摘 要

在网络越来越发达的今天，VPN代理的使用也越来越多。VPN能够使部署在内网中的资源安全地被外部网络终端访问。目前应用广泛的VPN有PPTP、OpenVPN和IPSec VPN等等。由于PPTP的容易部署和使用的特性，它也经常被用作代理服务器。为了保证VPN的安全，或者对VPN的使用进行监控，研究VPN代理的数据获取与解析就非常有必要。

本文实现了一个在Linux环境运行的PPTP VPN代理数据获取与解析工具，针对默认配置下的PPTP服务器，通过分析PPTP的流量，设计了一种能够检测用户上线和下线的方法，完成了对通过VPN的流量的基本解析功能，可以在通过PPTP服务器的流量中寻找HTTP请求和FTP命令，并记录相关的日志，在遇到管理员设定的不安全的HTTP请求时，能够实现即时的邮件报警。系统提供了HTTP请求日志、FTP使用日志和用户登录日志的管理功能，这些功能能够随时被管理员使用。

本系统使用Python开发，运行在Linux环境下，针对PPTP服务器的流量进行捕获和分析。使用了PyQt5框架构建图形界面，用PyQtGraph来绘制图表。使用一个对tShark的Python封装项目PyShark来获取流量并解析流量。使用SQLite3数据库引擎来存储产生的日志。

关键词：VPN；PPTP；流量截获；流量解析

**Abstract**

Nowadays, with the growth of Internet, the usage of VPNs is getting more. VPNs allow clients in the WAN to access a service which is in LAN safely. The most used VPNs are PPTP, OpenVPN and IPSec etc. Because of the attribute that PPTP is eazy to use and set up, it is usually used as a proxy server. To ensure that VPN’s security or to monitor the usage of VPN, it is necessary to do some research on VPN data intercepting and parsing.

This paper introduced a specified PPTP VPN proxy traffic intercepting and parsing tool which runs on Linux, and worked out a method to monitor VPN users’ log in and log out actions by analysising the traffic of PPTP VPN. This tool is able to seek HTTP requests and FTP commands in the traffic and generate associated log. Administrator may set some host domains which is considered insecure, of which the system will send warning email on found. The system provides HTTP log, FTP log and user login log manage function, which allows administrator to use at any time.

The system is developed using Python programming language, it runs only on Linux and only does parsing on PPTP traffic. To build graphic user interface, we make use of PyQt5 framework and draw diagram with PyQtGraph library. PyShark, which is a encapsulation of TShark, is used to intercept and parse traffic. Log is stored by using the SQLite3 DBMS.

**Key words:** VPN; PPTP; Traffic Intercepting; Traffic Parsing

目 录

[1 绪论 1](#_Toc485035816)

[1.1 课题介绍 1](#_Toc485035817)

[1.1.1 VPN代理数据获取与解析研究的背景 2](#_Toc485035818)

[1.1.2 VPN代理数据获取与解析研究的意义 2](#_Toc485035819)

[1.2 国内外发展现状 3](#_Toc485035820)

[1.2.1 国外发展现状 3](#_Toc485035821)

[1.2.2 国内发展现状 3](#_Toc485035822)

[1.3 可行性分析 4](#_Toc485035823)

[1.3.1 技术可行性 4](#_Toc485035824)

[1.3.2 经济可行性 4](#_Toc485035825)

[1.3.3 法律可行性 5](#_Toc485035826)

[2 需求分析 6](#_Toc485035827)

[2.1 系统功能模块需求 6](#_Toc485035828)

[2.1.1 层次方框图 6](#_Toc485035829)

[2.1.2 各模块功能说明 7](#_Toc485035830)

[2.2 模块工作时序图 8](#_Toc485035831)

[2.3 数据流图 9](#_Toc485035832)

[3 技术框架说明 10](#_Toc485035833)

[3.1 Python程序设计语言 10](#_Toc485035834)

[3.2 PyQt 5 10](#_Toc485035835)

[3.3 PyShark 10](#_Toc485035836)

[3.4 PyQtGraph和Numpy 11](#_Toc485035837)

[3.5 SQLite 11](#_Toc485035838)

[3.6 Pony ORM 11](#_Toc485035839)

[3.7 PPTP协议分析 11](#_Toc485035840)

[3.7.1 PPP协议 12](#_Toc485035841)

[3.7.2 MSCHAP Version 2 15](#_Toc485035842)

[3.7.3 IPCP 18](#_Toc485035843)

[3.7.4 MPPE 18](#_Toc485035844)

[4 系统设计 19](#_Toc485035845)

[4.1 概念结构设计 19](#_Toc485035846)

[4.2 逻辑结构设计 20](#_Toc485035847)

[4.2.1 关系表的范式与约束 20](#_Toc485035848)

[4.2.2 Pony ORM关系图 21](#_Toc485035849)

[4.3 软件结构设计 23](#_Toc485035850)

[5 系统实现 24](#_Toc485035851)

[5.1 用户界面实现 24](#_Toc485035852)

[5.1.1 国际化 24](#_Toc485035853)

[5.1.2 主界面 25](#_Toc485035854)

[5.1.3 设置界面 28](#_Toc485035855)

[5.1.4 日志管理界面 29](#_Toc485035856)

[5.1.5 报警网址设置界面 31](#_Toc485035857)

[5.1.6 用户登录日志界面 31](#_Toc485035858)

[5.2 流量截取进程实现 32](#_Toc485035859)

[5.2.1 LogInUser类 33](#_Toc485035860)

[5.2.2 Sniffer类 33](#_Toc485035861)

[5.2.3 PptpUserLoger类 33](#_Toc485035862)

[5.2.4 UserSniffer类 34](#_Toc485035863)

[5.3 PPTP模块 35](#_Toc485035864)

[6 系统测试 36](#_Toc485035865)

[6.1 测试前期准备 36](#_Toc485035866)

[6.2 用户统计和流量统计测试 38](#_Toc485035867)

[6.3 FTP和HTTP日志记录功能 39](#_Toc485035868)

[6.4 设置功能测试 40](#_Toc485035869)

[6.5 日志管理功能测试 42](#_Toc485035870)

[6.6 邮件报警功能测试 42](#_Toc485035871)

[7 结束语 43](#_Toc485035872)

[7.1 主要成果 43](#_Toc485035873)

[7.1.1 完成的功能 43](#_Toc485035874)

[7.1.2 系统特点 43](#_Toc485035875)

[7.2 对系统的展望 44](#_Toc485035876)

[7.2.1 未完成的功能 44](#_Toc485035877)

[7.2.2 系统的缺陷与不足 44](#_Toc485035878)

[谢 辞 45](#_Toc485035879)

[参考文献 46](#_Toc485035880)

# 绪论

当下的互联网技术发展得非常迅速。网络服务器存在于互联网的每一个角落，不断地为各行各业的工作者提供各种各样的服务。在公司中，为了节约网络地址，也为了公司的信息安全，通常在公司的内部搭建一个内网。这个内网与广域网之间隔着一道防火墙一些服务器，如Samba、FTP或者一些WEB服务，运行在这个内网里面，公司内部的人可以方便地访问到这些服务。

但是有可能出现一类情况，即某公司员工在物理上无法访问公司的网络，比如当员工出差的时候。由于防火墙的存在，所以上述的这些服务一般来说是无法从公共网络访问的。如果该员工想要访问这些服务，就要借助VPN（Virtual Private Netword，虚拟私人网络）在混乱的广域网上建立一条从客户端到处于公司内部网络的VPN服务器之间的一条隧道。

PPTP（Point-to-Point Tunneling Protocol，点对点隧道协议）是当前常常被用来解决这样一种问题的一种VPN协议，它可以通过PPP协议和GRE协议建立网络隧道，使用MSCHAPV2协议进行登录验证，通过MPPE协议加密隧道上的数据，从而为用户提供一种安全又稳定的VPN服务。随着这种服务的数量的增长，就出现了对服务器的访问进行监控的需求。

PPTP十分容易配置。在Linux操作系统上，已经有十分成熟的PPTP服务器开源项目poptop。在现在还在提供支持的Windows操作系统的所有版本上，都继承了PPTP客户端的功能，通过简单的设置，即可连接到一个PPTP服务器。PPTP由于上述的易于配置和易于使用的特性，许多人将它部署在Linux操作系统的VPS（Virtual Personal Server，虚拟私人主机）上，然后通过Linux自带的防火墙服务iptables配置转发，来把它用作一个代理服务器。由于它的保密性，它还经常被用作从中国境内突破互联网封锁的工具。

本文通过分析PPTP协议的流程，给出了一种独立于PPTP服务器之外的数据捕获与解析的方法，使用PyQt5和Python 3.5开发，使用pyShark来获取和解析流量，用SQLite3数据库来储存数据，完成了一个能够对用户的流量以及访问记录进行监控的应用程序。

## 课题介绍

VPN 代理，是在 VPN（Virtual Private Network）虚拟专用网络的基础上衍生出来 的提高网络访问速度和安全的技术。它利用 VPN 的特殊加密通讯协议在因特网位于不同 地方的两个结点间临时建立一条穿过混乱公用网络的安全稳定的专用隧道。所有的 VPN 均应保证通过公用网络平台传输数据的专用性和安全性。

为了保证 VPN 代理的安全，通过对 VPN 的流量进行捕获和分析，可以了解 VPN 的使 用情况，流量来源和去路。在发现危险流量时，就可能及时地对相应的情况作出响应， 从而避免更大的损失。

本课题旨在研究 VPN 通道上数据的安全特性，并通过技术手段实现对数据的捕获， 同时能对数据进行初步的解析。

### VPN代理数据获取与解析研究的背景

在互联网蓬勃发展的今天，计算机网络的结构越来越复杂，规模也越来越庞大。许多商业公司为了互联网服务的稳定性和安全性，会选择将服务器架设在属于公司内部的局域网中。当员工需要在无法物理接触局域网的情况下访问处于局域网中的服务器时，就要借助VPN技术在混乱的广域网中建立一条安全、稳定和快速的隧道。VPN这样的服务为上述情况提供了一个安全而又方便的解决方案，但是VPN这样一种从局域网内部向广域网开辟访问路径的服务，很容易就会成为不法分子入侵到局域网内部的首要入口。即便是对VPN服务的合法使用，也难以保证员工的操作是完全安全的。

另外，国家对境外一些不法分子建立的网站，使用“防火长城”来封锁了中国境内对这些网站的访问。但是“防火长城”有时候会将一些合法的境外网站也“误杀”掉。基于这个原因，许多了解计算机网络相关技术的人就在境外假设VPN服务器，通过使用VPN服务器来访问这些被“误杀”的网站。但是同时也会有一些不法分子在国内散播非法网站的URL，并提供VPN服务器让境内的网民访问。这些不法分子一般是处于政治目的或者怀有一些经济犯罪的目的，向网民提供VPN服务。对这些不法的VPN服务器进行监控，对国家来说非常有必要。

