# 目录

第一章 需求分析 4

1.1 设计目的 4

1.2 功能要求 5

第二章 概要设计 6

2.1 系统分析 6

2.1.1 架构设计 6

2.1.2 实体设计 7

2.1.3 分层设计 8

2.1.4 数据库设计 9

2.2 程序流程 10

2.2.1 程序结构图 10

2.2.2 调用关系 10

第三章 详细设计 12

3.1 功能类定义与实现 12

3.1.1 常量与参数的定义（Parameter类） 12

3.1.2 IP/ARP数据包显示信息的定义 13

3.1.3 捕获数据包的实现（HandleJpcap包） 14

3.1.4 Ping操作实现部分（HandlePing包） 19

3.1.5 TraceRoute操作实现部分（HandleTraceRoute包） 23

3.1.6 ARP-Attack操作实现部分（ARPAttack包） 26

3.2 图形界面类定义与实现 31

3.2.1 JFreeChart制图（JFreeChart类） 31

3.2.2 Dataset容器（JFreeChart制图所需容器） 35

3.2.3 界面设计与制作（JFrame继承子类） 36

3.3 函数调用关系图 36

第四章 调试分析 37

4.1 调试中遇到的问题 37

4.1.1ARP攻击实现时遇到的问题 37

4.1.2绘制表格中遇到的问题 37

4.1.3数据可视化遇到的问题 40

4.1.4其他问题 40

4.2 程序改进设想 40

第五章 测试结果 41

5.1登陆界面 41

5.2程序主界面 42

5.3数据包处理界面 43

5.4数据包抓取 43

5.5数据包整理与统计 45

5.6 Ping操作 47

5.7 TraceRoute操作 48

5.8 ARP攻击操作 50

第六章 课程设计总结 53

参考文献 55

# 第一章 需求分析

## 1.1 设计目的

1.基本要求

完成一个基于Libpcap的网络数据包解析软件，具有易用、美观的界面。

2.具体要求

（1）能够解析本地或局域网的数据包，例如TCP包，UDP包，ICMP包等，能分析报头以及数据包内容。

（2）能分析数据包的版本、头长度、服务类型、数据包总长度、数据包标识、分段标志、分段偏移值、生存时间、上层协议类型、头校验和、源IP地址和目的IP地址等内容。并将上述解析结果规范显示。

（3）能解析每个数据包的内容（如果为加密信息不能解析则备注不能解析）

（4）能在制定时间段内解析数据包，当程序解析停止时，能分类统计在这段时间内解析的各类数据包的数量。

（5）能够图形化显示数据包统计结果；

## 1.2 功能要求

1.功能概况

本次设计参考计算机网络基础知识，设计实现一个能够对本地局域网数据包进行捕获和分析的程序，并利用此程序实现网络监控和其他实用性功能。

2.获取数据包功能

利用Libcap，扫描设备网卡，选取不同网卡类型（WiFi／以太）进行对局域网数据包的监听和捕获，设计两种模式：混杂模式和非混杂模式，简单来说就是是否要过滤掉不是发送给自己的数据包，如果设置为混杂模式，则会收到整个局域网内的所有数据包并进行分析。

3.分析数据包功能

**程序能够对捕获的数据包进行分类整理，并提取数据包中内容进行分析，**包括数据包的版本、头长度、服务类型、数据包总长度、数据包标识、分段标志、分段偏移值、生存时间、上层协议类型、头校验和、源IP地址和目的IP地址等内容，并将上述解析结果规范显示。对于HTTP、ARP等报文，可以对其内容进行解析。

4.图形化显示功能

**利用表格组件，将获取到的数据包信息以表格形式直观地显示出来，并且加以交换牛可以查看深层内容。**

5.数据包统计分析功能

**可以对一段时间获取的数据包进行统计整理，并根据数据包类型进行分类，如：数据包版本类型（IPv4/IPv6）、数据包协议类型（TCP／UDP／ARP等），并以饼图直观显示；对于TCP、UDP、ICMP数据包，用直方图统计其最大、最小、平均生存期和数据包大小，可以直观显示。**

6.数据包清空功能

**可以清空之前获取的所有数据包。**

7.Ping功能

**调用终端相应，实现Ping操作，可以用来测试与目标主机的连通性。**

8.TraceRoute功能

**调用终端相应，实现traceroute操作，通过traceroute我们可以知道信息从本计算机到互联网另一端的主机是走的什么路径。**

8.ARP-Attack功能

**本程序可实现局域网内ARP攻击，经测试可以对指定IP地址的主机实现断网攻击，基本原理同上，是欺骗目标机器局域网网关地址，使得该机器ARP缓存表中记录的网关IP－MAC对应错误，从而无法把数据包发送给真正的网关，导致网络连接失败，如果将欺骗的MAC地址设置为自己的MAC地址，则会截获目标机器发送的数据包。**

# 第二章 概要设计

## 2.1 系统分析

### 2.1.1 架构设计

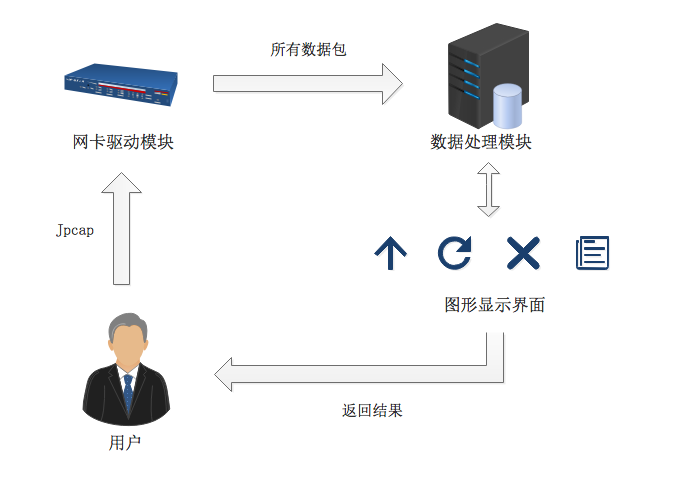
 本系统采用架构如图2.1所示，主要分为三个模块：用户操作界面、数据处理模块和网卡驱动模块。其中，网卡驱动模块是最底层模块，主要实现从硬件网卡中获取所有的数据包，利用Unix系统下的Libcap工具，采用jni动态链接库调用网络层之下的底层功能，集成jpcap用来实现对数据包的分析处理；数据处理模块主要用于对网卡驱动模块获得的数据包进行分类处理，根据数据包版本和协议不同用不同方式拆解分析数据包，并将所得到的结果整理分析，同时可以实现对数据包的重构和发送操作；用户操作界面为最顶层功能，便于以可视化的图形界面直接显示抓取数据包的内容，以及可视化其他操作所得到的结果，便于使用者更好的操作和使用。

图2.1 系统架构

### 2.1.2 实体设计

1.登陆实体

作为程序的进入窗口，用户登陆时，系统会要求用户输入账号和密码，并在数据库中进行匹配分析，若用户名存在且密码正确则通过身份认证，允许使用本软件。

2.主界面实体

登录成功后的功能选择界面，包含数据包抓取、Ping、TraceRoute、APR-Attack四个功能，其中，对于数据包抓取还细分为抓取数据包和分析数据包功能。

3.数据包抓取实体

从主界面中选择进入，负责调用Libpcap抓取被监听网卡的数据包，可选择是否设置为混杂模式（是否过滤），并可以分类解析所有数据包中的内容，并以图形化界面显示。抓取结束后还可以对抓取到的数据包做统计整理，并将结果用图形化界面显示。

4.Ping功能实体

从主界面中选择进入，输入要测试的网站和测试的次数，可以得到访问该网站的往返RTT，在程序运行结束后可以汇总最大、最小、中值RTT，并可以用折线图将所有RTT统计显示。

5. TraceRoute功能实体

从主界面中选择进入，输入要追踪的网站和测试的次数，可以得到访问该网站的路径（IP地址），如果中途站点拒绝相应会显示“＊”，在程序运行结束后可以汇总所经过的站点IP地址。

6. ARP-Attack功能实体

从主界面中选择进入，系统首先自动获取网管IP－MAC地址作为最低地址，同时得到广播地址以表示最高地址，并将该局域网内已连接设备的IP—MAC映射表先是在表格中。用户选择网卡类型后，在表格中选择相应的IP地址便可以对该设备实现APR攻击，使该设备无法连接到互联网。

### 

图2.2 实体设计

### 2.1.3 分层设计

1.Jpcap驱动层（数据层）

通过Java扩展jar－Jpcap调取Libpcap基本功能，以及jni包，对设备的底层网卡进行监听，将经过网卡的数据包捕获下来，可以应对不同网卡类型，如wifi、以太、网桥等，同时支持设置混杂模式以监控局域网内的所有数据包。

2.数据处理层（功能层）

主要是对接收到的数据包进行分类处理，将数据包分为IP数据包和ARP数据包，再对IP数据包协议进行解析从而分出TCP、UDP、ICMP、IGMP等数据包，同时对不用版本（IPv4、IPv6）的数据包也做区分，将分类结果

3.用户界面层（表示层）

接受来自数据处理层的处理结果：对于异常，采用消息框提示的方式展示给用户；对于抓取的数据包分析结果，以表格的形式显示在界面上，用户可以点击表格中某一项查看详细信息；对于数据包的统计结果，自动绘制图表显示给用户；其他提示信息同样采用消息框给用户反馈。同时用户可以直接在界面上选择相应功能，底层会自动实现相应操作并反馈给用户界面层。同样，Ping、TraceRoute、APR-Attack功能也有相应的图形化界面。

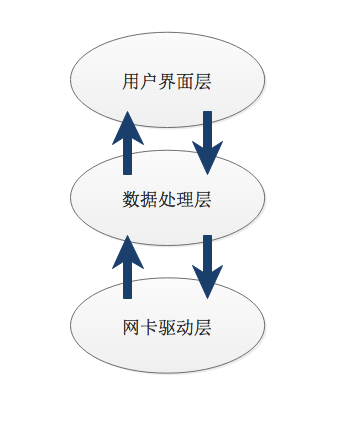


图2.3 系统层次结构

### 2.1.4 数据库设计

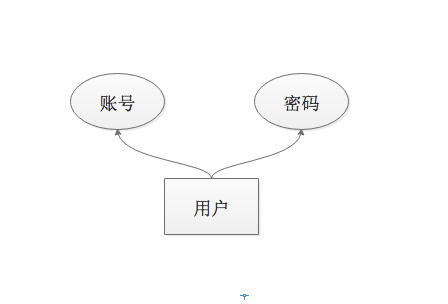
本程序中数据库仅做身份认证功能，不需要将统计到的数据包保存在数据库中，一方面是不需要检索和查询等操作，另一方面是数据库的使用并不会带来效率的提升，因此数据库仅仅保存用户名和密码，在登陆时对数据库进行匹配判断是否允许登陆。

