

**网 络 编 程 技 术**

**项 目 报 告**

课程名称： 网络编程技术

指导老师： 苏铓

学 院：计算机科学与工程学院

班 级： 9191062301

学 号： 919108320105

姓 名： 陈博宏

选 题： 包过滤防火墙

# 项目功能概述

本次项目选题为“包过滤防护墙”，完成了要求的三点功能：设计合理的用户页面；用户通过界面进行配置，实现制定IP的黑名单和白名单。

在此之外，提供多种协议包选择性过滤，如ICMP,IGMP,TCP,EGP,UDP,OSPF等丰富的协议选择过滤（协议不分先后以协议号排序），还提供对所有IP数据包的全局过滤，对于TCP、UDP端到端协议还提供选择端口过滤。

为方便用户体验操作记录，提供历史记忆功能，开启防火墙后能显示阻拦的数据包并将记录存放在文本文件中，能储存增添的防火墙记录，下次开启运行能自动加载。

# 开发运行环境

操作系统：1 Win10下VMWare虚拟机CentOS 7.9，内核版本Linux 3.10.0-1160.el7.x86\_64

2 腾讯云服务器CentOS 7.6

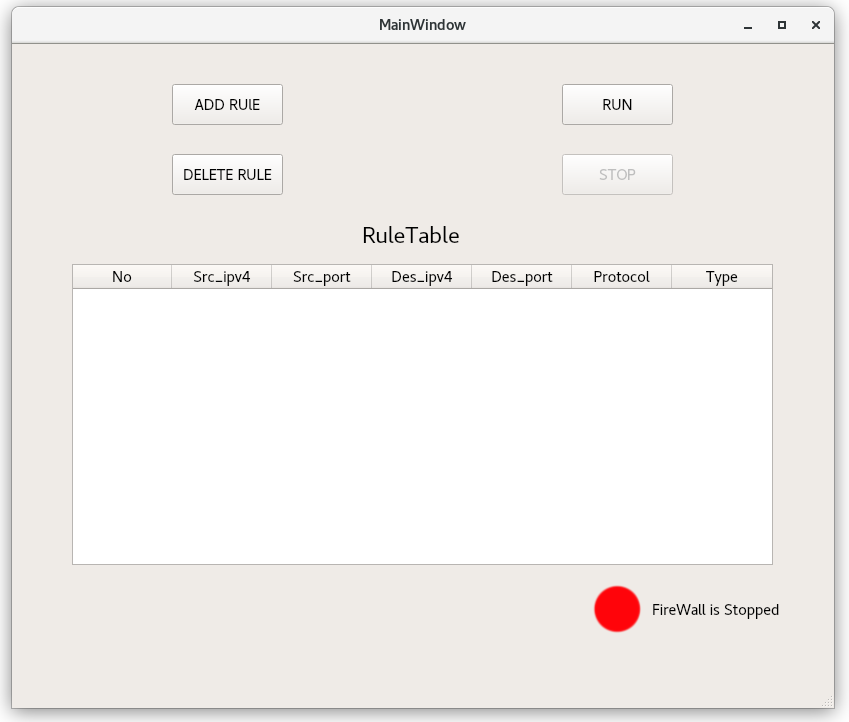
开发工具： Win10操作平台：VSCode，WSL，MobaXterm，VMWare，SourceInisght

Linux CentOS7操作平台：Qt5.9.9 , Vim

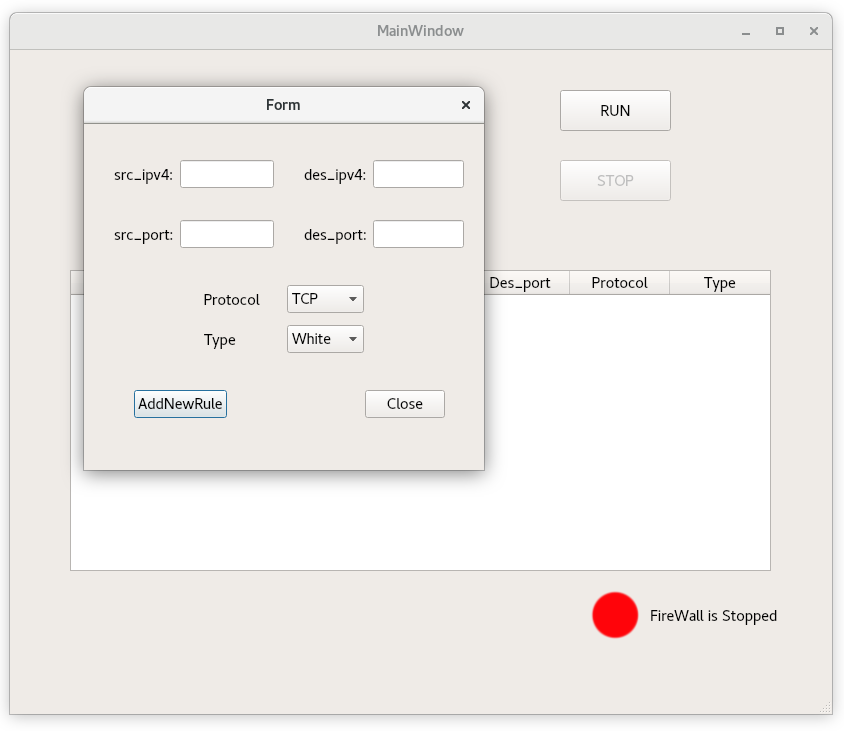
开发语言：C , C++

# 运行过程

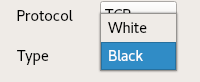
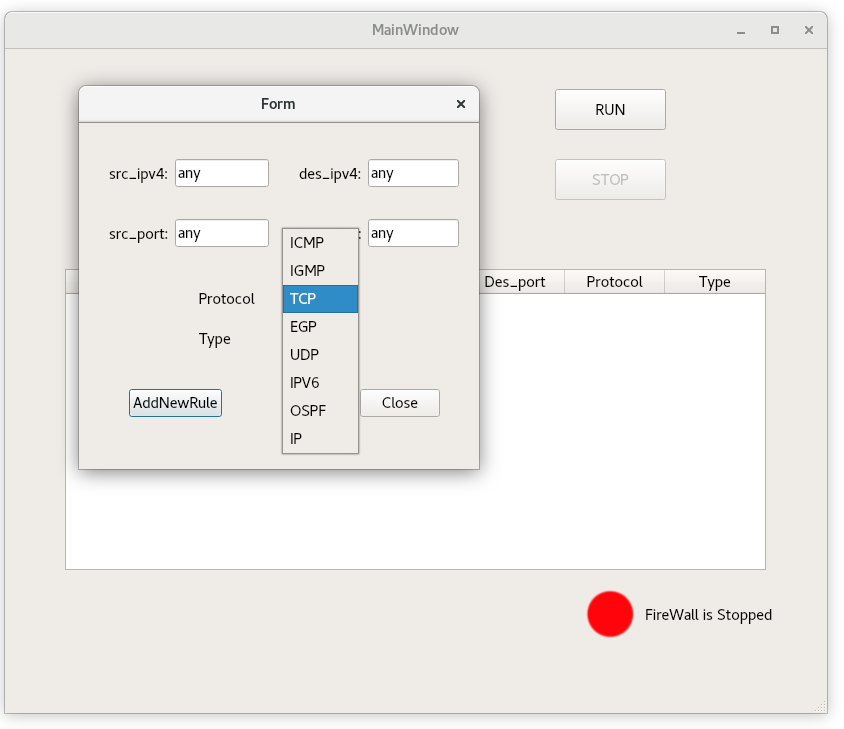
1.初始页面：



