School of Computer Engineering and Technology
Academic Year: 2023-2024 Sem V
Digital Forensics and Investigation

Lab Assignment : 05

Title: Email Header Analysis

Prepared By
Saurabh Jitendra Jadhav
Roll No:PA12
Batch A1
October 02,2023

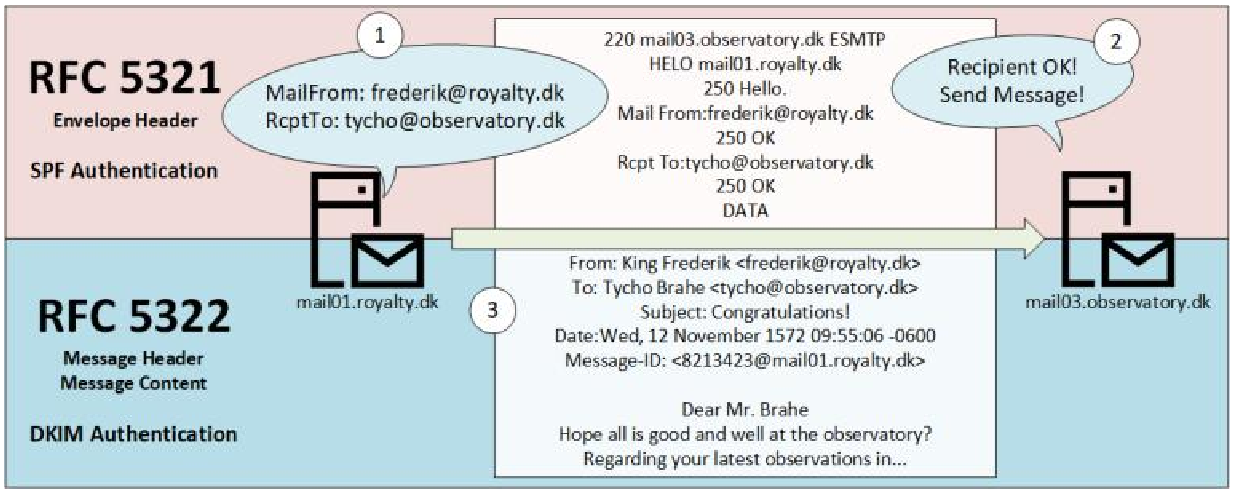
Aim: To write a C++/Java/python program to analyse email header.

Objective:

- ❖ To develop a program in [C++/Java/Python] that parses and extracts key information from email headers.
- ❖ To create a tool that identifies and displays sender details and routing paths within email headers.
- ❖ To gain hands-on experience in email header analysis for digital forensics through practical programming.

Theory:

RFC 5322:-


RFC 5321
Envelope Header
SPF Authentication
1
MailFrom: frederik@royalty.dk
RcptTo: tycho@observatory.dk
220 mail03.observatory.dk ESMTP
HELO mail01.royalty.dk
250 Hello.
Mail From:frederik@royalty.dk
250 OK
Rcpt To:tycho@observatory.dk
250 OK
DATA
2
Recipient OK! Send Message!
RFC 5322
Message Header
Message Content
DKIM Authentication
mail01.royalty.dk
3
From: King Frederik <frederik@royalty.dk>
To: Tycho Brahe <tycho@observatory.dk>
Subject: Congratulations!
Date:Wed, 12 November 1572 09:55:06 -0600
Message-ID: <8213423@mail01.royalty.dk>
Dear Mr. Brahe
Hope all is good and well at the observatory?
Regarding your latest observations in...
mail03.observatory.dk

RFC 5322 is a standard that defines the format of internet messages, such as email messages. It specifies the structure and content of email messages, including the headers, body, and attachments. The standard is maintained by the Internet Engineering Task Force (IETF) and is an important reference for anyone working with email or other internet messages. It is also known as the Internet Message Format Standard.

Message Header Fields:
The message header fields are essential components of an email message. They are located at the beginning of the email and contain metadata and control information necessary for the proper routing and delivery of the message.

Message header fields include elements like "From," "To," "Subject," "Date," "Message-ID," and "Received," among others. These fields provide critical information about the message's sender, recipients, subject, timestamp, and the message's path through various email servers.
Message header fields are key to understanding the provenance and route of an email, making them invaluable in digital forensics and email analysis.

Message Body:

The message body follows the message header and contains the actual content of the email, including text, images, attachments, and any multimedia elements. The message body is the part of the email that is visible to the recipient and contains the message's intended communication.

SPF:-
SPF or Sender Policy Framework is an authentication method used by senders to specify hosts that are allowed to send an email on behalf of the domain. MTA (mail transfer agent) checks the sender’s DNS records to confirm that the email received from a domain is sent by a host listed in the sender’s DNS records.

DKIM-Signature:-
DKIM-Signature or Domain Keys Identified Mail (DKIM) is another authentication method used to confirm that the email was authorized by the owner of the domain. The email is signed with a digital signature, which can be verified by checking the sender’s public key in the DNS records of the sender’s domain.

Message Header Fields in RFC 5322:
Email message header fields provide vital information about the email, its source, and its routing.some of them are as follows:-

From: This field identifies the sender of the email.

To: Specifies the primary recipient(s) of the email.

Subject: Contains a brief description of the email's content.

Date: Indicates the date and time when the email was sent.

Message-ID: Provides a unique identifier for the email message.

Received: Shows a history of the servers and timestamps through which the email has passed, helping trace its path.

CC (Carbon Copy): Lists additional recipients who receive a copy of the email.

