# 网络数据包抓取与分析程序

## 需求分析

* 1. **网卡选择模块**

功能：获取系统中的网卡列表，并允许用户选择其中一个网卡进行数据包抓取。

要求：显示系统中可用的网卡列表。允许用户选择一个网卡作为数据包抓取的目标。

* 1. **数据包抓取模块**

功能：从用户选择的网卡中实时抓取网络数据包。

要求：实时监听选定网卡上的网络流量。抓取数据包时应考虑高效性和准确。支持设置抓取数据包的过滤条件，以便捕获特定协议或源/目标地址等。

* 1. **TCP数据包分析模块**

功能：对抓取到的网络数据包进行解析和分析。如果是TCP数据包则读取信息并分析

要求：能够识别数据包所使用的协议，如IP或TCP。对于每个协议，解析其各个字段的含义和值，并进行简要说明。解析TCP数据包的头部字段，如源端口、目标端口、序列号、确认号等。分析TCP连接的状态、传输控制信息等。

## 概要设计

* 1. **算法设计说明**

计算TCP包中的校验和：首先根据解析出的包中的信息构建TCP伪首部，伪首部包括源IP地址（4字节），目标IP地址（4字节），保留（1 字节），协议（1字节），TCP 长度（2 字节），保留字节默认为空设为0，协议设为6，TCP长度用解析包中的首部长度和数据长度可得，将TCP首部，伪首部和数据部分视为一系列16位的字（如果数据部分的长度不是16位的整数倍，则需要在末尾填充0），然后将它们两两相加。如果相加的结果超过了16位，则将溢出部分回卷加到低位。对结果取反，即得到校验和。

数据偏移计算：将TCP的头部长度除以4可得。

* 1. **存储结构设计说明**

网络接口信息存储：包含以下成员变量表示网卡的名称的name。表示网卡的描述的description。一个布尔值，表示网卡是否是回环接口的loopback（loopback interface）。表示网卡所使用的数据链路层协议的名称的datalink\_name。表示网卡所使用的数据链路层协议的描述的datalink\_description。表示网卡的物理地址（MAC地址）的mac\_address。表示网卡的地址信息数组的addresses。

抓包信息存储：含有三个类Packet类，IPPacket类，TCPPacket类，从后往前依次继承实现，每个类存储该层包的数据信息，Packet类属性包含时间戳（秒和微秒部分），捕获的数据包的长度，数据包的总长度，数据链路层数据包，数据包的头部，数据包的数据部分，序列化版本号。IPPacket类属性包含IP协议的版本号，IP包的优先级，IP包的“D”标志（不分片），IP包的“T”标志（DF/TOS override），IP包的“R”标志（保留位），保留字段（TOS），IP包的长度，IP包的“Don't Fragment”标志，IP包的“More Fragments”标志，片偏移，IP包的生存时间（跳数限制），IP包的协议，IP包的标识，IP包的流标签，源IP地址，目标IP地址，IP选项，IP选项列表。TCPPacket类属性包含TCP包的源端口，TCP包的目标端口，TCP包的序列号，TCP包的确认号，TCP包的URG标志，TCP包的ACK标志，TCP包的PSH标志，TCP包的RST标志，TCP包的SYN标志，TCP包的FIN标志，保留字段，TCP窗口大小，紧急指针，TCP选项。