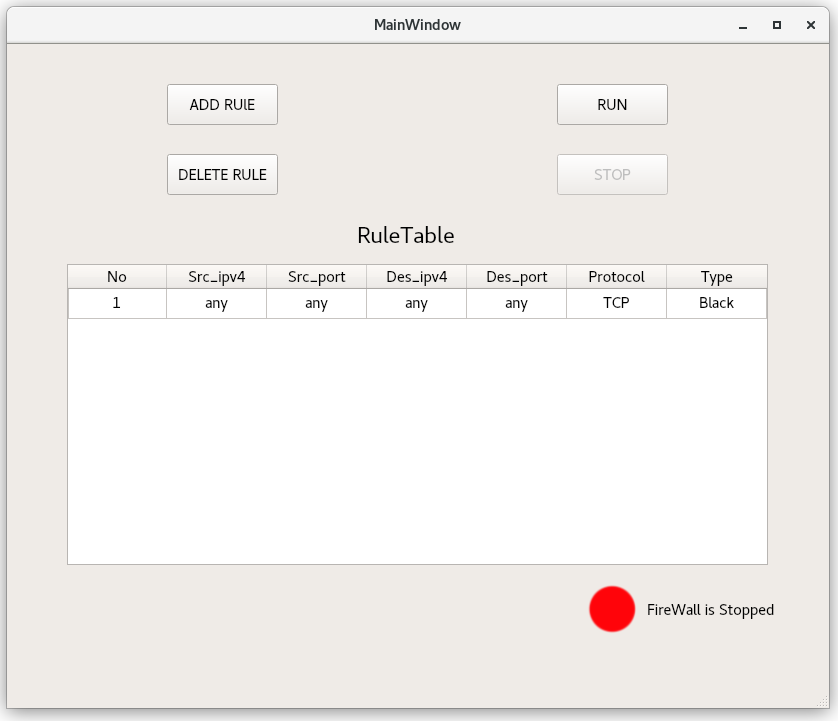
2.点击ADD RULE按钮后，弹出新增规则的模态框



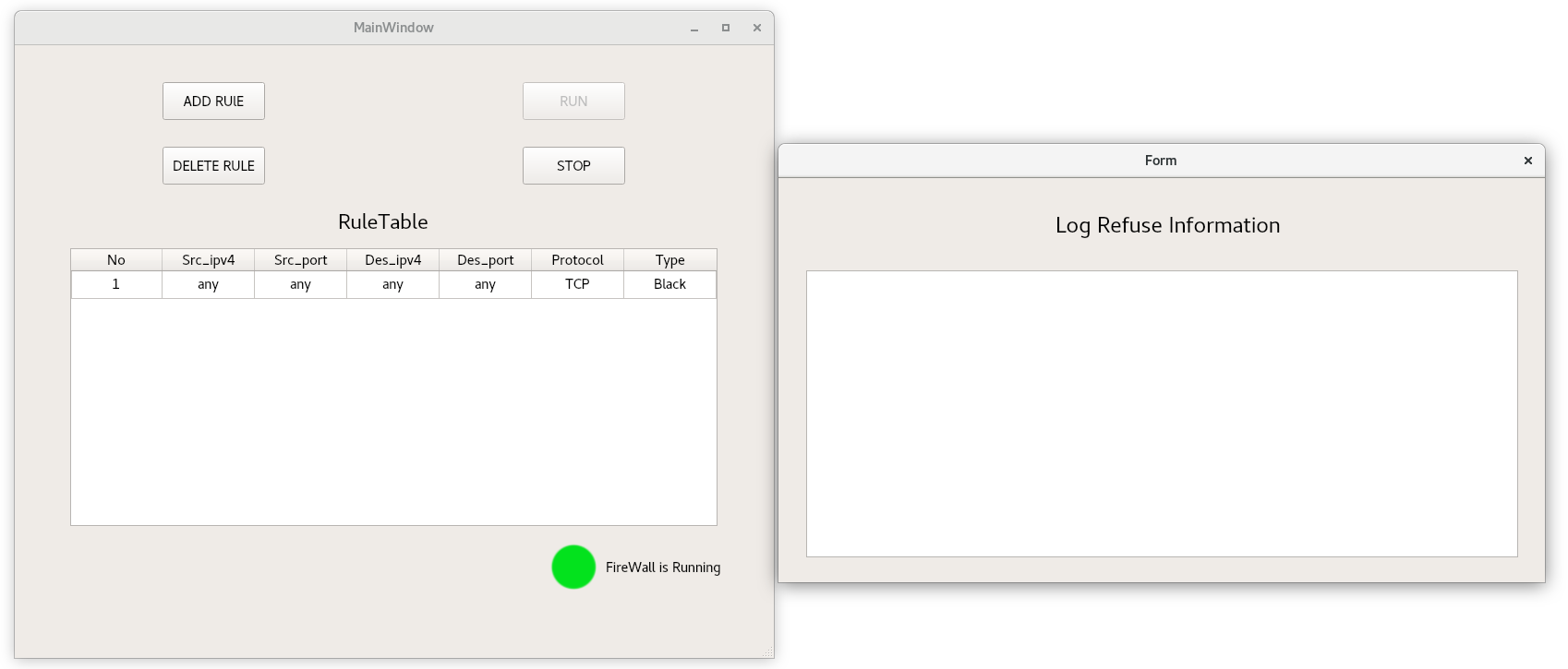
3.输入源和目的地址，源端口和目的端口，在此端口只对端到端协议起作用，便于举例说明，在此均输入any，即对所有地址和端口都起过滤作用，协议选取TCP，种类选择黑名单，选择完毕后点击AddNewRule按钮



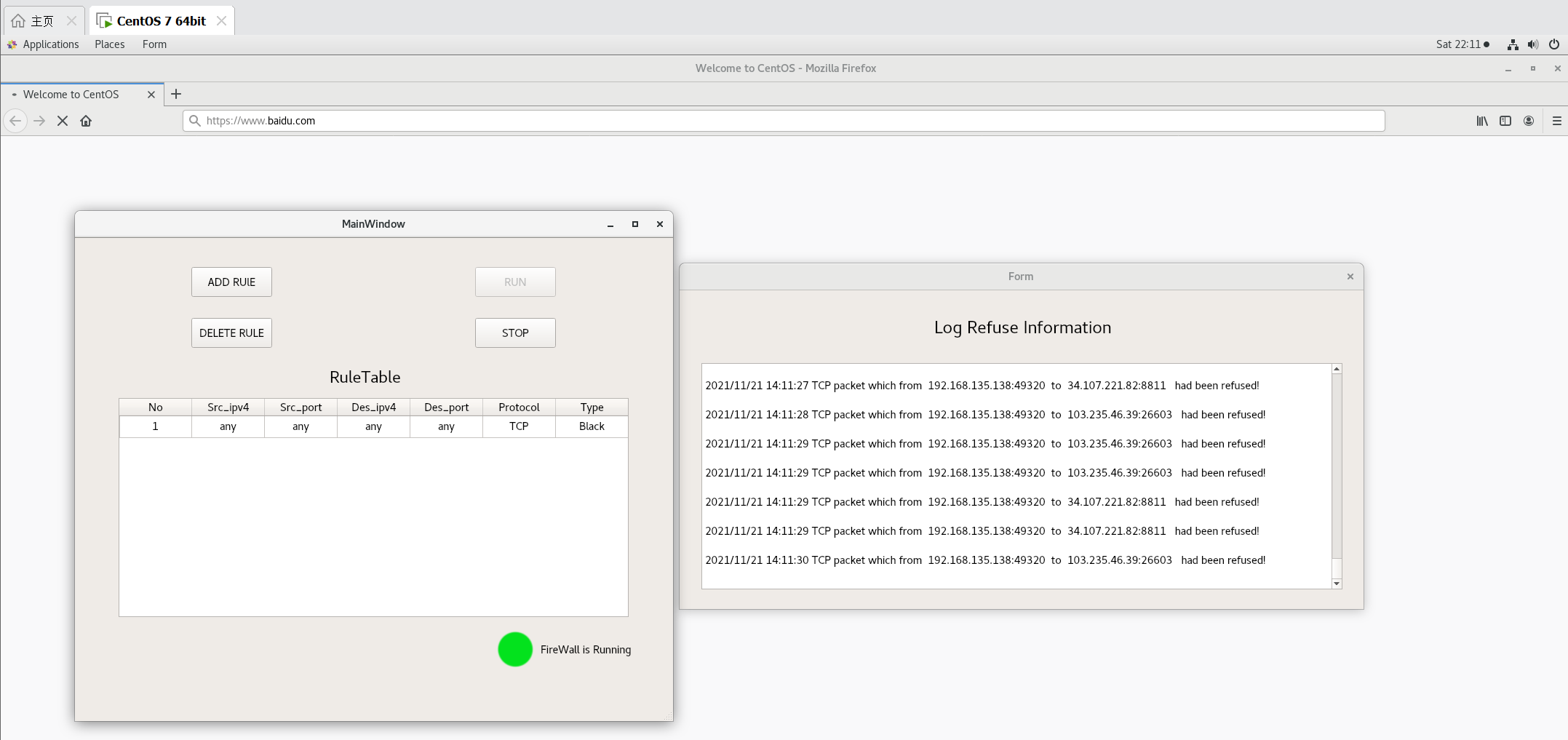
4.列表新增



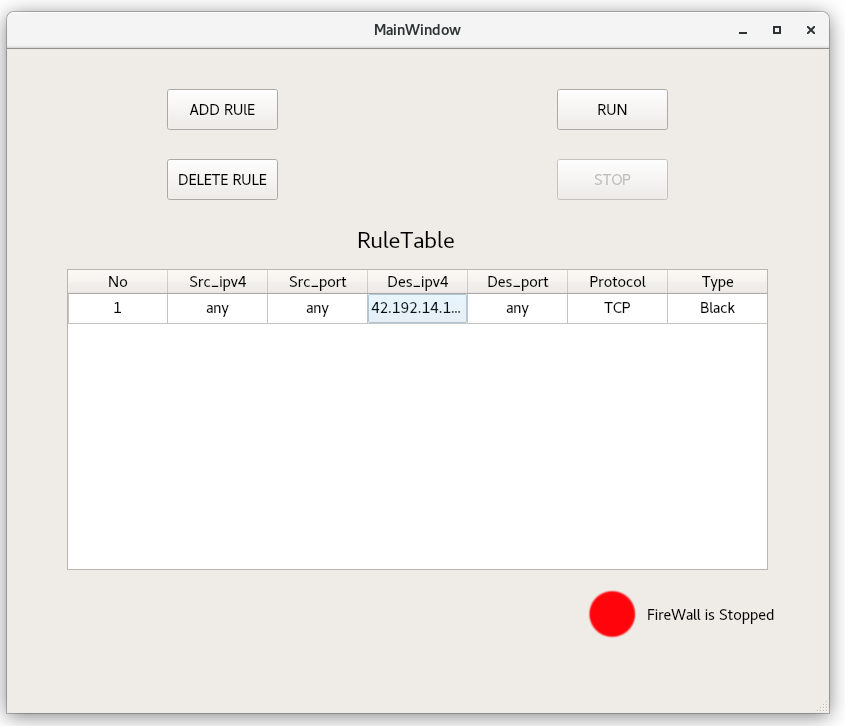
5.点击run，运行防火墙，下方状态区红色停止变为绿色运行，弹出非模态显示规则运行窗口



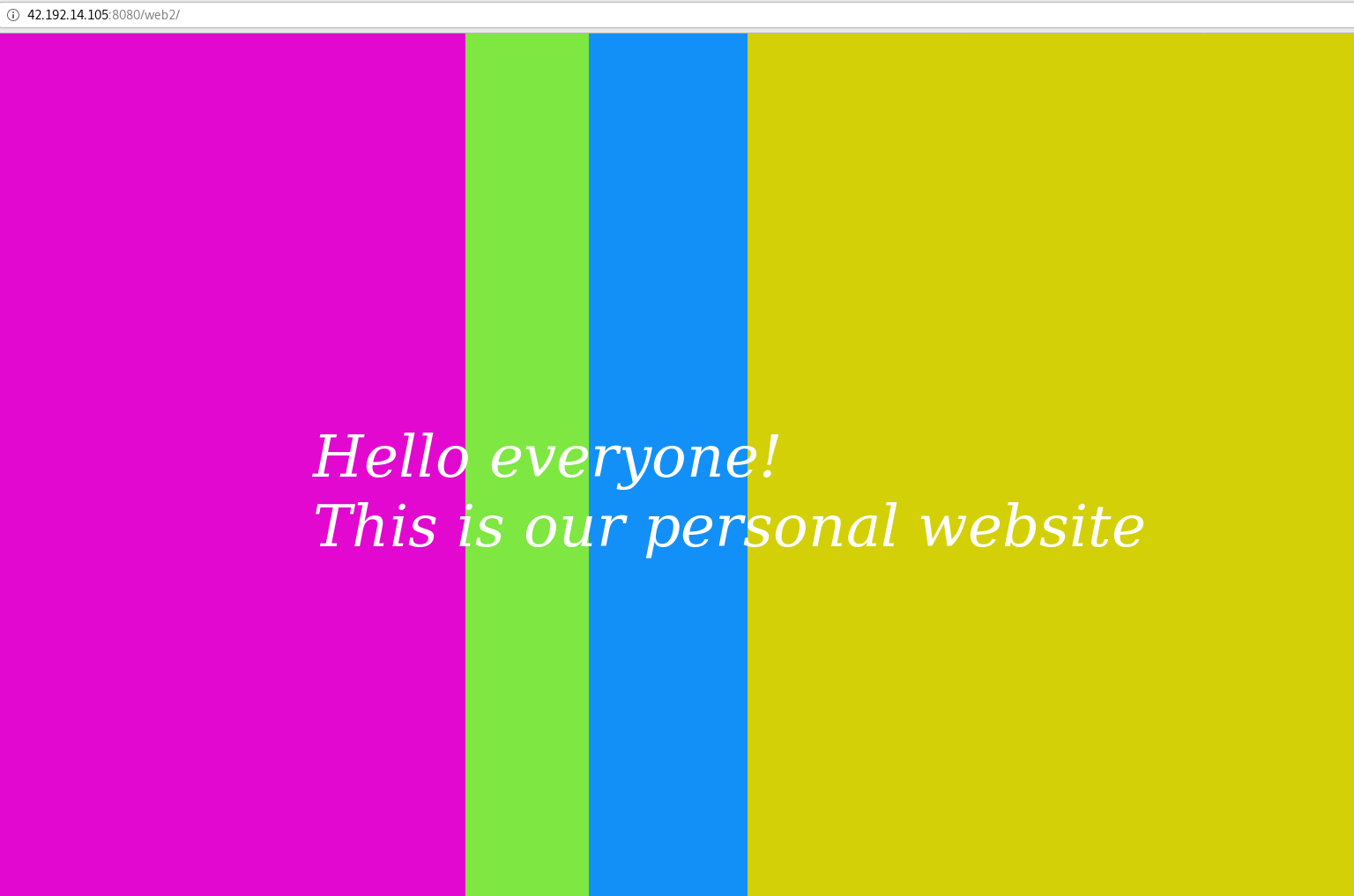
6.测试，打开浏览器，输入网址，比如[www.baidu.com](http://www.baidu.com)，可以看到TCP包正在被拦截，窗中信息持续更新弹出，浏览器始终处于加载状态，无法进入网站