### VPN代理数据获取与解析研究的意义

在上述背景下，开发一个用来监控VPN服务器运行状态和VPN使用者行为的软件就非常有必要。有了这样一个技术，若发现通过VPN的流量有异常，就可以及时地采取响应的措施，以免造成更大的损失。

本文尝试提出一种方法来将PPTP服务中的流量与其使用者账号对应起来的方法，并做出一个简单的实现。基于1.1.1中提到的一些不法分子非法传播VPN服务和非法境外网站行为，国家应该会做出应对的措施。不仅仅要将不法分子绳之以法，也要对非法使用VPN代理的人进行教育。如果这些使用VPN代理的人还通过这些做了一些违法犯罪的事，那么也要将他们绳之以法。为了找到这些人的真实身份，就需要对VPN的协议进行研究，尝试将流量与使用者对应起来。本文的工作对这样的研究也会起到一些积极的作用。

## 国内外发展现状

### 国外发展现状

随着互联网的规模日益壮大，信息安全技术相关的研究也变得更多了。目前在知名源代码托管网站上已经出现了许多MITM（Man-In-The-Middle，中间人攻击）的开源项目。

以最著名的MITMf（MITM framework）来说，这个框架的主要功能是通过在局域网中不断广播ARP欺骗的数据包进行ARP欺骗行为，让局域网中的其他逐句误以为发出广播的主机是网关。接下来，框架会模拟网关的行为让其他的计算机能够正常地访问网络，但是同时又截获来自其他主机的流量。这个框架实现了一些数据解析功能和对流量的篡改功能，例如它可以读取HTTP流量中的图片，可以对HTTP下载请求进行拦截，并将用户要下载的文件替换为更改过的文件来达到一些其他的目的。MITMf甚至还实现了ssltrip功能，这个功能可以通过伪装HTTPS服务器，同时转发流量，来达到窥探通过SSL加密的请求与响应的目的。如果用户在访问网络时，没有注意到浏览器地址栏中HTTPS证书异常的提示，那么就会在不知不觉中被攻击者获取机密的信息和个人隐私。

但是这些项目都不是针对VPN服务器的。它们没有将VPN服务中的流量与用户相对应的功能，也没有统计VPN流量的功能。

### 国内发展现状

根据项目前期的调查，目前国内已经有一些对VPN流量分析的研究工作论文。但是在这些论文中鲜有针对PPTP进行的研究。PPTP的配置简单，稳定性好，可以说是当下最著名也是最流行的VPN代理协议。可以说对PPTP协议的研究工作是势在必行的。

## 可行性分析

### 技术可行性

VPN服务器与客户端之间的流量是加密的，也就是说，不论在服务器与客户端之间的什么位置以什么样的方式来抓取流量，数据都是无法被解析的，至少是无法实现实时解析的。本文在研究过程中，提出了集中实现系统的思路：

1. 通过研究PPTP的协议，了解其加密流程，通过暴力破解密码来达到获取流量明文的目的。这种方法显然不现实，如果要实现对VPN服务器使用情况的实时监控，就需要一种快速的方法来获取流量的明文数据。但是暴力破解操作需要耗费大量的时间，所以不采用这个方法；
2. 通过对已有的PPTP VPN服务器的开源项目进行改造，将流量和对应的用户信息转发出来，再在另一个进程中进行解析操作。如果使用这种方法，那么将要耗费大量的时间来阅读开源项目的源代码来熟悉项目的架构，而且已有的项目的结构不一定适合扩展，所以也不采用这个方法；
3. 在PPTP VPN客户端与服务端之间，流量是被加密的，但是在流量经过服务端后，就已经是明文的数据了。只要在服务端上，抓取网卡上的流量即为明文，这种方法的问题在于，无法将流量与用户一一对应起来，因为在经过PPTP服务器后，流量中就不再含有用户的识别信息。而相应的解决方法，将在接下来提出。本文采用的就是这种方法。

在研究PPTP协议之后，发现在PPTP的登录流程中，有一些数据包包含了未加密的用户名明文，通过分析这些数据包，可以将用户名与流量中的IP地址一一对应，从而实现监控功能。

综上所述，该项目具有技术可行性。

### 经济可行性

目前的计算机网络规模越来越大，结构越来越复杂。很多公司也在内网中假设了私有的服务器。要通过复杂的网络环境来访问这些服务就要使用VPN技术，从而就有了对VPN监控软件的需求。

要开发这样一个软件，只需要一台配置足够的个人计算机，要求可以运行Vmware虚拟机软件，并至少能同时能运行两个虚拟机实例，和一个路由器即可。

综上所述，本项目具有经济可行性。

### 法律可行性

在网关上抓取数据并分析是具有两面性的。

如果是在公共网关上抓取流量，并分析其中的数据，那么这种行为无疑侵犯了该网络访问者的隐私权。如果将流量中的一些重要信息，例如银行卡卡号和密码、手机号和邮箱等等，以非法牟利的方式使用，在获利金额超过一定数目的情况下，还有可能触犯刑法。

但是如果是公司对公司内部VPN网关上的流量进行合理的监控，例如获取VPN流量负载，监控员工是否访问了非法的主机，记录终端是否有可能是危险的操作，那么就是合法的。

综上所述，本项目具有法律可行性。

# 需求分析

本章对软件系统进行需求分析，建立软件工程的模型。

## 系统功能模块需求

本节将系统的功能分为不同的模块，建立系统的层次方框图，并对各个模块的功能进行说明。

### 层次方框图



图 2‑1 系统功能模块图

在图 2‑1中列出了系统的功能模块，包括数据展示与分析、数据捕获与解析和系统设置。数据展示与分析包括日志管理模块，数据显示模块数据捕获与解析包括用户探测模块和数据解析模块。系统设置包括一般设置模块和监控规则设置模块。

### 各模块功能说明

1. 日志管理模块：日志管理模块提供了对日志的检索和删除功能。
2. 数据展示模块：数据展示模块提供了将在VPN服务器运行过程中被记录下的数据展示出来的功能。展示方法主要有图表展示和表格展示两种。
3. 数据捕获模块：数据捕获模块将数据从网卡中捕获出来，并交由数据分析模块和用户探测模块处理。
4. 用户探测模块：在流量中寻找用户登录时产生的MSCHAPV2（见3.7.2）数据包，并通过数据包中的信息探测用户的上线和下线行为，为流量与用户之间的关系判断提供依据。
5. 数据解析模块：数据解析模块将通过VPN服务器后的明文流量通过pyshark解析后，记录下结果。
6. 一般设置模块：一般设置模块提供对流量截取的目标网卡、报警E-Mail信息和邮件服务器地址以及账号密码进行设置的功能。
7. 监控规则设置模块：监控规则设置模块提供对日志记录、警报发送的规则进行设置的功能。

## 模块工作时序图



图 2‑2 模块工作时序图

图 2‑2展示了系统工作时，各模块互相合作的时序。

管理员发送开始捕获命令至数据包捕获模块之后，数据包捕获模块开始执行。数据包捕获模块捕获到数据包之后，就会将数据包交给用户探测模块和数据解析模块处理。在处理完毕后，两个模块将结果返回，并写入数据库。当管理员想查看日志时，应用程序就从数据库中获取已经储存的日志项目并展示在用户界面上。

## 数据流图



图 2‑3 数据流图

图 2‑3是本项目的数据流图。网络流量最开始是由PPTP服务器用户产生的，用户登录PPTP服务器后，访问互联网时，网络流量会通过PPTP客户端软件。此时流量会被加密，被加密后的流量会通过隧道发送至PPTP服务器。服务器解密后，会在服务器的对应PPP虚拟网卡上产生相应的明文流量，此时明文流量会被数据截获模块截取。被截取的流量会送入数据解析模块。数据解析模块在解析完数据之后，会将解析结果发送至用户界面模块，此时用户会将解析结果转换成数据库对象并存入数据库中，同时将解析结果通过日志表单的形式显示在图形界面上，使得PPTP服务器管理员能够查看。

# 技术框架说明

## Python程序设计语言

Python语言是在1980年吉多·范罗苏姆创建的面向对象的解释性语言。它用C语言编写，是一种胶水语言。因为使用C语言编写，它的执行效率非常高。因为它是一种胶水语言，内部直接提供了ctypes库来兼容C/C++中的类型和接口，而且其基本数据类型和类与对象的实现机制在底层也是用C语言实现的，所以它可以与所有的C/C++编写的库良好地交互。在Python 3中，还提供了修饰器、列表推导等等优秀的语法糖，让开发效率极大地提高了。

基于以上的原因，本项目选择使用Python语言编写。

## PyQt 5

PyQt5是Qt5在Python中的封装。使用Qt提供的Designer软件可以方便地设计图形界面，并保存为ui文件。ui文件是以XML形式表示图形界面的代码文件，使用pyuic5可以将ui文件编译至Python代码。通过继承生成的类，就可以对用户图形界面中的各种事件和信号进行处理。

另外，PyQt5是一个完整的Qt5封装。也就是说，所有在Qt5中可以使用的特性和组件都可以在PyQt5中找到封装。包括其Socket、Timer和Thread以及并发锁机制的封装，都能使用。这使得拥有Qt5开发经验的人可以无缝地转换到PyQt5。

基于以上的原因，而笔者也有Qt5的开发经验，所以本项目选择PyQt5。

## PyShark

PyShark是tShark在Python中的封装。tshark是著名抓包软件Wireshark的命令行版本。PyShark封装了数据包、网络分层和抓包实例等对象。

抓包实例对象可以读取已存在的抓包文件或者进行实时的抓包。不仅如此，抓包实例对象还实现了迭代器，可以通过for循环直接遍历抓到的数据包。实时抓包的抓包实例对象实现了环状缓冲区，只要将环状缓冲区的长度设置成合适的值，在实时抓包时就无需担心内存被占满。

在获取到数据包后，可以直接通过下标运算符或者获取成员运算符来查看数据包根据网络分层解析的结果，在各个网络层对象中，还能直接通过指定字段名来获取字段值。

## PyQtGraph和Numpy

PyQtGraph 是一个用于 python 的图形和用户界面库, 它提供了工程和科学应用中通常需要的功能。其主要目标是提供快速的用于显示数据的可交互式组件，和在Qt中进行快速应用开发的工具。它重度依赖于Qt平台和numpy，提供了对繁重数学运算的高性能解决方案。特别地，PyQtGraph使用了Qt的GraphView框架，这个框架本身就是一个拥有强大能力的框架，将优化的而又简洁的原语带入这个框架中，让使用者能够以最小的代价将数据可视化。

Numpy是一个Python中的数学库，它提供了线性代数中的许多运算和矩阵的表示方法，可以方便地对向量进行数学运算。

## SQLite

SQLite是世界上被使用得最广泛的SQL数据库引擎。在Python中，无需安装任何其他的附加软件包或者组件，也无需经过任何的配置，就可以使用SQLite3数据库引擎。这个数据库引擎使用C语言编写，而且十分的小巧，经常被用在许多嵌入式设备中。

