PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100 Feet Ring Road, BSK III Stage, Bengaluru-560 085

Department of Electronics and Communication Engineering

Course Title: Computer Communication Networks
Course Code: UE21EC351A

PROFESSOR: DR. VAMSHI KRISHNA T.

Project Title: CONFIGURING EMAIL SERVERS AND SHOWING END TO END FLOW OF EMAIL BETWEEN HOST

Done By:
Name: AKASH CHANABASU KATATE
SRN: PES1UG21EC022
NAME: ABHISHEK SHETTY
SRN: PES1UG21EC008
NAME: TEJAS V P
SRN: PES1UG21EC910
NAME: AKASH RAVI BHAT
SRN: PES1UG21EC025

Abstract

In today's digital era, emails remain an integral mode of communication, housing vital information critical to personal and professional spheres. This project focuses on harnessing the capabilities of Python's imaplib library to streamline the retrieval and potential processing of emails through the IMAP (Internet Message Access Protocol) protocol.

The project primarily aims to develop a Python script capable of securely connecting to diverse IMAP servers, accessing designated email accounts, and extracting pertinent information from received emails. Leveraging imaplib, the script establishes encrypted connections, enabling seamless communication between email clients and servers.

Table of Contents

1. Introduction

Emails serve as a crucial means of communication in today's digital landscape, housing valuable information and correspondence. To streamline email management and harness the power of automation, a Python project leveraging the imaplib library has been developed.

This project facilitates the retrieval and potential processing of emails from various email accounts using the IMAP (Internet Message Access Protocol) protocol. With Python's imaplib, the script connects securely to different IMAP servers, accesses inboxes, and extracts essential information from received emails.

1.1 The primary objectives of this project include:

Email Retrieval: Establishing secure connections to IMAP servers and retrieving emails from designated email accounts.

Content Extraction: Parsing email messages to extract specific details such as subjects, senders, and content.

Potential Automation: Offering a foundation for implementing automation, enabling tasks like email sorting, notification triggering, or data extraction based on predefined criteria.

Customization and Expansion: Providing a framework that can be expanded upon or customized to suit specific use cases or integrate with broader applications.

Throughout this project, the aim is to showcase the capabilities of Python's imaplib library in facilitating email management, demonstrating secure connections to IMAP servers and laying the groundwork for advanced email processing and automation.

1.2 Algorithms

Code 1: Retrieving Email Subjects

1. **Establish IMAP Connection:**
   - Initialize an IMAP connection to the specified server using `imaplib.IMAP4_SSL`.
   - Log in to the email account with the provided credentials.

2. **Select Mailbox:**
   - Select the "inbox" mailbox using `mail.select('inbox')`.

3. **Search for Emails:**
   - Search for all emails in the selected mailbox using `mail.search(None, 'ALL')`.

4. **Retrieve and Display Email Subjects:**
   - Iterate through the list of email IDs obtained from the search results.
   - For each email ID:
     - Fetch the email content using `mail.fetch(email_id, '(RFC822)')`.
     - Parse the raw email content using `email.message_from_bytes`.
     - Extract and decode the email subject using `decode_header`.
     - Print the decoded subject.

5. **Error Handling and Logout:**
   - Use a `try` block to catch any exceptions that may occur during the process.
   - Ensure a proper logout from the IMAP server in the `finally` block.

Code 2: Retrieving Entire Emails

1. **Establish IMAP Connection (Gmail):**
   - Initialize an IMAP connection to Gmail's server using `imaplib.IMAP4_SSL`.
   - Log in to the recipient's email account with the provided credentials.

2. **Select Mailbox:**
   - Select the "inbox" mailbox using `mail.select("inbox")`.

3. **Search for Emails:**
   - Search for all emails in the selected mailbox using `mail.search(None, "ALL")`.

