实验报告

实验题目：编写简单的木马监控一台电脑的上网进程

实现步骤及目的：

**木马的自启动程序**

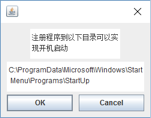
文件：Self.java;SelfInit.java

Self用来启动木马程序，SelfInit确保自启动目录下的脚本文件一直存在（每次运行都更新自启动目录下的脚本文件，确保脚本文件可以一直使用）

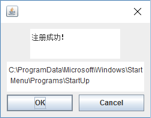
编写时，先生成一个可以启动主程序的bat文件，再在windows启动目录（此处使用windows操作系统）下生成一个可以启动这个bat的vbs脚本文件。因为vbs脚本用到的启动命令【createobject("wscript.shell").run "path(这里为bat文件的路径)",0】 只能用来启动bat，vbs，exe文件，所以对于jar文件，需要一个bat配套。

单独测试结果：

运行后会出现如下框图：



点击OK后会出现下面显示，表示已经在自启动的目录下创建好脚本文件了。



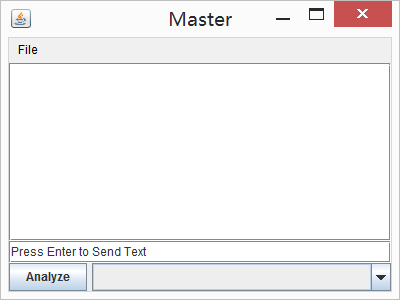
启动目录下会出现脚本文件，分别为SelfInit和Self对应生成的脚本文件。

当脚本文件创建好后，下次开机时自启动目录下的脚本文件自动运行。

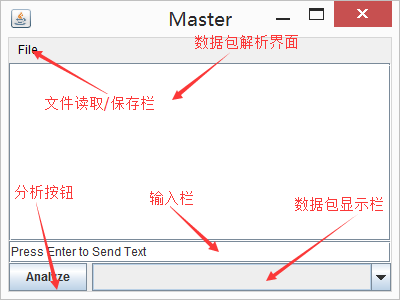
重启电脑，再次开机时会弹出上述框图，程序运行成功。（为了结果明显加入了GUI，加入木马时可以把GUI删掉）

**Master端（控制端）**

文件： masterTest.java; Master.jar



Master端结构概况：



Master端就是控制端。当受害者电脑被木马控制以后，控制人可以在输入栏中输入命令来观察控制情况，也可以通过输入命令获得被监视人的数据包并解析。分析完数据包以后可以点击File里面的保存将数据包保存下来。保存后的数据包可以通过读取再次读入，并且会在数据包栏内显示，此时的数据包仍可解析。

**建立与server端联系的两个socket**

1. 发送/接收命令的socket IP:120.77.168.152(服务器固定IP) 端口：1434

该socket用于与Server端进行指令交流，当输入指令以后会有回复对应的指令。

可用指令如下：

get cilentlist 获取被控制者的IP地址

get masterlist 获取控制者的IP地址

get maclist 获取被控制者的MAC地址

get (mac地址) start 对特定的MAC地址开始抓包

get (mac地址) stop 对特定的MAC地址停止抓包

send (ip) shutdown 使特定被控制者关机

该过程是通过BufferedReader和BufferedInputStream/BufferedOutputStream完成的。先通过BufferedReader从键盘读入数据，再通过BufferedOutputStream传输给Server端并将回复的指令用BufferedInputStream接收。

1. 发送/接收数据包的socket IP:120.77.168.152 端口：1438

该socket用于接收从Server端转发来的数据包并将其存入到抓包栏中。该过程是通过ObjectInputStream完成的。我们抓到的应该是一个完整的包而不是String类型的文字，所以我们使用了ObjectInputStream来接收数据包，这样方便保存和读取。因为Master端保存的不再是文字，而是一个个Object类型的包，读取的时候不会出现信息缺失。我们采用readObject（）方法读取数据包。

这两个socket是相互独立运行的，拥有两个不同的线程。

**建立输入栏**

输入栏采用BufferedReader监听系统输入，采用KeyListener类下的KeyPressed方法监听活动。若是按下回车键即输入BufferedReader中得到的从电脑输入的文字。