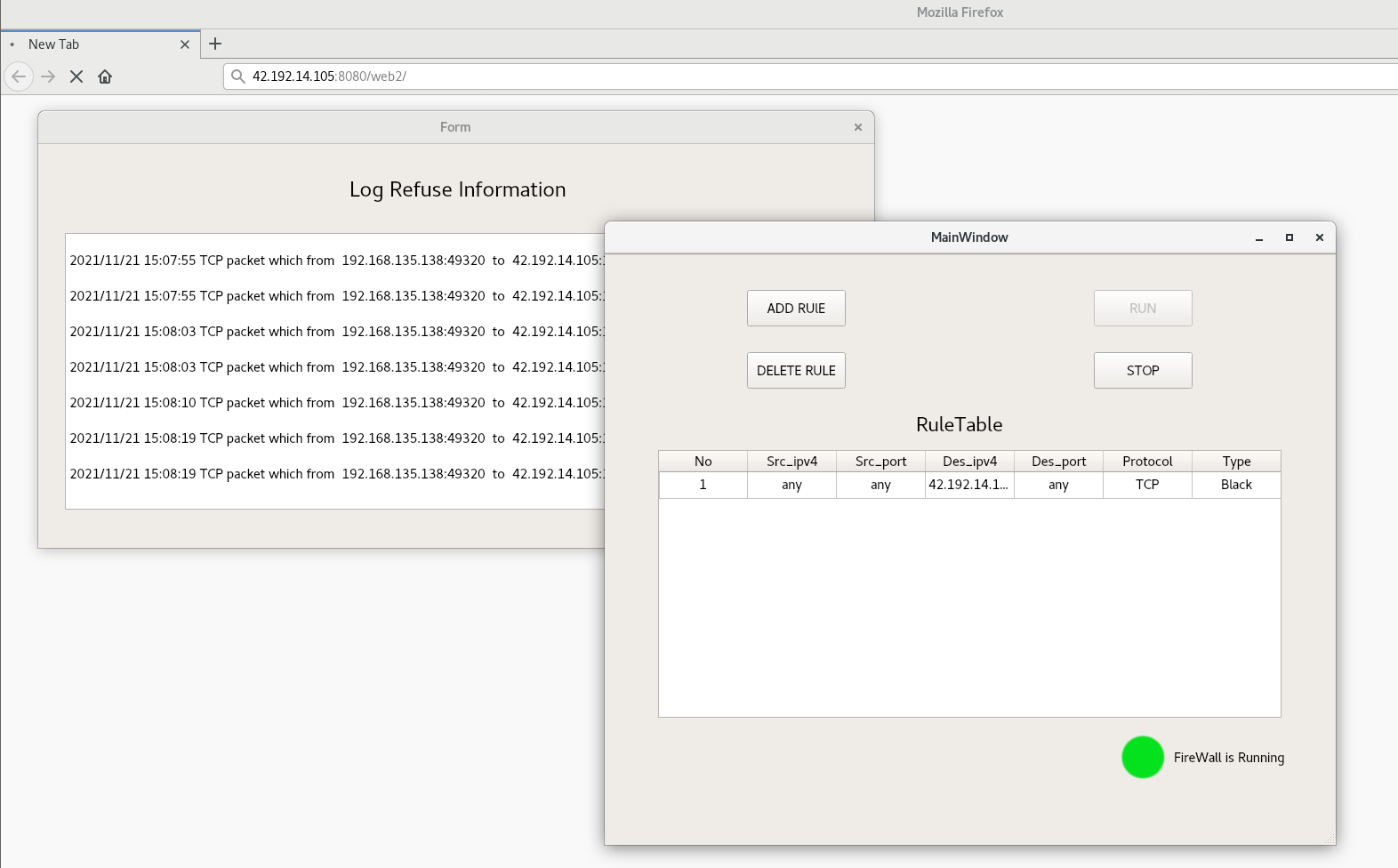
7.下面测试对定点IP地址的过滤，考虑到常见一个域名可能对应多个IP，方便起见我们访问部署在云服务器上的网站，点击规则，按下DELETE RULE按钮，即可删除前文增添的规则，选择目的IP地址为42.192.14.105，端口号为8080，协议同样为TCP，类型也为Black黑名单。



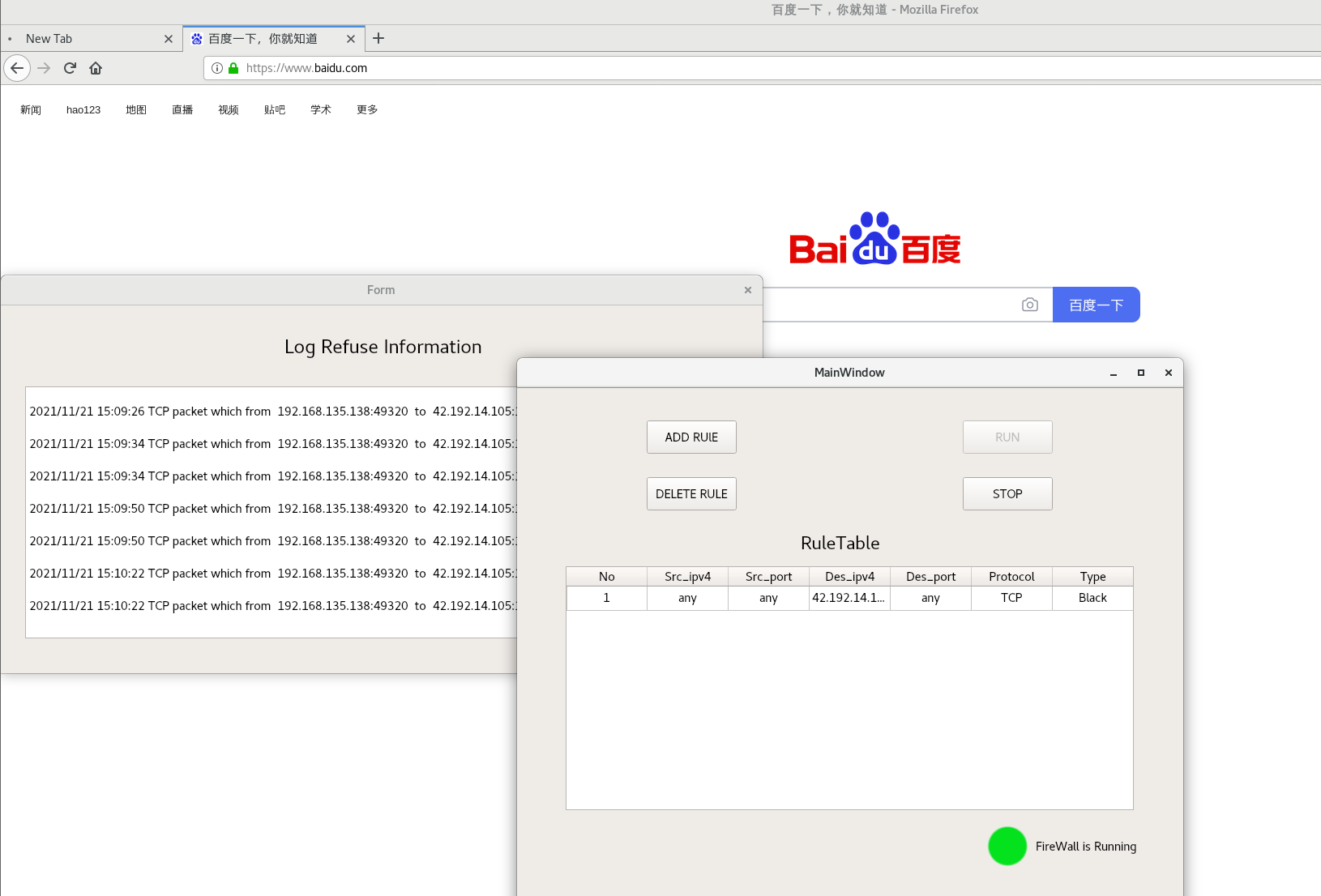
8.暂不开启防火墙，进入网站<http://42.192.14.105:8080/web2/>，进入成功