BCC (Blind Carbon Copy): Lists recipients who receive a copy without the knowledge of other recipients.

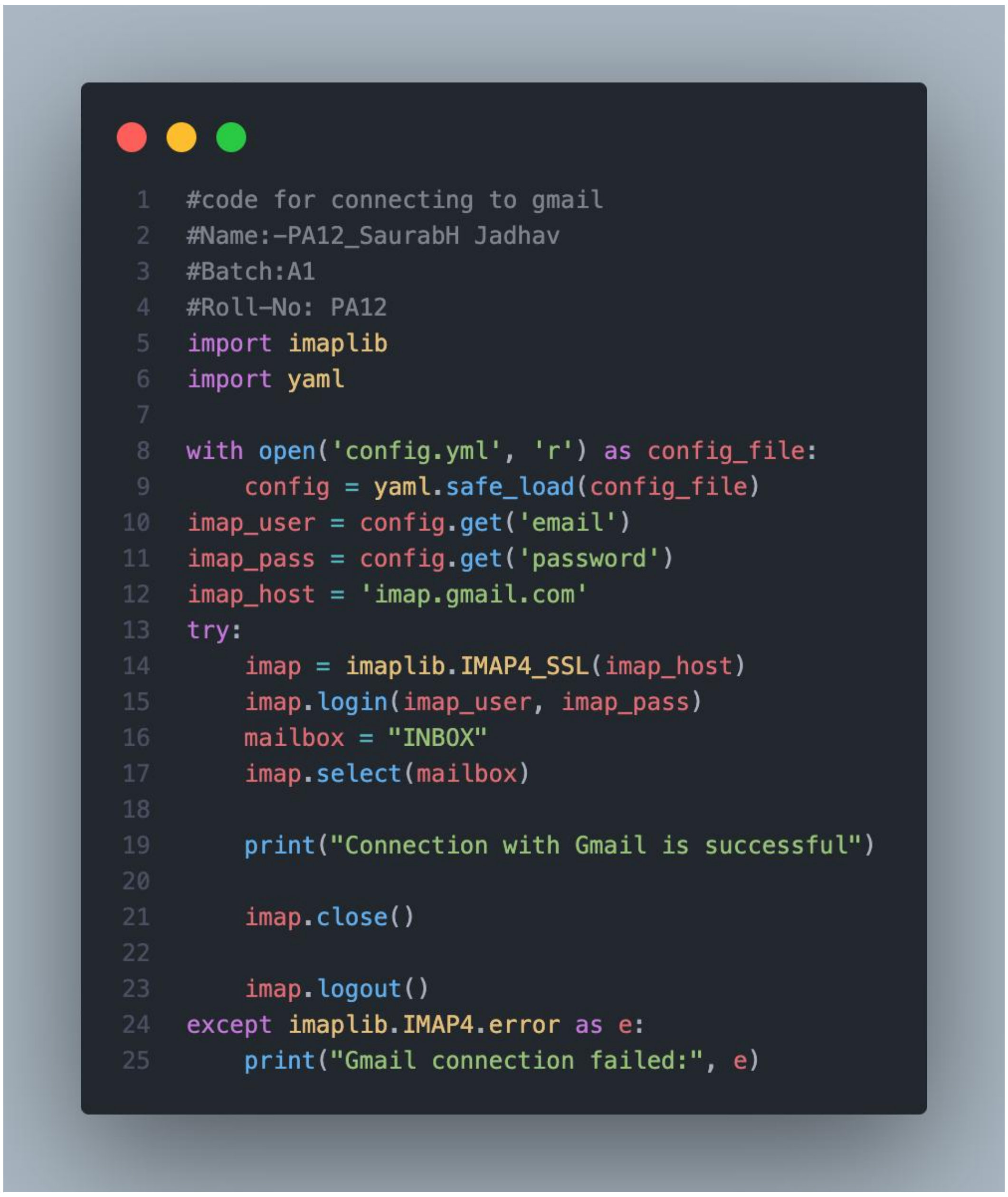
Reply-To: Specifies the email address to which replies should be sent.

MIME-Version: Identifies the MIME (Multipurpose Internet Mail Extensions) version used to format the message.

Implementation:-
To implement Email Header Analysis first we have to Enable IMAP:

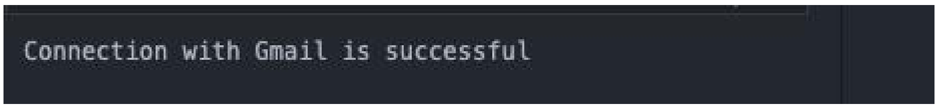
Configure Gmail for IMAP:
- ❖ 1)Log in to your Gmail account.
- ❖ 2)Click on the gear icon in the upper-right corner and select "See all settings."
- ❖ 3)Go to the "Forwarding and POP/IMAP" tab.
- ❖ 4)In the "IMAP Access" section, ensure that "Enable IMAP" is selected.
- ❖ 5)Save your changes.
- ❖ 6)Generate an App Password (If Two-Factor Authentication Is Enabled)

a) Go to your Google Account settings: https://myaccount.google.com/.
b) Click on "Security" on the left sidebar.
c) Under the "Signing in to Google" section, click on "App passwords."
d) In the "App passwords" section, select "Mail" for the app and "Other (Custom Name)" for the device.
e) Click "Generate."