**建立抓包栏**

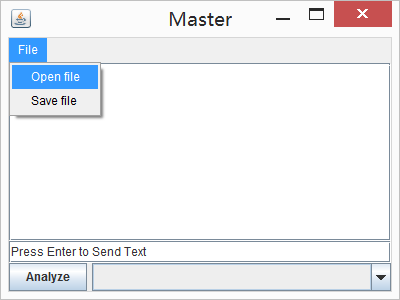
抓包栏是通过JComboBox完成的。我们将抓包栏设置为一个JComboBox，每当接收包的ObjectInputStream读入一个数据，我们就将其存入到JComboBox内。然后采用ItemListener类下的itemStateChanged方法监听选择包的变化，若变化则表示选中了一个数据包，此时就将选中的数据包解析出来输出到解析栏中。

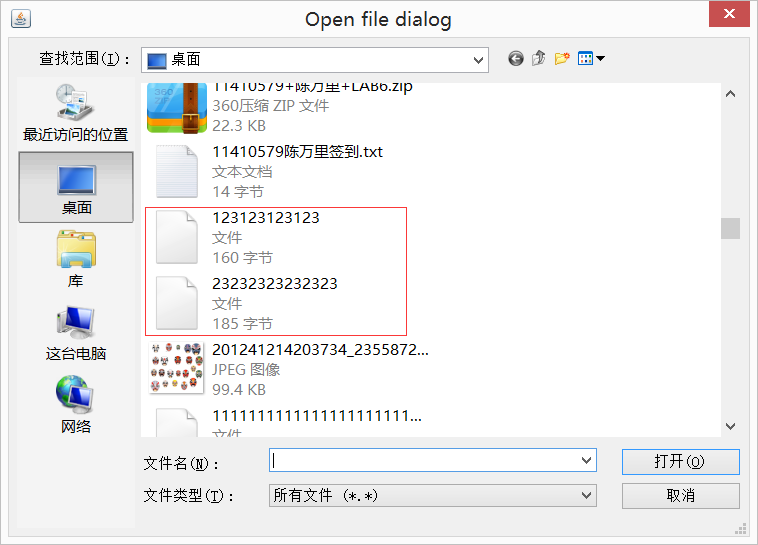
**建立解析栏**

解析栏采用JTextArea。其目的就是将选中的数据包解析，采用了toString方法将选中数据包的信息转化成文字输出到解析栏中。

**保存/读取文件栏**

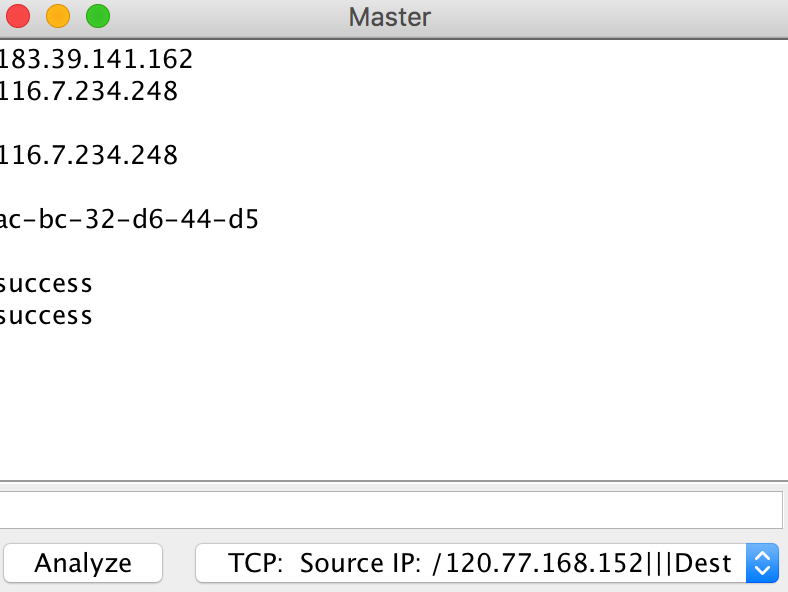
为了读取数据我们可以采用File栏下的Open file选项。选择之后会跳出一个选择框，选择文件的类型是一种特殊类型，只有该类型的文件才能打开。而生成该类型文件需要通过Save file。