图2.3 数据库ER图

## 2.2 程序流程

### 2.2.1 程序结构图

图2.4 程序结构图

### 2.2.2 调用关系

运行程序后，首先进入用户登陆界面，用户名与密码不匹配则登陆失败，成功则进入用户操作主界面。

在用户操作主界面中有四个按钮供用户选择，分别是：Packing、Ping、TraceRoute、Hack，分别对应网络数据包抓取分析功能、Ping功能、TraceRoute功能、ARP攻击功能，点击不同按钮跳转至不同功能实现界面。

网络数据包抓取分析功能分为数据包抓取、数据包分析和数据包重置功能：数据包抓取模块可以选取计算机指定网卡进行数据包的抓取操作，自动调用数据处理层的方法对数据包进行分类解析，并将解析结果显示在表格上，若用户想查看详细信息，可以点击表格中具体的某一项进行查看；数据包分析功能可以对本段时间抓取的所有数据包进行整理分析，并使用JFreeChart将所有数据可视化显示，直观的表现网络中流量的状态；数据包重置可以将本段时间抓取的数据包清空，便于下一次分析统计。

Ping操作调用Unix终端的ping指令，用于测试一个站点的往返延迟（RTT），本程序以图形化界面实现网络命令ping操作，并允许指定ping操作的执行次数，最终可以在程序运行后显示最大最小与均值RTT，点击绘图按钮可以以折线图统计本段时间所有的往返延迟。

TraceRoute操作调用Unix终端的trace指令，用于追踪访问一个站点所经过的路由器（IP弟子），本程序以图形化界面实现网络命令trace操作，并允许指定trace操作的测试包数目，最终可以在程序运行后显示所经过的全部站点。

ARP攻击操作可以实现ARP模拟攻击，可以搜索本机局域网内的所有设备ip地址，利用ARP诈骗原理实现对靶机的断网攻击，核心内容是使靶机无法找到网关IP地址所对应的MAC地址，从而使所有数据包无法发出局域网。

# 第三章 详细设计

## 3.1 功能类定义与实现

### 3.1.1 常量与参数的定义（Parameter类）

1.JDBC数据库连接所用参数与端口

JDBC（Java Data Base Connectivity,java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问，它由一组用Java语言编写的类和接口组成。JDBC为数据库开发人员提供了一个标准的API，据此可以构建更高级的工具和接口，使数据库开发人员能够用纯 Java API 编写数据库应用程序，并且可跨平台运行，并且不受数据库供应商的限制。

本系统采用的是MySQL数据库，编程语言为Java语言，编译器为Eclipse Oxygen，Java连接数据库时需要用到JDBC驱动器、数据库URL以及数据库管理员账号密码等。

根据数据库的URL、用户名和密码，创建一个JDBC Connection 对象。如：Connection connection = DriverManager.geiConnection(“连接数据库的URL", "用户名", "密码”)。其中，URL=协议名+IP地址(域名)+端口+数据库名称；用户名和密码是指登录数据库时所使用的用户名和密码。示例创建MySQL的数据库连接代码如下：

Connection connectMySQL = DriverManager.geiConnection(“jdbc:mysql://localhost:3306/myuser","root" ,"root" );

对应代码如下：

public static final String JDBC\_DRIVER = "com.mysql.jdbc.Driver";

public static final String DB\_URL = "jdbc:mysql://localhost:3306/java?useSSL=true&characterEncoding=UTF8";

public static final String ACC = "root";

public static final String PWD = "zxczxc";

2.统计部分参数定义

由于在统计部分需要用到已经捕获到的数据包的一些数据，例如：不同版本数据包的数量、不同协议数据包的数量、某协议数据包最大最小与均值生存期及数据包长度等，单独建立一个类用来统计所有的信息：

package Util;

public class Statistic {

// 不同协议总数

public static int IPv4 = 0;

public static int IPv6 = 0;

public static int OtherVersion = 0;

// 不同类型数据包总数

public static int TCP = 0;

public static int UDP = 0;

public static int ICMP = 0;

public static int IGMP = 0;

public static int ARP = 0;

public static int Others = 0;

// 数据包长度最大值、最小值、平均值

public static int TCP\_maxLen = 0;

public static int UDP\_maxLen = 0;

public static int ICMP\_maxLen = 0;

（其他变量定义不再举例）

### 3.1.2 IP/ARP数据包显示信息的定义

1.IP数据包

TCP/IP协议定义了一个在因特网上传输的包，称为IP数据包，而IP数据报(IP Datagram)是个比较抽象的内容，是对数据包的结构进行分析。 由首部和数据两部分组成，其格式如图所示。首部的前一部分是固定长度，共20字节，是所有IP数据报必须具有的。在首部的固定部分的后面是一些可选字段，其长度是可变的。首部中的源地址和目的地址都是IP协议地址。

IP数据包解析后的内容包括：版本、协议、数据包标识、生存时间、头长度、数据包总长度、优先权、服务类型、区分服务（最大吞吐量和最高的可靠性）、是否分段、分段偏移量、源IP地址与目的IP地址、源MAC地址与目的MAC地址、校验和；

针对UDP、TCP数据包，还设定了源端口号和目的端口号；

针对ICMP数据包（v4/v6）还设定了ICMP报文代码和报文类型；

除此之外可以详细显示DNS、HTTP等数据包的包内容。

2. ARP数据包

在网络通讯时，源主机的应用程序知道目的主机的IP地址和端口号，却不知道目的主机的硬件地址，而数据包首先是被网卡接收到再去处理上层协议的，如果接收到的数据包的硬件地址与本机不符，则直接丢弃。因此在通讯前必须获得目的主机的硬件地址。ARP协议就起到这个作用。源主机发出ARP请求，询问“IP地址是192.168.0.1的主机的硬件地址是多少”，并将这个请求广播到本地网段（以太网帧首部的硬件地址填FF:FF:FF:FF:FF:FF表示广播），目的主机接收到广播的ARP请求，发现其中的IP地址与本机相符，则发送一个ARP应答数据包给源主机，将自己的硬件地址填写在应答包中。这就是ARP数据包。

ARP数据包解析后的内容包括：版本、协议、头长度、数据包总长度、硬件类型、协议类型、硬件地址长度、协议地址长度、操作字段、源IP地址与目的IP地址、源MAC地址与目的MAC地址；

除此之外可以详细显示ARP数据包的包内容。

### 3.1.3 捕获数据包的实现（HandleJpcap包）

1.使用Jpcap捕获数据包并分类处理

Java的java.net包中，给出了传输层协议TCP和UDP有关的API，用户只能操作传输层数据，要想直接操作网络层，比如自己写传输层数据报（自己写传输层包头），或者自己写好IP数据包向网络中发则是无能为力的。 而Jpcap扩展包弥补了这一点，使我们可以支持从网卡中接收IP数据包，或者向网卡中发送IP数据包。

Jpcap实际上并非一个真正去实现对数据链路层的控制，而是一个中间件，JPCAP调用wincap/libpcap，而给JAVA语言提供一个公共的接口，从而实现了平台无关性。

下载libcap（Linux上的Wincap），使用jni连接C语言调用底层网卡功能，再建立Java动态链接库，实现jpcap.jar的生成。

使用jpcap后，编写程序按要求完整数据包的抓取，可以获得不同类型数据包，并根据数据包类型对不同数据包进行解析，可以在表格上显示不同数据包的类型、数据包标识、版本、源IP地址、目的IP地址以及数据包长度。对表格每一项可以选择查看其具体信息，对不同数据包有不同的描述，IP包为一大类，ARP包为另一类。

Jpcap包中主要是用到的方法如下：

JpcapHandler ：这个接口用来定义分析被捕获数据包的方法

ARPPacket ：这个类描述了ARP/RARP包，继承了Packet类

DatalinkPacket ：这个抽象类描述了数据链路层

EthernetPacket ：这个类描述了以太帧包，继承DatalinkPacket类

ICMPPacket ：这个类描述了ICMP包，继承了IPPacket类

IPAddress ：这个类描述了IPv4和IPv6地址，其中也包含了将IP地址转换为域名的方法

IPPacket ：这个类描述了IP包，继承了Packet类，支持IPv4和IPv6

Jpcap ：用来捕获数据包

JpcapSender ：它用来发送一个数据包

Packet ：这个类是所有被捕获的数据包的基类

TCPPacket ：这个类描述TCP包，继承了IPPacket类

UDPPacket ：这个类描述了UDP包，继承了IPPacket类

抓取数据包过程如下：

（1）首先，获得所有网卡列表

NetworkInterface[] devices = JpcapCaptor.getDeviceList();

（2）通过网卡，打开一个捕捉器。

openDevice(NetworkInterface intrface, int snaplen, boolean promics,

int to\_ms);

JpcapCaptor captor=JpcapCaptor.openDevice(device[index], 65535, false, 20);

|  |
| --- |
| 各个参数意义：  intrface 需要监听的网卡  snaplen 每次捕获的数据包最大长度（设置为IP包最大长度即可）  promics 是否过滤（Mac地址不是当前网卡的IP数据包）  to\_ms 超时时间 |

（3）通过捕捉器捕获数据：

本程序采用主动获取的方法。通过捕获器的getPacket() 可以主动的，一个数据包一个数据包的获取。主动获取，可以每次需要处理数据包的时候，自己去主动获取。然后写出处理逻辑。但是因为不知道何时去产生数据包，而一直去调用getPacket()，因此需要开辟多线程，让用户自行终止数据包的捕获。

（4）数据包分类：

利用 packet instanceof XXPacket对捕获到的数据包进行分类，并对不同类型的数据包进行分类拆解，具体方法举例如下（IP数据包）：

Packet packet = jpcap.getPacket();

if(packet instanceof IPPacket)