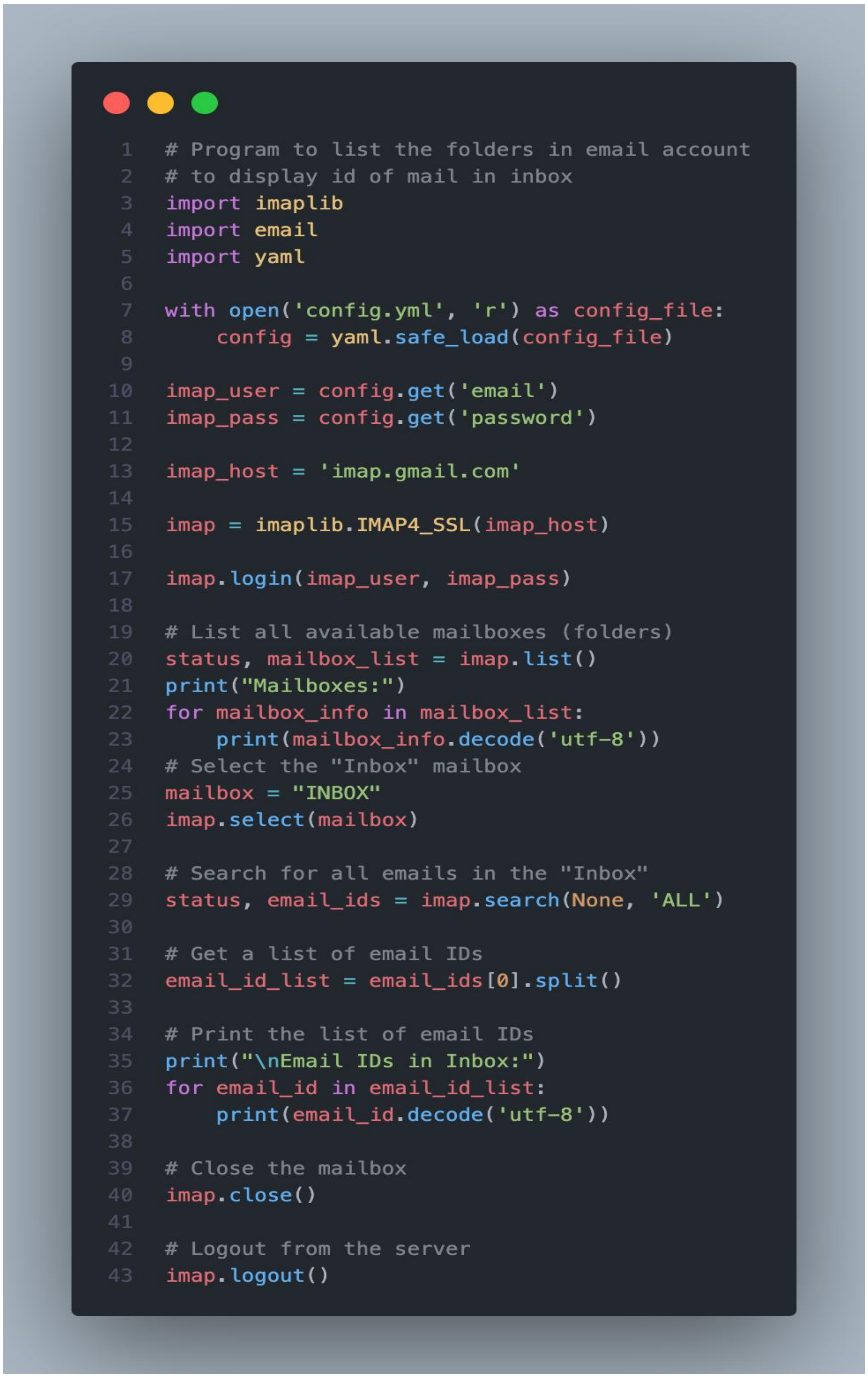


```python
#code for connecting to gmail
#Name:-PA12_SaurabH Jadhav
#Batch:A1
#Roll-No: PA12
import imaplib
import yaml

with open('config.yml', 'r') as config_file:
    config = yaml.safe_load(config_file)
imap_user = config.get('email')
imap_pass = config.get('password')
imap_host = 'imap.gmail.com'
try:
    imap = imaplib.IMAP4_SSL(imap_host)
    imap.login(imap_user, imap_pass)
    mailbox = "INBOX"
    imap.select(mailbox)

    print("Connection with Gmail is successful")

    imap.close()

    imap.logout()
except imaplib.IMAP4.error as e:
    print("Gmail connection failed:", e)
```

Output:

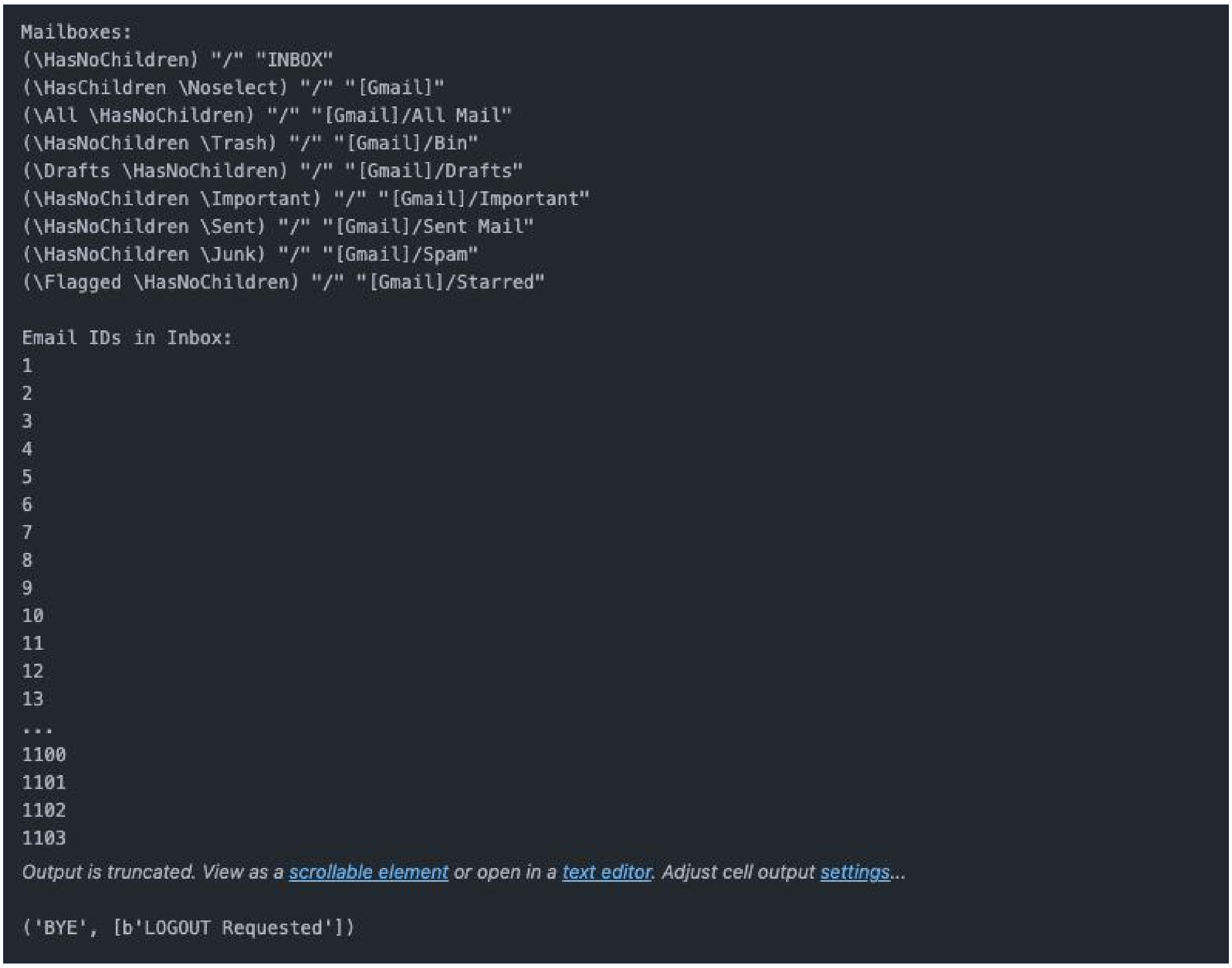


```
Connection with Gmail is successful
```

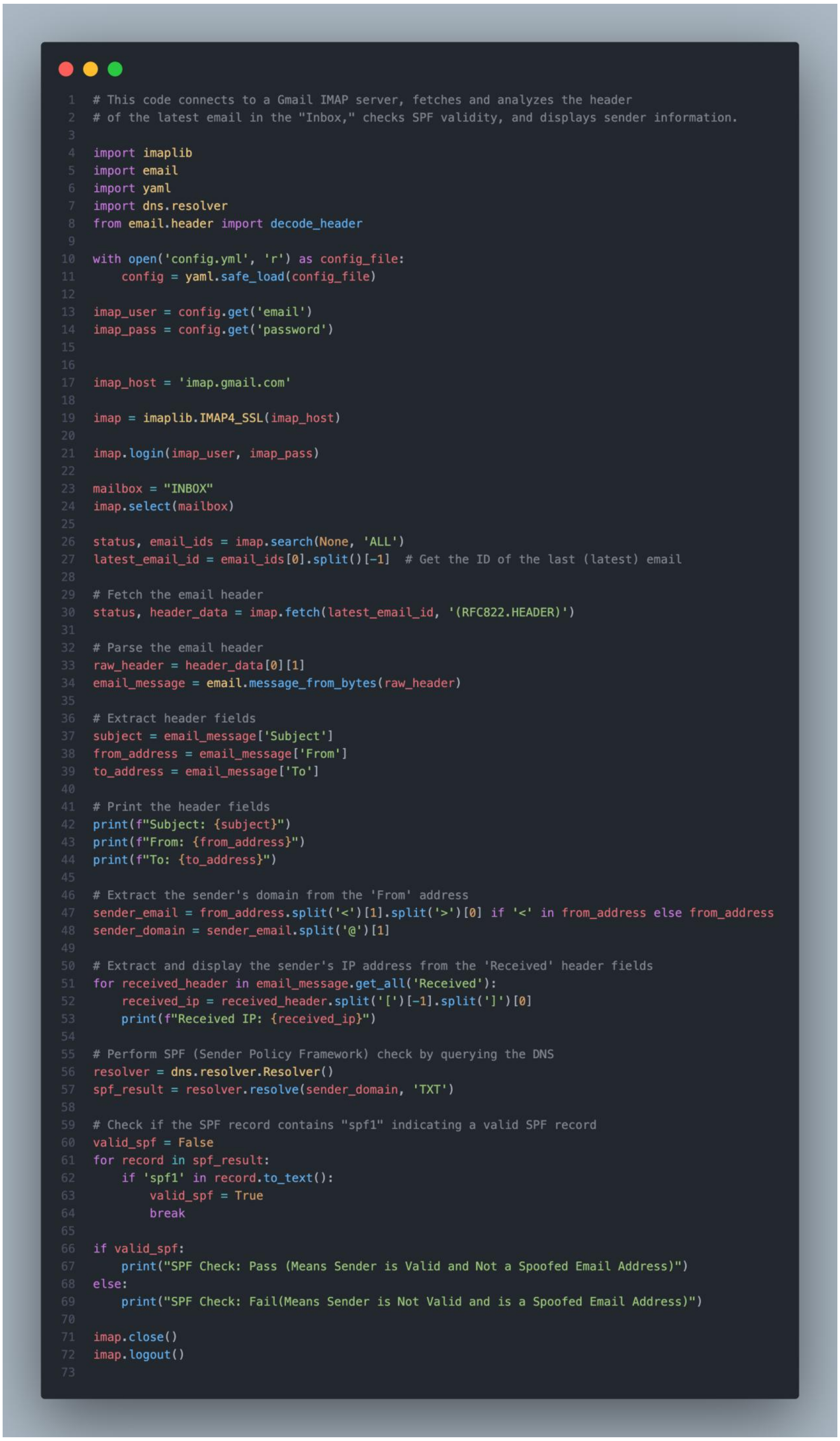


```python
# Program to list the folders in email account
# to display id of mail in inbox
import imaplib
import email
import yaml

with open('config.yml', 'r') as config_file:
    config = yaml.safe_load(config_file)

imap_user = config.get('email')
imap_pass = config.get('password')

imap_host = 'imap.gmail.com'

imap = imaplib.IMAP4_SSL(imap_host)

imap.login(imap_user, imap_pass)

# List all available mailboxes (folders)
status, mailbox_list = imap.list()
print("Mailboxes:")
for mailbox_info in mailbox_list:
    print(mailbox_info.decode('utf-8'))
# Select the "Inbox" mailbox
mailbox = "INBOX"
imap.select(mailbox)

# Search for all emails in the "Inbox"
status, email_ids = imap.search(None, 'ALL')

# Get a list of email IDs
email_id_list = email_ids[0].split()

# Print the list of email IDs
print("\nEmail IDs in Inbox:")
for email_id in email_id_list:
    print(email_id.decode('utf-8'))

# Close the mailbox
imap.close()

# Logout from the server
imap.logout()
```