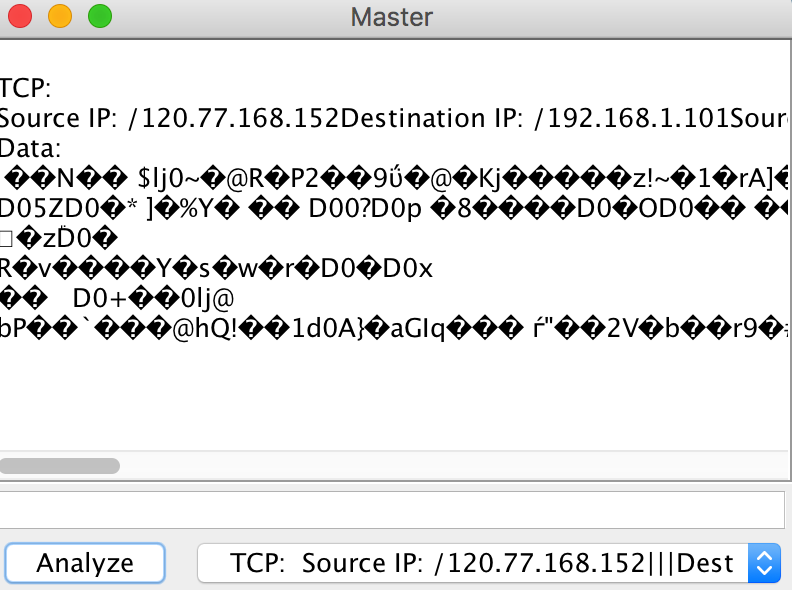


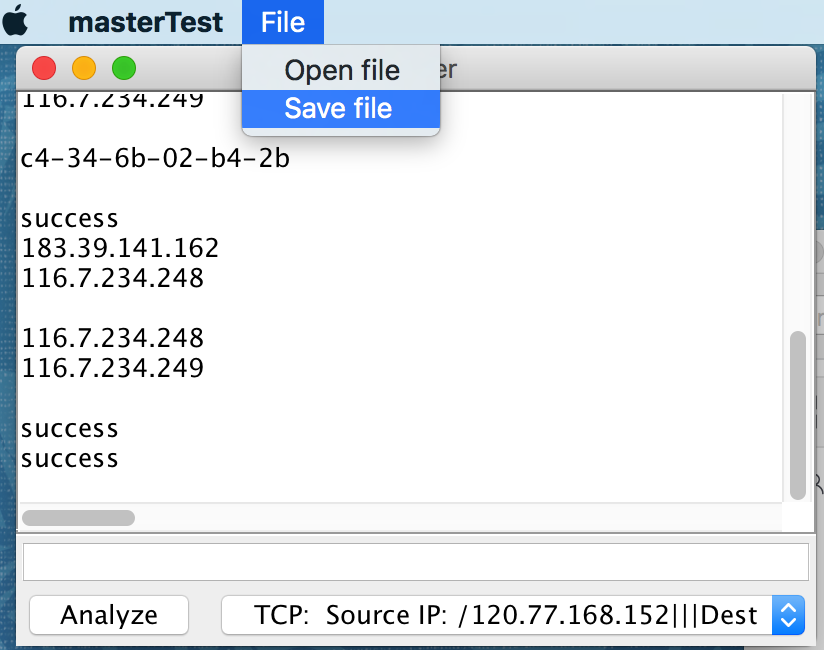


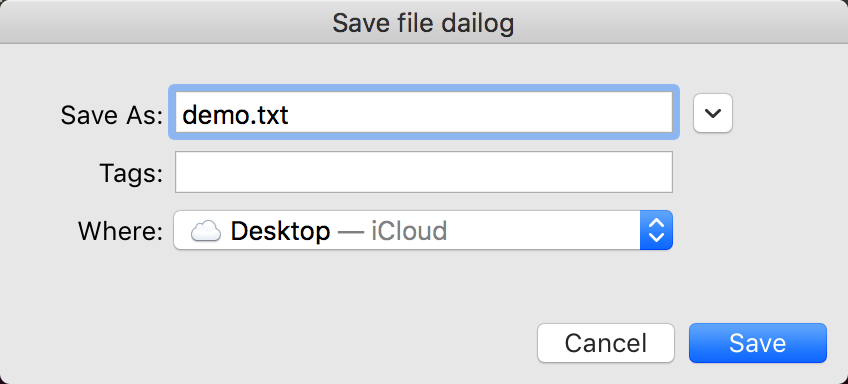
GUI上，保存/读取栏采用了Menubar Menu MenuItem等组件。功能上采用了 FileDialog组件和FileInuputStream/FileOutputStream和ObjectInuptStream/ObjectOutputStream。每当储存的时候就将抓包栏中的Object类型的文件通过ObjectO输出到FileOutputStream中，再由FileOutputStream保存到本地。读取的时候同理。

