

基于 Python 的 SMTP 协议信息还原

摘要

随着网络的普及和迅速发展,电子邮件作为一种具有传递速度快,可以群发,方便快捷等特点的通信方式,深受人们的喜爱。同时它已经深入到人们的生活中,成为日常通信的重要组成部分。但是在电子邮件给人类带来极大便利的同时,企业泄密,网络犯罪和黑客入侵等网络安全事件也迅速滋生,严重影响了人们正常的生活秩序甚至危害到企业和国家的安全。因此急需建立一系列的安全机制来控制 and 减少这些不安全事件的发生。本文设计了基于 Python 的 SMTP 协议信息还原系统,本系统的功能是监控局域网中的 SMTP 协议,从中提取电子邮件的信息存入数据库,方便用户对邮件内容的监控,并对可疑用户进行追踪。可以应用在小规模的局域网中监控电子邮件,能够保证监听网络中的全部电子邮件而不会发生漏包的情况,对局域网安全起到一定的作用。另外,此系统为局域网监控提供了一种模型,在稍微拓展的基础上可以实现对 POP、WWW、FTP 等其他协议的监控。在此基础上,还可以大力的发展数字传输监控,多媒体监控等。在安全领域有一定的应用前景。

关键词: 局域网; SMTP 协议; python; 信息还原

Abstract

With the popularization and rapid development of the Internet, e-mail, as a communication method with the characteristics of fast delivery, mass sending, convenience and quickness, is deeply loved by people. At the same time, it has penetrated deep into people's lives and become an important part of daily communication. However, while e-mail brings great convenience to human beings, cybersecurity incidents such as corporate leaks, cybercrimes, and hacker intrusions also increase rapidly, seriously affecting people's normal life order, and even endangering corporate and national security. Therefore, it is urgent to establish a series of safety mechanisms to control and reduce the occurrence of these unsafe events. This paper designs a Python-based SMTP protocol information recovery system. The function of this system is to monitor the SMTP protocol in the local area network, extract mail information and store it in the database, which is convenient for users to monitor the content of the mail and track suspicious users. It can be applied to mail monitoring in a small-scale local area network, which can ensure that all mails in the network are monitored, and no data packets are leaked, which plays a certain role in the security of the local area network. In addition, the system provides a local area network monitoring model, which can monitor other protocols such as POP, WWW, and FTP on the basis of a little expansion. On this basis, we can vigorously develop digital transmission monitoring, multimedia monitoring and so on. It has certain application prospects in the field of security.

Key words: local area network; SMTP protocol; Python; information recovery

目 录

摘要.....	I
Abstract.....	II
1 绪论.....	1
1.1 研究背景.....	1
1.2 研究目的及意义.....	1
1.2.1 研究目的.....	1
1.2.2 研究意义.....	1
1.3 文献综述.....	2
1.4 研究思路.....	3
2 系统实现基本原理.....	4
2.1 电子邮件工作原理.....	5
2.2 SMTP 的通信原理.....	5
2.2.1 SMTP 通信模型.....	5
2.2.2 SMTP 的命令与应答.....	6
2.2.3 SMTP 的会话过程.....	7
2.3 数据包的封装和拆封.....	10
2.3.1 数据包的封装.....	10
2.3.2 数据包的拆封.....	10
3 基于 Python 的 SMTP 数据包分析与还原的实现.....	11
3.1 捕获数据包.....	12
3.2 解析还原数据包.....	13
4 邮件还原、储存和查看的实现.....	16
4.1 邮件关键信息的提取.....	17
4.1.1 文件形式的邮件.....	17
4.1.2 HTML 形式的邮件.....	18
4.1.3 带图片的 HTML 邮件.....	19
4.1.4 带附件的邮件.....	20

4.2 邮件存储	21
4.3 邮件查询系统	23
5 系统测试与结果	24
5.1 邮件监听模块测试	25
5.2 邮件查询模块测试	28
总结	28
参考文献	29
致谢	31