Output:-

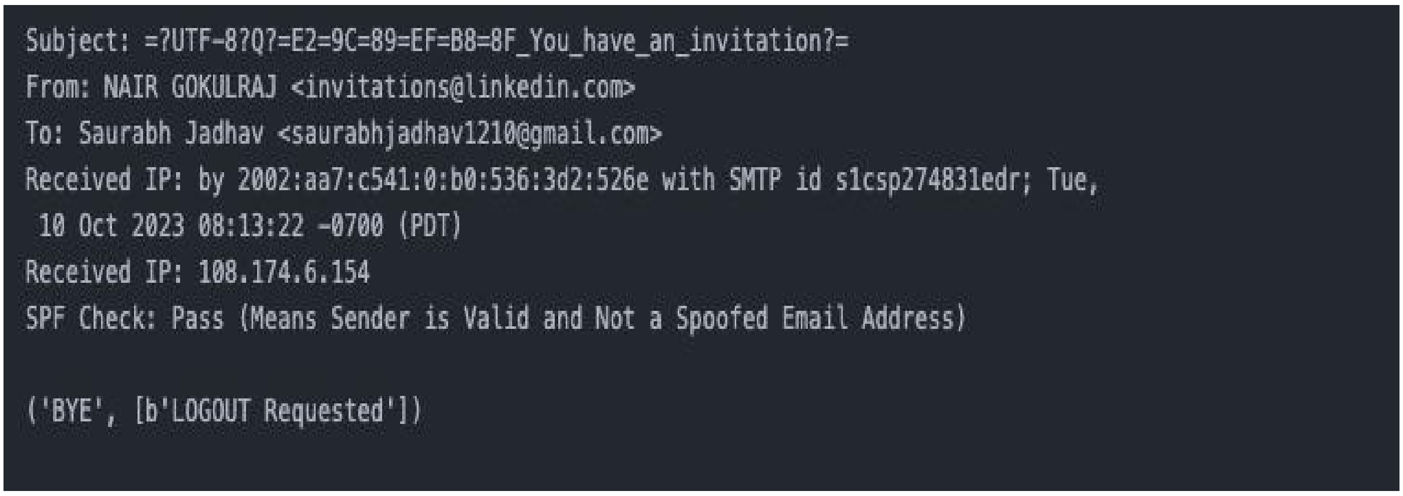


```
Mailboxes:
(\HasNoChildren) "/" "INBOX"
(\HasChildren \Noselect) "/" "[Gmail]"
(\All \HasNoChildren) "/" "[Gmail]/All Mail"
(\HasNoChildren \Trash) "/" "[Gmail]/Bin"
(\Drafts \HasNoChildren) "/" "[Gmail]/Drafts"
(\HasNoChildren \Important) "/" "[Gmail]/Important"
(\HasNoChildren \Sent) "/" "[Gmail]/Sent Mail"
(\HasNoChildren \Junk) "/" "[Gmail]/Spam"
(\Flagged \HasNoChildren) "/" "[Gmail]/Starred"

Email IDs in Inbox:
1
2
3
4
5
6
7
8
9
10
11
12
13
...
1100
1101
1102
1103
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

('BYE', [b'LOGOUT Requested'])
```