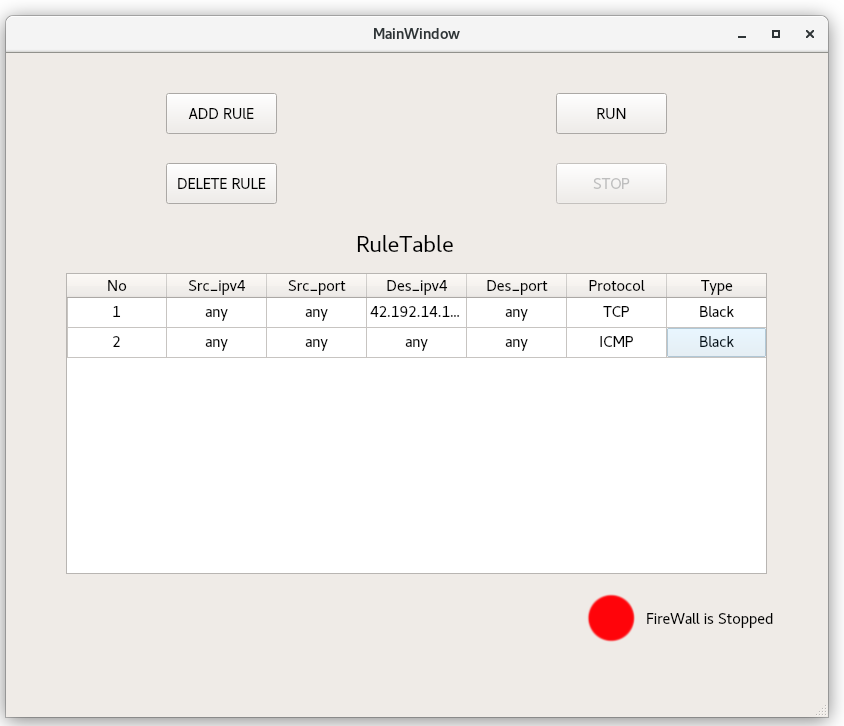
9.开启防火墙后，重新进入该界面，显示一直加载中，TCP数据包被拦截



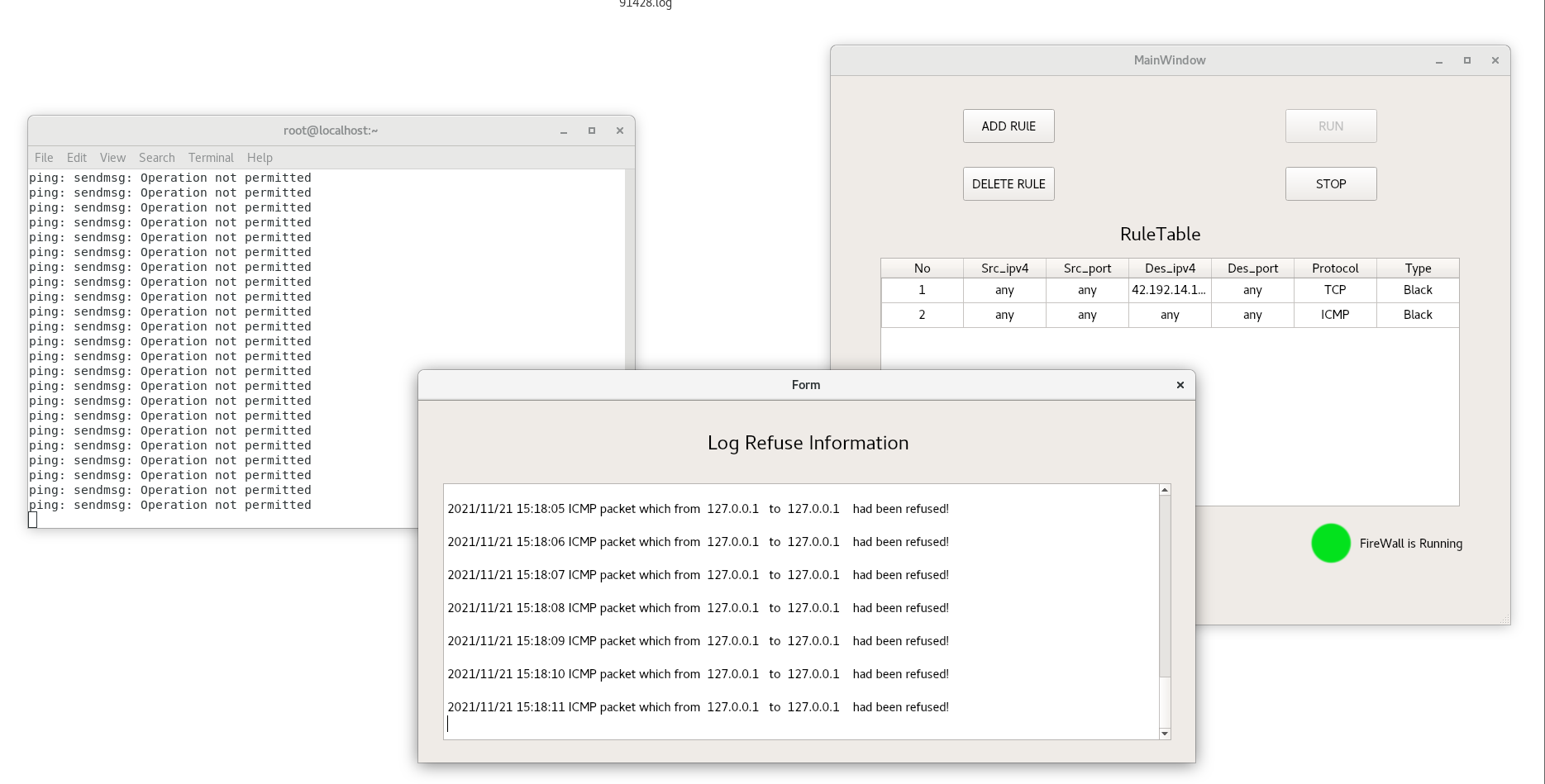
10.保持防火墙开启，进入其他网站，如百度，进入正常，说明可对特定IP地址进行拦截