控制端操作截图：







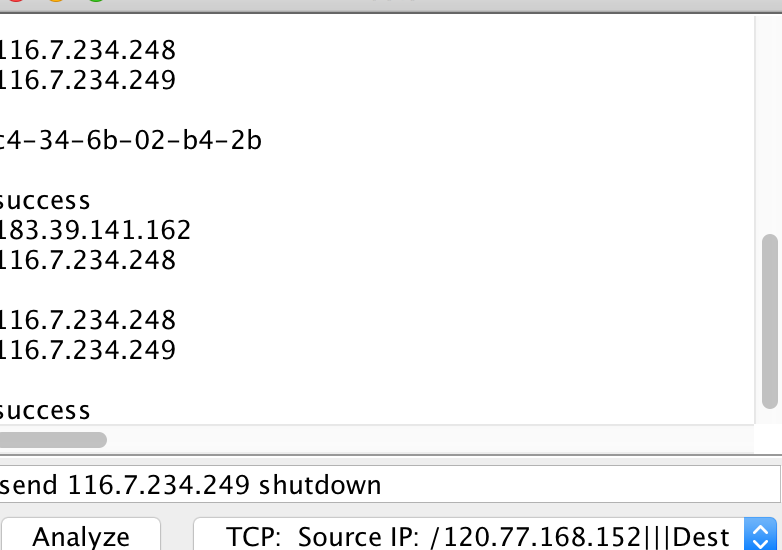


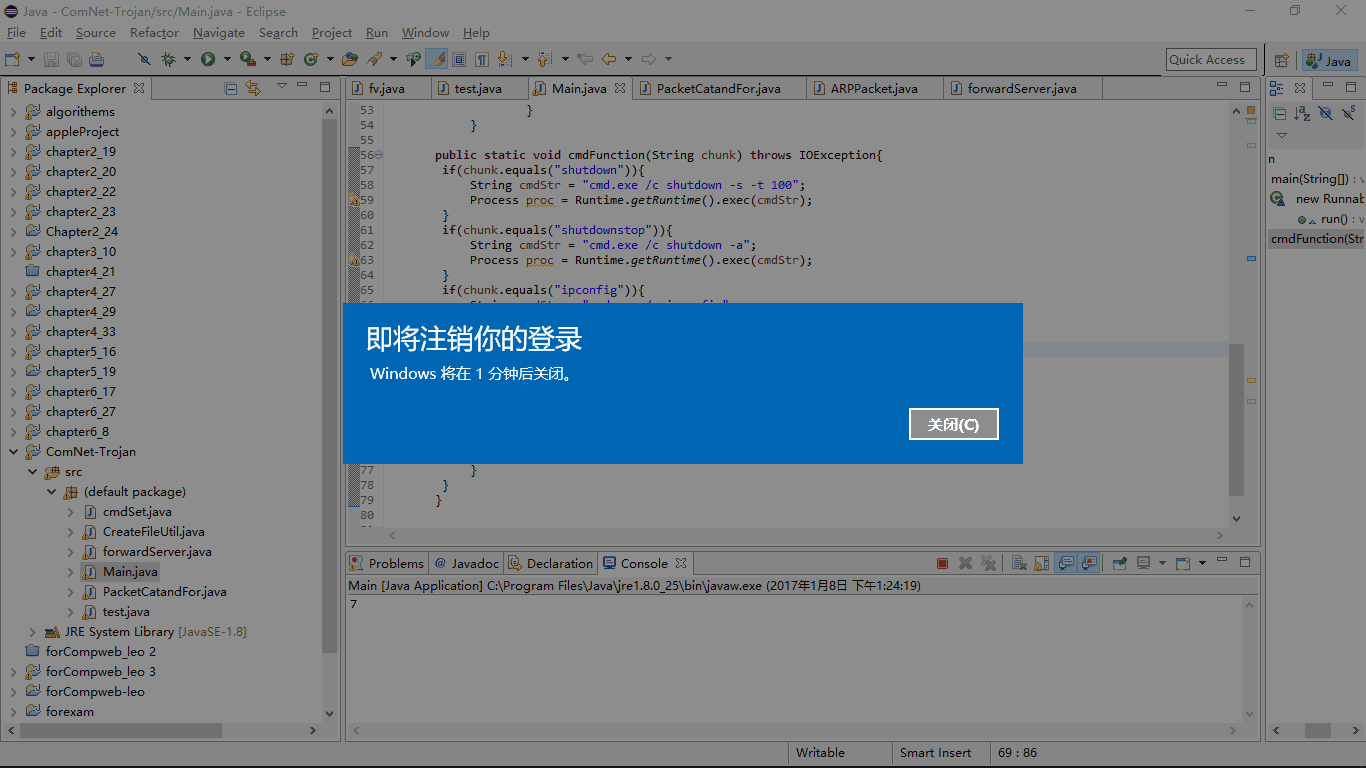
**目标主机**

文件： Main.java; GetMacAddress.java; PacketCatandFor.java; Main.jar

1.通过对Jpcap的使用，实现对受控主机的网络抓包。Jpcap是2003年日本开发的一套能够捕获、发送网络数据包的java类库。通过提供在windows或UNIX系统上进行底层的网络数据的访问API实现抓包发包的功能。特别的，为了实现代码的健壮性，我们设计程序将自动选择主要通信的网卡设备进行监听，获取本地MAC地址，采用多线程的方式发送经过分类与初步处理的数据和接受指令。

2.木马的功能应不止于网络活动的监视，在此基础上，我们设计了指令接收并执行的功能。控制端发送的指令在经过服务器的转发获目标主机接受后，目标主机根据字符串匹配特定的指令，调用命令行执行。我们采用的测试例子为关机与终止关机。额外的我们保留了ipconfig命令的执行，在简单的修改后，程序能够将命令行窗口产生的字符串结果发送到控制端读取。