1 绪论

1.1 研究背景

随着网络的迅速发展和普及，网络安全问题已经成为人们广泛关注的话题。网络安全是指网络系统的硬件、软件及其系统中的数据受到保护，不因为偶然的或者恶意的原因而遭到破坏、更改、泄露，系统连续可靠正常的运行，网络服务不中断。网络安全从其本质上说就是网络上的信息安全，即网络传播过程中确保信息的保密性、完整性、可用性、真实性和可控性。目前，众多的现代通信方式如电子邮件、即时通讯工具、聊天网页等已经深入了人们的生活。其中电子邮件凭借其方便快捷的特点成为了人们通信的重要组成部分，受到普遍青睐。但电子邮件给人们带来极大便利的同时，也带来了一系列不安全因素。目前国内网络安全监控产品还不够完善，功能单一。例如:邮件监控只对邮件收发者作记录，而邮件的具体内容和所带的附件、音频和图像都没有涉及;http 监控只捕获了浏览的网址，而没有涉及到图片，论坛发言等更为重要的上传数据;msn 监控只监控了即时信息，而没有涉及到传输的数据。事实上国外的大多产品也和国内一样存在这些问题。因此亟需建立一套高效的邮件监控系统，对网上的敏感邮件信息进行追踪和监控，及时地发现问题并解决问题，帮助公安机关控制网络犯罪，有效地保护用户局域网的信息安全。

1.2 研究目的及意义

1.2.1 研究目的

本文研究的目的在于通过研究当前网上用来传递邮件的 SMTP 协议的原理和会话过程，基于 Python 网络开发包实现局域网数据包的捕获。并对捕获到的数据包进行协议分析和协议还原，尽可能多的将邮件内容还原。帮助我们查看与监听网络邮件，从而减少网络犯罪，及时的发现问题和解决问题。能够通过监控技术对于网上有害或可疑邮件进行追踪并对可疑用户进行监控，使安全机关可以有效的控制网络犯罪。

1.2.2 研究意义

随着网络的普及，大量违法犯罪分子通过互联网盗取公司机要文件，传播暴力，反动，赌博等有害信息，所以网络安全变得越来越重要。急需建立一系列的

安全机制对网上信息的传输内容进行监控。国家公安部门已经大范围地对网络信息进行监控，追踪有害信息，从而减少和控制不安全因素的发生。

1.3 文献综述

甘胜玉（2009）在 HTTP、SMTP、POP3 等几种常见应用层协议的基础上文设计并实现了网络内容分析与还原系统，该系统能实时完成数据包捕获、协议分析、会话重组、应用层协议还原等一系列工作，最终准确实现网络协议的还原。王新宇（2010）分析了校园网面临的安全威胁，介绍了常见的网络安全防御产品，分析了网络内容监测系统的原型。深入研究了网络文本内容监测系统的关键技术，网络监听的原理、数据包的捕获、IP 分片重组、TCP 数据流重组、应用层协议和关键词匹配技术等。对校园网文本信息内容监测系统进行总体设计和详细设计，针对 WM 算法提出了一种改进的匹配算法。

丁岳伟（2002）通过对 SMTP 电子邮件报文格式与传输协议分析，给出了邮件信息的还原实现的有关技术和算法，即链路层类 CMyFrame 的设计与实现、网络层类 CMyIP__ Packet 的设计与实现传输层类 CMyTCP__ Packet 的设计与实现、应用层类 CMySMTP__ Mail 类的设计与实现为进一步分析网络中的情况，即流量情况、内容分析以及规则的匹配提供了分析的对象。

王娜（2020）通过借助 python 语言并 unittest 来实现接口自动化测试的框架的建设并将其应用于冒烟测试、回归测试，在敏捷开发过程中缩短测试时间，提高产品质量，该框架可复用性高，测试人员只需提供接口测试用例，配置 jenkins' 定时任务，生成测试报告发送到企业邮箱内，能够更快更方便到监控接口的正常运行。

文婷，何中全（2020）对军用软件测试过程中的测试环境局限性、测试效率低等问题，提出一种基于 python 的软件测试平台实现方法，实现被测软件与外部对象的网络通信模拟、接口数据自动生成及解析、测试结果自动判定等功能，能够降低对测试环境的依赖性，提高测试效率及充分性，有效保障测试质量。

