value 是否为用户输入的协议类型，若不相符直接进行下一个字典的比对工作，若相符则输入该字典的所有字段，并进行下一个字典的比对工作。

下面是对 ip 筛选的详细代码展示：

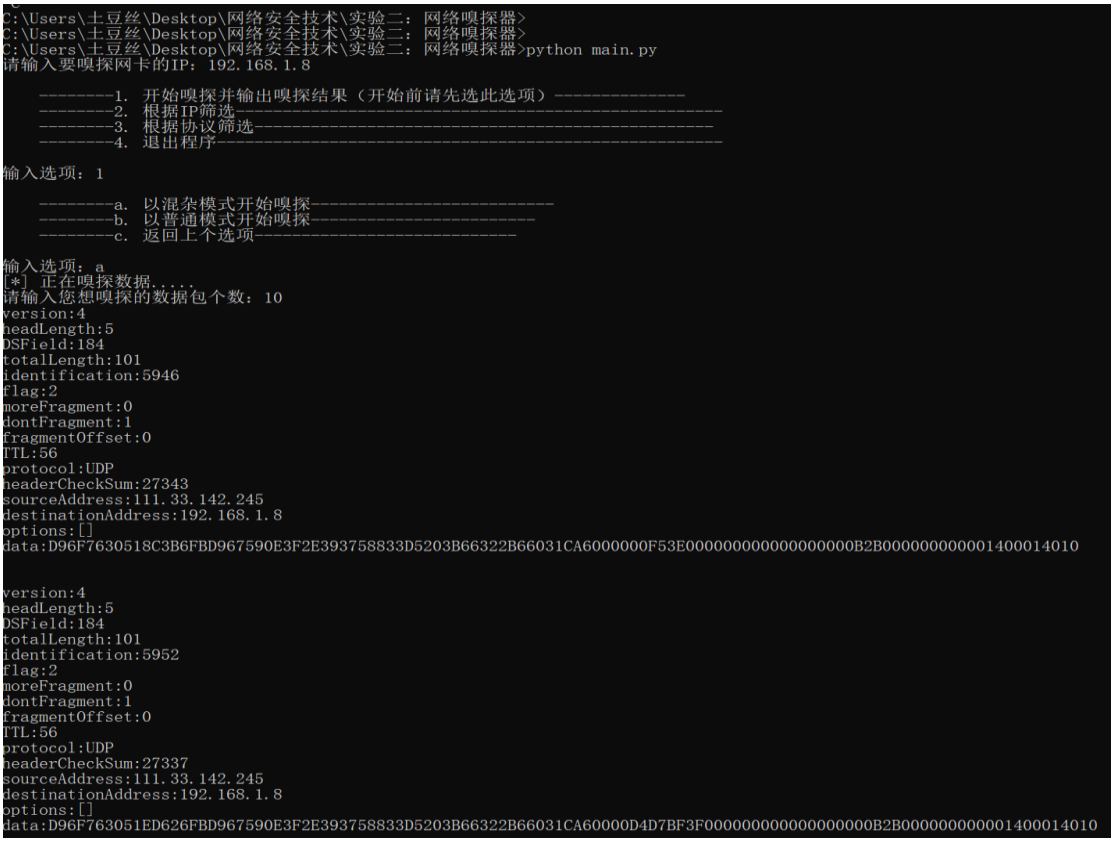
```
def ip_filter(choice):
    if choice == 'a':
        source = input("请输入源 IP 地址：")
        index = 0
        for pcap in pcaps:
            if pcap["sourceAddress"] == source:
                index += 1
                print("第{}条记录".format(index))
                for i in pcap.keys():
                    print("{}:{}".format(i, pcap[i]))
                print("\n")
        if index == 0:
            print("源 IP 输入错误或者无该记录！")
    if choice == 'b':
        dest = input("请输入目的 IP 地址：")
        index = 0
```

```
            for pcap in pcaps:
                if pcap["destinationAddress"] == dest:
                    index += 1
                    print("第{}条记录".format(index))
                    for i in pcap.keys():
                        print("{}:{}".format(i, pcap[i]))
                    print("\n")
            if index == 0:
                print("目的 IP 输入错误或者无该记录！")
        if choice == 'c':
            source = input("请输入源 IP 地址：")
            dest = input("请输入目的 IP 地址：")
            index = 0
            for pcap in pcaps:
                if pcap["sourceAddress"] == source and pcap["destinationAddress"] == dest:
                    index += 1
                    print("第{}条记录".format(index))
                    for i in pcap.keys():
                        print("{}:{}".format(i, pcap[i]))
                    print("\n")
            if index == 0:
                print("IP 输入错误或者无该记录！")
```