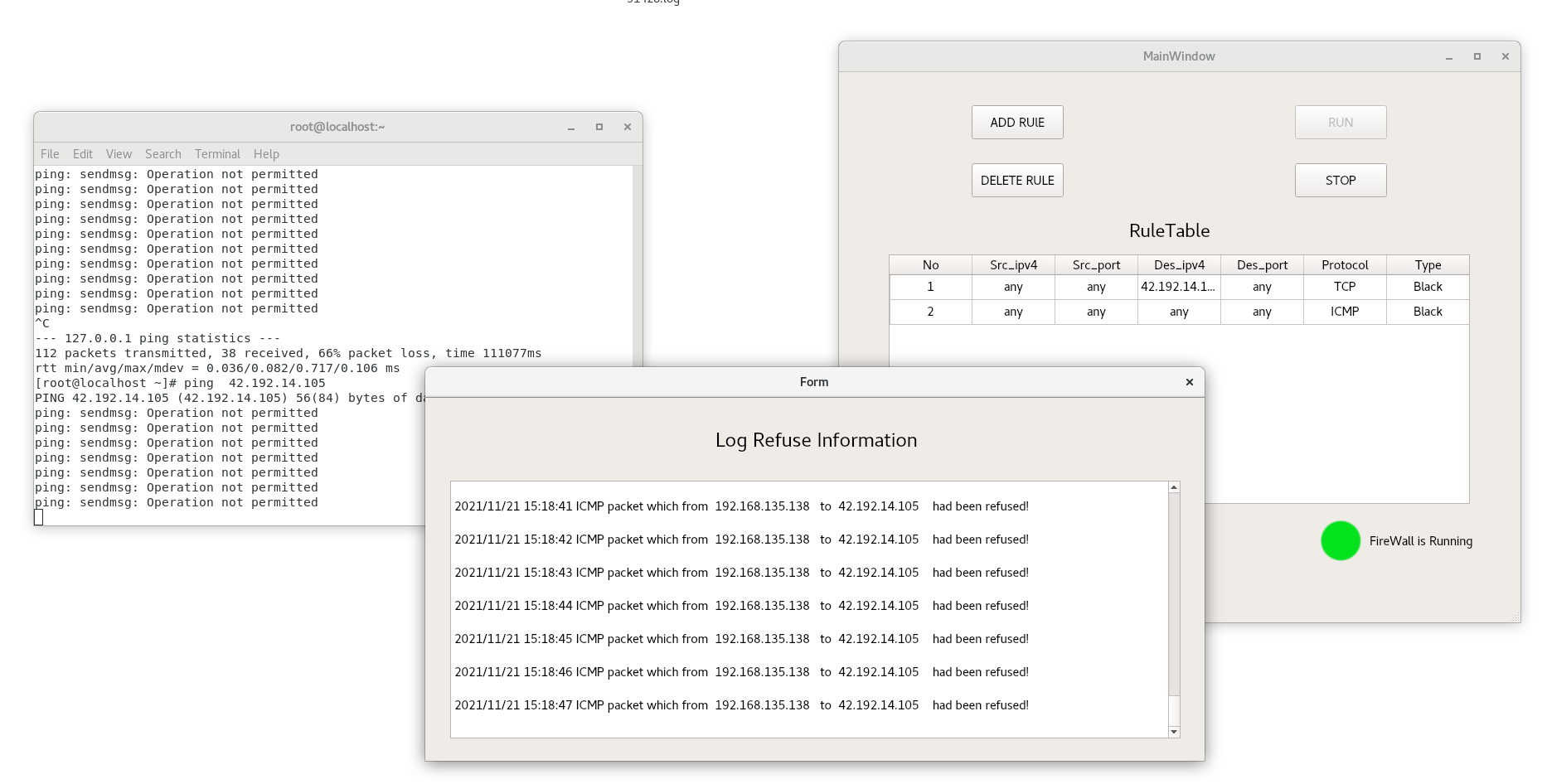


11.下面对拦截ICMP进行测试，增添拦截ICMP规则，运行防火墙

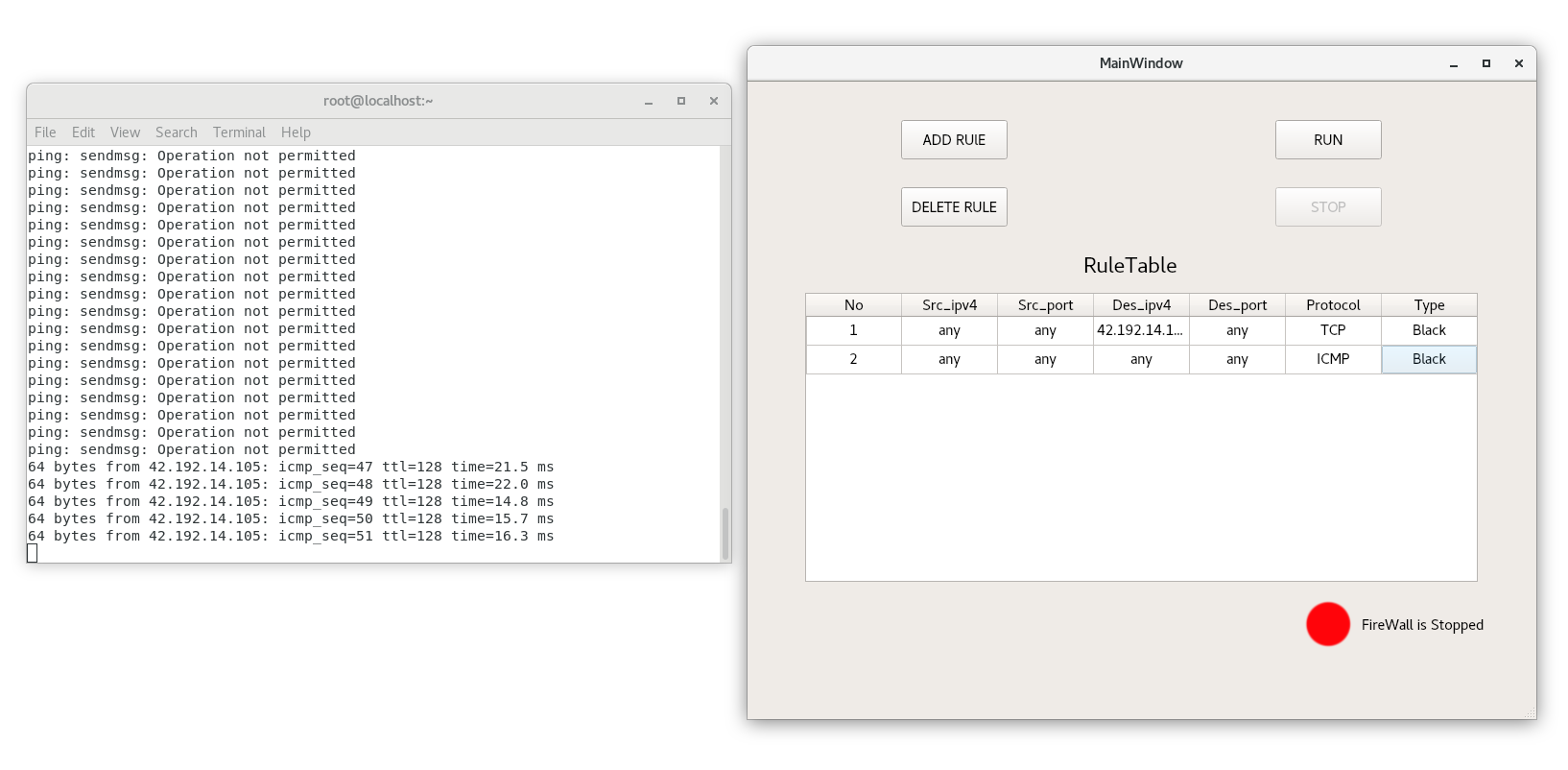


12.打开终端，ping 127.0.0.1，ping 42.192.14.105进行测试，如图，过滤成功

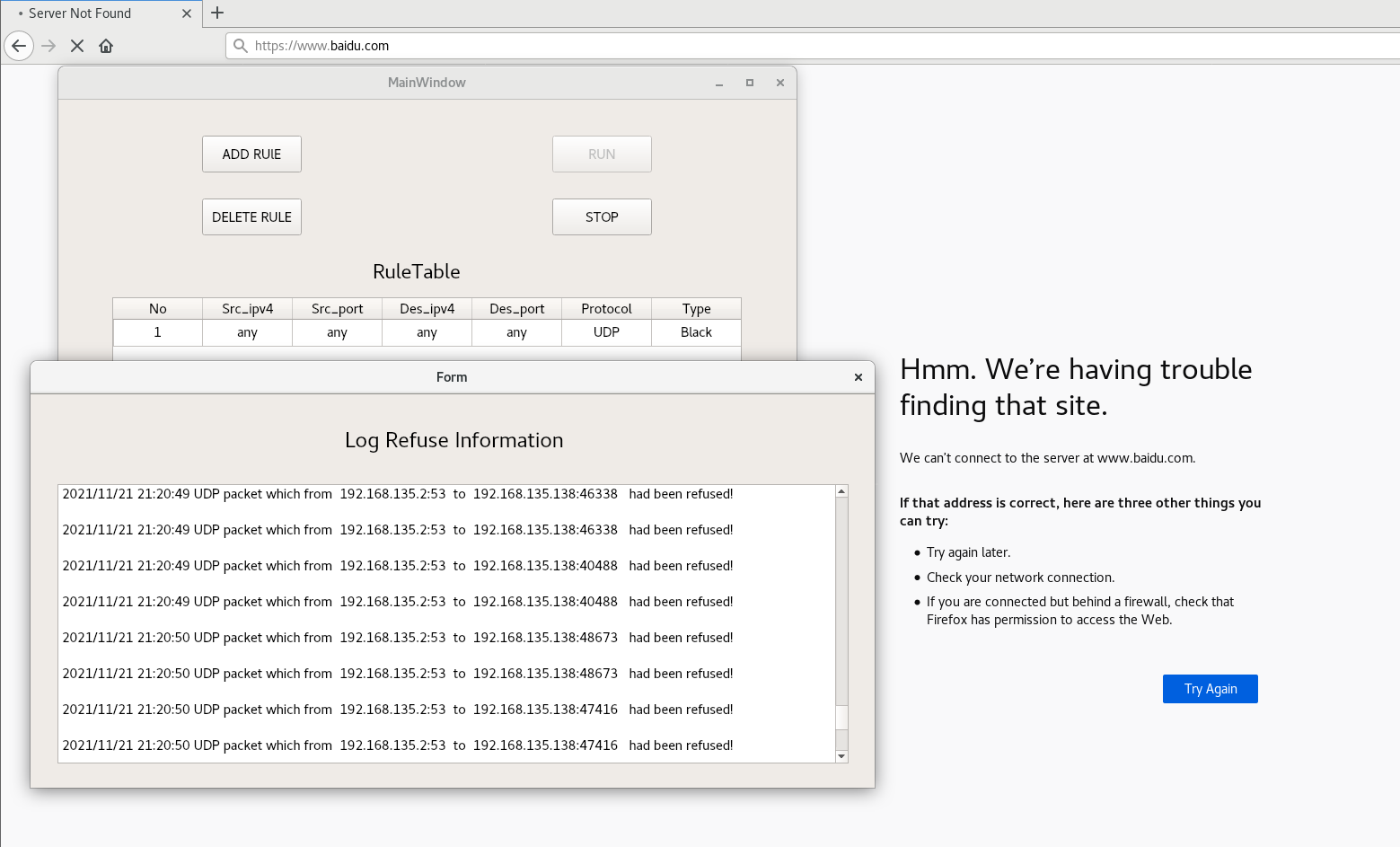




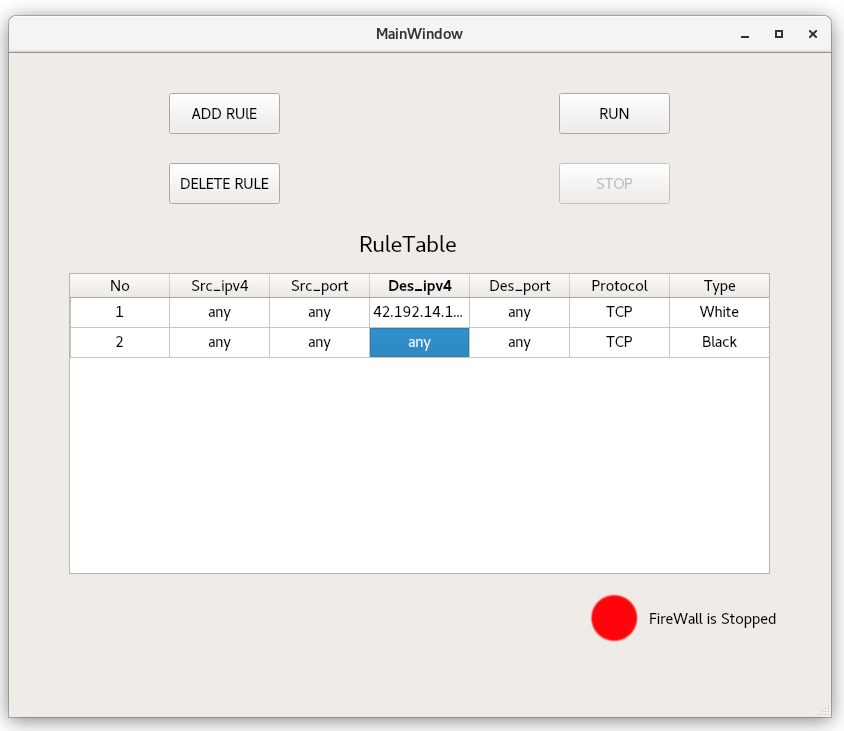
13.关闭防火墙后，ping正常



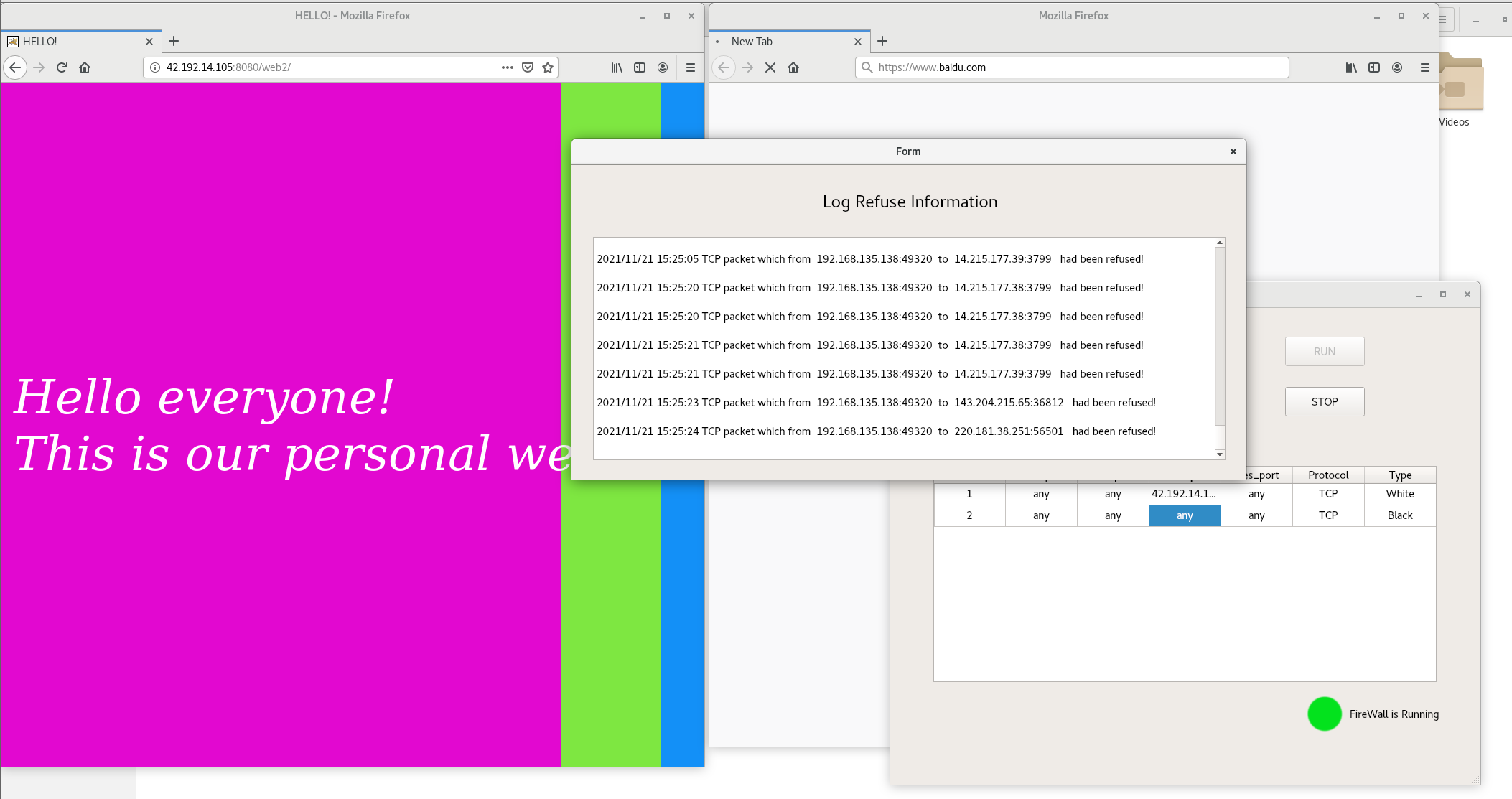
14 测试对UDP报文的拦截，同样还是输入[www.baidu.com](http://www.baidu.com)为例，如图，UDP数据包过滤成功