皇甫灵刚，彭俊超（2021）介绍了一种自动发送邮件的方法实践，搭建软件、数据库环境。Python 作为开发语言，利用 OpenOPC 协议读取 DCS OPC 服务器，同时将数据转存在 MySQL 数据库中，编写程序与设定值进行比较，超过报警时，自动发送邮件到指定地址。

张玉东（2008）总结了在 C# 环境下，基于底层的 SMTP（ESMPT）协议和 MIME 发送电子邮件，以及使用 MailMessage 类创建并发送邮件的过程，并重点解析了身份认证，邮件编码以及发送邮件附件等关键实用技术，然后给出了实现代码。

周彩兰、虞珊、张亚芳（2008）对 SMTP 协议的分析来识别垃圾邮件，同时进行拦截，可有效防治垃圾邮件。介绍了 SMTP 的工作模式和协议分析过程，及对 SMTP 协议命令的解释。从协议分析的角度出发，实现 SMTP 协议分析程序——鞠ifferSMTP。分析从网络中捕获邮件数据包，还原出邮件信息，依据规则进行模式匹配，检测出入侵邮件。

朱健、卢秉亮、王盛明、甄雷（2008）以 SMTP 协议为研究对象，在 Visual C++6.0 编程环境下应用 SMTP 协议，通过套接字发送 SMTP 命令，接收并处理邮件服务器的反馈信息，从而在客户端实现对电子邮件的发送。文章详细讨论了 SMTP 协议的命令、应答、会话流程、邮件格式化的实现以及在发送 SMTP 和接收 SMTP 之间的会话及应答过程。软件采用面向对象，消息驱动及多线程等新技术实现系统中的功能。

刘娅、张鹏博（2012）基于 SMTP 与 POP3 的“MailPower”电子邮件邮件客户端，研究了电子邮件的收发原理，应用 DES 加密算法与 MD5 相结合的方式保证安全，描述了系统功能，并在 Eclipse 开发平台实现了电子邮件收发系统。谢命坚（2007）认为电子邮件在现实世界中扮演着重要的角色。许多网站都提供免费电子邮箱，用户可方便、有效、免费地使用网站提供的这些免费邮箱，发送和接收电子邮件，因此，设计快速可靠邮件发送和接收软件十分重要。

1.4 研究思路

（1）利用 python 搭建一个系统，用来监控某个 ip 发送的 SMTP 协议

在本地的一台 Ubuntu 主机上搭建 SMTP 服务器，通过 Python 程序在本机上发送邮件到 Ubuntu 主机上，实现一个可控的操作环境。完成之后，需要验证是否搭建成功。

（2）捕捉到报文里面这些协议的字段，再翻译出来进行协议信息还原
数据流的抓取与

解析同样使用 Python 来编写，主要从网卡抓取到流量，先还原为完整的数据包，再做编码转换，将一个完整的邮件发送过程中的命令交互和邮件内容分别还原出来。

为了能够还原出附件，大致需要进行以下步骤：

准确分离出一封完整的邮件；准确分离出附件段；从附件段分离出附件的 base64 编码正文；将 base64 解码为二进制文件；获取当前文件类型，保存二进制文件为当前文件类型。

（3）有待解决的问题

TCP 发送过程中存在重复发送和乱序的问题，需要一定的机制去做判断；多个发送行为存在的时候，如何在可能交叉的发送包中匹配出一次完整的发送包；当发送程序使用不同的编码时，还需要获取到编码方式再对应进行解码。