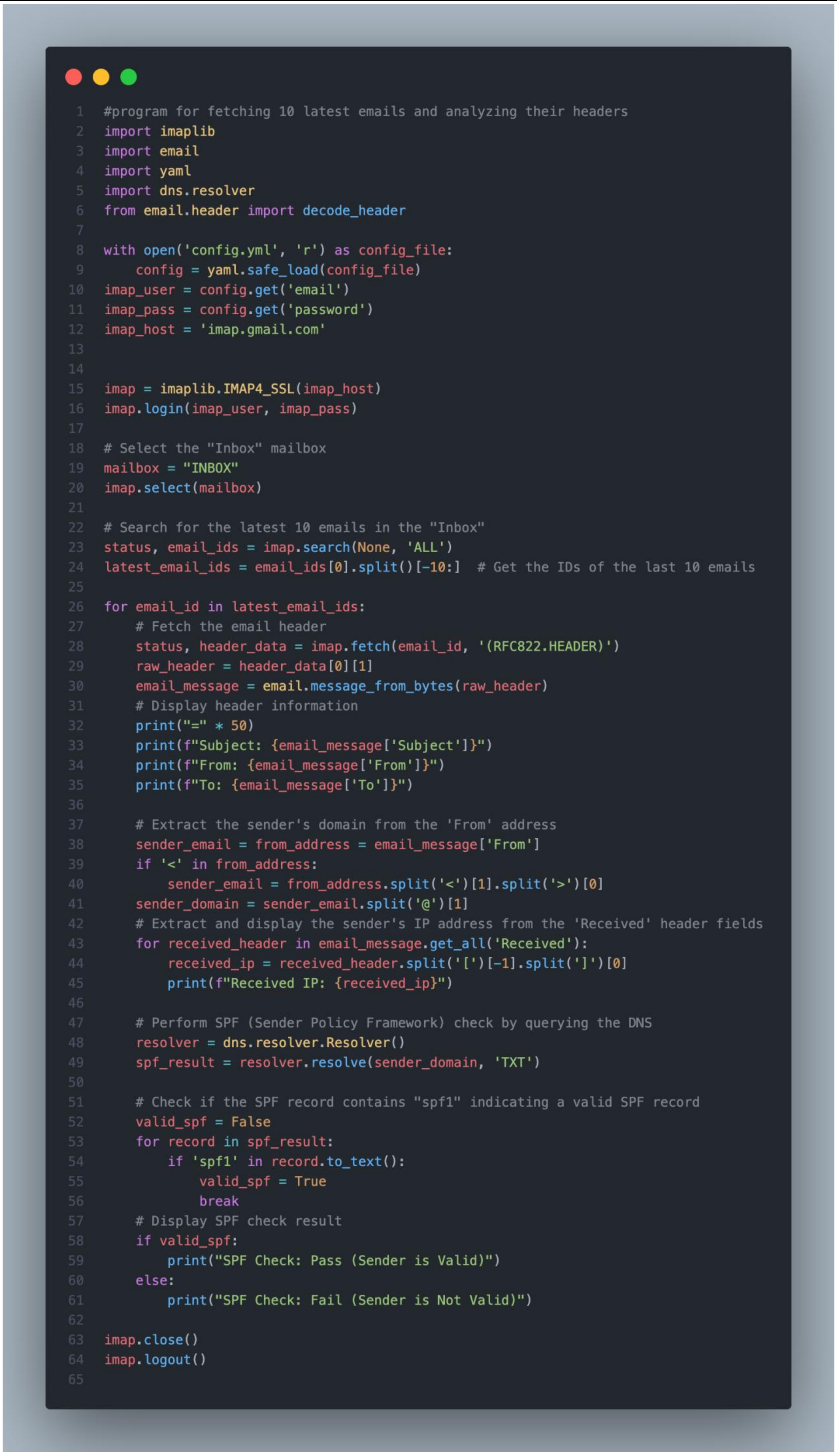


```python
# This code connects to a Gmail IMAP server, fetches and analyzes the header
# of the latest email in the "Inbox," checks SPF validity, and displays sender information.

import imaplib
import email
import yaml
import dns.resolver
from email.header import decode_header

with open('config.yml', 'r') as config_file:
    config = yaml.safe_load(config_file)

imap_user = config.get('email')
imap_pass = config.get('password')


imap_host = 'imap.gmail.com'

imap = imaplib.IMAP4_SSL(imap_host)

imap.login(imap_user, imap_pass)

mailbox = "INBOX"
imap.select(mailbox)

status, email_ids = imap.search(None, 'ALL')
latest_email_id = email_ids[0].split()[-1]  # Get the ID of the last (latest) email

# Fetch the email header
status, header_data = imap.fetch(latest_email_id, '(RFC822.HEADER)')

# Parse the email header
raw_header = header_data[0][1]
email_message = email.message_from_bytes(raw_header)

# Extract header fields
subject = email_message['Subject']
from_address = email_message['From']
to_address = email_message['To']

# Print the header fields
print(f"Subject: {subject}")
print(f"From: {from_address}")
print(f"To: {to_address}")

# Extract the sender's domain from the 'From' address
sender_email = from_address.split('<')[1].split('>')[0] if '<' in from_address else from_address
sender_domain = sender_email.split('@')[1]

# Extract and display the sender's IP address from the 'Received' header fields
for received_header in email_message.get_all('Received'):
    received_ip = received_header.split('[')[-1].split(']')[0]
    print(f"Received IP: {received_ip}")

# Perform SPF (Sender Policy Framework) check by querying the DNS
resolver = dns.resolver.Resolver()
spf_result = resolver.resolve(sender_domain, 'TXT')

# Check if the SPF record contains "spf1" indicating a valid SPF record
valid_spf = False
for record in spf_result:
    if 'spf1' in record.to_text():
        valid_spf = True
        break

if valid_spf:
    print("SPF Check: Pass (Means Sender is Valid and Not a Spoofed Email Address)")
else:
    print("SPF Check: Fail(Means Sender is Not Valid and is a Spoofed Email Address)")

imap.close()
imap.logout()