{

IPPacket ip = (IPPacket)packet;//强转

switch(new Integer(ip.protocol))

{

case 1:{

protocol = "ICMP";

//处理ICMP数据包

break;

}

case 2:protocol = "IGMP";Statistic.IGMP++;break;

case 6:{

protocol = "TCP";

//处理TCP数据包

break;

}

case 8:

{protocol = "EGP";

//处理IGP数据包;

break;

}

case 9:

{protocol = "IGP";

//处理IGP数据包

break;

}

case 17:{

protocol = "UDP";

//处理UDP数据包

break;

}

case 41:

{protocol = "IPv6";

//处理IPv6数据包

break;

}

case 58:{

protocol = "IPv6-ICMP";

//处理IPv6-ICMP数据包

break;

}

case 89:{

protocol = "OSPF";

//处理OSPF数据包

break;

]

default : {

protocol = String.valueOf(ip.protocol);

//处理其他数据包

break;

}

}

Jpcap还可以完成以下功能

（1）Jpcap是直接抓取经过数据链路层的数据包。 因此可以自己写IP数据包直接发送给数据链路层。

（2）Jpcap会对抓取到的数据包进行一定程序的解析，根据数据包内容，将数据包封装为对应的对象（）。

（3）根据用户设定的信息，过滤数据包（其实就是在解析的时候，对不需要的数据直接丢弃，不解析）

（4）Jpcap 只是直接从数据链路层上读取数据，并向数据链路层中发送数据，因此，Jpcap并不能操作 其他程序从数据链路层中读数据或者向网卡中发送数据。（IP层协议程序）

2.数据包存储数据结构的定义（Packet类与AllPackets类）

为方便整理统计各类数据包，我们需要采用一种数据结构用来存放前一段时间捕获到的所有数据包以便统计，为此单独设置一个Packet类用来描述单个数据包（ARP／IP数据包通用），对所有的数据包，创建一个集合ArrayList用来保存所有的Packet，放入AllPackets类中,可以对这个集合增添数据包，在某些时刻可以清除已有的数据包，具体描述如下：

public class Packages {

public int ID;

public String version; //版本

public String priority; //优先权

public String header\_length; //头长度

public String rsv\_tos; //服务类型

public String TF; //区分服务：最大的吞吐量

public String RF; //区分服务：最高的可靠性

public String total\_length; //数据包总长度

public String ident; //数据包标识

public String DF; //Don't Fragment

public String MF; //More Fragment

public String offset; //分段偏移量

public String TTL; //生存时间

public String protocol; //上层协议类型

public String srcIP; //源IP地址

public String dstIP; //目的IP地址

public String srcMAC; //源MAC地址

public String dstMAC; //目的MAC地址

public String srcHost; //源主机名

public String dstHost; //目的主机名

public String srcPort; //源端口号

public String dstPort; //目的端口号

public String checksum; //校验和

public String ICMP\_code; //ICMP 报文代码

public String ICMP\_type; //ICMP 报文类型

public String info=""; //包内容

}

public class AllPackages {

public static ArrayList<Packages> allpackages = new ArrayList<Packages>();

public static int tag = 1;

public static void Insert(Packages p) {

allpackages.add(p);

p.steID(tag);

tag++;

}

public static void RemoveAll() {

for(int i=allpackages.size()-1;i>=0;i--)

allpackages.remove(i);//循环清空

tag = 1;//指针重置

}

}

### 3.1.4 Ping操作实现部分（HandlePing包）

1.Ping 功能作用

Linux系统的ping命令是常用的网络命令，它通常用来测试与目标主机的连通性，我们经常会说“ping一下某机器，看是不是开着”、不能打开网页时会说“你先ping网关地址192.168.1.1试试”。它通过发送ICMP ECHO\_REQUEST数据包到网络主机，并显示响应情况，这样我们就可以根据它输出的信息来确定目标主机是否可访问（但这不是绝对的）。有些服务器为了防止通过ping探测到，通过防火墙设置了禁止ping或者在内核参数中禁止ping，这样就不能通过ping确定该主机是否还处于开启状态。

linux下的ping和windows下的ping稍有区别,linux下ping不会自动终止,需要按ctrl+c终止或者用参数-c指定要求完成的回应次数。

（1）命令格式：

ping [参数] [主机名或IP地址]

（2）命令功能：

ping命令用于：确定网络和各外部主机的状态；跟踪和隔离硬件和软件问题；测试、评估和管理网络。如果主机正在运行并连在网上，它就对回送信号进行响应。每个回送信号请求包含一个网际协议（IP）和 ICMP 头，后面紧跟一个tim结构，以及来填写这个信息包的足够的字节。缺省情况是连续发送回送信号请求直到接收到中断信号（Ctrl-C）。

ping命令每秒发送一个数据报并且为每个接收到的响应打印一行输出。ping命令计算信号往返时间和(信息)包丢失情况的统计信息，并且在完成之后显示一个简要总结。ping命令在程序超时或当接收到SIGINT信号时结束。Host参数或者是一个有效的主机名或者是因特网地址。

3.命令参数：

-d 使用Socket的SO\_DEBUG功能。

-f 极限检测。大量且快速地送网络封包给一台机器，看它的回应。

-n 只输出数值。

-q 不显示任何传送封包的信息，只显示最后的结果。

-r 忽略普通的Routing Table，直接将数据包送到远端主机上。通常是查看本机的网络接口是否有问题。

-R 记录路由过程。

-v 详细显示指令的执行过程。

<p>-c 数目：在发送指定数目的包后停止。

-i 秒数：设定间隔几秒送一个网络封包给一台机器，预设值是一秒送一次。

-I 网络界面：使用指定的网络界面送出数据包。

-l 前置载入：设置在送出要求信息之前，先行发出的数据包。

-p 范本样式：设置填满数据包的范本样式。

-s 字节数：指定发送的数据字节数，预设值是56，加上8字节的ICMP头，一共是64ICMP数据字节。

-t 存活数值：设置存活数值TTL的大小。

2.Java调用Unix终端实现Ping操作

本扩展功能使用Java调用终端Ping指令，在表格上显示Ping的相关信息，并绘制Ping操作往返延迟折线图，从而实现图形化操作的Ping指令：

//将结果暂存下来

PrintStream ps = new PrintStream(new FileOutputStream("Ping.txt", false));

//调用控制台操作

Runtime rt = Runtime.getRuntime();

Process pro = rt.exec("Ping -c " + times + " " + host);

BufferedReader buf = new BufferedReader(new InputStreamReader(pro.getInputStream()));

String line;

while ((line = buf.readLine()) != null) {

table.InsertRow(line);

ps.println(line);

}

ps.close();

3.利用正则表达式截取从终端返回的信息

为方便统计所有的ping结果，并获得最大、最小、均值RTT，我们需要对终端返回的信息进行筛选，由于java中已存在封装好的剪切字符串的方法，可以不用自己编写正则表达式，筛选方法如下：

if (index < num\_packets) {

if (result.split("time=").length < 2)//RTT超时

{

allstring.add(result);

rtt.add(-1.00);

index++;

}

else {//摘取每个字符串里的RTT

allstring.add(result);

String Rtt = result.split("time=")[1].split(" ms")[0];

rtt.add(Double.parseDouble(Rtt));

index++;

}

} else {

allstring.add(result);

}

}

//找到返回的round-trip min/avg/max/stddev和丢包率

for(int index2 = allstring.size()-1;index2>=0;index2--)

{

if(allstring.get(index2)!=null)

{

next = allstring.get(index2);

nextnext =allstring.get(index2-1);

break;

}

}

br.close();

try{

drop\_rate = nextnext.split("received, ")[1].split(" packet loss")[0];//丢包率

String info = next.split(" = ")[1];

max\_rtt = info.split("/")[2];//最大RTT

median\_rtt = info.split("/")[1];//均值RTT

min\_rtt = info.split("/")[0];//最小RTT

}catch (Exception e) {

drop\_rate = "100.0%";

max\_rtt="-1";

median\_rtt = "-1";

min\_rtt = "-1.0";

}

if (drop\_rate == null || drop\_rate.equals("100.0%")) {

drop\_rate = "100.0%";

max\_rtt = "-1.0";

median\_rtt = "-1.0";

min\_rtt = "-1.0";

}

经过以上方法，每一次ping操作的往返延迟便被记录下来，我们用折线图直观的展现，同时，最大最小与均值RTT也被记录了下来，表现如下：

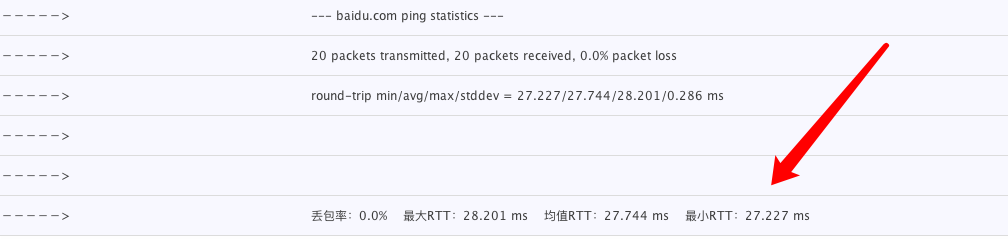


图3.1 Ping结果

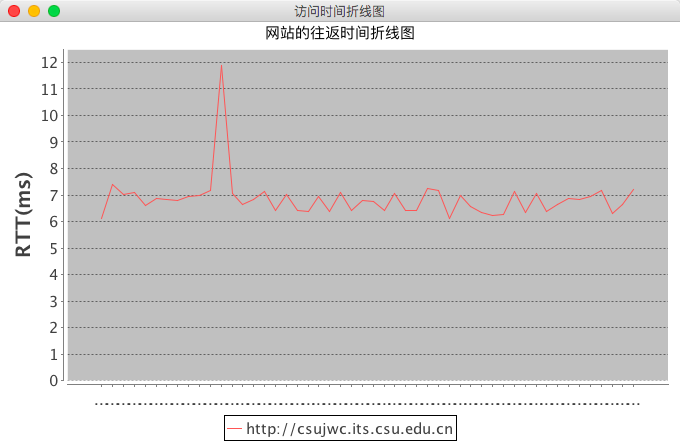


图3.2 Ping结果折线图

### 3.1.5 TraceRoute操作实现部分（HandleTraceRoute包）

1.TraceRoute 功能作用

通过traceroute我们可以知道信息从你的计算机到互联网另一端的主机是走的什么路径。当然每次数据包由某一同样的出发点（source）到达某一同样的目的地(destination)走的路径可能会不一样，但基本上来说大部分时候所走的路由是相同的。linux系统中，我们称之为traceroute,在MS Windows中为tracert。 traceroute通过发送小的数据包到目的设备直到其返回，来测量其需要多长时间。一条路径上的每个设备traceroute要测3次。输出结果中包括每次测试的时间(ms)和设备的名称（如有的话）及其IP地址。