4. **Retrieve and Display Entire Emails:**
   - Iterate through the list of email IDs obtained from the search results.
   - For each email ID:
     - Fetch the email content using `mail.fetch(email_id, "(RFC822)")`.
     - Decode the raw email content using `decode("utf-8")`.
     - Print the entire decoded email content.

5. **Error Handling and Logout:**
   - Use a `try` block to catch any exceptions that may occur during the process.
   - Ensure a proper logout from the IMAP server in the `finally` block.

These algorithms outline the logical flow of each section, from establishing a connection to the IMAP server, navigating mailboxes, searching for emails, retrieving and processing email content, handling errors, and finally, ensuring a clean logout. They provide a structured guide for implementing the functionality described in the code snippets.

2. Problem Statement

2.1 Requirement

This problem statement outlines the tasks and expectations for developing a Python script that interacts with IMAP servers to retrieve and process email data. It provides a clear set of functionalities and constraints to guide the implementation.

2.2 The goal

The script should be capable of autonomously connecting to IMAP servers, retrieving email subjects and entire email content, handling potential errors gracefully, and ensuring a secure logout. It serves as a tool for automating the initial steps of email processing, providing a foundation for more advanced email automation or integration into larger applications.

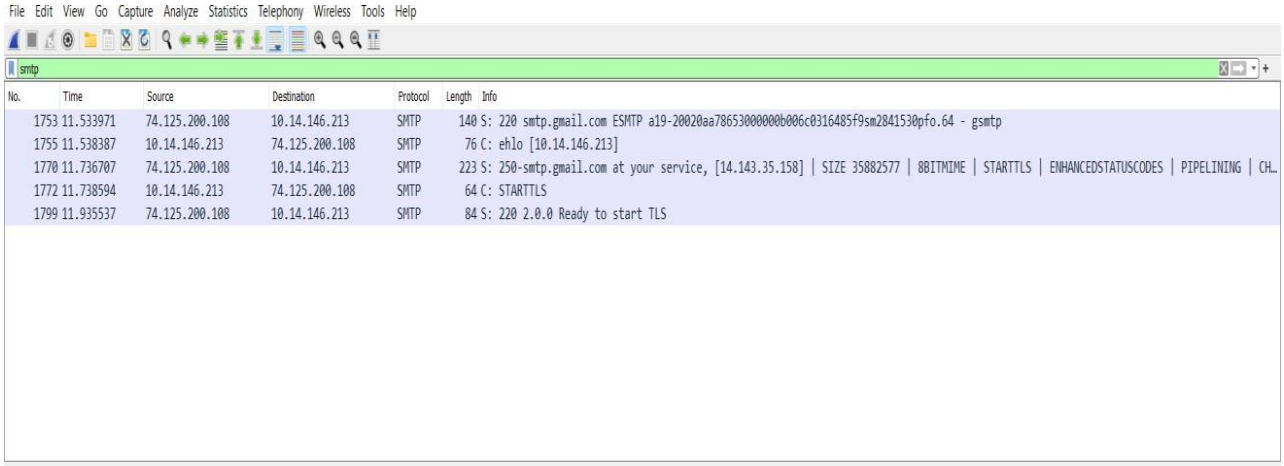
3. Methodology

SENDER'S OUTPUT

