# 第11讲 网络抓包与发包程序设计(下)

[需要2个教学单位完成，此为第2个单元]

## 一、完善抓包程序

上一讲的抓包程序显示的是抓包的原始信息，我们可以完善程序，当TCP包中的数据包含特定的关键词，就将TCP包中的数据部分显示出来，对于那些明文传输的数据，能进行更好的捕获分析。

**（1）完善设置对话框ConfigDialog.java**

如图11.1所示，增加设置数据关键字的功能，希望捕获的多个关键字用空格隔开，多个关键字建议使用“或”的关系。



图11.1 设置对话框

在ConfigDialog.java中，增加一个getKeyData()方法，用于给主程序PacketCaptureFX返回用户输入的关键字信息。

**（2）修改主程序PacketCaptureFX中的内部类PacketHandler**，使得当TCP包中的数据部分包含关键字，则将该包中的数据部分在显示区显示，如何获取包中数据，见上一讲的Jpcap的基本用法部分，部分核心代码如下：

class PacketHandler implements PacketReceiver {  
 @Override  
 public void receivePacket(Packet packet) {  
 ...... *//省略*

*//分析包中是否包含需要显示的数据* if (keyData == null || keyData.trim().equalsIgnoreCase(""))  
 return;  
 try {

*//将keyData按空格切分出包含多个关键词的字符串数组*

*//提示，split方法也可以使用正则表达式，*[*\s+*](file:///\\s+)*表示匹配一个或多个空白* String[] keyList = ......;  
 String msg = new String(packet.data, 0, packet.data.length, "utf-8");  
 for (String key : keyList) {  
 if (msg.toUpperCase().contains(key.toUpperCase())) {  
 Platform.*runLater*(() -> {  
 taDisplay.appendText("数据部分：" + msg + "\n\n");  
 });  
  
 break;  
 }  
 }  
  
 } catch (Exception e) {  
 System.*err*.println(e.getMessage());  
 }  
 }  
}

大家可以A、B两人一组，A首先打开抓包程序，然后A运行第5讲的TCPThreadServer服务器，然后B通过第3节的TCPClientThreadFX访问B的服务器，看看A是否能捕获指定的数据部分内容。

## 二、使用Jpcap发包

Java自带的类库对TCP/IP协议做了很好的封装，通过不同的Socket来处理TCP、UDP协议。但Java标准库并不能处理原始套接字(raw Socket，可以用来自行组装IP数据包，然后将数据包发送到其他终端。即允许直接发送/接收IP协议数据包而不需要任何传输层协议格式)。有时需要发送一个或多个IP包、TCP包等特定的构造包，以实现特定的目的，如网络安全检测、隐蔽扫描、网络泛洪攻击等。标准的Java套接字就无能为力。这时，就可以使用Jpcap开发包中提供的JpcapSender类。其发送TCP包的顺序大致为构造TCP包、设置其IP报头、填充TCP数据、构造以太网帧、发送包。构造包和发送包的详细过程如下：

### 1. 获取JpcapSender对象实例

先按照上一讲的知识，获取网络接口参数，然后选择有效网卡接口获得JpcapSender对象实例：

*//open a network interface to send a packet*NetworkInterface[] devices = JpcapCaptor.*getDeviceList*();  
JpcapSender sender=JpcapSender.*openDevice*(devices[0]);

*//或者用* *jpcapCaptor.getJpcapSenderInstance();获得对象实例*

### 2. 构造TCP包

TCP包的构造方法为：

public TCPPacket(int src\_port,int dst\_port,long sequence,long ack\_num, boolean urg,boolean ack,boolean psh,boolean rst,  
 boolean syn,boolean fin,boolean rsv1,boolean rsv2,  
 int window,int urgent)

构造方法中的参数实际就是TCP报头的参数。我们就可以按实际需求构造TCP包，例如：

*//create a TCP packet with specified port numbers, flags, and other parameters*

TCPPacket tcp=new TCPPacket (8000,80,56,78,false,false,false,false,true,false,true,true,200,10);

### 3. 设置IPv4头参数

使用setIPv4Parameter方法来设置Ipv4报头参数，该方法的方法头定义如下：

public void setIPv4Parameter(int priority, boolean d\_flag, boolean t\_flag, boolean r\_flag, int rsv\_tos, boolean rsv\_frag, boolean dont\_frag, boolean more\_frag, int offset, int ident, int ttl, int protocol, InetAddress src, InetAddress dst)