本项目对数据库系统的需求不大，主要是用来储存产生的日志，对象之间的关系也十分简单，但是又有一定的关系，所以选择支持SQL又无需繁琐配置的SQLite3数据库引擎是最好的。

## Pony ORM

ORM是Object Relational Mappin（对象关系映射）是一种被广泛使用在应用软件开发中的技术。它的主要功能是建立数据库中的关系表与软件系统中的对象的映射，使得在进行软件开发的时候可以方便地获取持久化对象，而无需关心底层数据库的实现细节与查询方式。在Python中，最常用的ORM是SQLAlchemy。

Pony ORM是一个高级的对象关系映射器。在使用Python这种高级的面向对象程序设计语言来编写软件系统时，使用对象的方式来访问持久化储存在数据库内部的数据就会把开发工作变得更加简单。Pony ORM在操作原语的设计上十分的优秀，它完美地结合了Python中的列表推导和Lambda表达式，使得查询的语句变得十分的优美和高效。

## PPTP协议分析

PPTP协议使得现有的网络接入服务器的功能可以以客户端-服务端的架构分离开来。PPTP被建立且仅被建立于PAC与PNS之间，其它的网络设备或应用程序都不应该知晓其存在。PPTP使用扩展的PPP数据包和GRE数据包来提供具有底层的拥塞控制和流控制的隧道，这使得PPTP可以有效地利用带宽并且避免无意义的重传[10]。

现在的PPTP服务器一般采用MPPE 128bits加密算法来加密隧道中的流量，在登录时使用MSCHAP Version 2来实现安全的挑战登录验证。本文就以这一种情况为例，设计了针对以这种方式部署的PPTP服务器的流量监控软件。因此在这一章中，要着重分析MSCHAP Version 2、用于建立隧道的PPP协议中的LCP部分和NCP中进行网络层IP地址协商的IPCP协议。

PPTP的功能被分离到C/S架构中，客户端被称为PAC（PPTP Access Concentrator），服务端被称为PNS（PPTP Network Server）。在本章中的其它部分，将会多次使用这两个术语。[4]

PPTP在连接建立时，会开启1723端口，使用基于TCP的控制协议来维护隧道。网络流量在通过网络层时，被加上PPP报头和GRE报头，使用IP数据报来传输。当主机收到一个PPTP数据报文时，首先去掉GRE头和PPP头，就获取了PPTP内网中的网络层报文。自此就可以将这个报文通过一般的协议栈进行解析。

表 3-1 PPTP数据报文结构

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IP | GRE | PPP | IP | PAYLOAD |

在PPTP连接建立后，对每一个终端，在PPTP服务器上都会有一个对应的虚拟网络接口卡。这个接口卡使用的IP地址是在协商建立连接时使用的地址。通过分析PPTP在建立PPP隧道时的数据包，就可以获取PPP虚拟网卡的地址，同时还能获取用户的用户名，并建立一一对应的关系。通过这个方法，本文得以实现一个不使用root权限就能区分用户流量的监控系统。

### PPP协议

PPP协议为在点对点连接之间传输不同协议的数据包提供了一种标准化的方法。它通过扩展LCP（Link Control Protocol，链路控制协议）和一系列NCP（Network Control Protocol，网络控制协议）来建立并配置处于不同网络层次的协议[5]。PPP协议的功能由三个主要的部分组成：

1. 在一系列链路上将数据报封装起来的功能；
2. 一个用于建立、配置和测试网络链路的协议：LCP（Link Control Protocol，链路控制协议）；
3. 一系列用于配置处于不同计算机网络层次的协议：NCP（Network Control Protocol，网络控制协议）。

本章将对这三个部分中的一些要点进行说明。

LCP用于建立、拆除和控制数据链路并且协商链路参数，认证方法、压缩方法等配置选项都由它管理。

LCP数据包分为三类，每一类有几种不同的数据包，关于数据包种类及其描述的说明，见表 3-2。因为链路维护数据包在本项目中未使用到，所以在表中省略了相关的信息，若有需要可以查看PPP的RFC文档。

表 3-2 LCP数据包类别与作用[1]

|  |  |  |  |
| --- | --- | --- | --- |
| 数据包种类 | 数据包名称 | 数据包作用 | |
| 链路配置数据包 | Configure-Request | 发送配置请求 | 用于在建立链路时进行配置协商 |
| Configure-Ack | 确认配置请求 |
| Configure-Nak | 请求修改配置 |
| Configure-Reject |
| 链路终止数据包 | Terminate-Request | 请求终止链路 | 用于终止链路 |
| Terminate-Ack | 确认终止链路 |
| 链路维护数据包 | 略 | 略 | |

链路配置数据包包含四种类型的数据包，用来协商链路的配置。下面将用一个例子来解释LCP在数据链路建立时的行为。

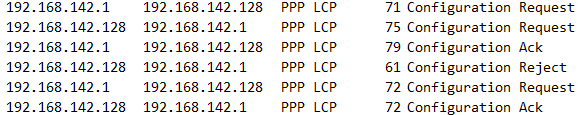


图 3‑1 LCP建立链路时Wireshark的抓包



图 3‑2 LCP工作的时序图

图 3‑1是通过Windows的VPN客户端功能连接运行在虚拟机中的PPTP服务器时，使用WireShark软件抓包的情况。为了简介明了，在此将图 3‑1中发生的数据包交换过程在图 3‑2中用时序图的方式表示。可以清晰地从中看到链路建立的协商过程：

1. 首先由客户端发送配置请求至服务器；
2. 服务器也发送自己的配置请求至客户端；
3. 客户端接受并确认了服务端的请求；
4. 服务端发现在来自客户端的请求中有一些无法识别或者错误的配置选项并把它封装在一个Configure-Reject中，返回客户端；
5. 客户端在修改了配置选项之后，重新向服务端发送配置请求；
6. 服务端接受了新的配置请求。

需要说明的是Configure-Reject和Configure-Nak两中数据包都是用来拒绝请求的，但是它们之间有一些不同。Configure-Reject是用来拒绝那些无法识别的选项的。如果所有的选项都可以被识别，但是有一些选项的值无法被接受，那么就要发送Configure-Nak来请求对方修改选项的值。



图 3‑3 链路终止时Wireshark的抓包

Terminate-Request由主动断开连接的一方发送至对方。收到Terminate-Request的一方必须回应一个Terminate-Ack。一旦收到了Terminate-Ack，表明链路已经处于关闭或者停止的状态，或者是处于需要重新协商的状态。

### MSCHAP Version 2

MSCHAP的全称是Microsoft PPP CHAP Extensions，CHAP指的是Challenge-Handshake Authentication Protocol，即询问握手协议，MSCHAP是微软对CHAP的扩展，而MSCHAP Version 2是其第二个版本。两者之间有一些差别，在表 3-3中列出。

表 3-3 MSCHAP Version1与Version2的区别[3]

| MSCHAP Version 1 | MSCHAP Version 2 |
| --- | --- |
| 在协商CHAP算法时，Algorithm字段值为0x80 | 在协商CHAP算法时，Algorithm字段值为0x81 |
| 服务端发送的挑战随机数长度为8字节 | 服务端发送长度为16字节的值至客户端，客户端使用这值来生成长度为8字节的挑战随机数 |
| 客户端对8字节的挑战随机数响应的是24字节的LANMAN数值和24字节的NT-Response随机数值 | 客户端生成一个长度为16字节的对等挑战随机数，这个随机数被用来生成8字节的挑战随机数和24字节的NT-Response |
| 服务端发送SUCCESS状态或FAILURE状态的响应 | 服务端发送状态为SUCCESS的响应，或者发送一个状态为FAILURE的响应，并捎带一个16字节的对等挑战随机数 |
| 客户端根据上述的SUCCESS状态或FAILURE状态的响应决定是继续还是结束连接 | 客户端根据上述的SUCCESS状态或FAILURE状态的响应决定是继续还是结束连接。另外，客户端将检查服务端的响应，如果返回值不正确，将断开连接 |

在LCP建立链路的过程中，将Algorithm字段设置为0x81说明请求在登录流程中使用MSCHAP Version 2协议。如果另一方不支持MSCHAP Version 2，但是正确地实现了PPP LCP中的Configure-Reject，那么在遭遇这个非标准的选项时，不会遇到任何问题。

MSCHAP Version 2含有四种数据包，接下来一一进行说明。

挑战数据包由长度字段、挑战随机数字段和名称字段组成。

挑战登录随机数是一个由服务器生成的随机数，长度是16个字节。因此长度字段的值应该为0x10。名称字段目前未被使用。



图 3‑4 NT-Response的生成

响应数据包由以下几个字段组成：

1. 16个字节：对等挑战随机数（Peer-Challenge）
2. 8个字节：保留字节，必须为0
3. 24个字节：NT响应（NT-Response）
4. 1个字节：标志位

第一个字段是由客户端生成的另一个随机数。这个随机数将在计算NT-Response的过程中被使用。生成NT-Response的流程如图 3‑4所示。

验证成功后，服务端将会发送登录成功的反馈。反馈信息由两部分组成。第一部分是20字节ASCII编码的16进制认证字符串，这个字符串由上文所述的一系列信息计算得出，16进制中的字母必须是大写。

当客户端收到登录成功的数据包后，就要检验认证字符串的有效性。若认证字符串有效，那么就可以认为登录成功，且登录完成。



图 3‑5 计算认证字符串步骤

在验证登录成功数据包中的认证字符串时，需要使用官方文档中提到的Magic块[1]，通过一系列哈希算法来计算出一个认证字符串，具体步骤见图 3‑5。若认证字符串中的值与数据包中的值相等，则认为登录成功。

登录失败数据包的格式与标准的CHAP登录失败数据包格式相同。数据包格式如下：

"E=eeeeeeeeee R=r C=cccccccccccccccccccccccccccccccc V=vvvvvvvvvv M=<msg>"

第一个字段E，是一个以十进制表示的错误编号（不一定是10位），用ASCII编码。第二个字段R，是一个标志字节，以ASCII表示，意义为是否允许重试，若允许重试，则为1，否则为0。

第三个字段C，是一个长度为16字节的挑战值，以十六进制表示，用ASCII编码。如表 3-3描述的那样，此字段只在MSCHAPV2中存在。当客户端的认证失败时，服务端将这个值捎带回客户端。客户端会验证这个值的正确性，以保证这个数据包是从服务端发送过来的。如果验证失败，那么有可能是遭到了中间人攻击，即这个数据包是由链路上的第三方发送的，这时候客户端会断开连接。

第四个字段V，是一个以ASCII编码表示的10进制版本号（不一定是10位），表示的是服务器支持的密码更改协议版本。对于MSCHAP Version2，这个值应该永远为3。

第五个字段M，是一个可读的以相应字符编码来编码的字符串信息。

### IPCP

IPCP（Internet Protocol Control Protocol，因特网协议控制协议）是NCP中的一种。在结束登录验证流程后，若验证通过，就进入NCP的工作流程。IPCP用于在验证结束后协商管道两端的IP地址。