## 详细设计

* 1. **UI模块**

采用JFram实现界面设计，代码如下

1. public class UI {
2. private JFrame frame;
3. private JPanel mainPanel;
4. private JButton selectButton;
5. private JButton captureButton;
6. private JLabel selectedInterfaceLabel;
7. private NetworkInterface[] devices = JpcapCaptor.getDeviceList();
8. private void initialize() {
9. frame = new JFrame("TCP数据包分析器");
10. frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
11. frame.setSize(1500, 800);
12. mainPanel = new JPanel();
13. mainPanel.setLayout(new BoxLayout(mainPanel, BoxLayout.Y\_AXIS));
14. frame.getContentPane().add(mainPanel);
15. selectButton = new JButton("选择网卡");
16. selectButton.addActionListener(e -> selectNetworkInterface());
17. mainPanel.add(selectButton);
18. selectedInterfaceLabel = new JLabel();
19. mainPanel.add(selectedInterfaceLabel);
20. captureButton = new JButton("开始捕获");
21. captureButton.addActionListener(e -> startCapture());
22. mainPanel.add(captureButton);
23. frame.setLocationRelativeTo(null);
24. frame.setVisible(true);
25. }
26. private void selectNetworkInterface() {
27. *// 获取网卡列表*
28. String[] interfaceNames = new String[devices.length];
29. for (int i = 0; i < devices.length; i++) {
30. interfaceNames[i] = devices[i].name;
31. }
32. String selectedInterface = (String) JOptionPane.showInputDialog(frame,
33. "请选择一个网卡：", "选择网卡", JOptionPane.PLAIN\_MESSAGE, null, interfaceNames, interfaceNames[0]);
34. if (selectedInterface != null) {
35. for (int i = 0; i < interfaceNames.length; i++) {
36. if (interfaceNames[i].equals(selectedInterface)) {
37. Main.Index = i;
38. break;
39. }
40. }
41. NetworkInterfaceAddress[] addresses = devices[Main.Index].addresses;
42. selectedInterfaceLabel.setText("已选择的网卡： " + addresses[1].address.toString() + "\n");
43. }
44. }
45. private void startCapture() {
46. try {
47. int i = 1;
48. while (true) {
49. JpcapCaptor captor = JpcapCaptor.openDevice(devices[1], 65535, true, 20);
50. Packet packet = captor.getPacket();
51. if (packet instanceof TCPPacket) {
52. TCPPacket tcpPacket = (TCPPacket) packet;
53. *// Your packet analysis code here*
54. break;
55. }
56. }
57. } catch (Exception exception) {
58. exception.printStackTrace();
59. }
60. }
61. }

添加两个按钮，并对按钮添加监听事件，选择网卡事件执行和开始抓包事件执行

* 1. **网卡选择模块**

调用NetworkInterface类获取本机所用可用网卡，并初始化一个网卡名字的数组，展示在UI界面的下拉列表里，用户选择后，获取选择网卡所在数组的下标，展示出该网卡的ip地址，然后将这个网卡传给下一个抓包模块，让其开始抓包。以下为代码