例如，我们就可以这样设置参数：

*//specify IPv4 header parameters*

tcp.setIPv4Parameter(0,false,false,false,0,false,false,false,0,1010101,100, IPPacket.IPPROTO\_TCP, InetAddress.getByName(源IP地址),  
 InetAddress.getByName (目的IP地址));

### 4. 填充TCP中的数据

tcp.data= "填充的数据".getBytes("utf-8");*//字节数组型的填充数据*

### 5. 构造以太网数据包（帧）

*//create an Ethernet packet (frame)*EthernetPacket ether=new EthernetPacket();

*//set frame type as IP*ether.frametype=EthernetPacket.*ETHERTYPE\_IP*;

*//set the datalink frame of the tcp packet as ether*tcp.datalink=ether;

*//set source and destination MAC addresses*

*//MAC地址要转换成十进制，ipconfig /all 查看本机的MAC地址  
//源地址是自己机器的MAC地址*

ether.src\_mac=new byte[]{(byte)00,(byte)27,(byte)185,(byte)177,(byte)74,(byte)70};

*//根据实际情况设置目的MAC地址， arp -a 可以查看相关的MAC地址*ether.dst\_mac=new byte[]{(byte)00,(byte)17,(byte)93,(byte)157,(byte)128,(byte)00};

### 6. 发送特定的TCP包

sender.sendPacket(tcp);

...... *//省略*

sender.close();

## 三、程序设计第一步：创建网卡选择对话框

如果没有网卡选择对话框，而是将选择的网卡接口写死在程序中，在不同机器上运行就非常不方便，而且容易忘记修改指定的网卡接口。

（1）在已有项目中新建包chapter11，然后按照上一讲创建网卡参数设置对话框的方法，创建网卡选择对话框NetworkChoiceDialog.java（可以拷贝上一讲的对话框程序，保留网卡选择列表，删除掉不需要的部分），后面的程序均创建在chapter11包中；

（2）在NetworkChoiceDialog类中提供一个getSender()方法，返回给主程序JpcapSender对象实例。

对话框参考界面见图11.2。

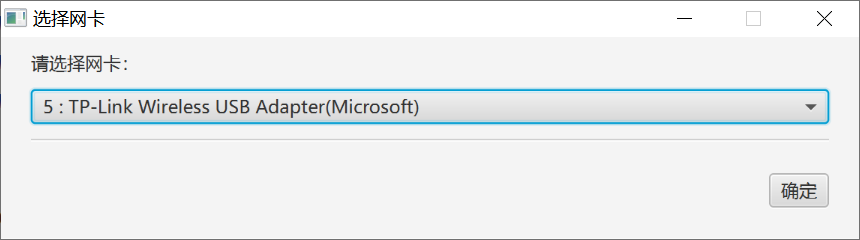


图11.2 网卡选择对话框

## 四、程序设计第二步：将发包操作封装为PacketSender类

发包的操作可以用在控制台程序中，也可以用在其他GUI界面的程序中。为了便于重用，将相关的发包操作封装为一个普通的类PacketSender.java，具体功能实现参看“[二、使用Jpcap发包](#_二、使用Jpcap发包)”中相应内容。

在PacketSender类中，提供一个静态方法sendTCPPacket用于发送报文，另外由于JpcapSender中使用的MAC地址是字节数组，每次转换会很麻烦，所以在该类中提供一个静态方法convertMacFormat，用于将常见的MAC地址形式转为Jpcap可用的字节数组。该类的框架代码如下：

*/\*\*  
 \* 封装jpcap发包功能  
 \*/*public class PacketSender {  
  
 */\*\*  
 \* @param sender JpcapSender类型  
 \* @param srcPort 源端口  
 \* @param dstPort 目的端口  
 \* @param srcHost ip地址形式或类似 www.baidu.com的域名形式  
 \* @param dstHost  
 \* @param data 填充到tcp包中的数据  
 \* @param srcMAC 格式为"dc-8b-28-87-b9-82"或"dc:8b:28:87:b9:82"  
 \* @param dstMAC  
 \* @param syn 这几个为常用标识位  
 \* @param ack  
 \* @param rst  
 \* @param fin  
 \*/* public static void sendTCPPacket(JpcapSender sender, int srcPort,   
 int dstPort,String srcHost, String dstHost, String data,  
 String srcMAC, String dstMAC,  
 boolean syn, boolean ack, boolean rst, boolean fin) {

try {  
 *//构造一个TCP包*  ......  
 *//设置IPv4报头参数，ip地址可以伪造*  ......  
  