IPCP数据包的分组与LCP相似。也有Configure-Request、Configure-Ack、Configure-Nak、Configure-Reject、Terminate-Request和Terminate-Ack等数据包分组，但是跟LCP中的字段有区别。

### MPPE

MPPE协议在PPTP中被用来加密管道中的数据包。MPPE使用RSA RC4方法加密数据。用来初始化加密矩阵的会话密钥长度是可以协商的。MPPE目前支持40位、56位和128位的会话密钥长度[2]。MPPE会话密钥会不断变化以保证传输过程中的加密安全性，其变化的频率是基于协商结果的，甚至可以对每一个数据包都有不同的会话密钥[2]。

如果协商结果为使用无状态的加密，则每一次数据包序号变化时，会话密钥都要随之变化。在无状态模式下，发送者必须在每一个数据包被加密和发送之前更改它的密钥；而接收者必须在收到每一个数据包之后，解密数据包之前更改密钥。

如果协商结果是使用有状态的加密，那么发送方则必须对任何数据包序号的低字节为0xFF的数据包在发送和加密之前，更改它的密钥；而接收方则必须在收到一个“标志数据包”之后，在解密它之前，更改自己的密钥。

# 系统设计

## 概念结构设计



图 4‑1 E-R图

本项目中数据库系统的作用主要是用来储存分析用户的流量后产生的日志。设计的数据库E-R图如图 4‑1所示。图中包含的实体有：流量截获会话、报警主机列表、用户、用户登录日志、用户HTTP请求日志和用户访问FTP服务器日志。

流量截获会话代表管理员的一次流量截获行为，每一次流量截获会话在系统内都有一个唯一的标识符，方便以后查找。

在每一次流量截获会话中，会产生大于等于零个HTTP访问日志或者FTP访问日志，流量截获会话与它们是一对多关系。

用户登录PPTP服务器时，会产生登录日志。一个用户会产生多个登录日志，但是一个登录日志只对应一个用户。

报警主机列表是一个设置项，与其他实体无关联，用于记录需要发送报警邮件的主机列表。

## 逻辑结构设计

### 关系表的范式与约束

在完成概念结构设计之后，实体之间的关系已经比较清晰。现在将实体对应到数据库中具体的表里，进行逻辑结构设计：

1. 用户（User）

表 4-1 User表结构

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段描述 | 数据类型 | 可否为空 | 约束 | 默认值 |
| id | 标识用户的唯一ID | int | 否 | 自增主键 |  |
| username | 用户登录服务器时的用户名 | str | 否 |  |  |

1. 登录日志（LoginLog）

表 4-2 LoginLog表结构

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段描述 | 数据类型 | 可否为空 | 约束 | 默认值 |
| id | 用户登录行为产生的日志的唯一标识符 | int | 否 | 自增主键 |  |
| timestamp | 用户登录的具体时间 | datetime | 否 |  |  |
| Ip | 用户登录的终端IP地址 | str | 否 |  |  |

1. FTP访问日志（FTPAccess）

表 4-3 FTPAccess表结构

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段描述 | 数据类型 | 可否为空 | 约束 | 默认值 |
| id | 用户访问FTP服务器生成的日志的唯一标识符 | int | 否 | 自增主键 |  |
| host | 用户访问的FTP服务器域名或者地址 | str | 否 |  |  |
| action | 用户访问的行为，1为登录，2为下载文件 | int | 否 |  |  |
| content | 若action的值为2，该字段的意义为下载的文件名，否则无意义 | str | 是 |  | 空字符串 |
| timestamp | 用户访问行为的时间戳 | datetime | 否 |  |  |

1. HTTP访问日志（HTTPAccess）

表 4-4 HTTPAccess表结构

| 字段名 | 字段描述 | 数据类型 | 可否为空 | 约束 | 默认值 |
| --- | --- | --- | --- | --- | --- |
| id | HTTP访问日志的唯一标识符 | int | 否 | 自增主键 |  |
| host | HTTP请求中的HOST字段，即用户访问的Web服务器 | str | 否 |  |  |
| timestamp | 时间戳 ，用户发起请求的具体时间 | datetime | 否 |  |  |
| traffic | 用户该次请求的流量 | int | 否 |  |  |
| method | HTTP请求中的method字段，即请求的方法 | str | 否 |  |  |
| uri | HTTP请求中的URI，指定了请求的路径 | str | 是 |  |  |

1. 报警主机

表 4-5 报警主机表结构

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段描述 | 数据类型 | 可否为空 | 约束 | 默认值 |
| id | 唯一标识符 | int | 否 | 自增主键 |  |
| host | 要告警的主机名 | str | 否 |  |  |
| method | HTTP请求中的方法 | str | 是 |  | GET |

### Pony ORM关系图

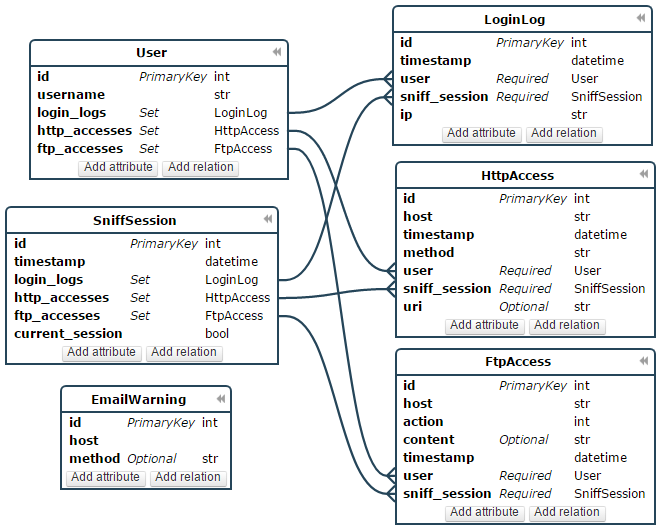


图 4‑2 Pony ORM图

Pony ORM为使用者提供了Web端的数据库关系编辑器。在编辑器中加入对象后，可以为对象添加属性，而且可以更改属性的类型和约束等。在对对象和其属性的编辑完毕之后，还能添加对象直接的关系映射。

如图 4‑2，根据图 4‑1 E-R图设计了4个关系表。User表与其他三个表之间都是1对多关系。

可以看到表中并没有外键。因为在Pony ORM中，表之间的关系直接由Pony ORM来管理，而不是交给数据库系统。这样做的好处是增加了ORM对多个数据库的兼容性，而且对锁的需求降低了，提高了并发性能。

Pony ORM是针对Python的ORM，所以数据类型用Python中的几种基础数据类型来表示。在运行程序时，ORM将会把Python中的数据类型自动地根据使用的数据库来选择相应的数据库系统支持的类型。

## 软件结构设计



图 4‑3 软件结构图

本系统由两个部分组成，一个是用户界面，一个是流量截取进程。系统在载入后首先打开用户界面，此时还不存在流量截取进程。当用户点击“Start Sniffer”（5.1.2 下方）时，系统启动流量截取进程，初始化共享内存空间。

流量截取进程运行的时候，会将用户登录的消息、FTP数据包截获分析后的消息和HTTP流量截获分析后的消息通过共享队列发送到用户界面进程，同时也会不断更新共享内存中的流量统计值。在用户界面中，设置了一个QTimer对象，每隔一秒钟就检查一次共享队列中是否有新的消息存在，然后更新流量统计值，在更新每秒流量统计值后，就将其清零。

由于Python中GIL（Global Interpreter Lock，全局解释器锁）的存在，同一时刻在同一个CPU上，只能运行一个线程，所以Python中的线程模型，更加适合I/O密集型的工作场景。本项目中，不仅有网络I/O的等待，更多的是在接到数据包后对数据包的分析运算。所以Python的多线程模型并不适合在本项目中使用。

由于上述的原因，本项目采用的是多进程编程模型。多进程编程模型有两个好处：第一个是由于新的子进程是一个独立的操作系统层面的进程，这个子进程就不再受到GIL的限制，多个进程可以并发地运行在不同的CPU核心上，从而达到最好的CPU利用率；第二个是由于进程是独立的实体，如果一个进程僵死，其他的进程不会受到影响，也就是说如果抓包进程僵死，用户界面并不会一起崩溃，甚至还能在未来添加自动杀死僵尸子进程并自动重启的功能。

# 系统实现

本章基于上述的技术背景，设计一个VPN监控软件。软件运行在Linux环境下，具有抓包、分析记录日志和展示数据的功能。本软件通过分析用户登录PPTP服务器的数据包，记录用户的IP和用户名，从而将IP与已登录的用户的用户名一一对应起来，并且无需从PPP的chap-secret文件中读取用户的密码等数据。

PPTP服务器可以是网络中的任意一台主机，但是这台主机必须要可以访问广域网，广域网上的终端要能够主动连接到这一台主机。

## 用户界面实现

所有的界面都使用Qt 设计师应用程序来设计。Qt设计师提供了良好的交互式图形界面，可以方便地通过控件来创建界面。除此之外，它还提供了对信号、槽与事件的处理绑定功能。通过点击某个控件，还能更改它的属性、布局等字段。

所有的后缀为ui的界面文件都放在项目的src/ui/ui\_files目录下。编写一个脚本放置在项目的根目录下，这个脚本可以批量使用pyuic5程序将ui界面文件转换为Python代码。每次改变界面设计或者加入了新的界面后，就运行一次这个脚本。

### 国际化

最初编写主界面时，使用的是英语。考虑到接触到这个软件的人大多数为中国人，在后期加入了国际化。使用了Qt提供的pylupdate5程序从项目的源代码和界面文件中提取可翻译的字符串，在linguist程序的帮助下，将提取出的字符串翻译为中文后，更改app.py文件中的代码，插入翻译器达到国际化的效果。当需要发布英语版本时，只需删除这部分代码即可回复英文界面。

表 5-1 国际化代码

|  |
| --- |
| mypath = 'src/ui/ui\_files/'  onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath, f))]  files = list(filter(lambda f: f.endswith('.qm'), onlyfiles))  files = list(map(lambda f: f[:-3], files))  translators = []  for f in files:  translator = QTranslator()  file = 'src/ui/ui\_files/' + f  translator.load(file)  translators.append(translator)  for t in translators:  app.installTranslator(t) |

代码如表 5-1所示，找到/src/ui/ui\_files下的所有的文件，找出所有文件名为“qm”结尾的文件，去掉后缀后，对每一个文件创建翻译器，并把它们加入到app对象中。