```

Output:

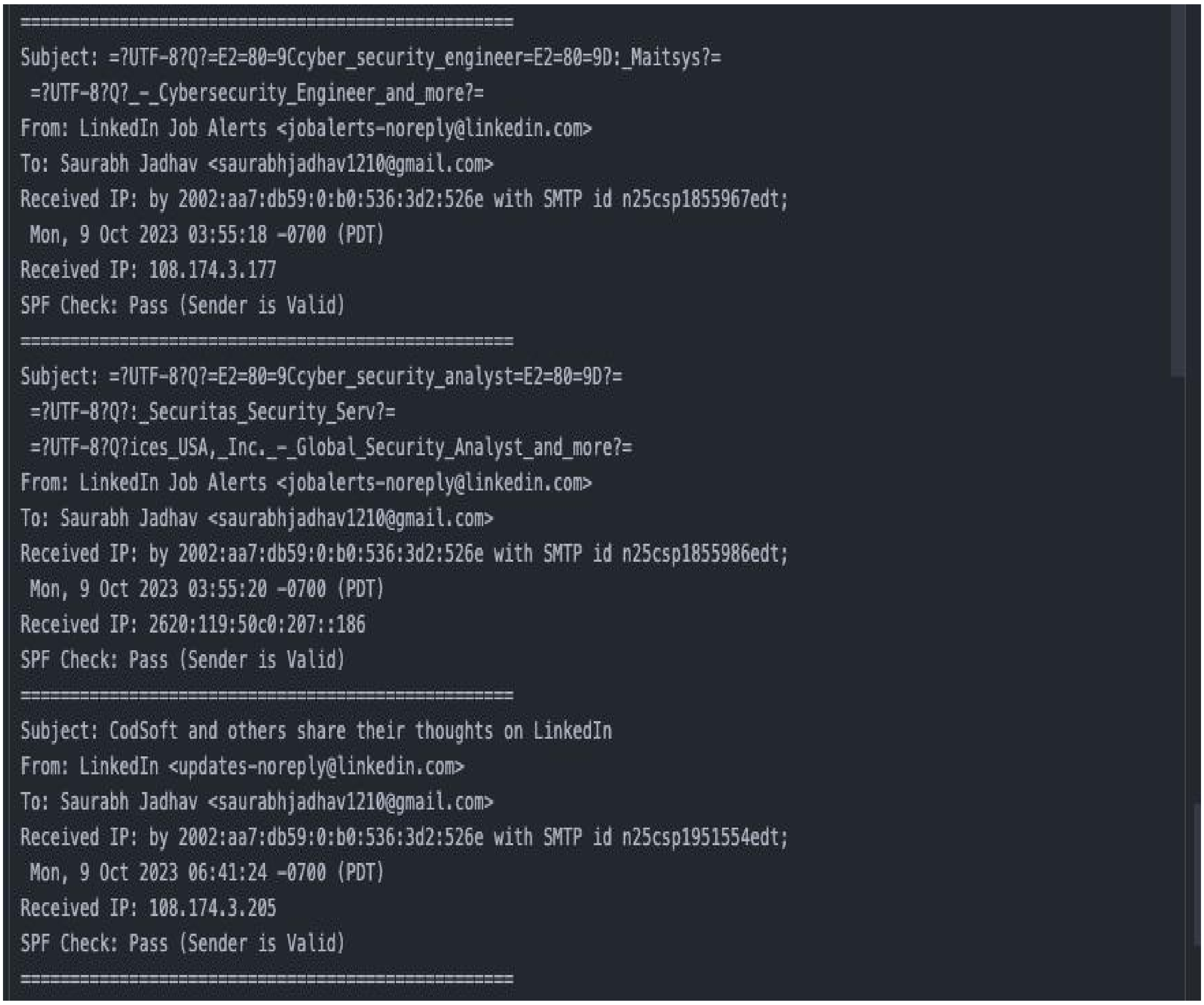


```
Subject: =?UTF-8?Q?=E2=9C=89=EF=B8=8F_You_have_an_invitation?=
From: NAIR GOKULRAJ <invitations@linkedin.com>
To: Saurabh Jadhav <saurabhjadhav1210@gmail.com>
Received IP: by 2002:aa7:c541:0:b0:536:3d2:526e with SMTP id s1csp274831edr; Tue,
 10 Oct 2023 08:13:22 -0700 (PDT)
Received IP: 108.174.6.154
SPF Check: Pass (Means Sender is Valid and Not a Spoofed Email Address)

('BYE', [b'LOGOUT Requested'])
```



```python
#program for fetching 10 latest emails and analyzing their headers
import imaplib
import email
import yaml
import dns.resolver
from email.header import decode_header

with open('config.yml', 'r') as config_file:
    config = yaml.safe_load(config_file)
imap_user = config.get('email')
imap_pass = config.get('password')
imap_host = 'imap.gmail.com'


imap = imaplib.IMAP4_SSL(imap_host)
imap.login(imap_user, imap_pass)

# Select the "Inbox" mailbox
mailbox = "INBOX"
imap.select(mailbox)

# Search for the latest 10 emails in the "Inbox"
status, email_ids = imap.search(None, 'ALL')
latest_email_ids = email_ids[0].split()[-10:]  # Get the IDs of the last 10 emails

for email_id in latest_email_ids:
    # Fetch the email header
    status, header_data = imap.fetch(email_id, '(RFC822.HEADER)')
    raw_header = header_data[0][1]
    email_message = email.message_from_bytes(raw_header)
    # Display header information
    print("=" * 50)
    print(f"Subject: {email_message['Subject']}")
    print(f"From: {email_message['From']}")
    print(f"To: {email_message['To']}")

    # Extract the sender's domain from the 'From' address
    sender_email = from_address = email_message['From']
    if '<' in from_address:
        sender_email = from_address.split('<')[1].split('>')[0]
    sender_domain = sender_email.split('@')[1]
    # Extract and display the sender's IP address from the 'Received' header fields
    for received_header in email_message.get_all('Received'):
        received_ip = received_header.split('[')[-1].split(']')[0]
        print(f"Received IP: {received_ip}")

    # Perform SPF (Sender Policy Framework) check by querying the DNS
    resolver = dns.resolver.Resolver()
    spf_result = resolver.resolve(sender_domain, 'TXT')

    # Check if the SPF record contains "spf1" indicating a valid SPF record
    valid_spf = False
    for record in spf_result:
        if 'spf1' in record.to_text():
            valid_spf = True
            break
    # Display SPF check result
    if valid_spf:
        print("SPF Check: Pass (Sender is Valid)")
    else:
        print("SPF Check: Fail (Sender is Not Valid)")

imap.close()
imap.logout()
```