 *//填充TCP包中的数据*  ......  
 *//构造相应的MAC帧* ......  
 ether.src\_mac = *convertMacFormat*(srcMAC);  
 ether.dst\_mac = *convertMacFormat*(dstMAC);  
 if(ether.src\_mac == null || ether.dst\_mac==null)  
 throw new Exception("MAC地址输入错误");  
  
 sender.sendPacket(tcp);  
 System.*out*.println("发包成功！");  
 } catch (Exception e) {  
 System.*err*.println(e.getMessage());  
 *//重新抛出异常，调用者可以捕获处理* throw new RuntimeException(e);  
 }  
 }

public static byte[] convertMacFormat(String MAC) {

*//（1）首先判断参数MAC中是否包含"-"或":";*

*//(2)通过split方法将MAC切分为字符串数组;*

*//(3)定义一个6字节的字节数组，循环将十六进制形式的字符串转为字节，赋值给字节数组;*

*//提示：通过(byte)Integer.parseInt("0F",16)方式将16进制表示的字符串转为字节*  
 }

}

## 五、程序设计第三步：创建主界面

（1）新建JavaFX程序SendPacketFX.java，参考界面如11.3所示。程序界面中有我们之前没用过的控件-复选框，复选框可以按如下方式定义为成员变量：

private CheckBox cbSYN = new CheckBox("SYN");  
private CheckBox cbACK = new CheckBox("ACK");  
private CheckBox cbRST = new CheckBox("RST");  
private CheckBox cbFIN = new CheckBox("FIN");

CheckBox可以通过setSelected方法来设置是否勾选，通过isSelected方法来判断是否被选中。

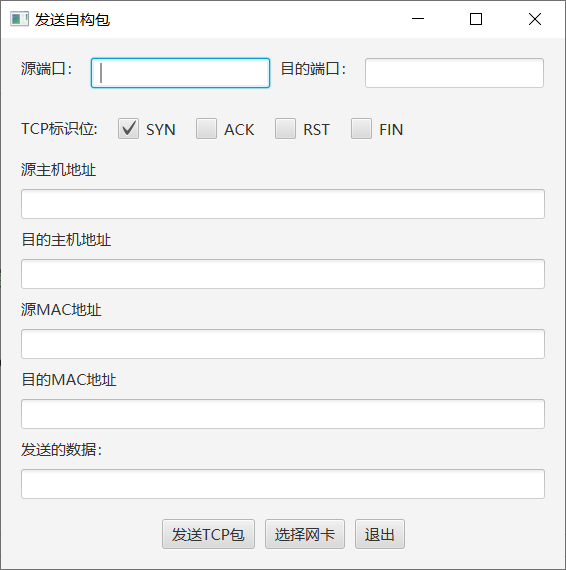


图11.3 发包程序主界面

建议程序一运行就先显示网卡选择对话框，选择后再显示主界面，部分参考代码如下：

......

primaryStage.setScene(scene);  
primaryStage.setTitle("发送自构包");  
primaryStage.setWidth(500);  
dialog = new NetworkChoiceDialog(primaryStage);  
dialog.showAndWait();  
sender = dialog.getSender();  
primaryStage.show();

......

（2）“发送TCP包”按钮动作事件响应：

btnSendTCPPacket.setOnAction(event -> {  
 try {  
 int srcPort = Integer.*parseInt*(tfSrcPort.getText().trim());  
 int dstPort = Integer.*parseInt*((tfDstPort.getText().trim()));  
 String srcHost = tfSrcHost.getText().trim();  
 String dstHost = tfDstHost.getText().trim();  
 String srcMAC = tfSrcMAC.getText().trim();  
 String dstMAC = tfDstMAC.getText().trim();  
 String data = tfData.getText();  
 *//调用发包方法*  
 PacketSender.*sendTCPPacket*(sender, srcPort, dstPort, srcHost,  
 dstHost, data, srcMAC, dstMAC,cbSYN.isSelected(),  
 cbACK.isSelected(),cbRST.isSelected(),cbFIN.isSelected());

new Alert(Alert.AlertType.*INFORMATION*, "已发送！").showAndWait();  
 } catch (Exception e) {  
 new Alert(Alert.AlertType.*ERROR*, e.getMessage()).showAndWait();  
 }  
});