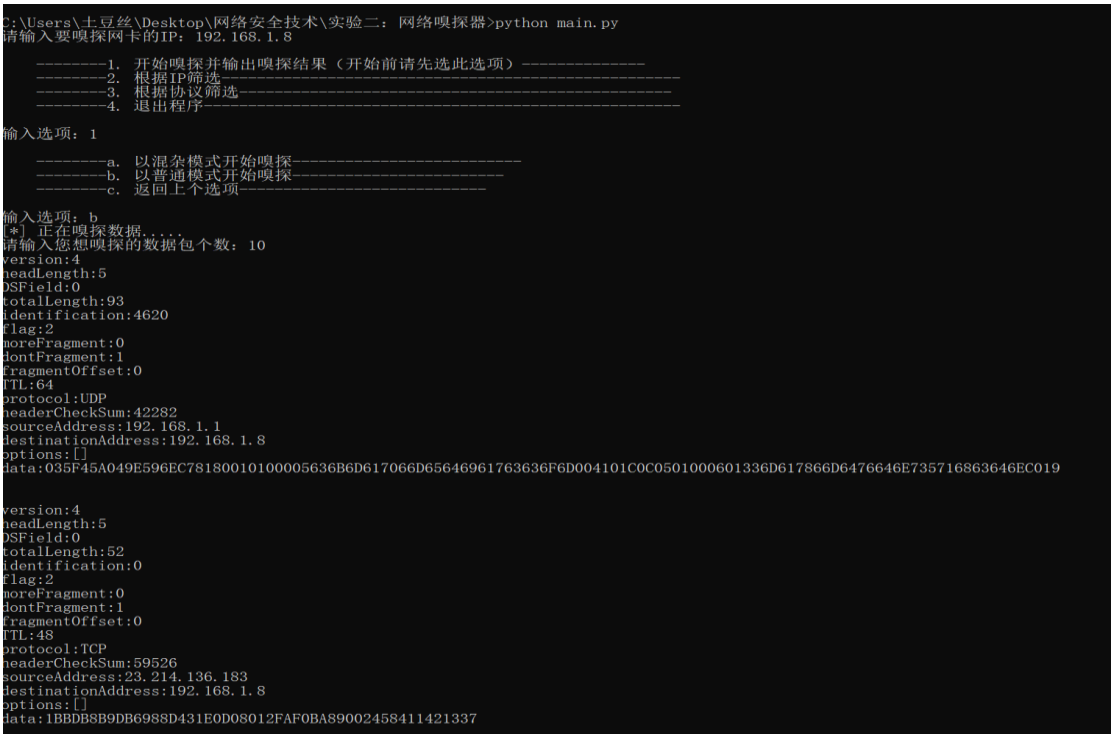
在对 ip 地址进行筛选的代码中可以实现按源 ip 地址筛选、按目的 ip 地址筛选、按源 ip 地址和目的 ip 地址筛选。对于用户输入的 ip 地址，程序将遍历全局列表 pcaps 中的所有字典 pcap 内 key 为 **sourceAddress** 或 **destinationAddress** 字段的 value 是否为用户输入的对应 ip 地址，若不相符直接进行下一个字典的比对工作，若相符则输入该字典的所有字段，并进行下一个字典的比对工作。