在大多数情况下，我们会在linux主机系统下，直接执行命令行：

traceroute hostname

1.命令格式：

traceroute[参数][主机]

2.命令功能：

traceroute指令让你追踪网络数据包的路由途径，预设数据包大小是40Bytes，用户可另行设置。

具体参数格式：traceroute [-dFlnrvx][-f<存活数值>][-g<网关>...][-i<网络界面>][-m<存活数值>][-p<通信端口>][-s<来源地址>][-t<服务类型>][-w<超时秒数>][主机名称或IP地址][数据包大小]

3.命令参数：

-d 使用Socket层级的排错功能。

-f 设置第一个检测数据包的存活数值TTL的大小。

-F 设置勿离断位。

-g 设置来源路由网关，最多可设置8个。

-i 使用指定的网络界面送出数据包。

-I 使用ICMP回应取代UDP资料信息。

-m 设置检测数据包的最大存活数值TTL的大小。

-n 直接使用IP地址而非主机名称。

-p 设置UDP传输协议的通信端口。

-r 忽略普通的Routing Table，直接将数据包送到远端主机上。

-s 设置本地主机送出数据包的IP地址。

-t 设置检测数据包的TOS数值。

-v 详细显示指令的执行过程。

-w 设置等待远端主机回报的时间。

-x 开启或关闭数据包的正确性检验。

2.Java调用Unix终端实现TraceRoute操作

本扩展功能使用Java调用终端traceroute 指令，在表格上显示traceroute 的相关信息，并绘制traceroute 操作经过的站点，从而实现图形化操作的traceroute 指令（显示\*为站点拒绝给以相应）:

//将结果暂存下来

String fileName ="Traceroute.txt";

PrintStream ps = new PrintStream(new FileOutputStream(fileName, false));

//调用控制台命令

Runtime rt = Runtime.getRuntime();

Process pro = rt.exec("traceroute -I -q " + times + " " + host);

BufferedReader buf = new BufferedReader(new InputStreamReader(pro.getInputStream()));

String line;

while ((line = buf.readLine()) != null) {

table.InsertRow(line);

ps.println(line);

}

ps.close();

3.利用正则表达式截取从终端返回的信息

为方便统计所有的trace到的站点结果，我们需要对终端返回的信息进行筛选，由于java中已存在封装好的剪切字符串的方法，可以不用自己编写正则表达式，筛选方法如下：

if(line.startsWith("traceroute"))continue;

String ip = null;

try {//切割出访问的站点名称

ip = line.split("\\(")[1].split("\\)")[0];

} catch (Exception e) {//如果为＊，说明没有回应，站点不可见

ip = "No Answer";

}

经过以上方法，每一次trace的站点便被记录下来，表现如下：

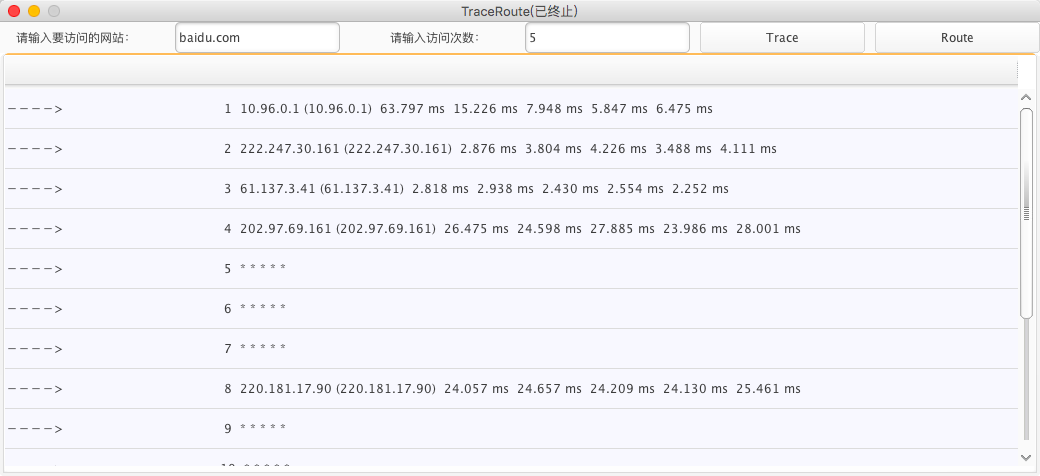


图3.3 TraceRoute显示

### 3.1.6 ARP-Attack操作实现部分（ARPAttack包）

1.ARP原理简介

每台主机都设有一个ARP高速缓存（ARP cache），里面有本局域网上各主机和路由器的IP地址和硬件地址的映射表，这些都是该主机目前知道的一些地址。

当主机A要向本局域网上的某个主机B发送IP数据报时，先在其ARP高速缓存中查看是否有主机B的IP地址。如果有，就在ARP高速缓存中查出其对应的硬件地址，再把这个硬件地址写入MAC帧，然后通过局域网把该MAC帧发往此硬件地址。也有可能查不到主机B的IP地址的项目。这可能是主机B才入网，也可能是主机A刚刚加电，其高速缓存还是空的。在这种情况下，主机A就自动运行ARP，然后按以下步骤找到主机B的硬件地址。

实现地址解析的第一步是产生ARP请求帧。在ARP帧数据部分的相应字段写入本地主机的物理地址、IP地址、待侦测的目的IP地址，在目的物理地址字段写入0，并在操作类型字段写入1，用以表示本数据帧是一个ARP请求数据帧。

该ARP请求帧以本地网络适配器的物理地址作为源地址，以物理广播地址（FF-FF-FF-FF-FF-FF）作为目的地址，通过物理层发送出去。

由于采用了广播地址，因此网段内所有的主机或设备都能够接收到该数据帧。除了目的主机外，所有接收到该ARP请求帧的主机和设备都会丢弃该ARP请求帧，因为目的主机能够识别ARP消息中的IP地址是否与本机相同。

与目的IP地址匹配的主机构造ARP应答帧。在ARP应答帧中，以请求分组中源物理地址、源IP地址作为其目的物理地址、目的IP地址，并将自身的物理地址、IP地址填入应答帧的源物理地址、源IP地址字段，并在操作字段中写入2，表示本ARP数据帧是一个应答数据帧。该分组通过数据链路层直接发给源主机。

源主机接收到ARP应答帧后，获得目的主机的物理地址，并将它作为一条新记录加入到ARP高速缓存表。

此外，如果源主机没有发送ARP请求而收到其他主机的ARP响应数据帧，源主机也会在本地ARP缓冲区中缓存该主机物理地址和IP地址的对应关系。

ARP高速缓存是非常有用的。如果不使用ARP高速缓存，那么任何一个主机只要进行一次通信，就必须在网络上用广播的方式发送ARP请求分组，这会使网络上的通信量大大增加。ARP把保存在高速缓存中的每一个映射地址项目都设置生存时间，超过生存时间的项目就从高速缓存中删除掉。

［ARP是解决同一个局域网上的主机或路由器的IP地址和硬件地址的映射问题。］

2.ARP数据包格式

在了解ARP原理的基础上，还必须了解ARP数据包的格式才能实现ARP诈骗。

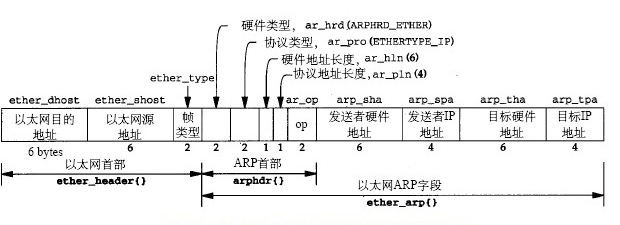


图3.4 ARP数据包格式（包头部分图片描述）

注意到源MAC地址、目的MAC地址在以太网首部和ARP请求中各出现一次，对于链路层为以太网的情况是多余的，但如果链路层是其它类型的网络则有可能是必要的。硬件类型指链路层网络类型，1为以太网，协议类型指要转换的地址类型，0x0800为IP地址，后面两个地址长度对于以太网地址和IP地址分别为6和4（字节），op字段为1表示ARP请求，op字段为2表示ARP应答。

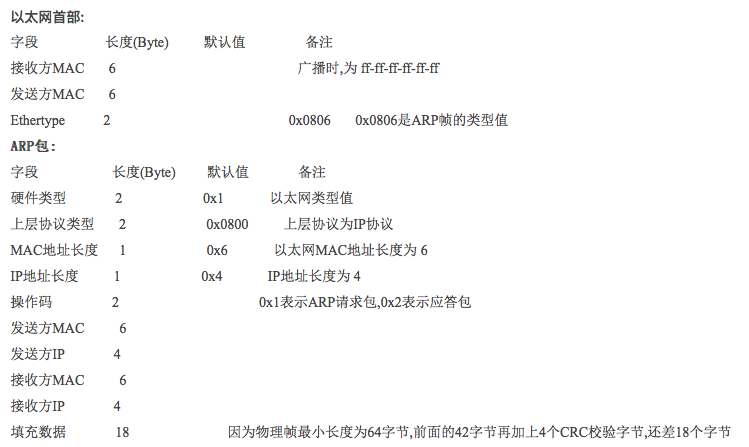


图3.5 ARP数据包格式（文字描述）

3.ARP诈骗原理

一台电脑通过网络访问另一台电脑的时候,在数据链路层需要知道对方的MAC地址进行真正的物理通信.

而电脑上的应用程序通常都是根据另一台电脑的IP地址来和对方建立通信,这时候就需要有一个协议将IP地址解析到MAC地址,这就是ARP协议.

