PA2 – Python SMTP Client Over SSL

Overview

For this project, you will write a Python script that sends an email message over TLS/SSL via Google's SMTP server. The script will not take any command-line arguments or user input, so it will be a boring program that sends the same email message every time that it runs! But that is OK – the real objective of this assignment is to familiarize yourself with the SMTP protocol (see Section 2.3 of the textbook) and SSL socket programming.

A code template is provided below. It includes several helpful comments to guide you. DO NOT CHANGE ANY OF THE CODE THAT IS IN THE TEMPLATE, INCLUDING THE COMMENTS! Your task is to fill in the pieces of missing code between the comments.

Your code will need to output every message that is sent and received. This is demonstrated in the sample dialogue below.

In order to send an email using Google's server, you will need to create a fake Gmail account. You will send the email from this account to your Cedarville email address. If you would like, in the Python code and in your console output, you can obscure the password that you chose for your account.

Below are several helpful links to get you started and to answer questions that may arise. **The two support.google.com links contain vitally important information**.

You may find that it is possible to send spoofed emails using the script that you create. DO NOT SEND SPOOFED EMAILS. This is a legal grey area and if it results in a complaint of any sort, you could be expelled from Cedarville.

What to Hand In

1. A screenshot of the console output of your program (see below – but with a screenshot)

2. A screenshot of your Cedarville email inbox showing the email (see below)

Practicing with a Telnet Session

Even though you need an SSL connection in order to talk with Google's SMTP server, it is still possible to have a to have a real-time dialogue like with Telnet. Use this command from the Linux command line:

```
john%landon>openssl s_client -connect smtp.gmail.com:465 -crlf -ign_eof
```

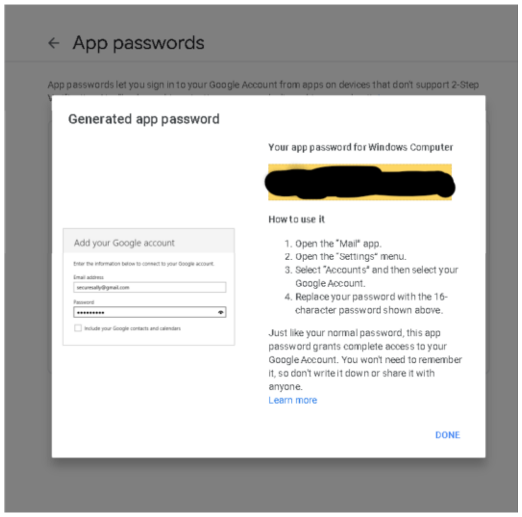
Helpful Links

https://en.wikipedia.org/wiki/Email_spoofing

https://support.google.com/mail/answer/7126229 (see outgoing mail server specs)

https://support.google.com/accounts/answer/185833

(Google has disabled authentication with "less secure" apps)
Please generate an App password from your google account for a "mail" app. Enable 2-step verification for your new gmail account. Then use the generated App password in the "Signing in to Google" account.



← App passwords

App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step

Generated app password

Add your Google account

Enter the information below to connect to your Google account.

Email address: securesally@gmail.com

Password: ••••••••••

☐ Include your Google contacts and calendars

Your app password for Windows Computer

[redacted]

How to use it

1. Open the "Mail" app.
2. Open the "Settings" menu.
3. Select "Accounts" and then select your Google Account.
4. Replace your password with the 16-character password shown above.

Just like your normal password, this app password grants complete access to your Google Account. You won't need to remember it, so don't write it down or share it with anyone.
Learn more

DONE

https://www.feistyduck.com/library/openssl-cookbook/online/ch-testing-with-openssl.html

http://www.samlogic.net/articles/smtp-commands-reference-auth.htm (see AUTH LOGIN)