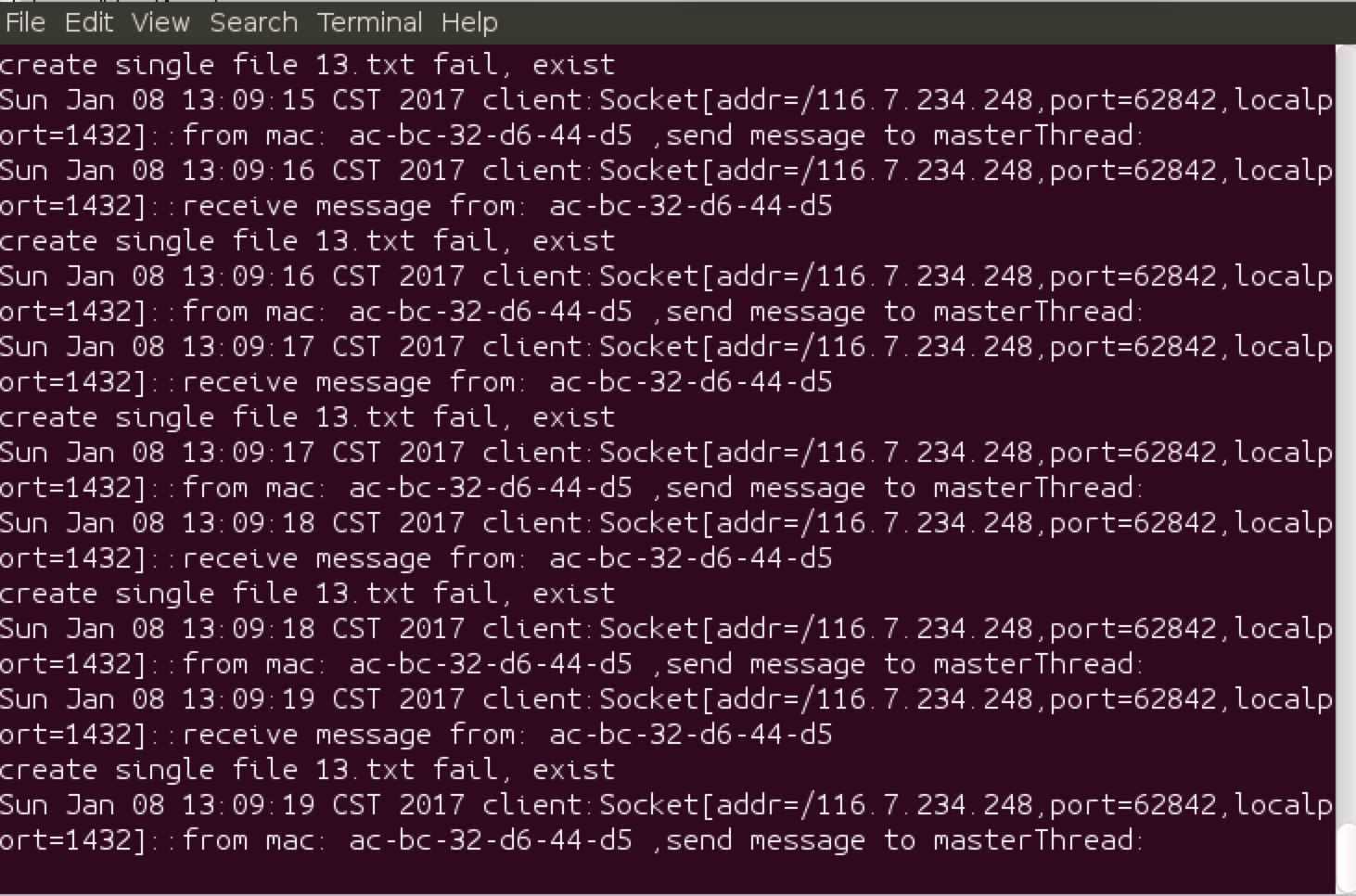
**服务器**

文件：forwardServer.java;forwardServer.jar

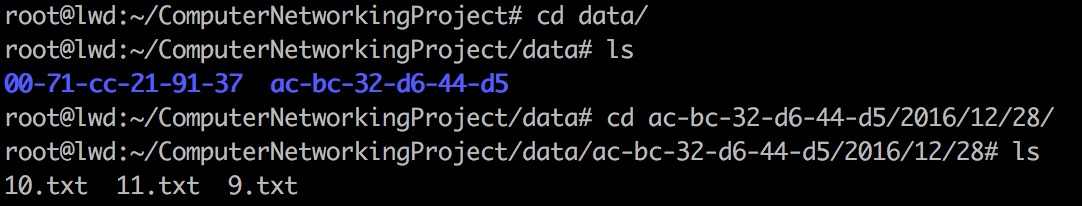
功能：目标主机通过TCP协议连接上阿里云服务器后分配2个线程给此连接，经过简单的密码验证后，1个线程接收这台目标主机发过来的网络活动信息，以mac地址及具体收到包的时间作为分类依据存储在data文件夹里，以便以后可以传输给控制端，进行更深层次的分析；另一个线程用来转发来自控制端的命令，例如查看ipconfig或者关机等等。控制端通过TCP协议连接上服务器后，同样分配2个线程。经过简单的密码验证后，1个线程用来接收来自控制端的指令，返回相应的信息，包括查看目标主机的相关信息和选择是否接收相应MAC地址发来的网络活动包信息，选择接收的网络活动包会通过另一个线程发送给这台控制端。如果指令是控制目标主机关机，则发送命令给相应目标主机。