### 主界面

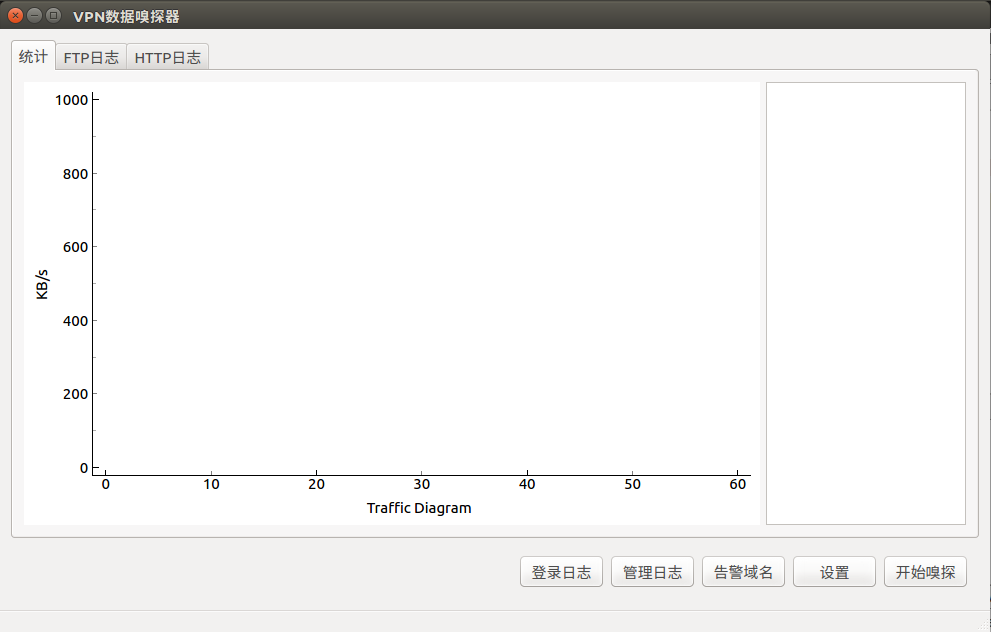


图 5‑1 用户界面首页

打开软件之后，可以看到软件有几个选项卡。如图 5‑1 用户界面首页所示，默认显示的是第一个“Statistics”选项卡。在这个选项卡中有两个部分，一个是当前通过PPTP服务器的实时流量图表，另一个是当前登录到PPTP服务器的用户列表。

图中的实时流量图表是使用PyQtGraph和Numpy（见3.4节）制作的。在Sniffer类（见5.2.2节）中，维护了两个变量，这两个变量储存在共享内存中，分别为每秒流量和总流量。在MainWindow类中设置了一个定时器，每隔一秒钟就会读取每秒流量中的数，加入到一个Numpy缓冲区中，并更新图表然后清零这个变量。

Numpy的缓冲区运用了环状缓冲区技术，即后来加入的数据会不断覆盖最老的数据，从而避免数据一直留存在内存中造成内存的浪费。实现Numpy的环状缓冲区使用了Python标准库中提供的deque，即双端队列，在初始化缓冲区时，设置一个长度为100的缓冲区，数据类型是浮点数。使用以下代码：

self.trafficData = deque(np.zeros(100, dtype='f'), 100)

当需要向缓冲区中加入新的数据时，就进行以下操作：

self.trafficData.append(self.sniffer\_process.data['traffic\_per\_second'] / 1024)

要把缓冲区的数据更新到图表中去时，就进行以下的操作：

self.sniffer\_process.data["traffic\_per\_second"] = 0

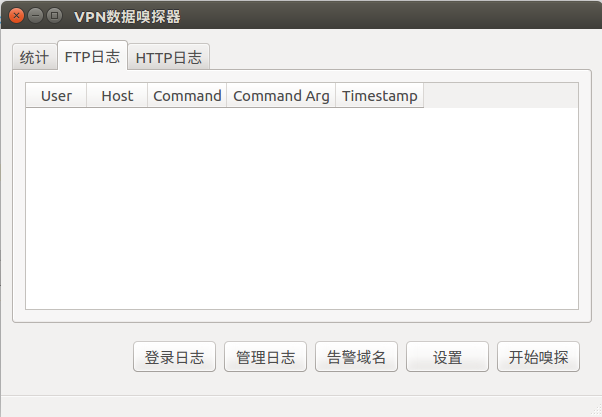


图 5‑2 FTP日志显示界面

如图 5‑2 FTP日志显示界面所示，切换到“FTP Log”选项卡之后，展示的是一个表格，表格会在开始流量截取之后，显示用户发起的FTP访问日志。“HTTP Log”日志界面与“FTP Log”界面大同小异，也是一个表格。表格使用的控件是QTableView。为此，还需要专门通过继承QAbstractTableModel类来分别编写模型类来承载日志数据。

下方的几个按钮的功能分别是电子邮件报警功能设置、日志搜索界面、设置界面和开始流量截取。在用户列表上方还有一个“Back To Traffic Diagram”按钮，这个按钮可以从用户详情界面回到流量统计图表界面。

主界面使用的是QMainWindow控件，这个控件的中央有一个名为centralWidget的QWidget，布局为GridLayout。这是一个两行的GridLayout，它包含两个BoxLayout，两个容器的布局都是水平布局。

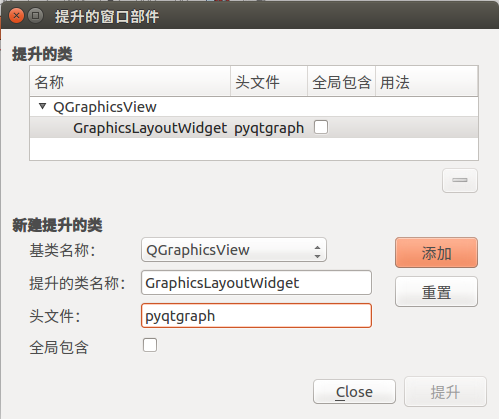


图 5‑3 提升组件至GraphicsLayoutWidget

在上方的水平布局容器中，左边是一个图表控件。在加入图表控件时，使用的控件是QGraphicsView控件，但是我们在实际编写代码时，使用的并不是简单的QGraphicsView，而是由PyQtGraph提供的GraphicsLayoutWidget。所以要如图 5‑3所示，进行以下操作：

1. 在“Display Widgets”分类下，找到“Graphics View”控件，把它拖到主界面中以创建一个QGraphicsView控件；
2. 在QGraphicsView控件上单击右键，选择“提升为…”命令；
3. 在“提升的类名称”中，输入需要使用的PyQtGraph组件名称，如“PlotWidget”、“GraphicsLayoutWidget”等等；
4. 在头文件中输入“pyqtgraph”；
5. 点击“添加”按钮，然后单击“提升”按钮。

右边的控件是一个垂直布局容器，包括一个按钮和一个ListWidget。

下方的水平布局容器中，有几个按钮控件，最左边有一个水平的Spacer控件，这个控件可以制造空间，把几个按钮挤到右边去，使得界面变得更加的美观。

### 设置界面

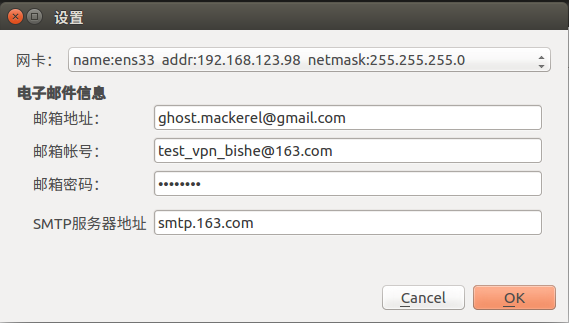


图 5‑4 设置界面

用户点击“Setting”按钮后，弹出如图 5‑4所示的设置界面。在这个界面中，用户可以设置要截取流量的网卡。用户在第一次设置要截取流量的网卡后，系统会记住该网卡，若下一次启动时该网卡仍然存在，就会继续使用这张网卡。在下方的“Email Information”中，用户可以设置接受警报右键的电子邮件地址和用于发送警报电子邮件的SMTP服务器、账号和密码。这些值也会被系统记住，在下一次打开程序时自动加载。

QComboBox控件中，选项数据是按照键、值对的形式储存的，用户看见的界面中显示的是文本，而这个文本有一个对应的值。在设置界面中，文本就是图 5‑4中显示的选项文本，而它对应了另一个含有网卡名称的字符串。在存储设置到配置文件时，就只存储这个字符串。

在载入设置界面时，程序会首先获取本机的所有网卡，并且同时获取网卡对应的IP地址和子网掩码。在获取本机网卡及其IP地址和子网掩码时，使用了Python中的一个psutil库，这个库对使用Python获取正在运行的进程和系统资源利用率的操作进行了跨平台的、易于使用的封装。获取信息的代码如表 5-2所示。

对话框的centralWidget布局为垂直布局，从上至下依次是一个垂直布局容器和一个QButtonBox控件。垂直布局容器中包含了一个表格布局容器和一个QGroupBox，QGroupBox的布局方式也是表格布局。两个表格中含有一些QLabel控件和一些QLineEdit控件，用于接受用户的输入。

表 5-2 使用psutil获取网络信息

|  |
| --- |
| def get\_net\_interfaces():  net\_interfaces = []  info = psutil.net\_if\_addrs()  for k,v in info.items():  name = k  address = netmask = None  for i in v:  if i.family == socket.AF\_INET:  address = i.address  netmask = i.netmask  if address is not None and netmask is not None:  net\_interfaces.append((name, address, netmask))  return net\_interfaces |

### 日志管理界面

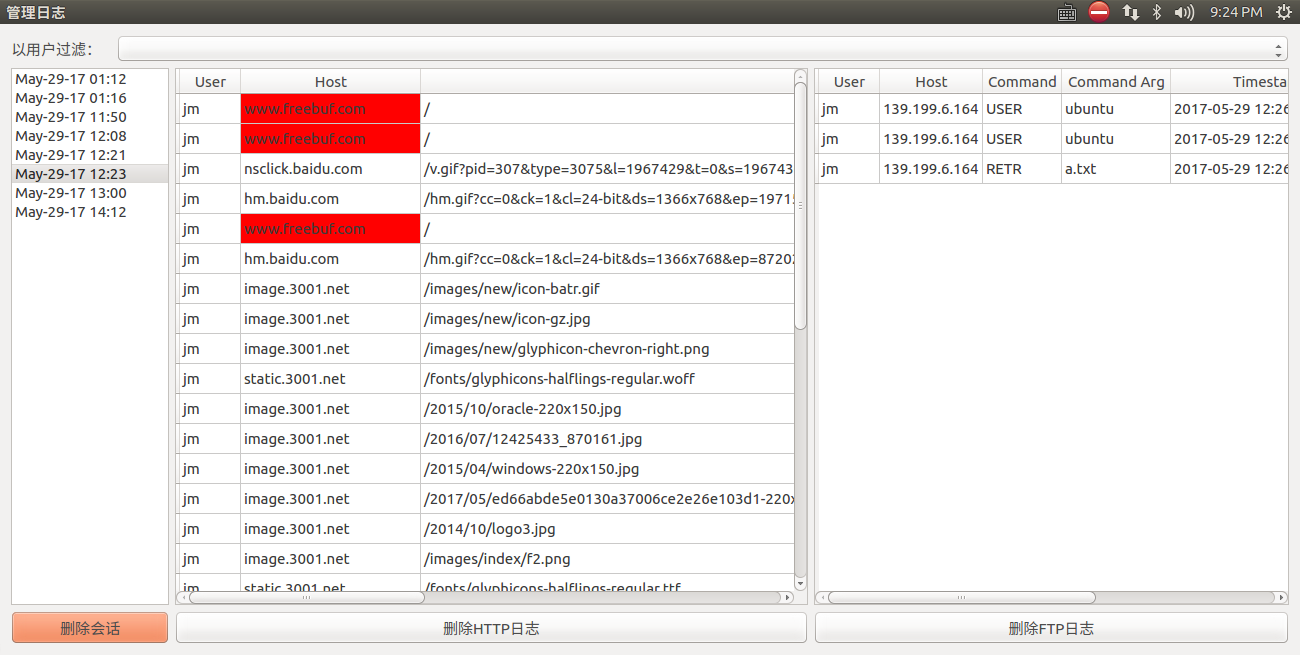


图 5‑5 日志管理界面

图 5‑5是日志管理界面，日志管理界面由一个下拉菜单，两个表单组成和三个按钮。在窗口被初始化时，系统从数据库中读取所有的流量截取会话记录，并把时间显示在最左边的QListWidget中。用户可以在列表中选取想要查询的会话记录。当系统发现列表中的选项被更改时，就会根据当前选定的列表项目获取对应的日志项目。