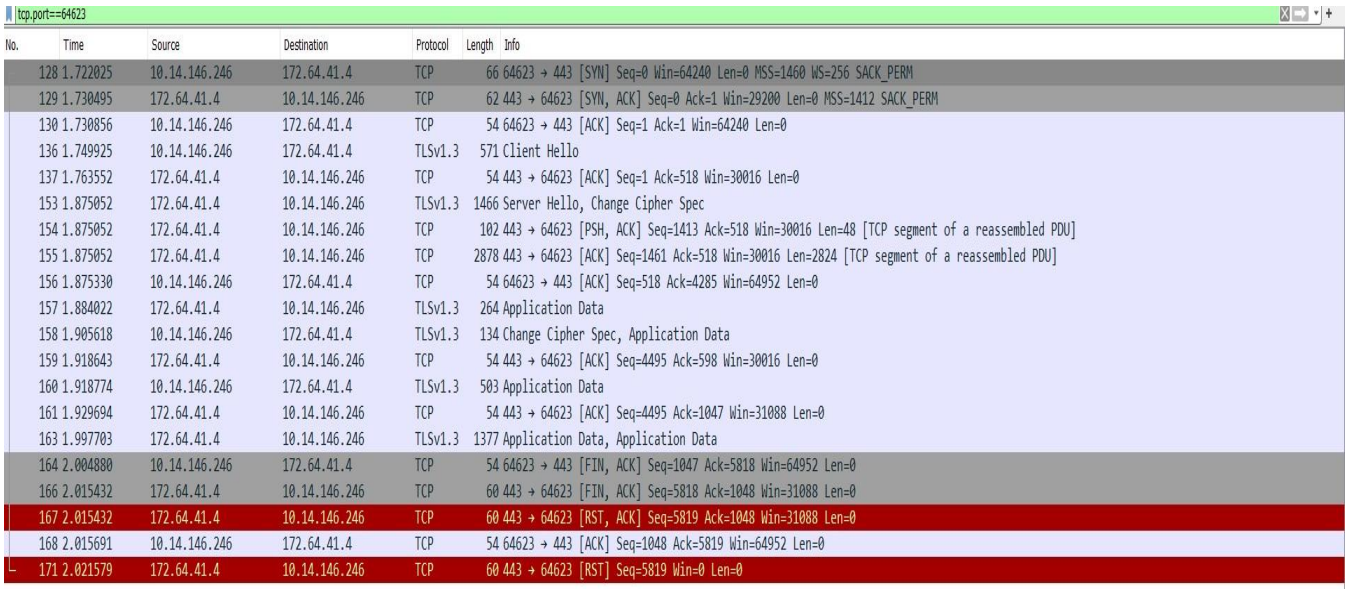
RECIVER'S OUTPUT

```
X-Google-Smtp-Source: AGHT+IHImU6anSvLLI+1FC5b+HBqcRUjktKVtUDTjlY6nLxfkP1Ny+U9PVlyxkHUvdHXx2P/thfLow==
X-Received: by 2002:a17:902:e5cd:b0:1cf:51c5:d427 with SMTP id u13-20020a170902e5cd00b001cf51c5d427mr8642292plf.65.1700555847101;
        Tue, 21 Nov 2023 00:37:27 -0800 (PST)
Return-Path: <akashkatate2015@gmail.com>
Received: from [10.14.146.246] ([1.6.180.189])
        by smtp.gmail.com with ESMTPSA id bj9-20020a170902850900b001b9e9edbf43sm7338195plb.171.2023.11.21.00.37.25
        for <techakashkatate@gmail.com>
        (version=TLS1_3 cipher=TLS_AES_256_GCM_SHA384 bits=256/256);
        Tue, 21 Nov 2023 00:37:26 -0800 (PST)
Message-ID: <655c6c46.170a0220.d581a.2f17@mx.google.com>
Date: Tue, 21 Nov 2023 00:37:26 -0800 (PST)
Content-Type: multipart/mixed; boundary="===============4985460977834331642=="
MIME-Version: 1.0
From: akashkatate2015@gmail.com
To: techakashkatate@gmail.com
Subject: Hello Akash kaatate how are you

--===============4985460977834331642==
Content-Type: text/plain; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit

This is a test email sent from Python.
--===============4985460977834331642==--
```

WIRESHARK ANALYSIS SMTP



WIRESHARK ANALYSIS FOR RECIVER



4. Result

Each line corresponds to an email in the inbox, and the text after "Subject:" is the decoded subject of that email. The actual subjects will vary based on the content of the emails in the specified inbox. If there is an error during the execution (e.g., connection issues, login failure), an error message will be printed instead.