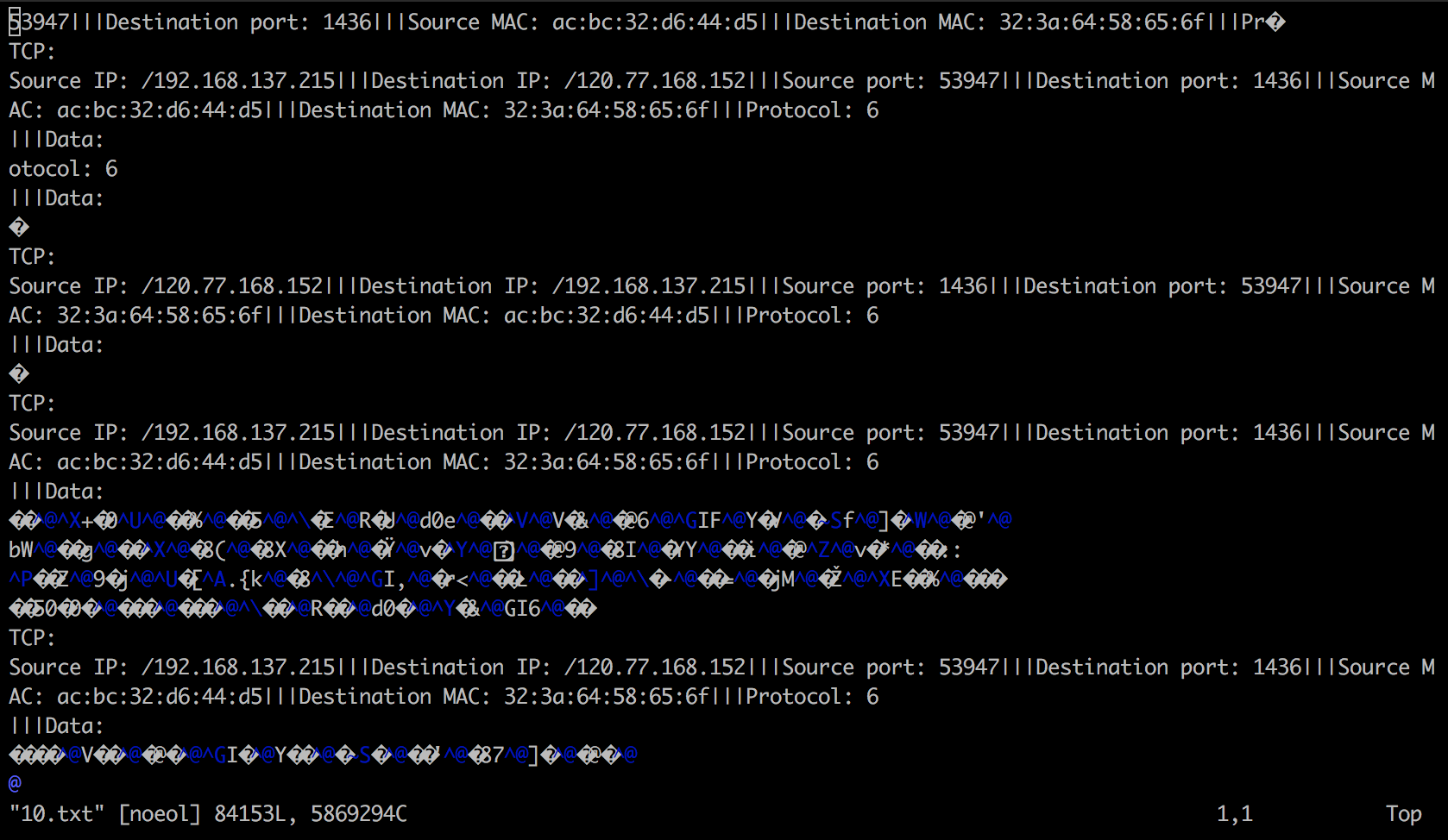
服务器截图：

为了运行效率等等原因，服务器没有桌面可视化，所以是用命令行



存储文件：





组员分工：

廖伟多：服务器端的多对多转发

吴昱含：服务器端的数据包存储

刘昊：目标主机发包与接收指令

李思远，韩菲：自启动

陈万里：控制端的GUI与发指令