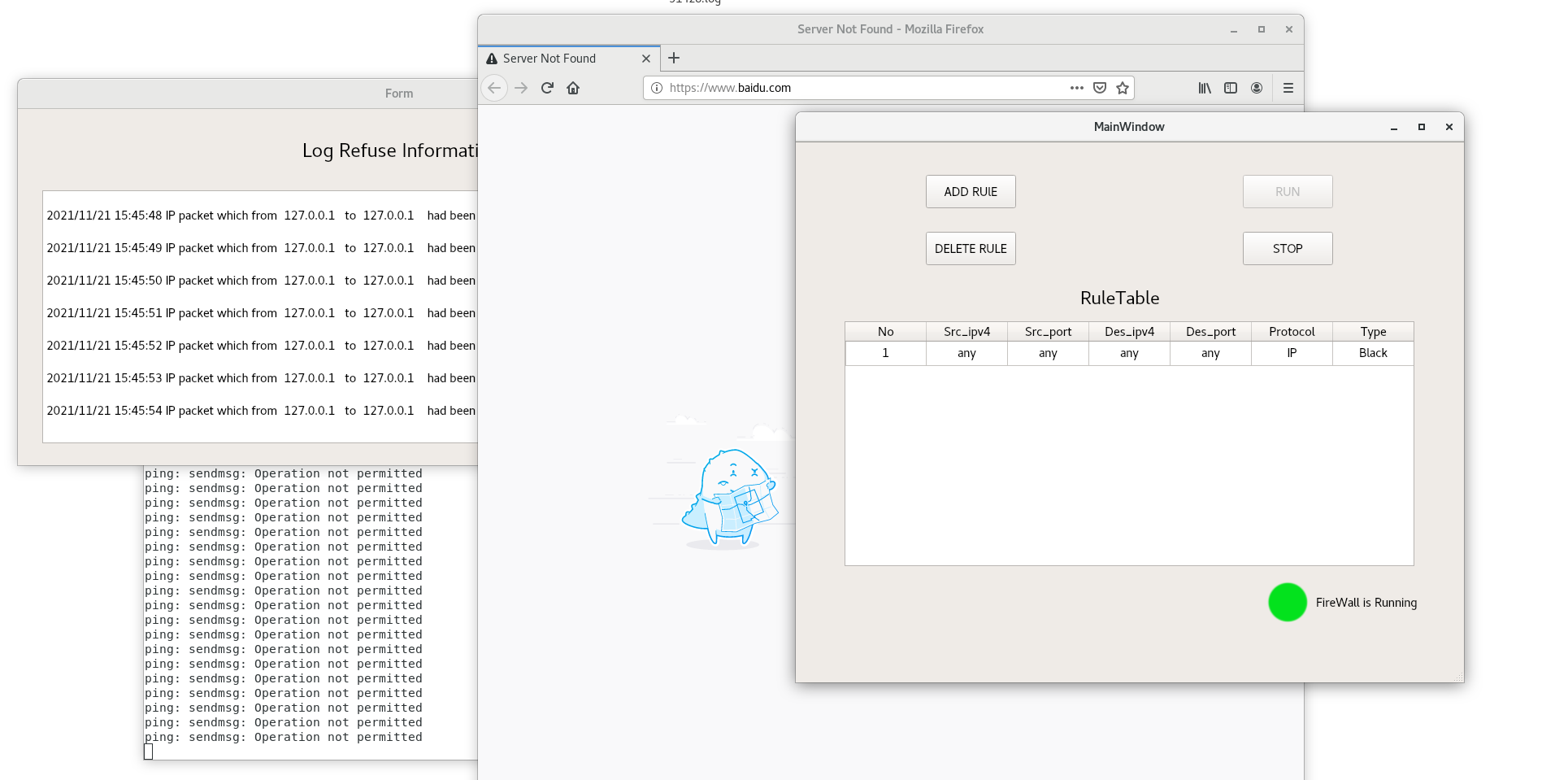
15 测试白名单，规则配置如下，白名单中目的地址设置为42.192.14.105，即云服务器的地址



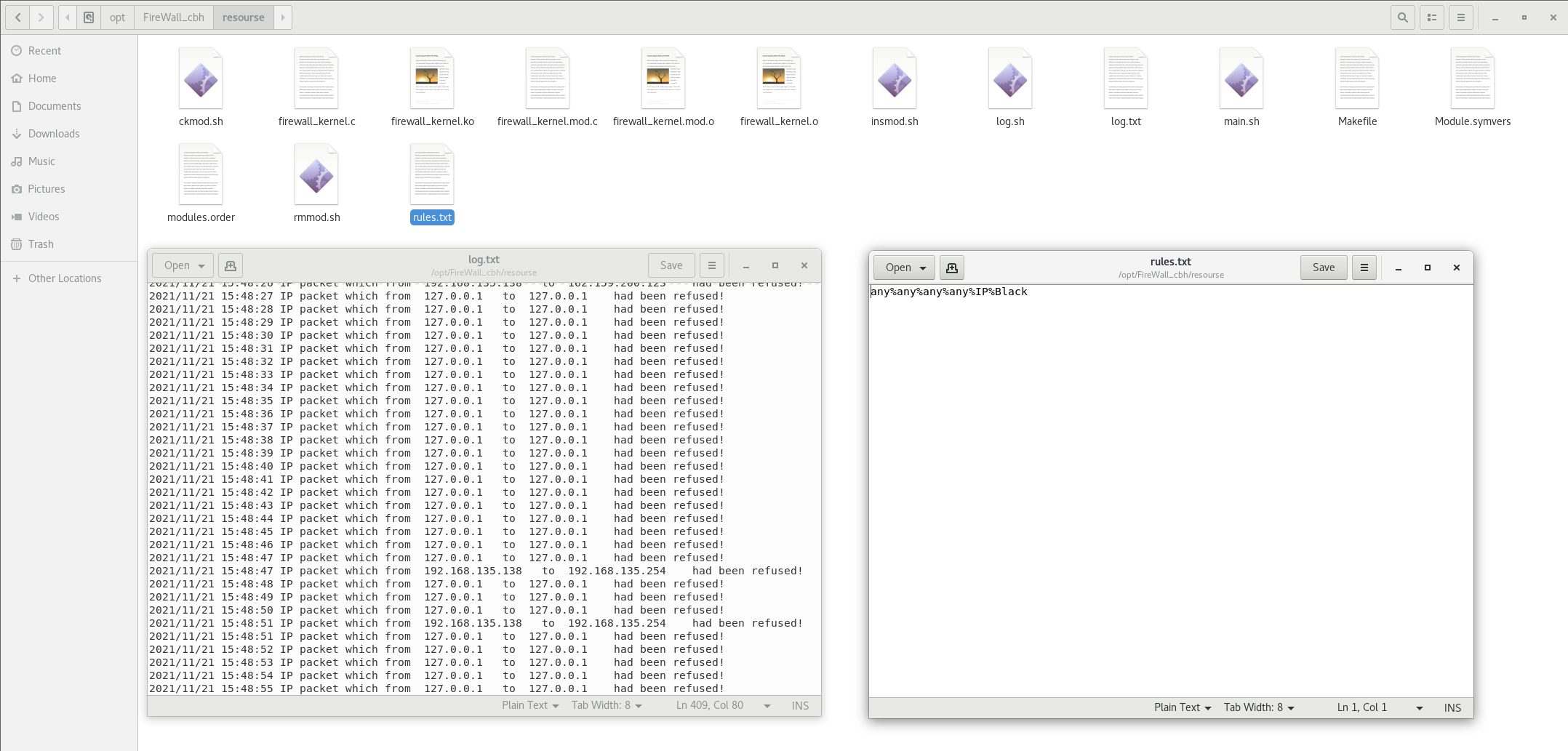
16.运行该防火墙，对比案例同样为百度网站www.baidu.com和部署在云服务器上的网站42.192.14.105:8080/web2，如下图，可以看到，白名单中42.192.14.105可正常访问，而位于黑名单中（any任意处理）[www.baidu.com](http://www.baidu.com)无法访问，一直加载中

