NTNU
Norwegian University of
Science and Technology
Department of Information Security
and Communications Technology

TTM4100

Communication – Services and Networks

Lab 6: “SMTP Client”

This lab is one of the 8 programming labs in this course. You must deliver and pass at least 6 of these labs to be qualified for the exam. Any questions about the exercises can be posted on the forum or you can come to the tuition. The exercises are designed to focus on learning and understanding – not debugging code. All lab-answers should be delivered on the blackboard for review.

Deadline of submission: **11.03.2018**

Lab 6: SMTP Client

By the end of this lab, you will have acquired a better understanding of SMTP protocol. You will also gain experience in implementing a standard protocol using Python.

Your task is to develop a simple mail client that sends email to any recipient. Your client will need to connect to a mail server, dialogue with the mail server using the SMTP protocol, and send an email message to the mail server. Python provides a module, called `smtplib`, which has built in methods to send emails using SMTP protocol. However, we will not be using this module in this lab, because it hides the details of SMTP and socket programming.

In order to limit spam and malicious behavior, most mail servers do not accept TCP connections from arbitrary sources (these "open mail relays" are quite scarce nowadays). Hence, the simple interaction you learnt in the textbook/theory classes (which was used in the early Internet) will not work in standard mail servers, as you'll need to authenticate in order to use the SMTP protocol and send an email.

However, you can test your SMTP client using a simple SMTP server running on your own machine. *Fake SMTP*, http://nilhcem.com/FakeSMTP/, is a very simple SMTP server with easy-to-use graphic user interface. You can run it on any platform provided that you had Java runtime installed. Normally, this program can be run from either command line or double-clicking its JAR file. You can change the port of this SMTP server and the directory to which emails are saved, then start it.

Once your local SMTP server has started, you can try crafting an email and have the server handle it by executing your Python script. You can trace the log and see the most recent email on the server interface and compare them to response messages on your client in order to check if your script is correct.

N.B. It is strongly recommended to read **Sect. 2.7.2** of the textbook before (and while) doing this programming lab.

Code

Below you will find the skeleton code for the client. You are to complete the skeleton code with any Python version of your choice. Beware there are some syntax differences in Python 2.7 and Python 3. The places where you need to fill in code are marked with **`#Fill in start`** and **`#Fill in end`**. Each place may require one or more lines of code. The same code can also be found in the file "Skeleton SMTP.py".

What to Hand in

In your submission, you are to provide the complete code for your SMTP mail client as well as a screenshot showing that you indeed receive the e-mail message. This screenshot could be of the *Fake SMTP* window showing it has got the message from your client correctly.

Skeleton Python Code for the Mail Client

The same code can also be found in the file "Skeleton SMTP.py". As this code requires command line arguments, -f <sender_email> and -t <recipient_email>, you have to run it on a terminal or a command prompt. For instance, `python SMTPClient.py -f bob@gmail.com -t alice@hotmail.com`, providing that your script is saved as `SMTPClient.py`.

```
# This skeleton is valid for both Python 2.7 and Python 3.

# You should be aware of your additional code for compatibility of the Python version
of your choice.



from socket import *

import argparse as ap

import getpass as gp



#Get sender_email and recipient_email as arguments to the program

parser = ap.ArgumentParser(description='A test SMTP client without authentication')

parser.add_argument('-f', '--from', dest='fromMail', required=True,
metavar='<sender_email>')

parser.add_argument('-t', '--to', dest='toMail', required=True,
metavar='<recipient_email>')

#If using the authentication of the SMTP server, also get a valid username (optional
exercise)

#parser.add_argument('-u', '--username', dest='username', required=True,
metavar='<username>')



args = parser.parse_args()

fromMail = args.fromMail #Sender's email address

toMail = args.toMail #Recipient's email address

#username = args.username #SMTP username in case you are implementing the optional
exercise



#If using the authentication of the SMTP server, ask for a valid password (optional
exercise)

#password = gp.getpass(prompt='Password: ')



# Message to send

msg = "\r\n I love computer networks!"

endmsg = "\r\n.\r\n"



# Our mail server is smtp.stud.ntnu.no but it allows only authenticated
communications. (optional exercise)

#mailserver = 'smtp.stud.ntnu.no'
```

```
# You can run a local simple SMTP server such as "Fake SMTP Server" and communicate
with it without authentication.

mailserver = 'localhost'


# Create socket called clientSocket and establish a TCP connection

# (use the appropriate port) with mailserver

#Fill in start

#Fill in end


recv = clientSocket.recv(1024)

print recv

if recv[:3] != '220':

      print '220 reply not received from server.'


# Send HELO command and print server response.

# Can use EHLO instead since HELO is obsolete, but the latter can still be used

heloCommand = 'EHLO Hey\r\n'

#clientSocket.send(heloCommand.encode()) #Python 3

#clientSocket.send(heloCommand) #Python 2.7

recv1 = clientSocket.recv(1024)

print recv1

if recv1[:3] != '250':

      print '250 reply not received from server.'


# Send MAIL FROM command and print server response.

# Fill in start

# Fill in end


# Send RCPT TO command and print server response.

# Fill in start

# Fill in end


# Send DATA command and print server response.
# Fill in start

# Fill in end


# Send message data.

# Fill in start

# Fill in end


# Message ends with a single period.

# Fill in start

# Fill in end


# Send QUIT command and get server response.

# Fill in start

# Fill in end


#Note that there are more communications if you implement the optional exercise.
```

Optional Exercise

1. Mail servers like office365 for your student accounts (address: smtp.office365.com, port: 587) and also smtp.stud.ntnu.no require your client to add a Transport Layer Security (TLS) or Secure Sockets Layer (SSL) for authentication and security reasons, before you send MAIL FROM command. Add TLS/SSL commands to your existing ones and implement your client using either office365 or smtp.stud.ntnu.no mail server with appropriate address and port.

Additional Notes

If you try this optional exercise and use *smtp.stud.ntnu.no* or *smtp.office365.com* to deliver the message to a recipient outside NTNU domain (e.g., gmail, yahoo, etc…), your email will most likely be blocked. It won't even appear on the spam folder, it will simply be rejected by the receiving mail server. This is intentionally done by some ISPs and mail servers to block spam and malicious behaviors, and does NOT mean your code is wrong. As long as you receive the correct reply code from *smtp.stud.ntnu.no* when sending the email, your code is correct: you email has been sent and received. The fact that this receiving mail server will accept the email after receiving it is a different matter, in which other protocols in addition to SMTP are involved. However, if you send to a recipient inside NTNU domain, e.g. yourself, the recipient should receive the message.