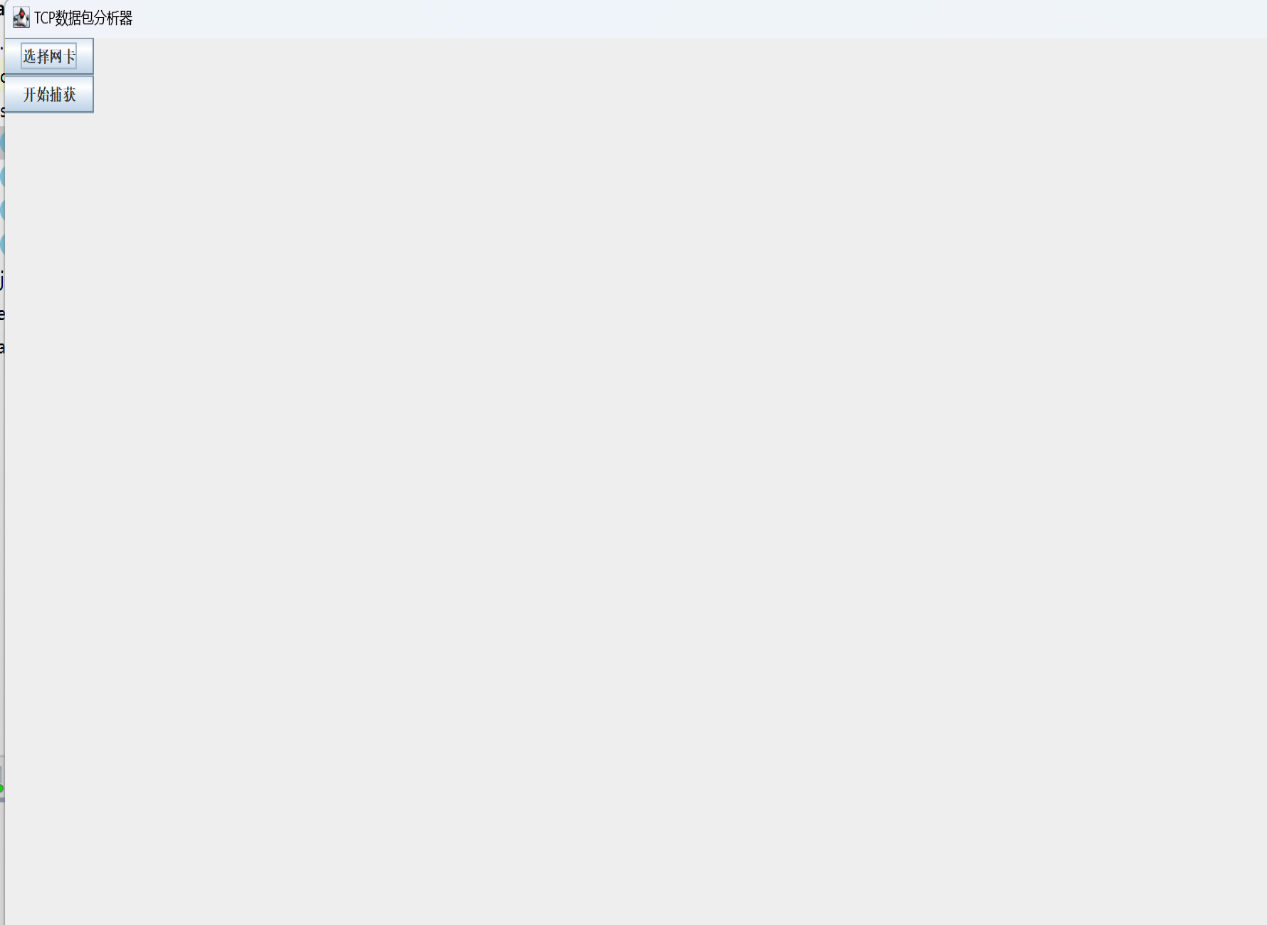
1. *//监听事件*
2. jButton.addActionListener(new ActionListener() {
3. @Override
4. public void actionPerformed(ActionEvent e) {
5. *// 弹出选择网卡对话框*
6. String[] interfaceNames = new String[devices.length];
7. for (int i = 0; i < devices.length; i++) {
8. interfaceNames[i] = devices[i].name;
9. }
10. String selectedInterface = (String) JOptionPane.showInputDialog(frame,
11. "请选择一个网卡：", "选择网卡", JOptionPane.PLAIN\_MESSAGE, null, interfaceNames, interfaceNames[0]);
12. *// 如果用户选择了网卡，则更新按钮下方的标签*
13. if (selectedInterface != null) {
14. *// 寻找选择的网卡在interfaceNames数组中的下标*
15. for (int i = 0; i < interfaceNames.length; i++) {
16. if (interfaceNames[i].equals(selectedInterface)) {
17. Index = i;
18. break;
19. }
20. }
21. *//显示选择的网卡名*
22. NetworkInterfaceAddress[] addresses = devices[Index].addresses;
23. selectedInterfaceLabel.setText("已选择的网卡： "+addresses[1].address.toString()+"\n");
24. }
25. }
26. });
    1. **TCP包解析模块**