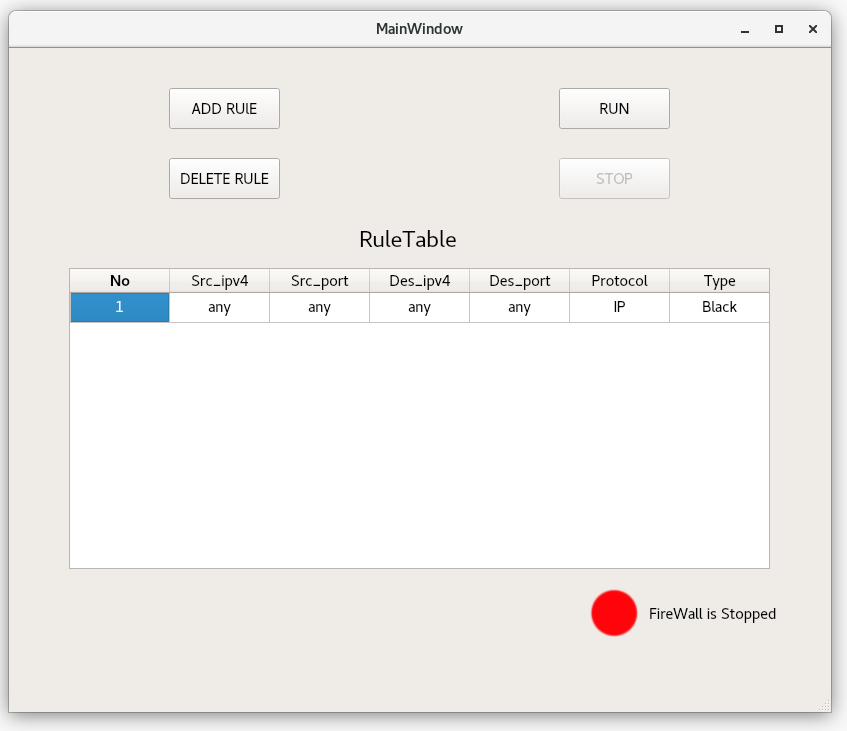


17. 测试拦截IP数据包，该策略会将如TCP，UDP，IGMP，ICMP等协议数据包全部过滤掉。演示如下，此时无论是ping，还是访问网站，均无法做到



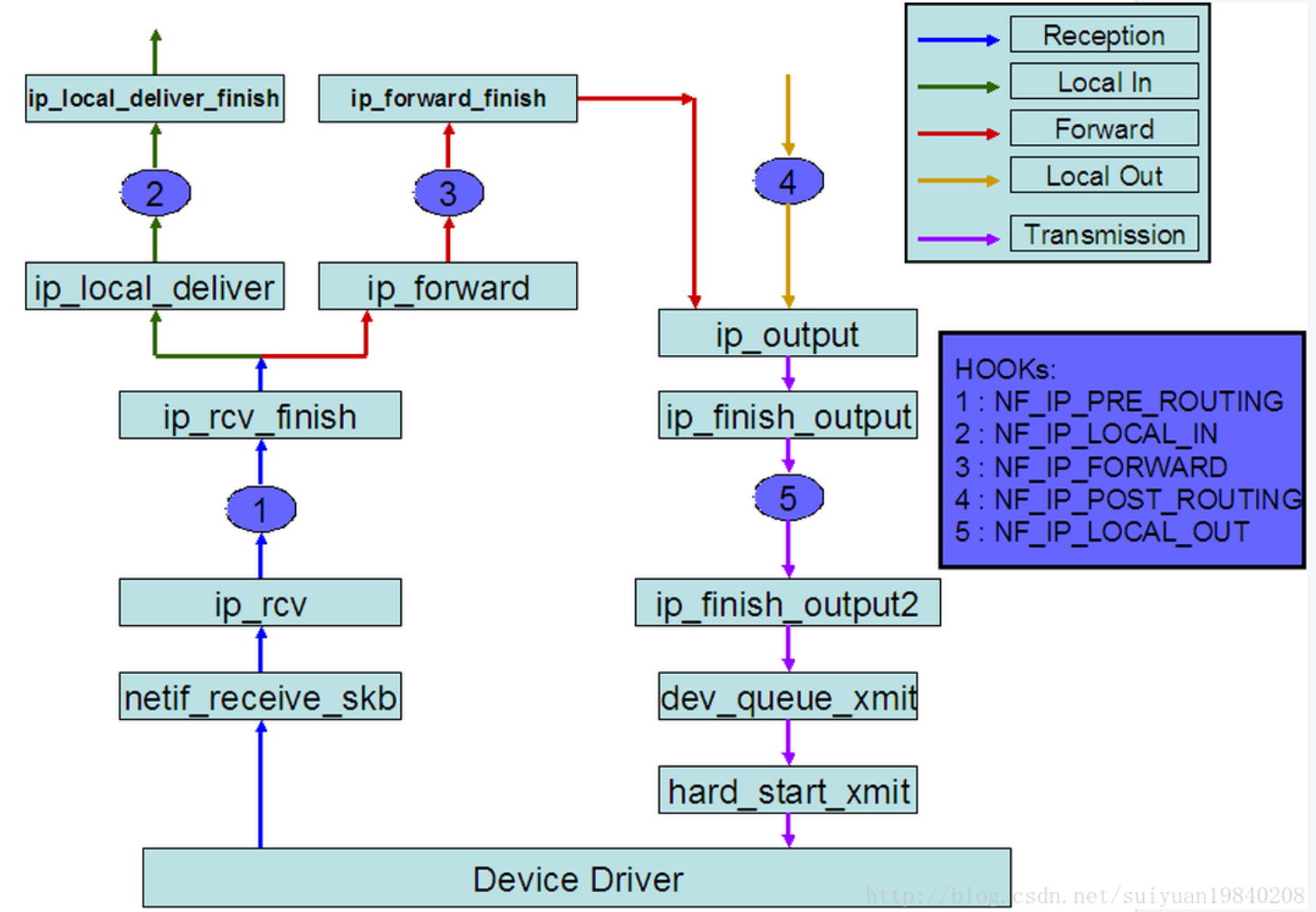
18. 历史记录功能：存放的过滤数据包记录以及存放规则，均在resourse目录下的log.txt和rules.txt中，由于紧接着第16步，故此时rules中仅有上述的一条规则，即过滤所有IP数据包，重启应用后，自动加载历史规则





# 开发及技术介绍

本次选择在Linux平台上开发的主要原因是Linux内核开源且Linux网络编程方面资源丰富便于学习。该防火墙实现的核心的原理是利用netfilter提供的钩子(hook)来进行对数据包的接收或阻拦，在此由于是防火墙，所以选取NF\_INET\_PRE\_ROUTING作为其的hooknum来对其的接收数据包进行组织，由于大多协议均存在应答，经测试选取NF\_INET\_POST\_ROUTING作为其的hooknum也能起到相同的效果。



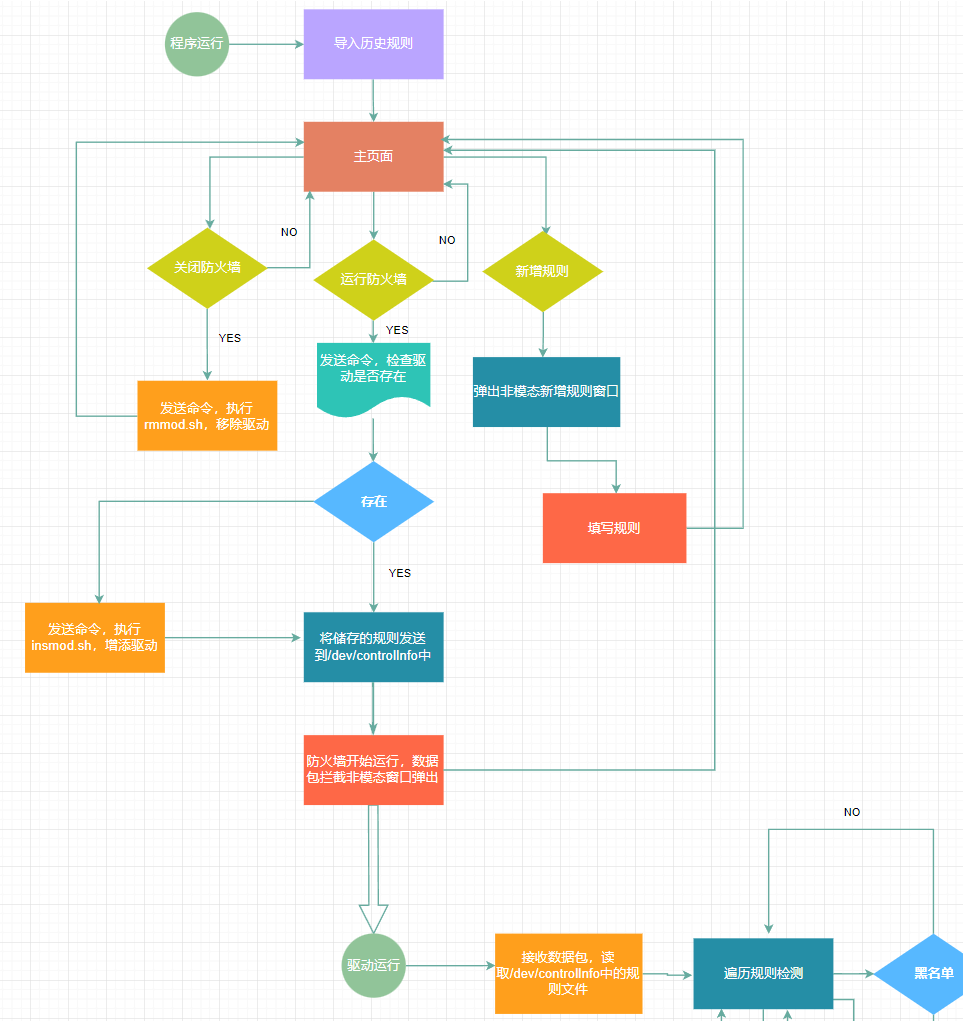
(图转自<https://blog.csdn.net/suiyuan19840208/article/details/19684883>)

关于该hook函数的源码在/resourse/firewall\_kernel.c中，该文件通过编写的Makefile进行make后生成ko驱动文件，使用Qt编写的有图形化页面程序的核心要点是将规则写入dev/controlInfo文件中便于该驱动程序读取规则。

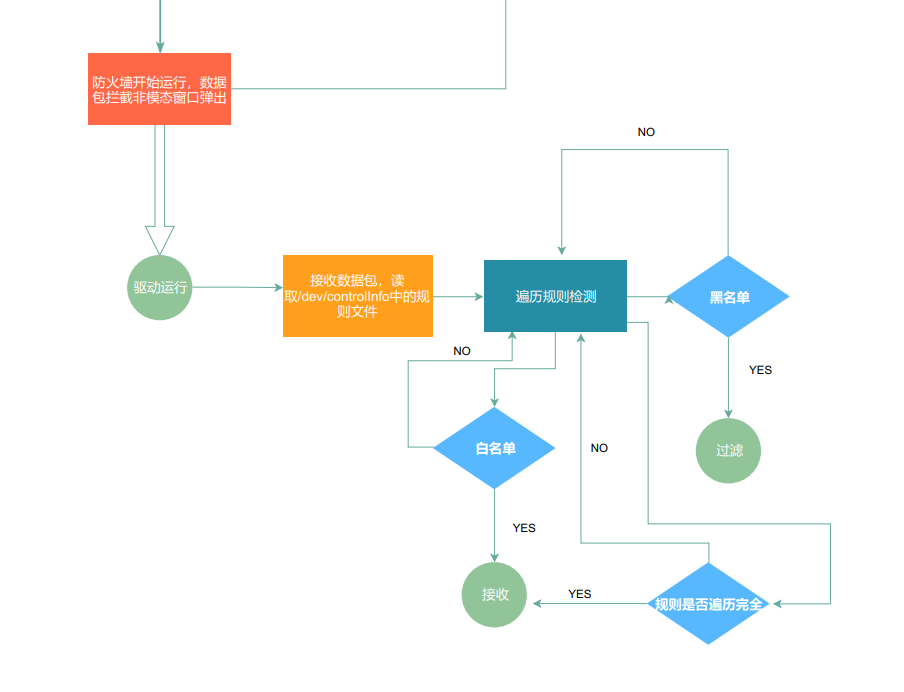
驱动程序（即firewall\_kernel.c生成的firewall\_kernel.ko）中hook\_func函数提取sk\_buff中的IP数据包后，对通过qt里的函数输出在controlInfo里的规则协议进行判断，选取适宜的判断函数对数据包进行选择接收或者丢弃拒绝，从而达成防火墙的目的。