在查出一次截取会话中的日志后，用户还可以通过最上方的用户过滤功能来筛选当前日志显示界面中的日志。在初始化对话框的时候，程序会把所有的用户都读入到顶部的下拉菜单中。

两个QTableView都有各自的模型类。模型类提供了更新当前数据的功能。系统根据当前选择的流量截取会话从数据库中选择所有的日志记录，不断地读取保存到内存中，并在最后触发layoutChanged信号。具体实现见表 5-3。

表 5-3 更新模型类的实现代码

|  |
| --- |
| def new\_select(self, s):  self.logData = []  with db\_session:  logs = s.ftp\_accesses.select()[:]  for l in logs:  self.logData.append(l)  self.layoutChanged.emit() |

用户点击“Delete”按钮可以删除选中的表格项。模型类还提供了删除指定行的功能。为了让用户只能选择一行，而且一次性选中一行而不是一个单元格，需要在Qt设计师的设计界面中，更改两个QTableView的“SelectionMode”和“SelectionBehavior”属性，将其设置为“SingleSelection”和“SelectRows”，详见图 5‑6。

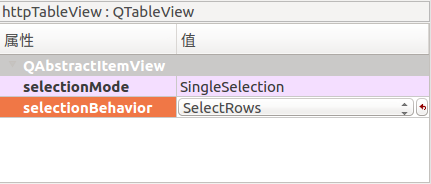


图 5‑6 在Qt设计师中设置QTableView的属性

### 报警网址设置界面

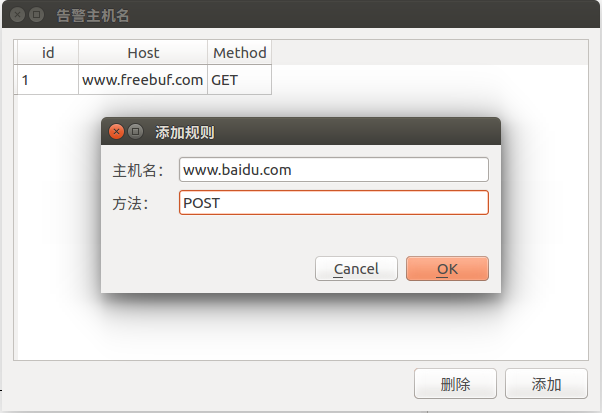


图 5‑7 告警主机名设置

用户点击“添加”按钮，软件就弹出“添加规则”对话框询问用户要设置的主机名和请求方法。用户将告警主机规则输入完毕后就可以点击确定按钮来添加告警主机规则。告警主机规则被储存在数据库中，用户点击删除可以删除选中的告警主机规则。

### 用户登录日志界面

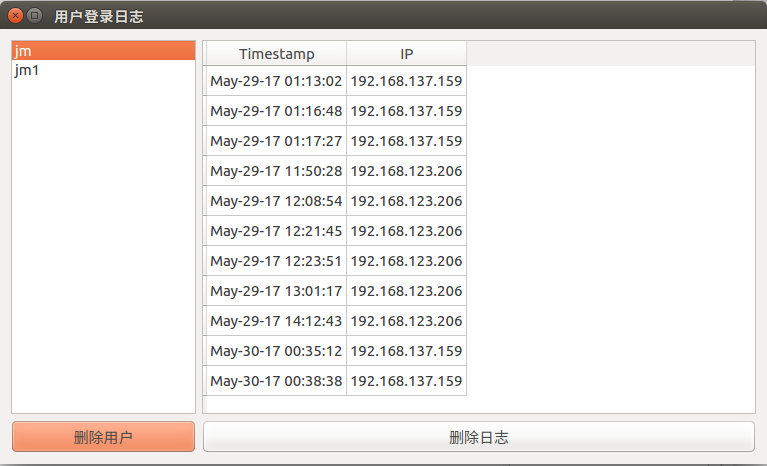


图 5‑8 用户登录日志界面

系统在检测到用户登录到PPTP服务器后，会生成用户登录日志以记录用户登录的时间和远程IP地址并保存到数据库中。在这个界面中，系统将数据库中储存的用户登录日志读取出来。用户在左边的用户列表中点击用户之后，就可以查看该用户在什么时候使用什么IP地址登录了PPTP服务器。

## 流量截取进程实现



图 5‑9 流量截取模块类图

流量截取进程实际上是一个封装好的模块，这个模块使用了PyShark库，使用Python提供的multiprocess库新建了进程来运行流量截取和分析逻辑。该模块包含四个类，第一个是Sniffer类，这个类封装了流量截取和消息传递的功能；第二个是LoginUser，这个类用来保存已经登录用户的信息；第三个是PptpUserLogger，这个类提供了数据包流的追踪功能，可以在数据流中找出那些与PPTP登录和下线相关的数据包，分析后给出用户的上下线通知；第四个是UserSniffer，这个类继承了Python中的threading.Thread类，是一个线程，当用户登录后，Sniffer进程就会启动一个UserSniffer线程的实例来对这个用户的流量进行截获和分析。

### LogInUser类

LogInUser类是一个实体，表示登录到PPTP服务器的用户，储存了用户的相关信息。当用户登录到PPTP服务器后，PptpUserLogger类就创建一个LogInUser的实例，并加入users字典中，别的类通过查询这个字典就能获取到已登录用户的相关信息。

### Sniffer类

Sniffer类封装了流量截取和消息传递的功能。成员变量worker保存了工作进程的引用，user\_manager是一个PptpUserLogger的实例，data和data\_queue分别是共享内存空间和共享队列的引用。

在用户界面中始终保存着一个Sniffer类的实例。当用户点击“Start Sniffer”按钮时，实际上调用了start()函数，这个函数启动了流量捕获进程。流量捕获进程运行的是Sniffer类中的do\_sniff()函数，这个函数简单地开启一个LiveRingCapture实例，然后通过LiveRingCapture提供的迭代器接口遍历捕获到的数据包，对每一个数据包都调用do\_parse()函数。这是一个死循环，如果用户点击停止流量截获按钮，则会通过调用Process.terminate()函数来结束抓包进程。

在do\_parse()函数中，会判断每一个数据包的所属的类型，并且根据不同的数据包类型来将数据包分发到不同的解析逻辑代码段中去。

send\_user\_login()和send\_user\_logout()是两个事件处理函数，分别绑定到PptpUserLogger的实例上的user\_login\_handler和user\_logout\_handler事件处理器上去。当PptpUserLogger发现有用于上线或者下线时，就会出发这两个事件处理器。这两个函数在用户登录或退出时，生成一个dict作为消息放入共享队列中，用来通知图形界面加入或移除用户。

sniff\_user也被绑定到PptpUserLogger实例上的user\_login\_handler事件处理器上，这个函数用来创建一个SniffUser线程的实例并开始线程。相对应的,stop\_sniff\_user()也被绑定到user\_logout\_handler事件处理器上，在用户下线时，就把SniffUser中的exit\_flag变量设为True来通知线程退出。

### PptpUserLoger类

在Sniffer类的do\_parse()函数中，数据包被传入PptpUserLoger对象中。user\_login\_handler和user\_logout\_handler是两个列表，它们被用于存储用户登录事件和用户退出事件的事件处理函数。log\_in\_user用于存储发起了登录请求但是还没有确认登录成功的用户；users是一个字典，用来存储已经成功登录的用户，键是用户的IP，值是LogInUser的实例。PptpUserLogger通过以下方法找到登录的用户：

1. 当发现Code字段为Response（0x2）的MSCHAPV2数据包时，说明有用户试图登录，这时候新建一个LogInUser实例，填充它的用户名（name）字段、call\_id字段、标识符（identifier）字段，把这个实例加入到log\_in\_user中；
2. 在接下来的流中判断是否有Code字段为Success（0x3）的MSCHAPV2数据包，如果发现了这样的包，说明有一个用户登录成功，但是这个数据包可能是恶意的第三者发送的，所以要进行验证。在验证auth\_string字段的值之后，就可以认为这个用户登录成功了。通过用户的标识符（Identifier）来确定这个登录成功的数据包指定的是哪一个用户，此时将该用户的LoginUser加入到log\_in\_user字典中，供其他的模块查询。
3. 最后记录用户登录的日志，并触发user\_log\_in\_handler事件，调用其中的事件处理函数。

当用户断开连接时，会发送PPP LCP中的Terminate-Request分组数据包，在流量中找到这样的数据包时，就说明用户已经下线了。通过PPP LCP分组中的call-id字段来判断是否存在这样一个用户，如果存在，则从users中将其移除，并触发user\_log\_out\_handler事件，调用其中的事件处理函数。

### UserSniffer类

这个类继承了threading.Thread类，是一个线程类的子类，它的初始化需要从Sniffer类传入用于放置消息的共享队列对象和LogInUser的对象。UserSniffer类通过LogInUser的实例来确定要抓取流量的网卡，相关的代码在表 5-4中给出。

表 5-4 获取用户对应的ppp网卡，启动抓包实例

|  |
| --- |
| # 在\_\_init\_\_函数中  from utils import get\_ppp\_interfaces  interfaces = get\_ppp\_interfaces()  for i in interfaces:  if i[1] == user.local\_ip:  self.interface = i[0]  user.capture\_session = self  # 在do\_sniff函数中  cap = pyshark.LiveRingCapture(interface=self.interface, bpf\_filter='not udp and ip') |

## PPTP模块

PPTP模块实现了PPTP协议中的一些算法。根据PPTP RFC、MSCHAPV2 RFC和MPPE RFC中的描述，使用了Python中自带的hashlib和一个第三方DES加密解密算法库pyDES，实现了在MSCHAP Version2中用到的所有加密流程和哈希流程。