其他按钮动作响应代码自行完成。

## 六、课堂计分

同学之间先互相练习发包、抓包是否成功（同学局域网之间直接发包、以及给服务器发包，目的MAC的选择，要理解清楚，因为服务器是在其它的网段）。程序验证成功后，进入计分过程：

使用SendPacketFX向服务器 202.116.195.71：8008发送自己封装的TCP包，包的数据部分内容为自己的学号&姓名，例如20181111111&程旭元，要求你的发包端口必须设置为：8000。

用本讲中完善版本的抓包程序查看是否有返回成功信息。（提示：服务器返回信息总是包含"服务器"，所以在包中数据查找的关键字中设置"服务器"）

## 七、扩展练习一：隐蔽扫描

第9讲介绍过端口扫描技术，本质上是与指定端口通过三次握手建立TCP连接，如果连接成功，则说明端口开放，但这种方式会在系统日志上留下大量记录。

如果使用本讲的发包技术，则没有必要建立完整的三次握手过程，而是向指定端口发包实现第一次握手（SYN=1），使用抓包程序，看是否能捕获服务器返回的信息，如果有返回信息，判断是否为第二次握手过程。如果符合二次握手特征，可以判断端口开放。这样没有建立完整TCP连接，服务器上留下的痕迹就更少。（这种方式准确性不够标准扫描高）

## 八、扩展练习二：ARP欺骗

### 1. ARP协议与ARP欺骗

ARP欺骗又被称为ARP重定向技术，ARP地址作为一个局域网的协议，它是建立在各主机之间互相信任基础之上的，因此存在一定的安全问题：

（1）主机地址映射表是基于高速缓存动态更新的，这是ARP协议的特色，也是安全问题之一。由于正常的主机间MAC地址刷新都是有时限的，这样假冒者如果在下次更新之前成功的修改了被攻击机器上的地址缓存，就可以进行假冒；

（2）ARP请求以广播方式进行。这个问题是不可避免的，因为正是由于主机不知道通信对方的MAC地址，才需要进行ARP广播请求。这样攻击者就可以伪装ARP应答，与广播者真正要通信的机器进行竞争。还可以确定子网内机器什么时候会刷新MAC地址缓存，以确定最大时间限度的进行假冒；

（3）可以随意发送ARP应答包。由于ARP协议是无状态的，任何主机即使在没有请求的时候也可以做出应答，只要应答有效，接收到应答包的主机就无条件的根据应答包的内容更新本机高速缓存；

（4）ARP应答无需认证。由于ARP协议是一个局域网协议，设计之初，出于传输效率的考虑，在数据链路层就没有作安全上的防范。在使用ARP协议交换MAC地址时无需认证，只要收到来自局城网内的 ARP应答包，就将其中的MAC<->IP映射刷新到本主机的高速缓存中。

### 2. ARP欺骗方案

假设攻击者A和被监控目标B在同一个局域网，A尝试拦截或监控B访问外部网络的数据包：

（1）首先伪造一个ARP REPLY包，数据链路层头及ARP内容部分的源MAC地址填入A的MAC地址，而源IP部分填入网关IP，目的地址填入B的MAC、IP，然后将这个包发送给B，而B接收到这个伪造的ARP REPLY包后，由于源IP为网关IP，于是在它的ARP缓存表里刷新了一项，将（网关IP，网关MAC）刷新成（网关IP，A的MAC）。而B要访问外部的网都需要经过网关，这时候这些要经过网关的包就流到A的机器上来了；

（2）接着，再伪造一个ARP REPLY包，数据链路层头及ARP内容部分的源MAC地址填入A的MAC地址，而源IP部分填入B的IP，目的地址填入网关MAC、IP，然后将这个包发给网关，网关接收到这个伪造的ARP REPLY包后，由于源IP为B的IP，于是在它的ARP缓存表里刷新了一项，将（B的IP，B的MAC）刷新成（B的IP，A的MAC）。这时候外部传给B的数据包经过网关时，就通通转发给A；

（3）完成上面两个步骤，已经可以拦截B的数据包——这时B已经无法上网，如果只是监控，A可以作为中间人，将进出的数据包目的地址部分修改，其它部分原封不动进行转发。

### 3. 用Jpcap实现拦截监听

就如上面说的，为了实现监听，我们必须做四件事：

（1） 发送ARP包修改B的ARP缓存表；

（2）发送ARP包修改网关ARP缓存表；

（3） 转发B发过来的数据包；

（4） 转发网关发过来的数据包

参考代码可见[ARPTrap.java](https://oc.our-web.org/download?file=ARPTrap.java)。