而ARP具体过程就是当需要通过IP获取一个远端的的MAC地址的时候，系统会首先检查ARP表中是否存在对应的IP地址，如果没有，则发送一个ARP广播，当某一个拥有这个MAC地址的节点收到ARP请求的时候，会创建一个ARP reply包，并发送到ARP请求的源节点，ARP Reply包中就包含了目的地节点的MAC地址，在源节点接受到这个reply后，会将目的地节点的MAC地址保存在ARP缓存表中，下一次再次请求同一 IP地址的时候，系统将会从ARP表中直接获取目的地MAC地址，而不需要再次发送ARP广播。

用ping命令举例来说, 在机器A,比如说IP是192.168.1.2中发出下面的ping命令:

ping 192.168.1.3,如果192.168.1.3这个MAC地址在机器A的ARP缓存表中不存在,这时候机器A就会发送一个ARP广播,当192.168.1.3接到广播后,会给机器A回一个ARP Reply包,其中包含了192.168.1.3的MAC地址,这是正常的ARP过程.

假设IP--MAC的对应关系为(192.168.1.3--00-1C-23-2E-A7-0E)

但是局域网内的其他机器也可向机器A发一个假的IP--MAC对应关系ARP Reply包,机器A接收到这个假的包后同样会更新自己的ARP缓存表.假设发送的为(192.168.1.3--00-1C-23-2E-A7-0A,0A实际上为192.168.1.4的MAC地址)那么机器A再通过IP往192.168.1.3发送数据的时候,实际上却发到了192.168.1.4这台机器.这样就到达了ARP攻击或者叫做ARP欺骗.ARP攻击只能发生在局域网内。

ARP欺骗分为两种，一种是针对路由器进行欺骗，使路由器的ARP缓存中建立错误的IP与MAC地址映射表，结果就是从路由器发送的数据都发给了一个错误的MAC地址，造成主机无法正常接收信息。另一种是对主机中的ARP缓存进行欺骗，伪造主机路由器IP和MAC地址映射表，使主机发送的数据都发送到伪造后的MAC地址对应的主机上，这种情况不仅会使主机不能正常上网，而且还能用来窃取信息。

4.本程序ARP诈骗方式

我们用程序实现第二种ARP欺骗，并对第二种欺骗方式进行举例。

假设现在主机A和主机B通过路由器C连接在同一个局域网内，主机A是欺骗方，主机B是被欺骗方。主机A在自己的ARP缓存中查到路由器C的IP地址IP\_C，MAC地址MAC\_C，查到主机B的IP地址IP\_B。接下来，主机A自己生成一个ARP回复，该回复的源IP并不是主机A的IP地址，而是冒充路由器C的IP地址IP\_B，同时编造一个不存在的MAC地址作为回复的源MAC地址（可以设置为自己主机的MAC地址用来窃听），目标IP是IP\_B，目标MAC地址是MAC\_B。这个ARP回复以单播的形式发送给主机B之后，主机B并不会验证回复的真实性，而是会把路由器的IP\_C和一个错误的MAC地址写入自己的ARP缓存，此时，所有主机B通过路由器发送给外界的数据，都会因为找不到路由器而发送失败。（在此，路由器也称作为网关）。

总结一下：主机A冒充路由器C不断向主机B发送一个包含错误的IP和MAC映射表的ARP回复，使主机B经由路由器发给外界的数据无法到达目的地，从而致使主机B无法正常上网。

5.ARP攻击实现流程

本实验ARP攻击实现如图3.6所示：



图3.6 ARP攻击流程

在本机接入局域网时，首先调取终端功能查询本机IP地址和MAC地址，并获取局域网网关IP地址和MAC地址，同时获取广播地址查看网段大小；

接入局域网监听一段时间后，可自动获取局域网内所有设备的IP地址与MAC地址，这些信息缓存在系统arp表中，可以将其查询出来：



图3.7 自动获取的局域网信息

然后选择要攻击设备的IP地址，选定伪装的MAC地址（如果设置为本机MAC地址则会截获所有数据包，起窃听作用），设置网卡模式（WiFi／以太），设置攻击间隔（多长时间攻击一次），便可开始ARP攻击。

6.ARP诈骗注意事项

（1）由于存在ARP代理，对于没有配置缺省网关的计算机要和其他网络中的计算机实现通信，网关收到源计算机的 ARP 请求会使用自己的 MAC 地址与目标计算机的 IP地址对源计算机进行应答。因此不可以随意指定IP地址，因为如果是错误的IP地址，网关会用自己的MAC地址做应答，然而实际无意义，因此需要判断返回的ARP应答对应的MAC地址是否为网关地址。

（2）需要在进行攻击之前发送心跳包，测试指定的IP地址是否存在，是否是一个真正的设备而不是预留的设备，因此需要测试：

if(Arrays.equals(p.target\_protoaddr,senderIP.getAddress())){

if(Arrays.equals(p.sender\_hardaddr,stomac(GatewayMAC))) {

//攻击失败

jc.close();

return null;

}

## 3.2 图形界面类定义与实现

### 3.2.1 JFreeChart制图（JFreeChart类）

1.柱形图制作

传入图表的名称（标题）、Dataset容器（存放图形参数）、以及所要绘制图形的颜色，返回绘制的Chart图形。

public class CategoryChart {

public static JFreeChart createChart(String name, CategoryDataset dataset,Color color) {

JFreeChart chart = ChartFactory.createBarChart3D(name, null, null, dataset, PlotOrientation.VERTICAL, true,

false, false);

CategoryPlot plot = chart.getCategoryPlot();// 设置图的高级属性

BarRenderer3D renderer = new BarRenderer3D();// 3D属性修改

plot.setNoDataMessage("无 信 息");

renderer.setSeriesPaint(0,color); //设置柱的颜色

// 设置网格竖线颜色

plot.setDomainGridlinePaint(Color.gray);

plot.setDomainGridlinesVisible(true);

// 设置网格横线颜色

plot.setRangeGridlinePaint(Color.gray);

plot.setRangeGridlinesVisible(true);

// 图片背景色

plot.setBackgroundPaint(Color.LIGHT\_GRAY);

plot.setOutlineVisible(true);

// 设置墙体颜色

renderer.setWallPaint(Color.gray);

// 对X轴做操作

CategoryAxis domainAxis = plot.getDomainAxis();

// 对Y轴做操作

ValueAxis rAxis = plot.getRangeAxis();

renderer.setBaseItemLabelGenerator(new StandardCategoryItemLabelGenerator());

renderer.setBaseItemLabelsVisible(true);

renderer.setBaseItemLabelFont(new Font("黑体", Font.CENTER\_BASELINE, 12), true);

renderer.setBasePositiveItemLabelPosition(

new ItemLabelPosition(ItemLabelAnchor.OUTSIDE12, TextAnchor.CENTER\_LEFT));

renderer.setItemLabelAnchorOffset(15);

//设置距离图片左端距离

domainAxis.setUpperMargin(0.2);

//设置距离图片右端距离

domainAxis.setLowerMargin(0.2);

//设置柱的透明度

plot.setForegroundAlpha(0.6f);

// 最后还得将renderer 放到plot 中

plot.setRenderer(renderer);// 将修改后的属性值保存到图中

return chart;

}

}

2.饼图制作

传入图表的名称（标题）、Dataset容器（存放图形参数），饼图颜色自动分配，返回绘制的Chart图形。

public class PieChart {

public static JFreeChart createChart(String name,PieDataset piedataset) {

JFreeChart jfreechart = ChartFactory.createPieChart3D(

name, piedataset, true, true, false);

PiePlot3D pieplot3d = (PiePlot3D) jfreechart.getPlot();

pieplot3d.setNoDataMessage("无 信 息");

pieplot3d.setLabelGenerator(new StandardPieSectionLabelGenerator("{0}:{1}({2})"));//设置标签显示的格式。

pieplot3d.setSectionOutlinesVisible(false);

pieplot3d.setLabelBackgroundPaint(new Color(220, 220, 220));//设置标签的背景颜色。

//设置旋转角度

pieplot3d.setStartAngle(180.0);

//设置旋转方向，Rotation.CLOCKWISE)为顺时针。

pieplot3d.setDirection(Rotation.CLOCKWISE);

//设置图表透明图0.0~1.0范围。0.0为完全透明，1.0为完全不透明。

pieplot3d.setForegroundAlpha(0.9F);

return jfreechart;

}

}

2.折线图制作

public static void drawRTT(String host,double[] RTT) throws Exception{

StandardChartTheme mChartTheme = new StandardChartTheme("CN");

mChartTheme.setLargeFont(new Font("黑体", Font.BOLD, 20));

mChartTheme.setExtraLargeFont(new Font("宋体", Font.PLAIN, 15));

mChartTheme.setRegularFont(new Font("宋体", Font.PLAIN, 15));

ChartFactory.setChartTheme(mChartTheme);

DefaultCategoryDataset mDataset = new DefaultCategoryDataset();

int i=1;

for(double rtt:RTT)

{

mDataset.addValue(rtt, host,String.valueOf(i++));

}

JFreeChart mChart = ChartFactory.createLineChart("网站的往返时间折线图", "", "RTT(ms)", mDataset,

PlotOrientation.VERTICAL, true, true, false);

CategoryPlot mPlot = (CategoryPlot) mChart.getPlot();

//mPlot.setBackgroundPaint(Color.PINK);

mPlot.setRangeGridlinePaint(Color.black);// 背景底部横虚线

mPlot.setOutlinePaint(Color.WHITE);// 边界线

ChartFrame mChartFrame = new ChartFrame("访问时间折线图", mChart);

mChartFrame.setLocation(390, 160);

mChartFrame.pack();

mChartFrame.setVisible(true);