设定为Action类，下为代码

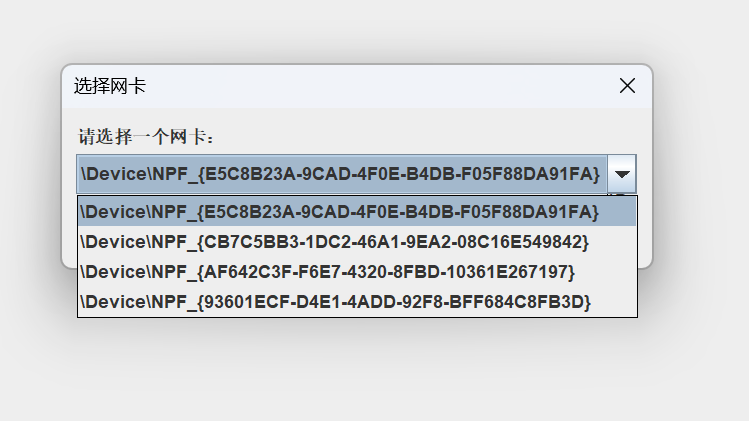
1. import jpcap.packet.TCPPacket;
2. import java.util.zip.CRC32;
3. public class Action {
4. public static String showurgent(short urgent){
5. if (urgent == 0){
6. return "无紧急数据";
7. }else{
8. return "有"+urgent+"字节的紧急数据";
9. }
10. }
11. public static String panProtorl(short protocol){
12. switch (protocol){
13. case 1:
14. return "ICMP协议";
15. case 2:
16. return "IGMP协议";
17. case 4:
18. return "IP协议";
19. case 6:
20. return "TCP协议";
21. case 17:
22. return "UDP协议";
23. default:
24. return "其他";
25. }
26. }
27. *//在控制台展示信息*
28. public static void showInConsole(TCPPacket tcpPacket) {
29. System.out.println("Source Port: " + tcpPacket.src\_port);
30. System.out.println("Destination Port: " + tcpPacket.dst\_port);
31. System.out.println("Sequence Number: " + tcpPacket.sequence);
32. System.out.println("Acknowledgment Number: " + tcpPacket.ack\_num);
33. System.out.println("Data Offset: " + tcpPacket.header.length);
34. if(tcpPacket.urg){
35. System.out.println("URG Flag: 该报文有紧急数据" );
36. }
37. if(tcpPacket.ack){
38. System.out.println("ACK Flag: 确认字段有效" );
39. }else{
40. System.out.println("ACK Flag: 确认字段无效" );
41. }
42. if(tcpPacket.psh){
43. System.out.println("PSH Flag: 立即推送" );
44. }
45. if (tcpPacket.rst){
46. System.out.println("RST Flag: 重置连接" );
47. }
48. if (tcpPacket.syn){
49. System.out.println("SYN Flag: 同步连接" );
50. }
51. if (tcpPacket.fin){
52. System.out.println("FIN Flag: 结束连接" );
53. }
54. System.out.println("Window Size: " + tcpPacket.window);
55. System.out.println("Urgent Pointer: " + tcpPacket.urgent\_pointer);
56. System.out.println("Options: " + tcpPacket.option);
57. System.out.println("Protocol: " + tcpPacket.protocol);
58. System.out.println("Data: " + new String(tcpPacket.data));
59. System.out.println("------------");
60. }
61. public static int calculateDataOffset(TCPPacket tcpPacket) {
62. *// TCP头长度存储在第12字节的前4位*
63. *// TCP 标头长度字段以 4 字节为单位，因此乘以 4 即可得到字节*
64. return (tcpPacket.header[12] & 0xF0) >> 4;
65. }
66. *// 计算 TCP 校验和的函数*
67. public static long calculateChecksum(TCPPacket tcpPacket) {
68. *// 初始化校验和*
69. CRC32 crc = new CRC32();
70. *// 准备计算校验和的数据：TCP伪头+TCP头+TCP数据*
71. byte[] pseudoHeader = preparePseudoHeader(tcpPacket);
72. byte[] tcpHeader = tcpPacket.header;
73. byte[] tcpData = tcpPacket.data;
74. *// 将伪标头、TCP 标头和 TCP 数据合并到单个字节数组中*
75. byte[] combinedData = new byte[pseudoHeader.length + tcpHeader.length + tcpData.length];
76. System.arraycopy(pseudoHeader, 0, combinedData, 0, pseudoHeader.length);
77. System.arraycopy(tcpHeader, 0, combinedData, pseudoHeader.length, tcpHeader.length);
78. System.arraycopy(tcpData, 0, combinedData, pseudoHeader.length + tcpHeader.length, tcpData.length);
79. *// 使用组合数据更新 CRC32*
80. crc.update(combinedData);
81. *// 获取计算出的校验和*
82. return crc.getValue();
83. }
84. private static byte[] preparePseudoHeader(TCPPacket tcpPacket) {
85. *// 伪标头包括：*
86. *// 源IP地址（4字节）*
87. *// 目标IP地址（4字节）*
88. *// 保留（1 字节）*
89. *// 协议（1字节）*
90. *// TCP 长度（2 字节）*
91. byte[] pseudoHeader = new byte[12];
92. *// 获取源IP地址和目的IP地址*
93. byte[] srcAddress = tcpPacket.src\_ip.getAddress();
94. byte[] dstAddress = tcpPacket.dst\_ip.getAddress();
95. *// 将源和目标 IP 地址复制到伪标头中*
96. System.arraycopy(srcAddress, 0, pseudoHeader, 0, 4);
97. System.arraycopy(dstAddress, 0, pseudoHeader, 4, 4);
98. *// 将保留字段设置为 0*
99. pseudoHeader[8] = 0;
100. *// 将协议字段设置为 6 (TCP)*
101. pseudoHeader[9] = 6;
102. *// 获取TCP长度（报头长度+数据长度）*
103. int tcpLength = tcpPacket.header.length + tcpPacket.data.length;
104. *// 设置伪标头中的 TCP 长度*
105. pseudoHeader[10] = (byte) ((tcpLength >> 8) & 0xFF);
106. pseudoHeader[11] = (byte) (tcpLength & 0xFF);
107. return pseudoHeader;
108. }
109. }