# 系统测试

## 测试前期准备



图 6‑1 测试系统结构图

如图 6‑1的测试系统结构所示，首先在VMware虚拟机1中运行一个Ubuntu 16.04 LTS操作系统，然后在终端中使用apt-get命令安装pptpd。在apt的源中，pptp服务器指的是poptop开源项目。安装完成后按照以下步骤配置PPTP服务器：

1. 在系统的/etc/目录下找到pptpd.conf文件，编辑位于文件最后的ip分配设置，如表 6-1。将主机IP设置为10.0.0.1，而分配给客户端的地址池为10.0.0.100-200；

表 6-1 pptp.conf设置IP分配

|  |
| --- |
| localip 10.0.0.1  remoteip 10.0.0.100-200 |

1. 在系统的/etc/ppp目录下找到chap-secrets文件，加入两个用户，如表 6-2所示；

表 6-2 chap-secrets文件中的用户

|  |  |  |  |
| --- | --- | --- | --- |
| Client | Server | Secret | IP Addresses |
| jm | pptpd | 123 | \* |
| jm1 | pptpd | 123 | \* |

1. 更改/etc/ppp/目录下的pptpd-options文件，使得PPTP服务器只接受MSCHAPV2和MPPE 128 bits的验证和加密算法。另外，将分配给用户的DNS设置为阿里云的DNS，配置文件

表 6-3 pptpd-options文件配置

|  |
| --- |
| name pptpd  refuse-pap  refuse-chap  refuse-mschap  require-mschap-v2  require-mppe-128  ms-dns 223.5.5.5  ms-dns 223.6.6.6  proxyarp  nodefaultroute  lock  nobsdcomp  novj  novjccomp  nologfd |

1. 为了能够让linux内核启用转发功能，需要在/etc/sysctl.conf文件中找到net.ipv4.forward选项，并设为1。接下来在终端以管理员权限运行systcl -p，来使ipv4转发功能生效。
2. 启动PPTP服务器，并在iptables中添加规则，使得用户可以连接到PPTP服务器，而且可以正常访问局域网内的网络，同时又能将PPTP服务器作为代理服务器访问外部网络。PPTP服务器默认配置PPP网卡的MTU为1396，计算MSS时将其减去20字节的IP头部和20字节的TCP头部，结果为1356。如果没有在TCP连接握手建立时将MSS设置为这个值，网络丢包率将会大增，导致几乎无法通过PPTP服务器访问互联网。iptables在每一次重启之后都会清空规则，为了简化测试操作，编写bash脚本，并在每一次重启虚拟机后以管理员权限运行。脚本内容如表 6-4所示。

表 6-4 添加iptables规则并启动PPTP服务器

|  |
| --- |
| iptables -t nat -A POSTROUTING -s 10.0.0.0/24 -o ens33 -j MASQUERADE  iptables -A FORWARD -p tcp --syn -s 10.0.0.0/24 -j TCPMSS --set-mss 1356  service pptpd start |

此时，PPTP服务器已经启动，并且所有经过系统路由功能后的数据包，只要源地址为10.0.0.0/24，就会被转发至ens33网络接口卡上。用户可以通过Windows自带的PPTP客户端功能连接到服务器。若PPTP服务器运行在可以访问广域网的主机上，那么客户端也能够访问广域网。

在操作系统3上，使用VSFTPD软件搭建一个FTP服务器。因为服务器假设在公网上，为了安全性，设定匿名用户无法登录 。在操作系统3上的用户用户名为ubuntu，密码为520Mrblood!，用户家目录为/home/ubuntu。在家目录下放置一个文本文件a.txt。启动VSFTPD服务器，使得FTP服务能够被访问。

## 用户统计和流量统计测试

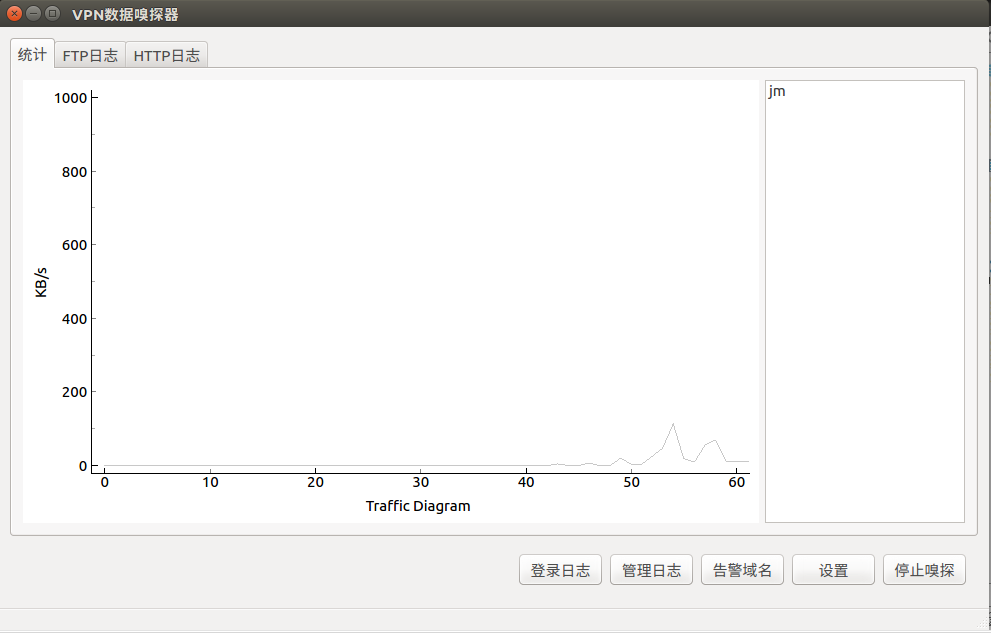


图 6‑2 用户登录测试图

在操作系统2和操作系统3中，使用Windows自带的PPTP客户端功能，以用户jm和用户jm1的身份连接运行在操作系统1中的PPTP服务器，可以看到在主界面的统计选项卡右方的用户列表中出现了6.1节中测试前期准备的用户。在客户机2上访问一些较大的含图片较多的网页，产生流量，可以看到流量统计表中的曲线在上下浮动，中间出现了一些峰值，说明客户机本项目成功地检测到了用户的登录行为，也可以统计流量。

当用户从客户端断开连接后，可以看到列表中的用户项消失了，这说明本项目可以成功地检测到用户的下线行为，并做出响应。

## FTP和HTTP日志记录功能

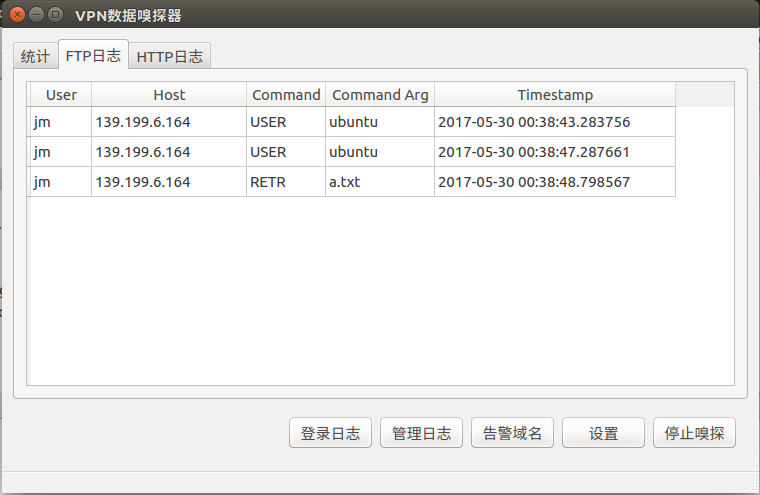


图 6‑3 FTP日志功能测试界面

在操作系统3上运行FTP服务器，然后在主机2中打开FileZilla应用软件。FileZilla是一个具有图形界面、易用的FTP客户端应用程序。用FileZilla连接运行在操作系统3上的FTP服务器，然后下载位于FTP服务器根目录下的a.txt。这时候可以看见软件主界面的“FTP Log”选项卡中的TableView里，出现了一些日志，如图 6‑3所示。可以通过日志看到用户访问FTP服务器的时间，执行的命令和命令参数。

命令参数对于不同的命令具有不同的意义，当命令是USER时，参数的意义是登录的用户名，而当命令为RETR时，参数的意义则是下载的文件的文件名。本次测试中出现的日志表示拥护登录了FTP服务器，FTP服务器的地址是139.199.6.164，登录用户为ubuntu，下载了一个文件，文件名为a.txt。

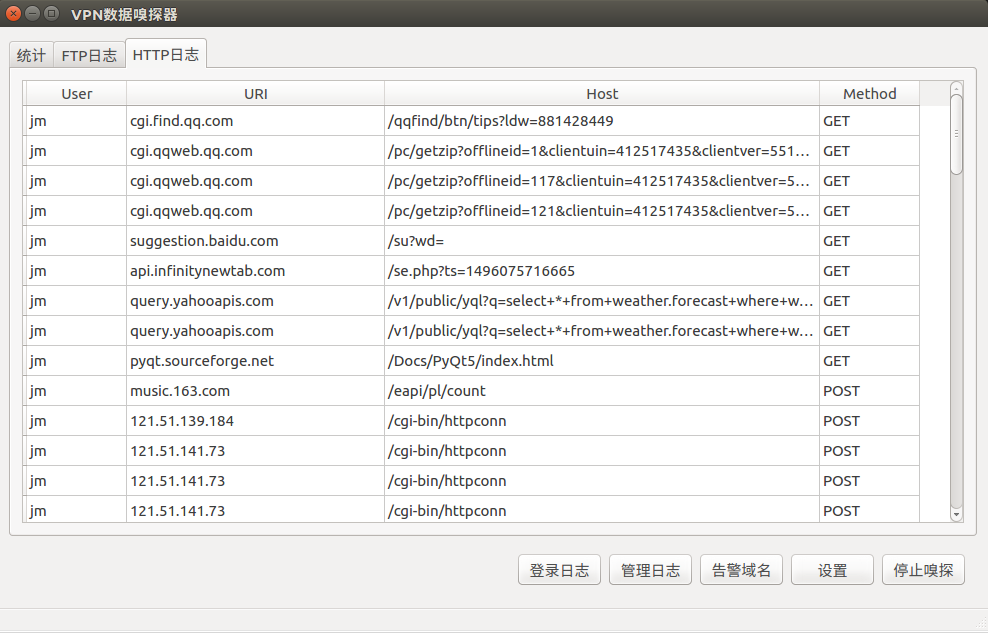


图 6‑4 HTTP日志记录界面测试

在操作系统2中打开浏览器，在地址栏输入www.freebuf.com。这个网站的内容较丰富，会产生许多请求，而且它是没有使用HTTPS的网站，由于HTTPS在传输层与应用层之间加入了一个SSL协议层，将所有的流量都通过一个强加密的SSL隧道协议传输，所以只有使用HTTP的网站，本软件才能解析它的HTTP请求。