}

### 3.2.2 Dataset容器（JFreeChart制图所需容器）

1.柱形图数据集

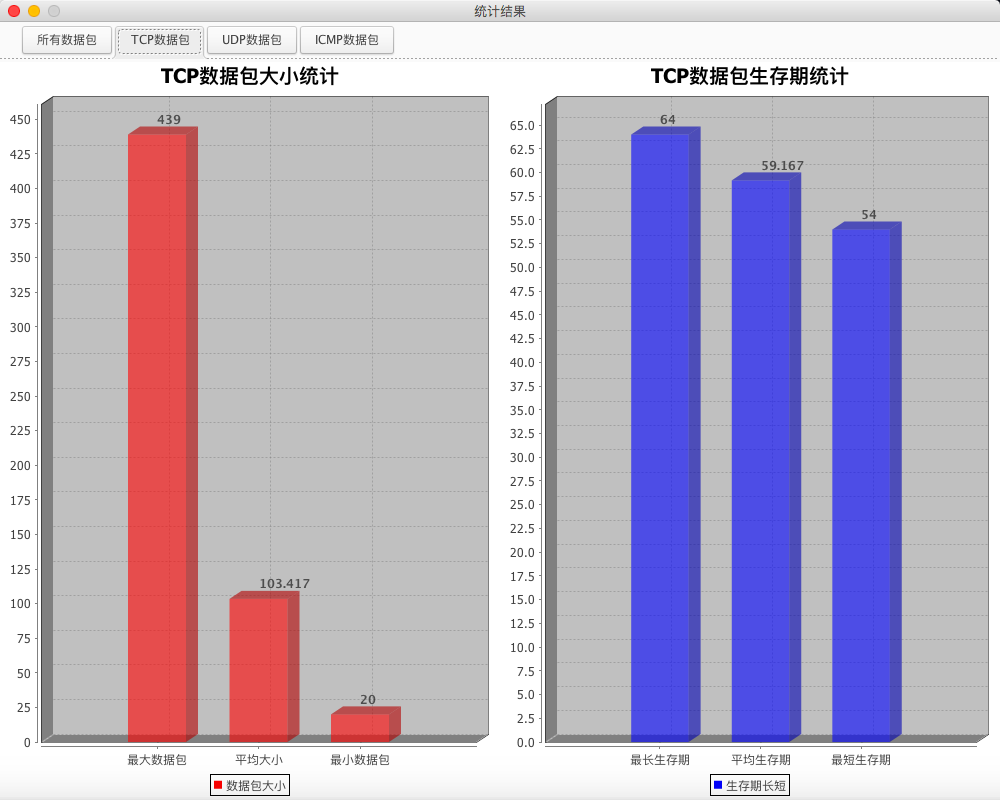
主要包括数据包大小统计的数据集和UDP数据包生存期统计的数据集，对于数据包大小和生存期，本程序统计最大值、平均值和最小值，对于这三项数据，采用柱形图绘制比较合适。

图3.8 柱形图

2.饼图数据集

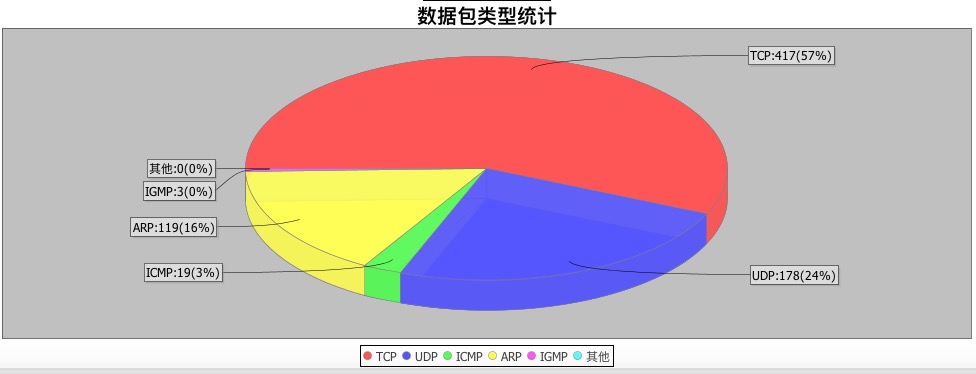
主要包括数据包版本的数据集和数据包协议类型的数据集，本系统绘制饼图直观的展现不同数据包版本和数据包协议的比例关系。

图3.9 饼图

### 3.2.3 界面设计与制作（JFrame继承子类）

主要包括登陆界面、主界面、数据包抓取处理界面、Ping界面、TraceRoute界面、抓取数据包（表格显示）、统计整理数据包（自动制作柱形图、饼图）、ARP攻击界面，详情见第五章。

设计界面主要使用Java Swing和Java AWT，采用框架Java BeautyEye，同时采用事件处理等接口完成图形界面设计，做到良好的美观性以及实用性、易用性。便于用户操作。

## 3.3 函数调用关系图

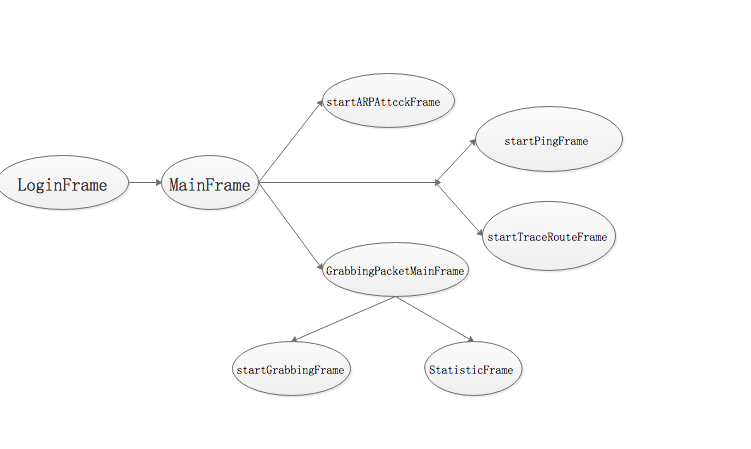


图3.3 函数调用关系图

程序由LoginFrame()进入，登陆成功进入MainFrame()，Mainframe可以选择进入GrabbingPacketMainFrame、startPingFrame、startTraceRouteFrame和startARPAttcckFrame，其中GrabbingPacketMainFrame调用startGrabbingFrame和StatisticFrame函数，StatisticFrame调用HandleDate、JFreeChart和JDBC函数完成统计并自动将数据可视化。

# 第四章 调试分析

## 4.1 调试中遇到的问题

### 4.1.1ARP攻击实现时遇到的问题

1.攻击有时会失效

如果攻击设备的IP不存在或不在同一个局域网内，应当通过程序判断并显示攻击失败，避免无效的攻击。

2.ARP代理

可能存在ARP代理情况，需要判断攻击的设备MAC地址是否真正存在，而不是网关代理的MAC地址。

3.网段大小

应通过此局域网的子网掩码判断局域网IP地址的范围，方便选择IP进行攻击。

### 4.1.2绘制表格中遇到的问题

1.表格显示

在未使用JTable前，我本来打算采用GridLayout解决表格显示问题，但尝试后发现效果不理想，差了JDK文档后发现有JTable类可以产生好看的表格，于是采用了JTable类。

2.在表格中添加按钮

最早设计的时候，一次性在表格显示的信息太多会导致样式很不好看，如果不是一次性显示的话，有没有很好的方法先是详细信息，因此我选择在表格中添加按钮以实现此功能。

然而添加按钮没有那么容易，最开始的时候直接将button添加到表格中，表格不能显示按钮，只显示了Java.Swing.Button的代码，后来查文献知道需要重构表格内容（见代码），重构之后可正常在表格中显示按钮，并可以绑定相应事件。

class TableRender extends DefaultTableCellRenderer {

public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus,

int row, int column) {

if (column != 6)

return super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row, column);

else {

JButton label = new JButton("选择");

label.setPreferredSize(new Dimension(table.getColumnModel().getColumn(column).getPreferredWidth(), 20));

return label;

}

}

}

class TableEditor extends DefaultCellEditor {

public TableEditor(JCheckBox checkBox) {

super(checkBox);

}

public TableEditor(JComboBox comboBox) {

super(comboBox);

}

public TableEditor(JTextField textField) {

super(textField);

}

public Component getTableCellEditorComponent(JTable table, Object value, boolean isSelected, int row,

int column) {

if (column != 6)

return super.getTableCellEditorComponent(table, value, isSelected, row, column);

else {

JButton label = new JButton("选择");

label.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String result = JDBC.searchByID(table.getValueAt(row, 0).toString());

if (result == null || result.equals(null)) {

} else if (result.equals("Fail")) {

return;

} else {

}

}

});

label.setPreferredSize(new Dimension(table.getColumnModel().getColumn(column).getPreferredWidth(), 20));

return label;

}

}

}

3.表格调整

如表格内容应该是不可以自己随意修改的、表格每一项大小等，这些都比较容易实现。

4．表格刷新

刚开始从表格进入编辑操作界面的时候，修改或删除之后，原表格不会更新，这样违背了同步性，因此我查询了一些资料了解了表格刷新的方案，主要是更新TableModel的内容，再对表格进行刷新即可，代码如下：

for(int i=tableModel.getRowCount()-1;i>=0;i--)

tableModel.removeRow(i);

for(int i=0;i<rows;i++)

tableModel.addRow(data[i]);

tableModel.fireTableDataChanged();