流程图如下

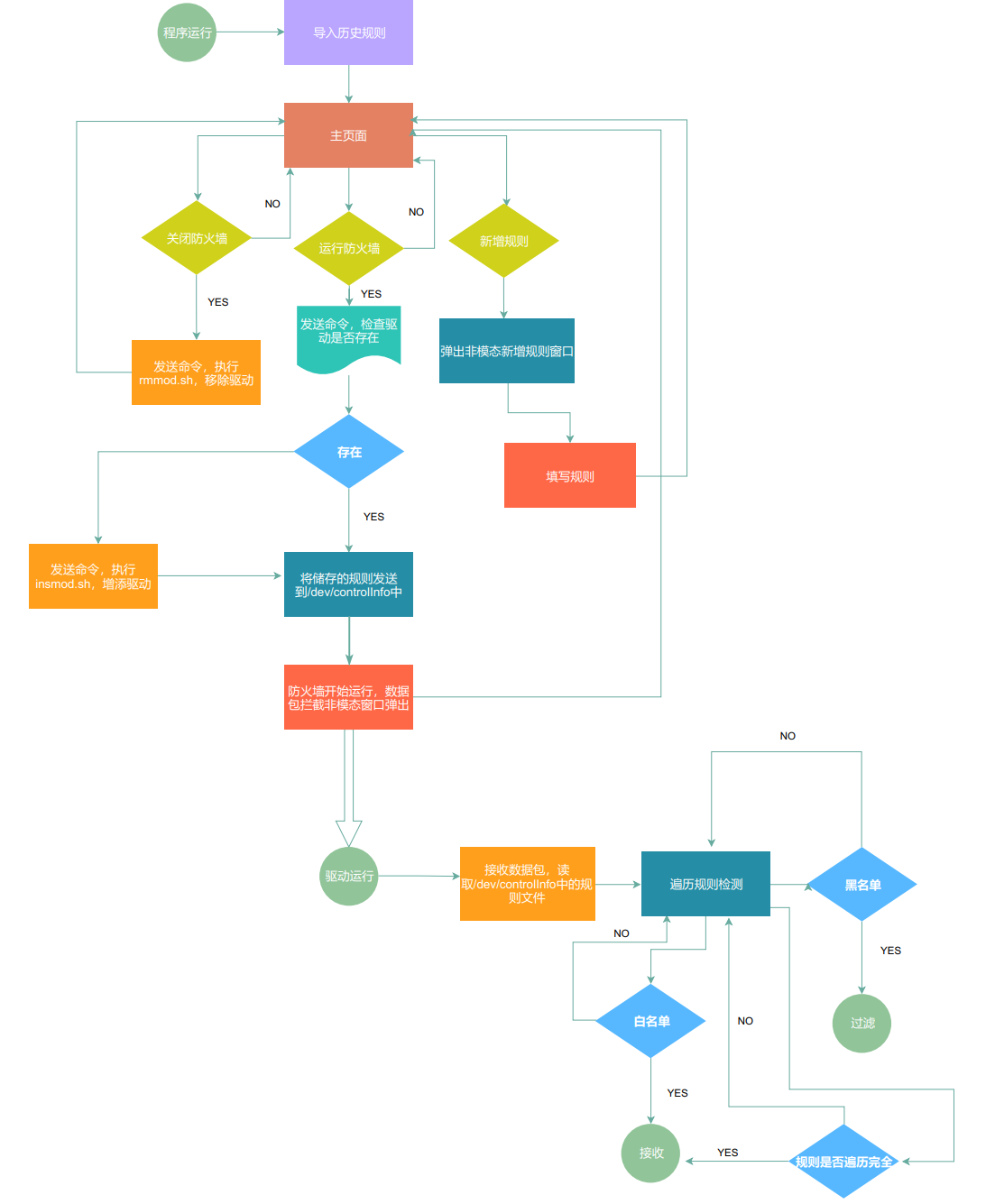
Qt交互逻辑部分



内核驱动部分

、

全流程图（大箭头表示Qt程序于内核驱动程序的交互）



上述仅为防火墙功能实现的简要关键介绍。

下面将对运行过程中的关键函数进行说明，由于代码实现并没有涉及比较难的算法，并且在一些关键点写了注释，故在此仅对函数原型及其功能进行介绍，具体实现请看文件夹resouse和firewall，其中resourse文件夹主要涉及linux内核编程，一些历史存放记录，图片文件，方便运行的sh文件，以及关键的firewall\_kernel.c也在其中；而firewall中的文件，为qt编程的文件，主要是交互逻辑的实现。

**resourse/firewall\_kernel.c**

unsigned int hook\_func(void \* priv,struct sk\_buff \*skb,const struct nf\_hook\_state \* state)

最重要的钩子函数，配合

bool ipaddr\_check(unsigned int saddr, unsigned int daddr)

bool port\_check(unsigned short srcport, unsigned short dstport)

bool icmp\_check(void)

bool tcp\_check(void)

bool udp\_check(void)

bool igmp\_check(void)

bool ip\_check(void)

等函数完成对sk\_buff提取的数据包的接收或丢弃。

static ssize\_t write\_controlinfo(struct file \* fd, const char \_\_user \*buf, size\_t len, loff\_t \*ppos)

读取dev/controlInfo中写入的规则信息放入自己的char数组中，便于hook\_func等函数对防火墙设定规则的读取使用

static int \_\_init initmodule(void)

驱动注册

static void \_\_exit cleanupmodule(void)

驱动清除

**resourse/Makefile**

将firewall\_kernel.c编译生成firewall\_kernel.ko驱动文件

**resourse/insmod.sh** 插入驱动

**resourse/ckmod.sh** 检测驱动是否插入

**resourse/rmmod.sh** 移除驱动

**resourse/log.sh** 根据定义的标签提取hook\_func函数中printk到内核终端的包过滤信息，将这些信息输出到resourse/log,txt中

**firewall/mainwindow.cpp**

void MainWindow::addRuleToTable(Rule rule)

将增添的规则写入表中

void MainWindow::writeRuleToTextFile()

将用户输入的规则写入resourse/rules.txt中

void MainWindow::getRuleFromTextFile()

每当程序启动时，读取resourse/rules.txt中的规则内容，实现读取历史数据的功能

QString MainWindow::runShell(QString cmd)

在qt程序中执行shell命令

void MainWindow::sendRuleToFireWall()

将用户定义的规则写入到/dev/controlInfo中，使得驱动程序读取规则

void MainWindow::ruleFormatChange(Rule ruleString, char \*p\_controlinfo)

将规则转换处理，本质等价形式发生改变，比如定义的规则协议类型为TCP，将其转化成6（其协议序号），便于驱动程序处理，处理ip地址，将QString ip地址转换为一个32位的网络序列ip地址

**firewall/refuse.cpp**

void Refuse::logRefuse()

读取resourse/log,txt中内容输入到窗口中，实现显示阻拦的数据包信息功能

# 心得体会

本次网络编程作业，初始选题准备做VPN，因为感觉做个VPN的话，会很有实际意义，比如在租个在港澳台或者国外的云服务器，将服务端部署在上面就可以实现翻墙上外网。但实操了才发现，VPN不光光仅涉及代理还涉及加密处理，在Github上找的几个项目关于密钥生成部分代码都不是我现在的水平能够看懂的，在仿照一个项目做,尝试代理的实现过程中便放弃了。

由于FTP网上资源多而且之前有过一定的基础，为了保证作业能够如期写完便保底写了个简简单单的单纯命令行页面的FTP课题。写罢感觉还可以再尝试下，感觉防火墙看上去能够实现虽说感觉没有VPN那么有意思，并且这段时间还有一定的空闲精力，开始转战防火墙项目了，事后来看难度还是超出了当初的预估，但还在接受范围内。

由于Windows下的防火墙相关资源太少，找到可供参考的文章、视频、代码等几乎都是Linux操作系统下的，下定决心跳转到Linux环境下进行开发。在此之前，自己关于Linux的经验，也仅仅只有在软件课设上在Linux云服务器上搭网页。

开始是直接用VNC远程连接自己的云服务器开发的，但延迟真的太高，迫不得已在自己电脑上安装了虚拟机，突然感觉还行，延迟差强人意。

Qt上提供了很多网络编程的案例，同时也在Github上找到了几个防火墙项目，一个是单纯的命令行的，一个是用Qt开发的。大概顺着看了5个项目吧，3个是Qt提供的案例，2个是刚在提到的在Github上找到项目，于是乎开始了自己阅读源码仿照整合增添的过程。

主体开发过程耗时一个多星期，初始想拿着Unix网络编程两卷本，TCP/IP详解三卷本这在网上饱受盛名的书，结果发现几乎完全无法在其中提取我想要的能应用在项目的知识。抱着随便翻翻的心态，找到几本挺老的，看着像是小时候报亭卖的书，比如《Linux网络编程》，《鸟哥的Linux私房菜》，跳到我想知道的部分，比如vim编辑器的使用，常见的操作命令以及Linux网络内核编程，出乎意料的收获颇丰。

开发过程中首先是以Qt交互逻辑操作实现进行开发，这个过程很是顺利，因为之前做过两个Qt开发的项目，学到的东西也便只有读写文件，权限赋予，以及在Qt程序内实现向终端发送命令行，加深对槽函数的理解和使用也算吧，毕竟第一次学会在不同窗口间用槽函数发送信息。

最核心也是最困难的部分就是前文里提到的firewall\_kernel.c程序的撰写了，首先最让人头疼的是，阅读参考代码的过程中，我无法像在Windows的IDE那样，看到陌生的函数或者结构体，直接能看它的源码实现。开始是直接在目录下的内核文件找，但是又嫌太慢，直到发现并学会使用SourceInsight这个程序，阅读过程才变得舒服许多。其次的困难便是，写程序的过程中没有以往在IDE上的自动代码提示和补全功能，想要调用一些结构体的函数（比如iphdr，tcphdr）只能去定义那的文件下找以及去网上查；并且过程不会报错，只会在开始编译的时候才会报错，但好在也告诉了错误的地方。

写程序的过程中也感慨C指针的厉害，只要知道了数据结构的实现就能随意修改了，也纠正了学习过程中的一个误区，一直以为TCP包和IP包是两种东西，写程序调错才知道，TCP包是在IP包的数据里面的，IP数据包自己一个头，里面数据部分TCP包自己也还有一个头，因为这改错许久。

整合运行过程中关于权限问题也让人头疼了一会，以往在win下顶多就是以管理员身份运行就行，开始想着按着规范走，但权限问题层出不穷，比如无法写入文件，无法读取文件，无法执行文件…最后写了个开机自动以root用户登录，权限问题一切就都好了。

也曾纠结于内核代码的格式，Makefile的语法和作用机理，数据包在操作系统内核的传递，钩子函数的实现等，现在回想起来感觉自己也许不过多久都会忘了，但好歹也有个印象，未来拾起的时候应该会很快。

除去课程中学到的知识，收获最大的也是学习别人项目源码的过程，学习指令的应用，学习内核代码的写法，也惊讶于其对过程中错误的处理，学会在内核终端抓取错误信息，这是自己之前从未注意的，之前的编程经历都是靠在IDE的控制台的报错纠错。

相较于参考的几个项目，该项目代码更易读，精简了程序结构，关键部分给与注释，结构体和函数的命名更加易懂，增添了UI页面，丰富了协议的选择，增加了白名单的设置。

在本次大作业的完成过程中，加深了对网络通信理解，学会了在Linux系统上软件开发、内核驱动动开发，动手实现了防火墙项目，收获颇丰。

# 参考资料

工程代码参考：

1. <https://github.com/mryuan0428/Firewall-Based-on-Netfilter>

书籍参考：

1.《Linux网络编程》 宋敬彬，孙海滨等编著 清华大学出版社

2.《Unix网络编程 卷一》 （美）史蒂文斯，（美）芬纳，（美）鲁道夫 著 人民邮电出版社

3.《TCP/IP详解 卷一》 （美）凯尔 R.福尔 （美）W.理查德 著 机械工业出版社

4 《鸟哥的Linux私房菜》 鸟哥 著 人民邮电出版社

网页参考：

网页参考资料过多，主要在菜鸟教程，百度，CSDN，StackOverFlow，简书等