2 系统实现基本原理

2.1 电子邮件工作原理

Internet 电子邮件系统是基于客户机/服务器方式，客户端也叫用户代理（User Agent），提供用户界面，负载邮件发送的准备工作，如邮件的起草、编辑以及向服务器发送邮件或从服务器取邮件等。服务器端也叫传输代理（Message Transfer Agent），负责邮件的传输，它采用端到端的传输的传输方式，源端主机参与邮件传输的全过程。工作原理主要包括以下三个步骤：（1）当用户需要发送电子邮件时，首先利用客户端的电子邮件应用程序按规定格式起草、编辑一封邮件，指明收件人的电子邮件地址，然后利用 SMTP 将邮件送往发送端的邮件服务器。（2）发送端的邮件服务器接收到用户送来的邮件后，接收人地址中的邮件服务器主机名，通过 SMTP 将邮件送到接收端的邮件服务器，接收端的邮件服务器根据收件人地址中的账号将邮件投递到对应的邮箱中。（3）利用 POP3 协议或 IMAP，接收端的用户可以在任何时间、地址利用电子邮件应用程序从自己的邮箱中读取邮件，并对自己的邮件进行管理。

2.2 SMTP 的通信原理

2.2.1 SMTP 通信模型

SMTP（Simple Mail Transfer Protocol，简单邮件传输协议）是 Internet 上基于 TCP/IP 的应用层协议，使用于主机与主机之间的电子邮件交换，其模型如图 2-1 所示。SMTP 的特点是简单，它只定义了邮件发送方和接收方之间的连接传输，将电子邮件有一台计算机传送到另一台计算机，而不规定其他任何操作，如用户界面的交互、邮件的接收、邮件存储等。Internet 上几乎所有主机都运行着遵循 SMTP 的电子邮件软件，因此使用非常普遍。另一方面，SMTP 由于简单，因而有其一定的局限性，它只能传送 ASCII 文本文件，而对于一些二进制数据文件需要进行编码后才能传送。

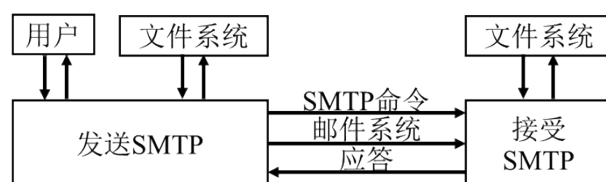


图 2-1 SMTP 模型

2.2.2 SMTP 的命令与应答

从前面的 SMTP 通讯模型可以看出发送端和服务端之间的会话是靠发送端 SMTP 命令和服务端反馈的应答来完成的。SMTP 命令是发送于 SMTP 主机之间的 ASC II 信息，可能使用到的命令如表 2-1 所示。

表 2-1 SMTP 协议命令

SMTP 命令	描述
DATA	开始信息写作
EXPN<string>	验证给定的邮箱列表是否存在，扩充邮箱列表，也常被禁用
HELO<domain>	向服务器标识用户身份，返回邮件服务器身份
HELP<command>	查询服务器支持什么命令，返回命令中的信息
MAIL FROM<host>	在主机上初始化一个邮件会话
NOOP	无操作，服务器应响应 OK
QUIT	终止邮件对话
RCPT TO< user >	标识单个的邮件接收人，常在 MAIL 命令后面可有多个 RCPT TO
RSET	重置会话，当前传输被取消
SAML FROM<host>	发送邮件到用户终端和邮箱
SEND FROM<host>	发送邮件到用户终端
EHLO<domain>	和 HELO 命令作用相同，支持 SMTP 服务扩展的客户用该命令
TURN	接收端和发送端交换角色
VERFY<user>	用于验证指定的用户/邮箱是否存在，由于安全方面的原因，服务器常禁止此命令

SMTP 协议的每一个命令都会返回一个应答码，应答码的每一个数字都是有特定含义的，如第一位数字为 2 时表示命令成功；为 5 表示失败；3 表示没有完成。一些较复杂的邮件程序利用该特点，首先检查应答码的首数字，并根据其值来决定下一步的动作。SMTP 的应答码列表如表 2-2 所示：

表 2-2 应答码列表

应答码	含义	应答码	含义
501	参数格式错误	251	用户非本地
502	命令不可实现	450	要求的邮件操作未完成，邮箱不可用
503	错误的命令序列，接受邮箱格式错误	550	要求的邮件操作未完成，邮箱不可用
504	命令参数不可实现	451	放弃要求的操作，处理过程中出错
211	系统状态或系统帮助响应	551	用户非本地
214	帮助信息	452	系统存储不足，要求的操作未执行
220	服务就绪	554	操作失败
221	服务关闭	552	过量的存储分配，要求的操作未执行