该类中，含有对TCP包的抓取方法，并将相关信息处理后把含义显示在UI表格中，还有计算TCP校验和和偏移量的方法。

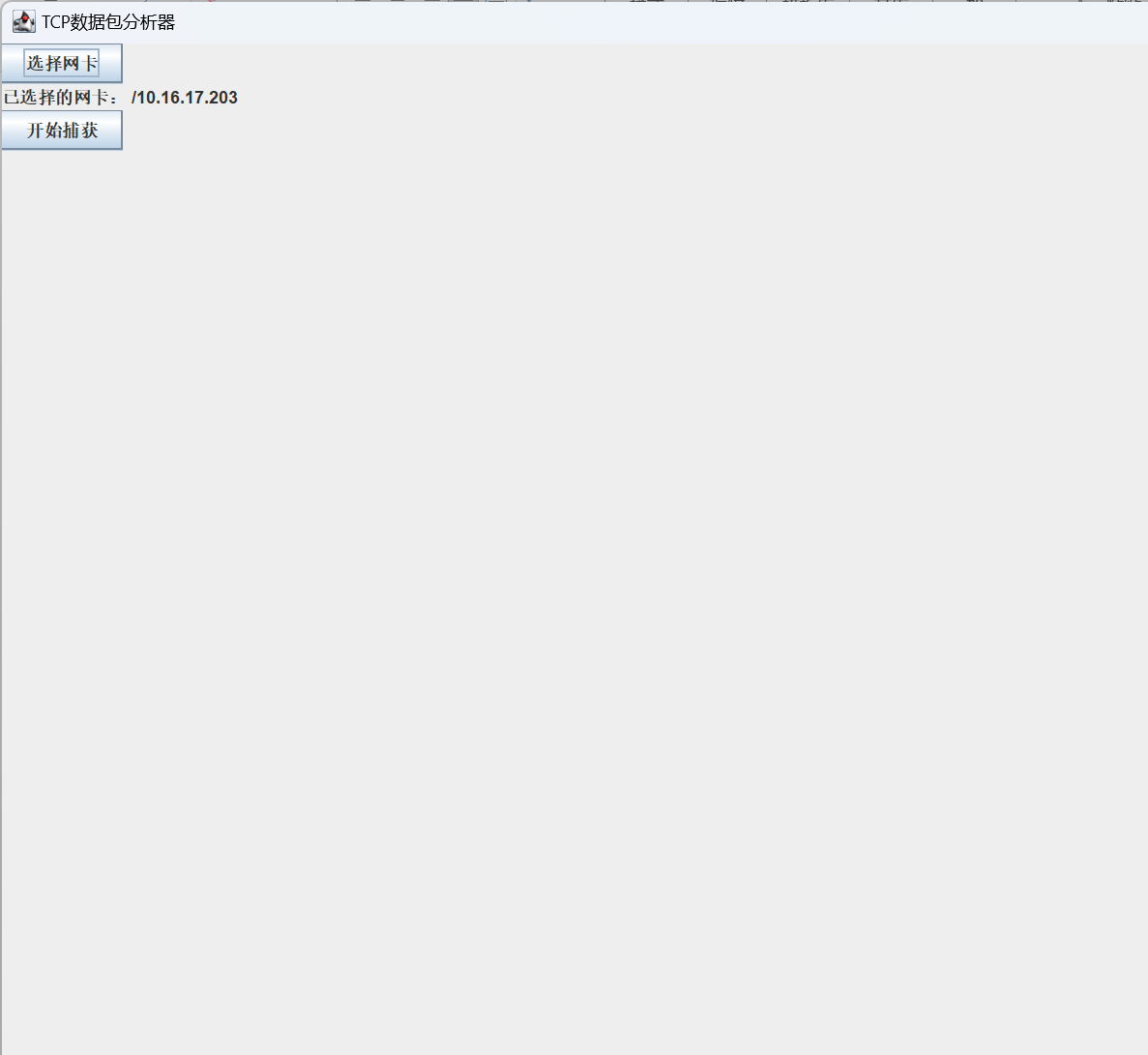
## 调试分析

点开界面时的UI

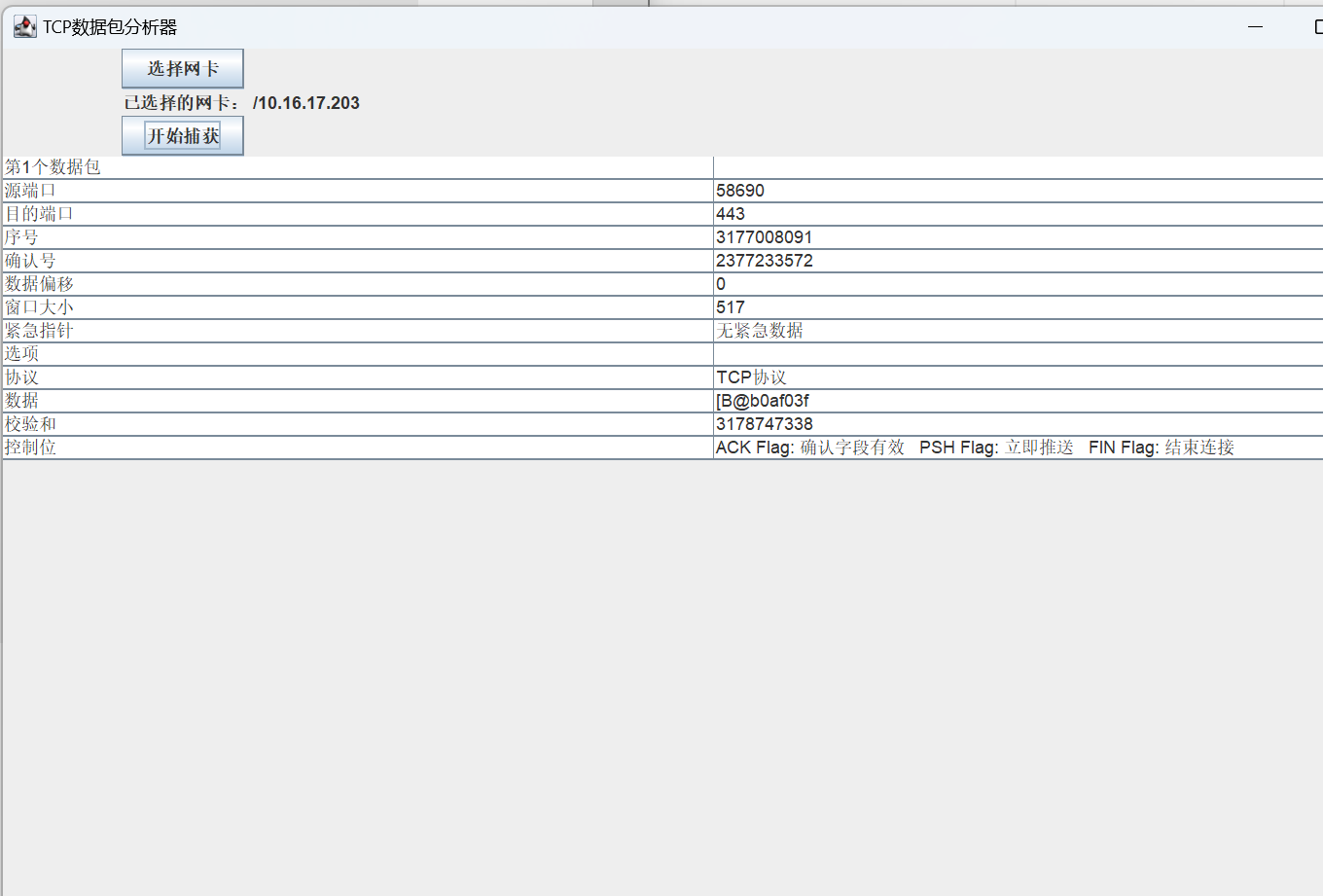
选择网卡



选择网卡后



捕捉包后显示的信息



## 课设分析

在完成网络数据包抓取与分析程序的课程设计过程中，我获得了许多宝贵的经验和收获，下面我将对整个过程进行总结和反思。

收获与经验：

1. 深入理解计算机网络原理：通过这个课程设计，我不仅加深了对计算机网络原理的理解，还学习了如何通过编程实践来应用所学的理论知识。特别是在实现数据包捕获和分析过程中，深入理解了网络协议的结构和工作原理。

2. 提高了编程能力： 通过编写代码实现网络数据包的抓取和分析，我加深了对Java编程语言的掌握，尤其是在使用第三方库进行网络编程时，学会了如何有效地利用外部资源和工具。