Keep in mind that you need to replace 'your_imap_server.com', 'recipient@example.com', and 'your_password' with your actual IMAP server address, email account, and password for this code to work with your email account.

5. Conclusion and future scope

Conclusion:

The provided Python script demonstrates a basic functionality for connecting to an IMAP server, logging in, and retrieving and printing the subjects of emails in the inbox. The script effectively uses the `imaplib` library to interact with the IMAP server and the `email` library to parse and decode email content. Error handling is incorporated to handle potential exceptions during the IMAP operations, ensuring a graceful exit from the script.

Future Scope:

1. Email Content Retrieval:
   - Extend the script to retrieve and display the entire content of emails, including the message body and any attachments.

2. Filtering and Sorting:
   - Implement functionality to filter and sort emails based on criteria such as date, sender, or specific keywords.

3. Automated Actions:
   - Integrate automated actions, such as marking emails as read, moving emails to different folders, or flagging important messages.

4. User Interface:
   - Develop a graphical user interface (GUI) to enhance user interaction and provide a more user-friendly experience.

5. Multiple Mailbox Support:
   - Modify the script to handle multiple mailboxes and allow users to choose the mailbox they want to interact with.

6.Reference:

- Python **imaplib** Documentation
- Python **email** Documentation

7,TEAMATES CONTRIBUTIONS

Written by Abhishek A Shetty

We started the Project in GJBC library (2nd floor) on 4/11/23 after college at 2pm, I and Akash katate started doing analysis from sender side size, Akash bhat and tejas did analysis from receiver side .

All we were present at that date we divided the team work equally, Akash katate coded code for sender side i did analysis for smtp in the wireshark we both uploaded the mail to the server.

Tejas did coding part for receiver side and Akash Bhat did wireshark analysis for that, as a team we have analysed smtp and imap protocol.

Everyone tried their best to in analysing the code and wireshark.

Written by Akash katate

Akash Chanabasu Katate:We have started the Project in GJBC library (2nd floor) on 6/11/23 after college at 2pm .I helped in doing the Python script for sending an email using Smtp lib in python.

Abhishek Shetty:He helped in capturing the packets through wire shark of sender's side he analysed the end-end smtp components in sender's side SMTP packets and reported it

Tejas V P : He done with the receiver's side python script using imap

library

Akash Ravi Bhat: He done with wire shark analyses part of receiver's side and observed the various components how actually emails are receiving using imap using tcp connections

Written by Tejas V P

We have started the Project in GJBC library (2nd floor) on 6/11/23 after college at 2pm .
All the teammate were present that day so we decided to divide the project into 2 parts as sender and receiver side , so Akash chanabasu katate and abhishek shetty started work for sender side , me (tejas) and akash bhat started doing for receiver side .
as we faced some error we decided to do it in next day after college ,

and we done that part .
but it took one more day to understand the wireshark captures , we have joined through google meet , as it was Saturday . and we finished the project by 8pm .
akash katate was successfully able to write complete code of sender side and abhishek joined hands with him and also he done wireshark capture and he explained us about it.

I am able to complete receiver side python code and during that time akash bhat helped me and he also wireshark capturing successfully and he was able to explain it to all of us.

WRITTEN BY AKASH RAVI BHAT:

We chose our project name as configuring email servers on the day of distribution of lists of projects then we started our project on November 6th.

we divided our project into two parts , one is about smtp and another one is about tcp connection eshtablsihments.

we started this using different smtp servers but we got error in some them , we analysed Wireshark using different online platforms and our lab manual

we analysed different type of secure connections during tcp connection using imap.

Me and Tejas complete receiver side python code and we analysed different flags occurred during Wireshark captures Akash Katate and Abhishek shetty did sender side python code and wireshark analysis

We completed report on 22 november 2023

FINALLY IT WAS TOTALLY TEAM WORK

THANK YOU ALL!