### 4.1.3数据可视化遇到的问题

1.柱形图

在柱形图遇到的问题主要有坐标上乱码、如何在柱子上方显示数字等问题，这些在一些博客上看到了解决方案得到了良好的解决。

2.饼图

饼图问题较少，主要是如何分配图形的空间，主要采用的是四张图一组的方式，这样现实能避免不同比例的缩放问题。

### 4.1.4其他问题

如异常的抛出和捕获等问题，主要采用的是Log生成日志追踪，得以解决

## 4.2 程序改进设想

1.ARP查询

应可以自动获取局域网内设备的IP地址，并可以通过IP地址和MAC地址的关系显示该设备类型。

2.表格操作

更为方便的表格操作应该是双击进入编辑界面，本系统采用的按钮操作可能不如双击便捷，可考虑今后完善。

3.界面背景

程序界面仍可以继续美化。

# 第五章 测试结果

## 5.1登陆界面

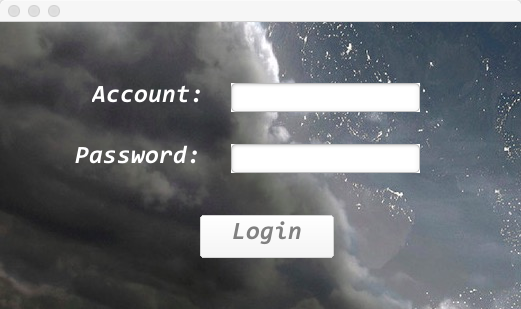


图5.1 登陆界面

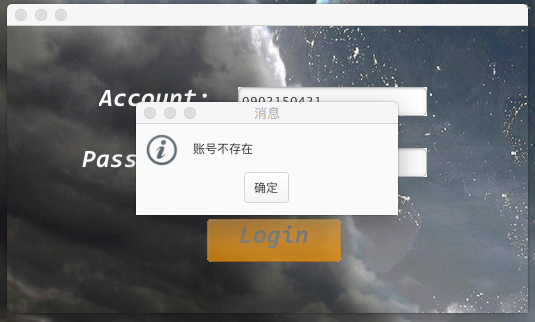


图5.2 账号错误

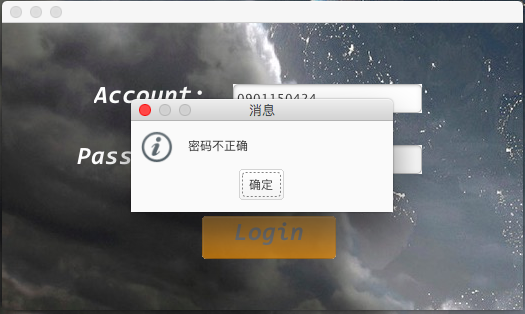


图5.3 密码错误

## 5.2程序主界面

如果登陆失败，如账号不存在或密码错误，则会给以相应提示（如图5.2，图5.3），登陆成功则跳转至程序主界面。



图5.4 程序主界面

其中，四个按钮分别对应四种操作：数据包处理、Ping、TraceRoute和ARPAttack。

## 5.3数据包处理界面

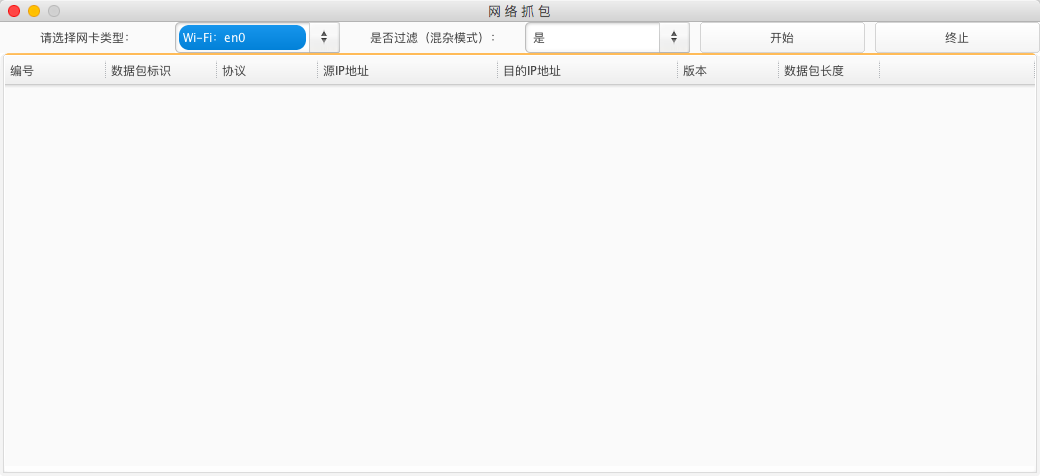


图5.5 数据包处理界面

其中，四个按钮分别对应四种操作：抓取数据包、统计数据包、数据包重置以及返回上层菜单。

## 5.4数据包抓取

点击数据包抓取按钮，显示出一个表格，首先需要选择网卡类型，即现在上网的方式是使用WiFi还是以太网；选择网卡模式后选定是否设置为混杂模式，即是否过滤掉不是发送给自己主机的数据包，如果不过滤的话意味着可以监听整个局域网内的数据包。点击开始自动抓取数据包，点击暂停停止数据包的捕获。

 图5.6 抓取数据包主界面

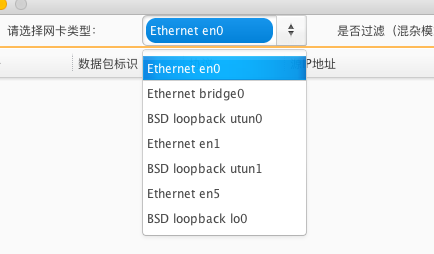


图5.7 本机网卡设备列表

图5.8 TCP数据包详情

在表格中只显示数据包标识、数据包协议类型、版本类型、源IP和目的IP地址以及数据包总长度,其他详细信息点击某一项的按钮即可查看：



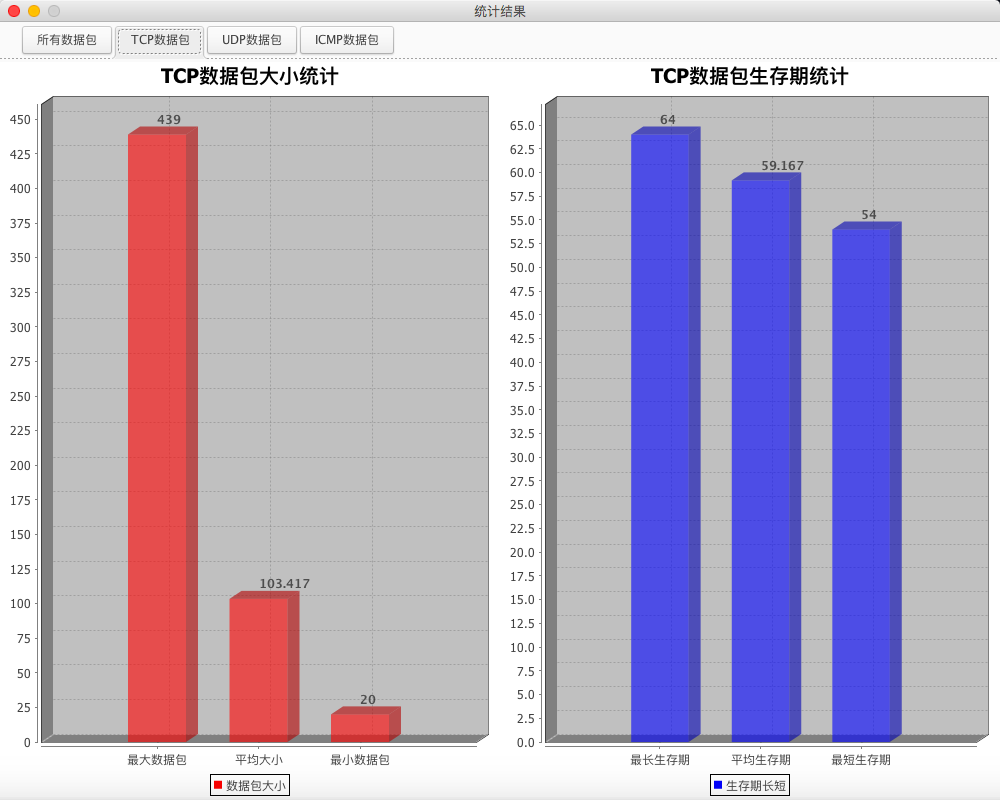
图5.9 TCP数据包详情

从图中可以看到这是一个HTTP数据包，可以看到包中的内容是一个HTTP报文。

## 5.5数据包整理与统计

在抓取数据包一段时间之后，可以对所有数据包进行分类统计整理，并将整理结果以图像化的方式直观显示，如：不同类型协议数据包的数量、不用版本（v4，v6）数据包的数量及类型、最大、平均、最小生存期与数据包大小等信息，并采用JFreeChart绘图。若想重新统计，可以返回上层界面选择“数据包重置”以清空这段时间补货到的数据包。具体显示如下：