421	服务未就绪，关闭传输信道	553	邮箱名不可用，要求的操作未执行
250	要求的邮件操作完成	354	开始邮件输入，以“.”结束

2.2.3 SMTP 的会话过程

发送端发送一封邮件时，会与其服务器进行 SMTP 会话，会话过程大致有发送端与服务器建立 TCP 连接，邮件传输和终止 TCP 连接三个过程，如图 2-2 所示。

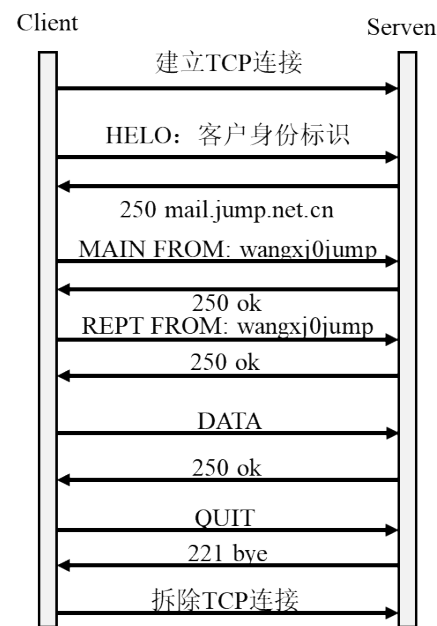


图 2-2 SMTP 会话过程

(1) 建立连接

SMTP 发送端和 SMTP 接收端建立连接即 TCP 三次握手。其连接过程如图 2-3 所示。

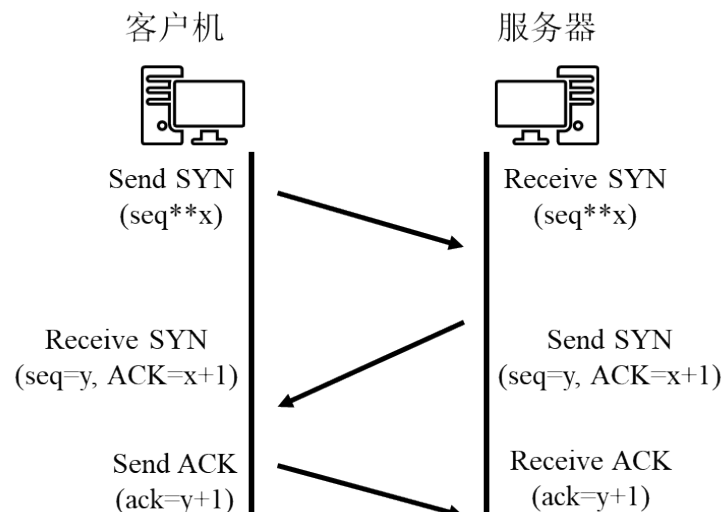


图 2-3 TCP 连接建立

(2) 邮件传输

客户端发送 EHLO 命令以标识发件人自己的身份，表示要发送一封邮件，服务器返回同意应答 OK。客户端向服务器发送 AUTH 命令，声明自己要求使用的认证方式，再服务端确认之后，输入以 BASE64 方式编码的用户名和口令，服务端认证通过后，客户端就可以开始发送邮件了。然后客户端发送 MAIL FROM 命令，表明邮件发送端的邮箱地址，如果服务器端接收，则以 OK 作为应答。客户端发送 RCPT TO 命令，以标识该电子邮件的计划接收人，可以有多个 RCPT 行；服务器端则表示是否愿意为收件人接收邮件，如果愿意则返回 OK，否则返回拒绝接受的应答，然后会继续重复这个操作。协商结束，客户端发送 DATA 命令，服务端返回确认应答，开始输入邮件内容并以“.”号表示结束输入，结束此次发送，服务端返回确认应答。客户端最后用 QUIT 命令退出。