https://www.base64encode.org/

https://www.base64decode.org/

https://tools.ietf.org/html/rfc5321

https://tools.ietf.org/html/rfc5322

https://docs.python.org/3/library/ssl.html

https://docs.python.org/3/library/base64.html

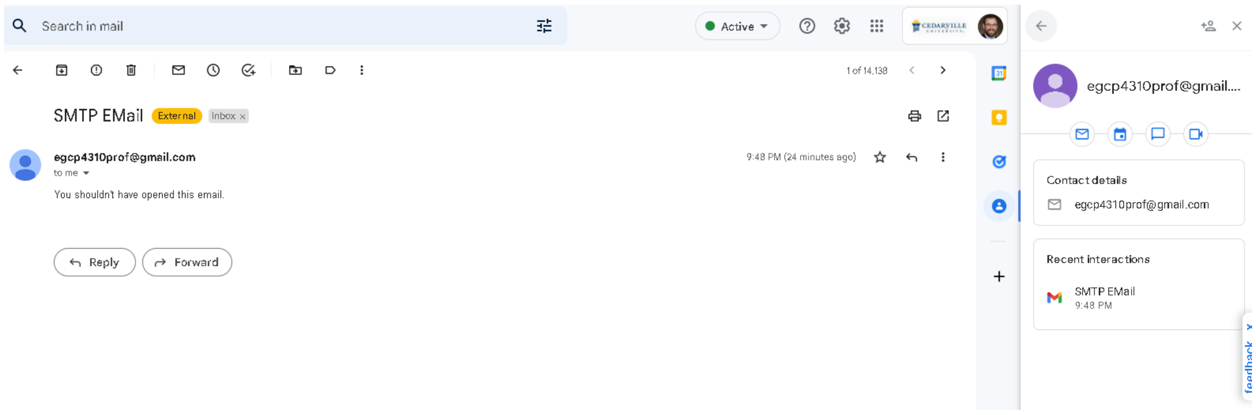
Python Template Code (also posted on Canvas for download)

```
# ASK BEFORE ADDING IMPORTS
from socket import *
from ssl import *
from base64 import *

NL = '\r\n'          # newline
OK_CODE = "250"      # OK message number
SMTP_SERVER  =  ""  # use the links to look this up!
SSL_TLS_PORT =  ""  # use the links to look this up!
# do not copy in the Base64 encodings of these 2 strings!!!
# use the Python library to derive the encoding from the normal strings
USERNAME = ""        # your fake gmail account
PASSWD = ""          # you can obscure this when you hand in your code

# create a clientSocket and then the sslSocket
# this requires you to use the wrap_ function a little differently
# than the example in the API -- play around with it and you'll figure it out

# print cipher being used

# use these functions to send and recv
def send(s) :
    sslSocket.send(s.encode())
    print("SENT:" + NL + s)

def recv(s="") :
    s += sslSocket.recv(1024).decode()
    print("RCVD:" + NL + s)

# send hello command...beware of NL gotcha!

send("test")
# here the server sends two messages with a short time gap in-between.
# the first is a greeting and the second is a series of 250 messages.
# make sure you have received a 250 message before you “speak” again.
# to accomplish this, create a loop to receive 1 byte at a time (do not use the
# recv function above) until you receive the string "250"
# build the bytes you receive onto a string, and then call the recv() function
# passing the string you've accumulated as an argument.
# after this, for the remainder of the script, use the recv() function above

# now that it is your turn in the dialogue “to speak” again...
# send login command because Google makes you authenticate...

# now follow the login prompts...

# now that you are authenticated, you can start the normal SMTP dialogue

# send quit command

# close connection
sslSocket.close()
```

Dialogue of Successful Run of the Program

```
('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
SENT:
EHLO localhost

RCVD:
220 smtp.gmail.com ESMTP bs38-20020a05620a472600b006b615cd8c13sm6281813qkb.106 - gsmtp
250-smtp.gmail.com at your service, [24.123.56.45]
250-SIZE 35882577
250-8BITMIME
250-AUTH LOGIN PLAIN XOAUTH2 PLAIN-CLIENTTOKEN OAUTHBEARER XOAUTH
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-CHUNKING
250 SMTPUTF8

SENT:
AUTH LOGIN

RCVD:
334 VXNlcm5hbWU6

SENT:
RUdDUDQzMTBQcm9mQGdtYWlsLmNvbQ==

RCVD:
334 UGFzc3dvcmQ6

SENT:
e[NOPE]=

RCVD:
235 2.7.0 Accepted

SENT:
MAIL FROM: <EGCP4310Prof@gmail.com>

RCVD:
250 2.1.0 OK bs38-20020a05620a472600b006b615cd8c13sm6281813qkb.106 - gsmtp

SENT:
RCPT TO: <georgelandon@cedarville.edu>

RCVD:
250 2.1.5 OK bs38-20020a05620a472600b006b615cd8c13sm6281813qkb.106 - gsmtp

SENT:
DATA

RCVD:
354  Go ahead bs38-20020a05620a472600b006b615cd8c13sm6281813qkb.106 - gsmtp

SENT:
From: Someone
To:georgelandon@cedarville.edu
Reply-To: EGCP4310Prof@gmail.com
Subject: SMTP EMail

SENT:
You shouldn't have opened this email.
.

RCVD:
250 2.0.0 OK  1665280777 bs38-20020a05620a472600b006b615cd8c13sm6281813qkb.106 - gsmtp

SENT:
QUIT
```

Screenshot of Email Inbox



SMTP EMail External Inbox

egcp4310prof@gmail.com
to me

9:48 PM (24 minutes ago)

You shouldn't have opened this email.

egcp4310prof@gmail....

Contact details
egcp4310prof@gmail.com

Recent interactions
SMTP EMail
9:48 PM