三、实验过程及截图

混杂模式嗅探：



普通模式嗅探：



按源 IP 地址筛选：

```
    --------1. 开始嗅探并输出嗅探结果（开始前请先选此选项）---------------
    --------2. 根据IP筛选-----------------------------------------------------
    --------3. 根据协议筛选-------------------------------------------------
    --------4. 退出程序-----------------------------------------------------

输入选项：2

    --------a. 按源ip地址筛选---------------------------
    --------b. 按目的IP地址筛选-------------------------
    --------c. 同时按源ip地址和目的ip地址筛选----------
    --------d. 返回上个选项-----------------------------

输入选项：1

    --------a. 按源ip地址筛选---------------------------
    --------b. 按目的IP地址筛选-------------------------
    --------c. 同时按源ip地址和目的ip地址筛选----------
    --------d. 返回上个选项-----------------------------

输入选项：a
请输入源IP地址：111.33.142.245
第1条记录
version:4
headLength:5
DSField:184
totalLength:101
identification:5946
flag:2
moreFragment:0
dontFragment:1
fragmentOffset:0
TTL:56
protocol:UDP
headerCheckSum:27343
sourceAddress:111.33.142.245
destinationAddress:192.168.1.8
options:[]
data:D96F7630518C3B6FBD967590E3F2E393758833D5203B66322B66031CA6000000F53E0000000000000000000B2B0000000000001400014010

第2条记录
version:4
headLength:5
DSField:184
totalLength:101
identification:5952
flag:2
moreFragment:0
dontFragment:1
fragmentOffset:0
TTL:56
protocol:UDP
```

按目的 ip 地址筛选：

```
    --------a. 按源ip地址筛选---------------------------
    --------b. 按目的IP地址筛选-------------------------
    --------c. 同时按源ip地址和目的ip地址筛选----------
    --------d. 返回上个选项-----------------------------

输入选项：b
请输入目的IP地址：192.168.1.8
第1条记录
version:4
headLength:5
DSField:184
totalLength:101
identification:5946
flag:2
moreFragment:0
dontFragment:1
fragmentOffset:0
TTL:56
protocol:UDP
headerCheckSum:27343
sourceAddress:111.33.142.245
destinationAddress:192.168.1.8
options:[]
data:D96F7630518C3B6FBD967590E3F2E393758833D5203B66322B66031CA6000000F53E0000000000000000000B2B0000000000001400014010

第2条记录
version:4
headLength:5
DSField:184
totalLength:101
identification:5952
flag:2
moreFragment:0
dontFragment:1
fragmentOffset:0
TTL:56
protocol:UDP
headerCheckSum:27337
sourceAddress:111.33.142.245
destinationAddress:192.168.1.8
options:[]
data:D96F763051ED626FBD967590E3F2E393758833D5203B66322B66031CA60000D4D7BF3F0000000000000000000B2B0000000000001400014010

第3条记录
version:4
headLength:5
DSField:184
totalLength:101
identification:6076
flag:2
```