(3) TCP 会话终止

终止 TCP 会话原则是当一方完成它的数据发送任务后就能发送一个 FIN 来终止这个方向的连接。收到一个 FIN 只意味着这一方向上没有数据流动，一个 TCP 连接在收到一个 FIN 后仍能发送数据。首先进行关闭的一方将执行主动关闭，而另一方执行被动关闭。TCP 采用四次挥手关闭连接如图 2-4 所示。

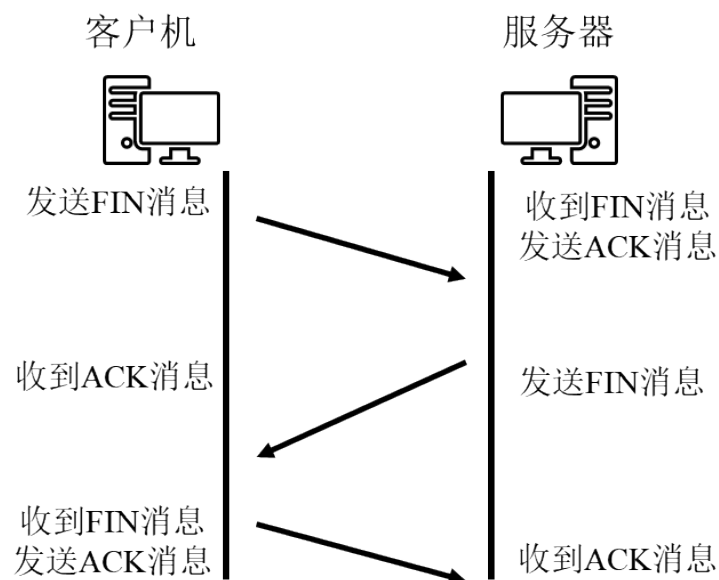


图 2-4 TCP 连接的关闭过程

2.3 数据包的封装和拆封

2.3.1 数据包的封装

封装是从应用层向物理层一层层封装下去的，电子邮件必须被封装成帧的形式才能在网络上传输。数据包的封装，如图 2-5 所示，从图中我们可以看出，应用程序接收的数据包不仅包含邮件本身的内容，还有从应用层向下一层层打上的数据包头信息。

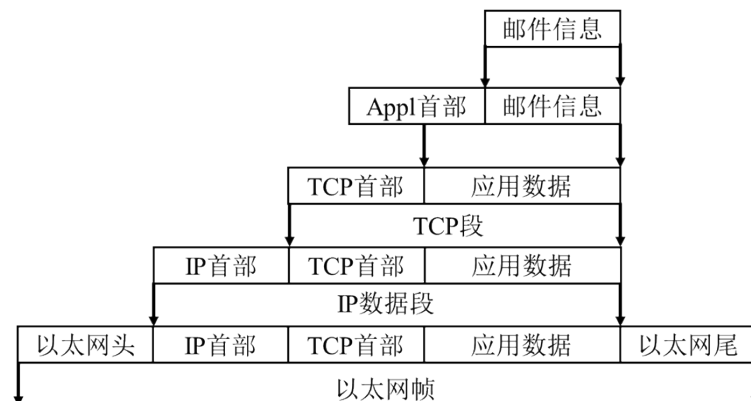


图 2-5 数据包封装

2.3.2 数据包的拆封

因为数据包捕获程序工作在网络底层，所以将网卡设置为混杂模式以后，从网络底层捕获到的数据包包含有数据内容和包头信息。需要对数据包进行拆封，去掉包头信息。邮件的拆封就是封装的逆过程。具体邮件拆封如图 2-6 所示。