 图5.10 数据包版本和类型统计

图5.11 TCP数据包统计

同理也可以统计UDP、ICMP数据包。

## 5.6 Ping操作

使用Java调用终端Ping指令，在表格上显示Ping的相关信息，并绘制Ping操作往返延迟折线图，从而实现图形化操作的Ping指令：

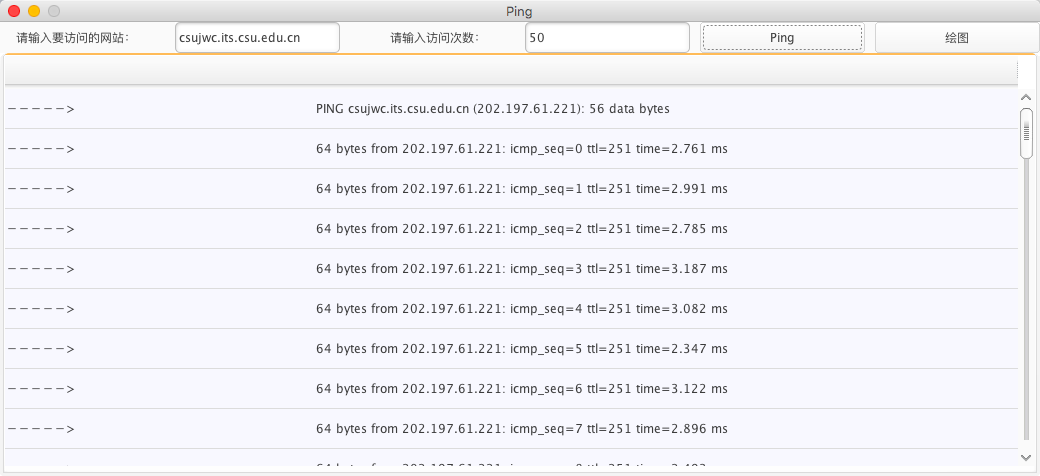


图5.12 Ping操作界面

在Ping操作结束后，会统计本次Ping操作的相关信息，如丢包率、最大RTT、均值RTT与最小RTT。

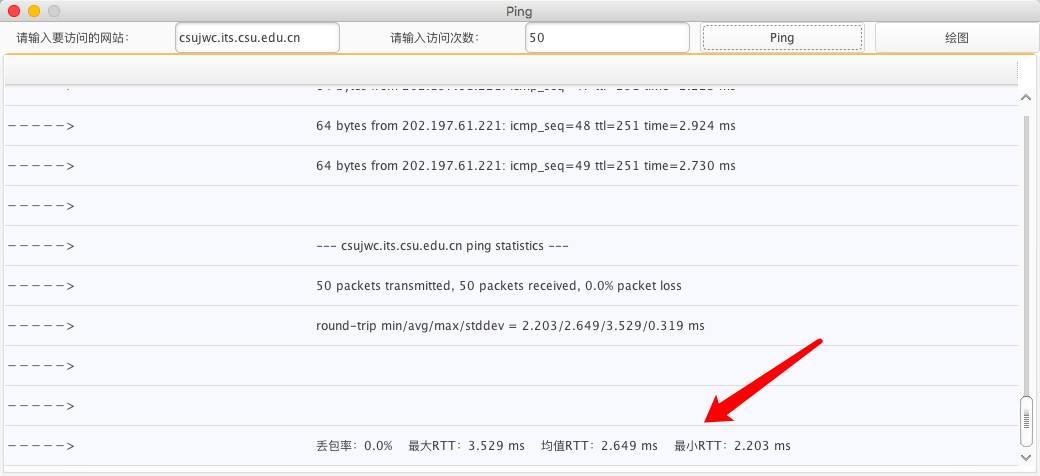


图5.13 Ping统计

点击绘制按钮可以绘制本次Ping操作的折线图：

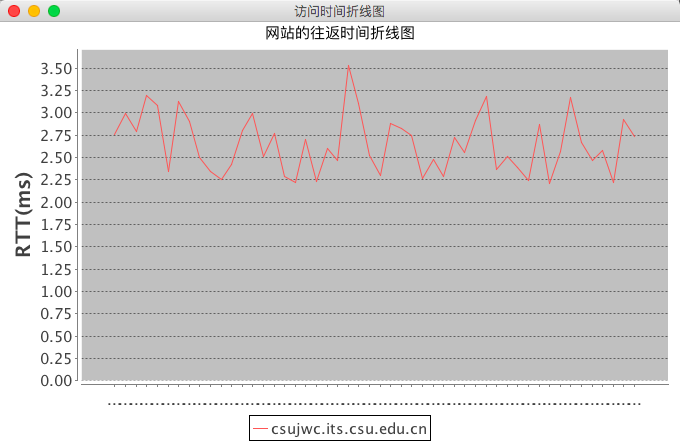


图5.14 Ping折线图

## 5.7 TraceRoute操作

使用Java调用终端traceroute 指令，在表格上显示traceroute 的相关信息，并绘制traceroute 操作经过的站点，从而实现图形化操作的traceroute 指令（显示\*为站点拒绝给以相应）：

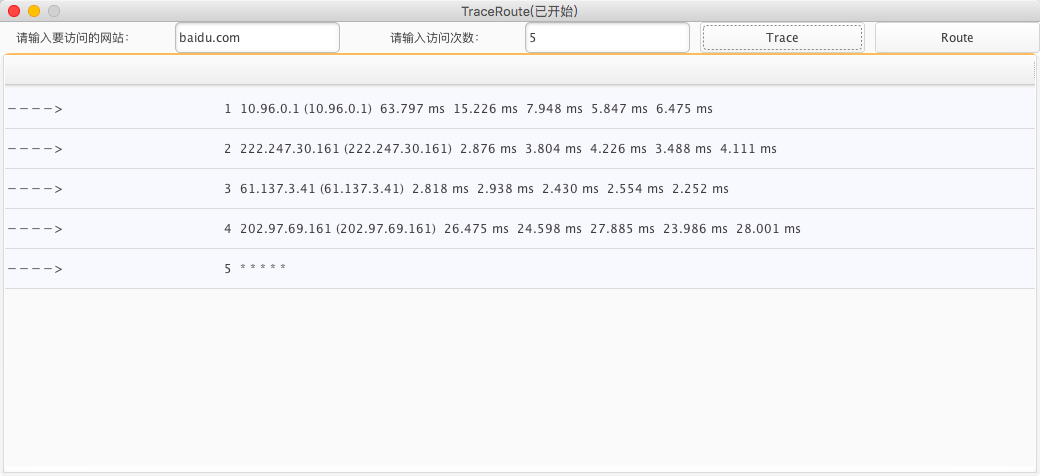


图5.15 TraceRoute显示（1）

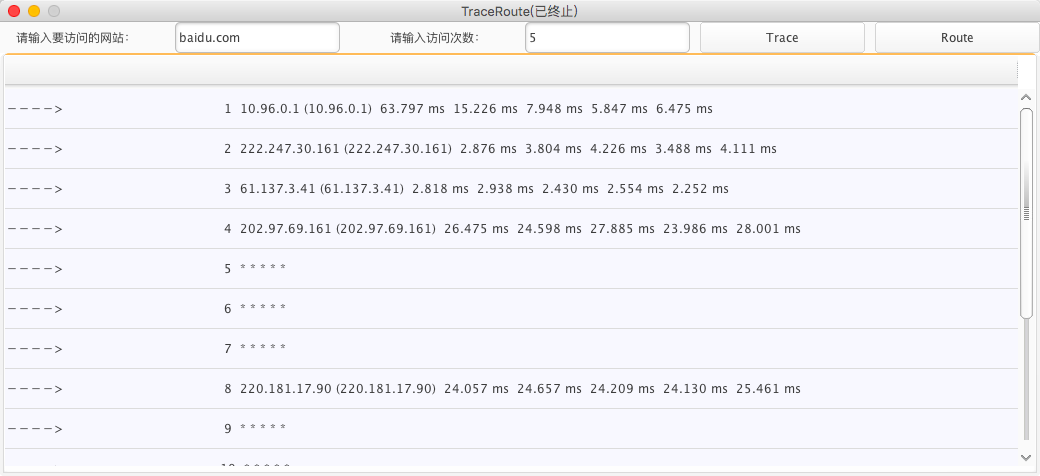


图5.16 TraceRoute显示（2）

在TraceRoute操作结束后，可以显示经过站点。



图5.17 经过的站点

## 5.8 ARP攻击操作

本程序可实现局域网内ARP攻击，经测试可以对指定IP地址的主机实现断网攻击，基本原理同上，是欺骗目标机器局域网网关地址，使得该机器ARP缓存表中记录的网关IP－MAC对应错误，从而无法把数据包发送给真正的网关，导致网络连接失败，如果将欺骗的MAC地址设置为自己的MAC地址，则会截获目标机器发送的数据包。



图5.18 ARP攻击界面

经测试，只要IP正确，就可以对设备实现断网攻击。由于ARP表过一定时间会自动更新，所以要隔一段时间就发送一个欺骗包。攻击停止后一段时间设备会自动恢复网络畅通。

程序打开时可以自动获取本机IP地址与MAC地址、网关IP地址与MAC地址、局域网广播地址以及局域网内其他设备IP地址与MAC地址：



图5.19 ARP攻击准备

下面以攻击手机为例（与电脑在同一个局域网中），展示ARP攻击与恢复：



图5.20 ARP攻击开始

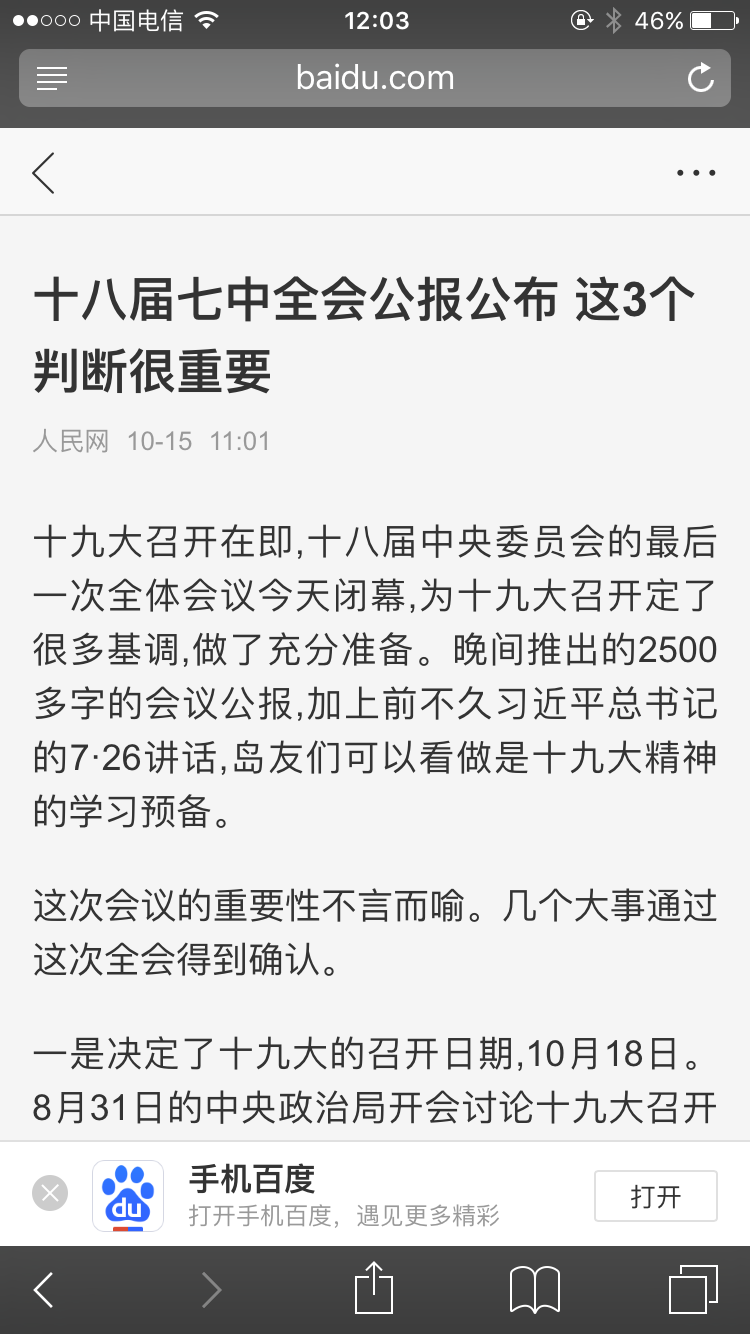


图5.21 攻击前（图1）攻击中（图2）停止攻击一段时间后恢复（图3）

把目的MAC设置为本机MAC，可以捕获靶机数据包：



图5.22 中间人截获数据包

# 参考文献

[1] 郭克华：《用Java语言描述面向对象》2011.11；

[2] Bruce Eckel：《Thinking in Java》2006.2

[3] Andrew S.Tanenbaum,David J.Wetherall：《计算机网络》2012.3