可以看见在jm1连入PPTP服务器后，产生了三个HTTP请求。其他的都是jm用户的请求。这说明系统可以准确区分流量来自哪一个已经登录的用户。

## 设置功能测试

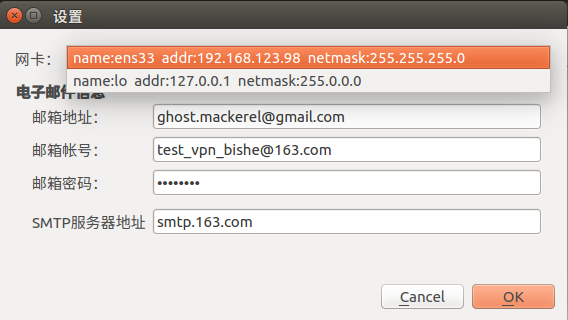


图 6‑5 网卡设置截图

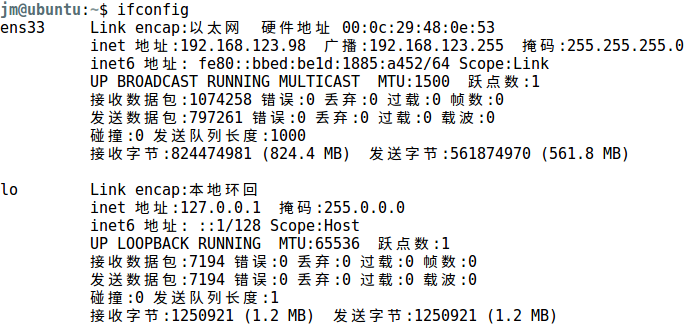


图 6‑6 系统ifconfig命令结果

在图 6‑5 网卡设置截图中可以看见程序成功获取了系统当前的网卡名称、网卡地址和子网掩码。如图 6‑6所示，打开Ubuntu终端，执行ifconfig命令，可以看见返回的结果与设置界面中的选项是一一对应的。这说明网卡信息获取功能正常。

图 6‑7 保存的配置文件

如图 6‑7所示，在点击“OK”按钮之后，再打开软件目录中的“config.ini”文件，可以看到配置都被保存在了配置文件中。而在点击“Settings”按钮后弹出的窗口中也可以看见已经保存的项目被填充在用户界面中，如图 6‑5所示。这可以说明设置功能是可以正常执行的。

## 日志管理功能测试

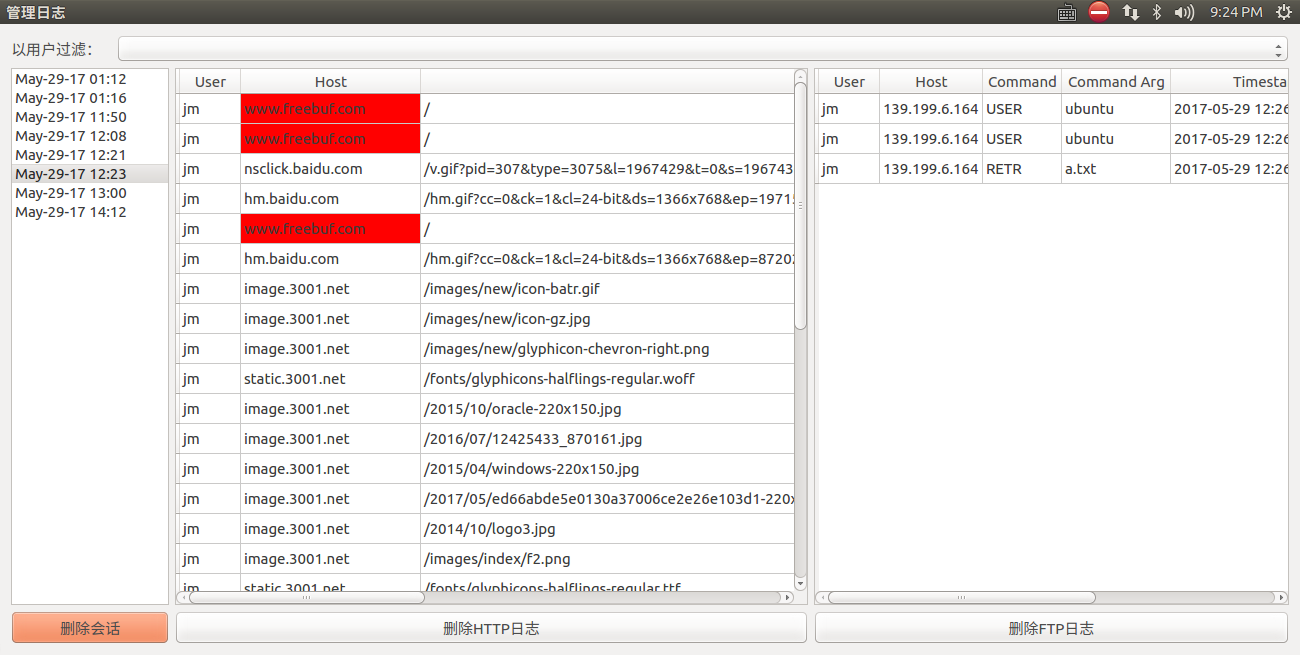


图 6‑8 日志管理测试界面

点击“Manage Log”按钮，会弹出日志管理对话框。左边是流量截取会话的记录列表，这说明系统可以成功地记录流量截取会话，并把它查询出来。

选择某一个流量截取会话项目后，下方的两个表单就发生变化，显示的是当前选取的流量截取会话中产生的所有日志。这可以证明系统能够正确地保存流量截取会话中产生的日志，并把它查询出来。

选择日志表单中的某一行，点击Delete按钮，该行消失。重新加载这一次会话后也没有再出现。说明系统可以成功实现删除日志项目的功能。

在HTTP日志的表单中，可以看到被设置为危险域名的请求被标识为红色，可以证明危险日志标记功能正常。

## 邮件报警功能测试

图 6‑9 收到邮件的截图

在设置www.freebuf.com为不安全的域名后，使用工具开启流量截取，在操作系统2中用浏览器访问该域名，发现邮箱中已经受到了报警邮件，警告有用户访问了不安全的域名。如图 6‑9，域名为[www.freebuf.com](http://www.freebuf.com)，方法是GET，用户名为jm。

# 结束语

## 主要成果

### 完成的功能

本项目完成了一个可以运行在Linux操作系统下的能够获取与解析PPTP VPN流量的工具。主要功能如下：

1. 用户上线和下线检测功能：工具能够通过分析通过网卡的流量，获取用户登录和下线的情况；
2. 可以统计通过PPTP VPN服务器的实时流量；
3. FTP使用记录功能：软件可以分析流量中的FTP数据包，记录FTP服务器登录和文件下载的行为，形成日志；
4. HTTP请求记录功能：软件可以分析流量中的HTTP数据包，记录HTTP请求的主机名和请求方法；
5. 不安全域名报警功能：软件能够让管理员设置需要特别注意的域名，在发现指向该域名的请求时通过邮件向管理员发送警告；
6. 日志管理功能：可以在每次流量截获时将日志储存在SQLite3数据库中，管理员可以随时通过日志管理功能查看已经储存的每次流量截取会话中记录的日志，还可以通过日志管理功能删除指定的日志项目。

### 系统特点

本项目采用Python语言编写，代码清晰明了简介。通过合理的模块划分，项目被组织得易于维护，可扩展性强。

在图形界面上，使用了PyQt5作为界面框架，通过Qt设计师应用程序来组织界面，通过pyuic5程序自动生成界面代码，既保证了可维护性，又能得到PyQt5强大的特性，而且因为PyQt5是Qt5.8的原生库在Python中的封装，所以几乎没有执行效率上的损失。

项目采用了多进程的软件架构，使得软件在CPU核心数大于等于2的计算机上运行效率更加高，并且具有更好的稳定性。

## 对系统的展望

### 未完成的功能

由于时间上的限制，和本人能力有限的原因，本系统在设计与开发上存在一些缺陷与不足。

在截获流量生成日志时，没有对流量的内容做进一步的解析。例如在发现有用户下载FTP服务器上的文件时，只能记录用户下载的行为和下载的目标文件名，并没有将用户下载的文件内容进行分析；在记录HTTP流量时，只能记录HTTP请求中的主机字段、请求方式字段和URI等简单的字段，并没有对服务器的响应进行进一步的分析。

### 系统的缺陷与不足

本系统通过分析用户登录的流量，可以检测出用户的上线和下线操作。在用户上线后，进行网络操作而产生流量时，系统能知道流量中存在HTTP请求或者FTP请求。在捕获流量并解析后，系统并没有对日志进行进一步的统计分析处理。

在抓取到HTTP流量后，系统只是记录了HTTP请求的日志，并没有在这之后对TCP流进行跟踪并解析返回的响应。

由于时间比较紧迫，在编写代码之前，本人将大量的精力和时间花费在了PPTP、PPP、MSCHAPV2和MPPE协议的文档上，通过不断地试错，才找到了不使用root权限读取用户密码对流量进行解密即可区分用户流量的功能，在编写代码时，时间比较紧迫，所以没有实现对流量更进一步的解析功能。

希望在将来的工作中，上述的缺陷和不足能够得以完善。

# 参考文献

1. G. Zorn, RFC 2759: Microsoft PPP CHAP Extensions, Version 2, IETF, January 2000
2. G. Pall and G. Zorn, RFC 3078: Microsoft Point-To-Point Encryption (MPPE) Protocol, IETF, March 2001
3. B. Schneier, Mudge. Cryptanalysis of Microsoft's PPTP Authentication Extensions (MS-CHAPv2)[J]. Springer-Verlag, 1999, (99): 192-203
4. K. Hamzeh, RFC 2637: Point-to-Point Tunneling Protocol (PPTP), IETF, July 1999
5. G. McGregor, RFC 1332: The PPP Internet Protocol Control Protocol (IPCP), IETF, May 1992
6. S. Hanks, RFC 1701: Generic Routing Encapsulation (GRE), IETF, October 1994
7. 吴杰歆，赵丽红，杨德俊. 基于RC4和RSA混合算法的加密卡设计[J]. 电子质量, 2007第7期:12-14
8. 程思,程家兴. VPN中的隧道技术研究[J]. 计算机技术与发展,2010,(02):156-159.
9. 王丹. 动态P2P VPN系统的研究与设计实现[D].北京邮电大学,2010.
10. 郝辉,钱华林. VPN及其隧道技术研究[J]. 微电子学与计算机,2004,(11):47-51.
11. 秦晨洋. 基于PPTP客户端的防火墙技术研究[D].郑州大学,2012.
12. 程思,程家兴. VPN中的隧道技术研究[J]. 计算机技术与发展,2010,(02):156-159.
13. 郝辉,钱华林. VPN及其隧道技术研究[J]. 微电子学与计算机,2004,(11):47-51.
14. 林宏刚. MS—PPTP协议安全性研究[D].四川大学,2003.
15. 成思迥. VPN在LINUX下的实现[D].上海海运学院,2000.