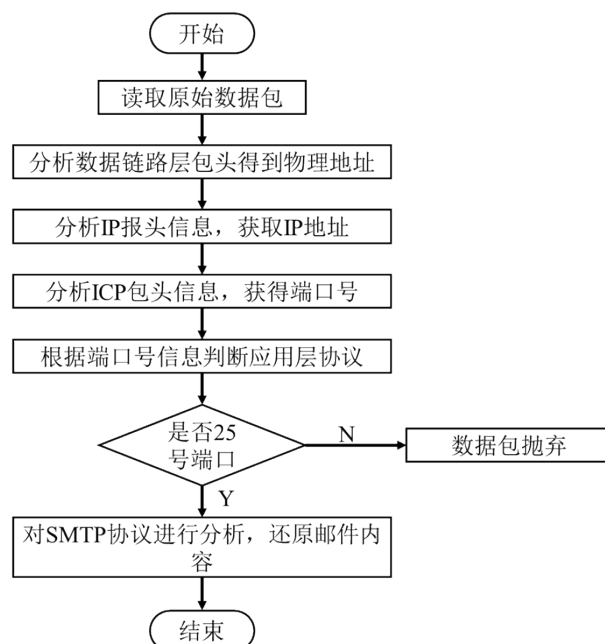


图 2-6 数据包的拆封

3 基于 Python 的 SMTP 数据包分析与还原的实现

收取邮件就是编写一个 MUA 作为客户端，从 MDA 把邮件获取到用户的电脑或者手机上。收取邮件最常用的协议是 POP 协议，目前版本号是 3，俗称 POP3。Python 内置一个 poplib 模块，实现了 POP3 协议，可以直接用来收邮件。注意到 POP3 协议收取的不是一个已经可以阅读的邮件本身，而是邮件的原始文本，这和 SMTP 协议很像，SMTP 发送的也是经过编码后的一大段文本。要把 POP3 收取的文本变成可以阅读的邮件，还需要用 email 模块提供的各种类来解析原始文本，变成可阅读的邮件对象。所以，收取邮件分两步：第一用 poplib 把邮件的原始文本下载到本地；第二用 email 解析原始文本，还原为邮件对象。

3.1 捕获数据包

通过 POP3 下载邮件，POP3 协议本身很简单，以下面的代码为例，来捕获一封邮件内容：

```
import poplib

# 输入邮件地址， 口令和 POP3 服务器地址:
email = raw_input('Email: ')
password = raw_input('Password: ')
pop3_server = raw_input('POP3 server: ')

# 连接到 POP3 服务器:
server = poplib.POP3(pop3_server)

# 可以打开或关闭调试信息:
# server.set_debuglevel(1)

# 可选:打印 POP3 服务器的欢迎文字:
print(server.getwelcome())

# 身份认证:
server.user(email)
server.pass_(password)

# stat()返回邮件数量和占用空间:
print('Messages: %s. Size: %s' % server.stat())

# list()返回所有邮件的编号:
```

```

resp, mails, octets = server.list()
# 可以查看返回的列表类似['1 82923', '2 2184', ...]
print(mails)
# 获取最新一封邮件， 注意索引号从 1 开始:
index = len(mails)
resp, lines, octets = server.retr(index)
# lines 存储了邮件的原始文本的每一行，
# 可以获得整个邮件的原始文本:
msg_content = '\r\n'.join(lines)
# 稍后解析出邮件:
msg = Parser().parsestr(msg_content)
# 可以根据邮件索引号直接从服务器删除邮件:
# server.delete(index)
# 关闭连接:
server.quit()

```

用 POP3 获取邮件其实很简单，要获取所有邮件，只需要循环使用 `retr()` 把每一封邮件内容拿到即可。

3.2 解析还原数据包

解析邮件的过程和上一节构造邮件正好相反，因此，先导入必要的模块：

```

import email
from email.parser import Parser
from email.header import decode_header
from email.utils import parseaddr

```

然后加入如下所示的一行代码就可以把邮件内容解析为 `Message` 对象：

```
msg = Parser().parsestr(msg_content)
```

但是这个 `Message` 对象本身可能是一个 `MIMEMultipart` 对象，即包含嵌套的其他 `MIMEBase` 对象，嵌套可能还不止一层。所以要递归地打印出 `Message` 对象的层次结构：

```
# indent 用于缩进显示:
```

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。

如要下载或阅读全文，请访问：

<https://d.book118.com/435303310304012032>