3. 解决问题的能力： 在编程过程中，我遇到了许多问题，如如何选择网卡、如何正确解析数据包、如何进行界面设计等等。通过查阅文档、搜索资料、尝试不同的方法，我逐渐解决了这些问题，并且对问题的解决过程也有了更深入的理解。

4. 提高了调试能力： 在调试过程中，我学会了如何利用调试工具、打印日志信息以及逐步调试等技巧，提高了自己解决bug和调试程序的能力。

遇到的问题与思考：

1. 网络编程的复杂性： 实现网络数据包的抓取和分析涉及到许多复杂的网络编程概念和技术，如套接字编程、数据包解析等。在遇到问题时，我深感网络编程的复杂性，需要不断学习和实践才能掌握。

2. 界面设计的挑战： 在设计程序界面时，我发现需要考虑用户的操作习惯和需求，使界面简洁明了、操作简单方便。这需要综合考虑用户体验和功能实现的平衡，是一个相对较难的挑战。

3. 数据包解析的准确性： 在对数据包进行解析时，我发现需要考虑到不同协议和不同厂商的实现差异，以确保解析结果的准确性。这需要对网络协议有深入的理解，并且进行充分的测试和验证。

对计算机网络课程的思考：

1. 理论与实践结合：通过这个课程设计，我深刻体会到了理论与实践相结合的重要性。理论知识是实践的基础，而实践又可以加深对理论的理解和应用。因此，我认为计算机网络课程应该注重理论与实践相结合，通过实践项目来巩固和应用所学的知识。

2. 培养实际应用能力： 计算机网络是一门应用性很强的课程，学生除了要掌握理论知识外，还需要具备实际应用的能力。因此，课程设计应该注重培养学生的实际应用能力，让他们通过实践项目来解决实际问题，提高解决问题的能力和创新意识。

对计算机网络课程的认识：

1. 重要性： 计算机网络是计算机科学与技术中的重要基础课程，它不仅是连接计算机的基础设施，也是支撑互联网和信息社会的基础。因此，学习计算机网络对于计算机专业的学生来说至关重要。

2. 广泛性： 计算机网络涉及到许多领域和技术，如网络协议、网络编程、网络安全等，涵盖面非常广泛。因此，学习计算机网络不仅可以提高学生的专业素养，还可以为他们未来的工作和学习打下坚实的基础。

总的来说，通过这个课程设计，我不仅学到了许多关于计算机网络的知识和技能，还提高了自己的解决问题和实践能力，这对我的专业发展和个人成长都具有重要意义。我会继续深入学习和实践，不断提高自己的专业水平和能力。