Output:-



```
==================================================
Subject: =?UTF-8?Q?=E2=80=9Ccyber_security_engineer=E2=80=9D:_Maitsys?=
 =?UTF-8?Q?_-_Cybersecurity_Engineer_and_more?=
From: LinkedIn Job Alerts <jobalerts-noreply@linkedin.com>
To: Saurabh Jadhav <saurabhjadhav1210@gmail.com>
Received IP: by 2002:aa7:db59:0:b0:536:3d2:526e with SMTP id n25csp1855967edt;
 Mon, 9 Oct 2023 03:55:18 -0700 (PDT)
Received IP: 108.174.3.177
SPF Check: Pass (Sender is Valid)
==================================================
Subject: =?UTF-8?Q?=E2=80=9Ccyber_security_analyst=E2=80=9D?=
 =?UTF-8?Q?:_Securitas_Security_Serv?=
 =?UTF-8?Q?ices_USA,_Inc._-_Global_Security_Analyst_and_more?=
From: LinkedIn Job Alerts <jobalerts-noreply@linkedin.com>
To: Saurabh Jadhav <saurabhjadhav1210@gmail.com>
Received IP: by 2002:aa7:db59:0:b0:536:3d2:526e with SMTP id n25csp1855986edt;
 Mon, 9 Oct 2023 03:55:20 -0700 (PDT)
Received IP: 2620:119:50c0:207::186
SPF Check: Pass (Sender is Valid)
==================================================
Subject: CodSoft and others share their thoughts on LinkedIn
From: LinkedIn <updates-noreply@linkedin.com>
To: Saurabh Jadhav <saurabhjadhav1210@gmail.com>
Received IP: by 2002:aa7:db59:0:b0:536:3d2:526e with SMTP id n25csp1951554edt;
 Mon, 9 Oct 2023 06:41:24 -0700 (PDT)
Received IP: 108.174.3.205
SPF Check: Pass (Sender is Valid)
==================================================
```

Conclusion:- Thus we learned and implement Email Header Analysis using Imaplib in Python.

FAQ'S:-

1)What is SMTP,Explain the basic function.
-->SMTP, or Simple Mail Transfer Protocol, is a communication protocol used for the transmission of email messages over the Internet. SMTP serves as the backbone of email communication, facilitating the transfer of emails from the sender's email client or server to the recipient's email server.
Basic Functions of SMTP:

a) **Message Routing:** SMTP is responsible for routing an email message from the sender's email client or server to the recipient's email server. It determines the most appropriate path for the message to reach its destination.
b) **Message Submission:** When a user sends an email from their email client, the client communicates with an SMTP server to submit the message. The SMTP server takes care of the transmission to the recipient's server.
c) **Relaying:** SMTP servers often relay messages through a series of intermediate servers to reach the final destination. This relay process involves multiple SMTP servers cooperating to transfer the message to its recipient.
d) **Authentication:** SMTP can require authentication to ensure that only authorized users or devices can send email through a particular server. This helps prevent unauthorized use and email spoofing.
e) **Error Handling:** SMTP includes mechanisms for error handling and notification.If a message cannot be delivered, the sender or sender's server will receive a bounce-back message (a Non-Delivery Report or NDR) indicating the reason for the failure.
f) **Data Transfer:** SMTP transfers email data between servers in a standardized format. It defines how email messages are structured, specifying the format for email headers and body content.
g) **Port:** SMTP uses port 25 for unencrypted communication and port 587 for secure communication (SMTPS). The latter ensures that data is transmitted securely over the internet, protecting against eavesdropping and data tampering.
h) **Text-Based Protocol:** SMTP is a text-based protocol, which means it uses plain text commands to initiate and manage the email transmission process.

This makes it human-readable and relatively straightforward for email clients and servers to communicate using SMTP commands.

Q.2: How to view Email Header ?
-->
i. Log in to your Gmail account.
ii. Open the email you want to examine.
iii. Click the three dots (More options) in the upper-right corner of the email.
iv. Select "Show original" from the dropdown menu.
v. A new tab or window will open with the full email, including its header.
vi. Examine the email header for details.
vii. Close the "Show original" tab when done.