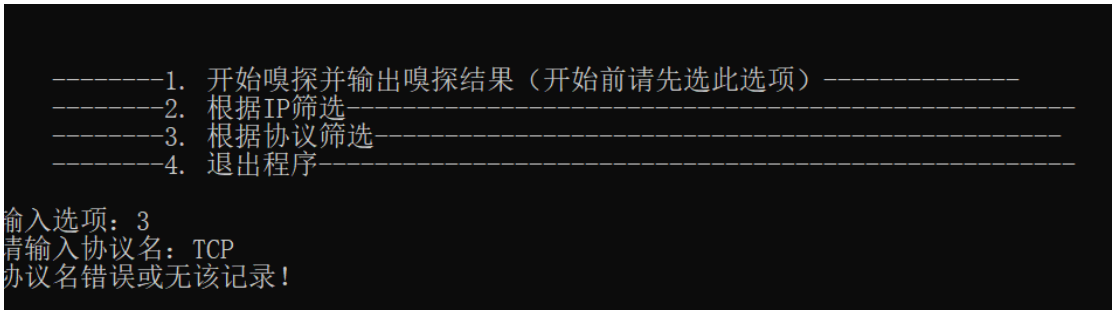
同时按源 ip 地址和目的 ip 地址筛选：

```
    --------a. 按源ip地址筛选--------------------------
    --------b. 按目的IP地址筛选------------------------
    --------c. 同时按源ip地址和目的ip地址筛选---------
    --------d. 返回上个选项--------------------------

输入选项：c
请输入源IP地址：111.33.142.245
请输入目的IP地址：192.168.1.8
第1条记录
version:4
headLength:5
DSField:184
totalLength:101
identification:5946
flag:2
moreFragment:0
dontFragment:1
fragmentOffset:0
TTL:56
protocol:UDP
headerCheckSum:27343
sourceAddress:111.33.142.245
destinationAddress:192.168.1.8
options:[]
data:D96F7630518C3B6FBD967590E3F2E393758833D5203B66322B66031CA6000000F53E0000000000000000000B2B0000000000001400014010


第2条记录
version:4
headLength:5
DSField:184
totalLength:101
identification:5952
flag:2
moreFragment:0
dontFragment:1
fragmentOffset:0
TTL:56
protocol:UDP
headerCheckSum:27337
sourceAddress:111.33.142.245
destinationAddress:192.168.1.8
options:[]
data:D96F763051ED626FBD967590E3F2E393758833D5203B66322B66031CA60000D4D7BF3F0000000000000000000B2B0000000000001400014010


第3条记录
version:4
headLength:5
DSField:184
totalLength:101
identification:6076
flag:2
```

按协议类型筛选—UDP：

```
    --------a. 按源ip地址筛选--------------------------
    --------b. 按目的IP地址筛选------------------------
    --------c. 同时按源ip地址和目的ip地址筛选---------
    --------d. 返回上个选项--------------------------

输入选项：d

    --------1. 开始嗅探并输出嗅探结果（开始前请先选此选项）--------------
    --------2. 根据IP筛选-----------------------------------------------
    --------3. 根据协议筛选---------------------------------------------
    --------4. 退出程序-----------------------------------------------

输入选项：3
请输入协议名：UDP
第1条记录
version:4
headLength:5
DSField:184
totalLength:101
identification:5946
flag:2
moreFragment:0
dontFragment:1
fragmentOffset:0
TTL:56
protocol:UDP
headerCheckSum:27343
sourceAddress:111.33.142.245
destinationAddress:192.168.1.8
options:[]
data:D96F7630518C3B6FBD967590E3F2E393758833D5203B66322B66031CA6000000F53E0000000000000000000B2B0000000000001400014010


第2条记录
version:4
headLength:5
DSField:184
totalLength:101
identification:5952
flag:2
moreFragment:0
dontFragment:1
fragmentOffset:0
TTL:56
protocol:UDP
headerCheckSum:27337
sourceAddress:111.33.142.245
destinationAddress:192.168.1.8
options:[]
data:D96F763051ED626FBD967590E3F2E393758833D5203B66322B66031CA60000D4D7BF3F0000000000000000000B2B0000000000001400014010
```

按协议类型筛选—TCP



四、实验感悟及分析

在本次实验中普通模式的嗅探工作并不顺利，一开始在测试的时候时常程序没有任何反应，为此我尝试了包括打开浏览器网页，发送微信消息等多种手段，一时程序仍不打印任何的数据报内容。后来经过查询相关资料得知可能是我打开网页的数据报经过加密操作，而我写的程序不含解密操作，致使在对数据报的解析工作无法顺利展开。而微信 app 的通信采用隧道传输的方式同样进行了加密操作，而网页版微信则为典型的 https 传输。后再尝试打开常规的网页后通过普通模式捕获到了数据报。