CSC 255: Project 1

Socket Programming Assignment
Email Spoofing

Submission Last Date: 7 October 2018
Submitted on: 5 October 2018

Ramandeep Chumber

Index 2

Introduction

SMTP stands for Simple Mail Transfer Protocol. It is an internet standard for email transmission. In this lab we will develop a simple mail client and send the email to any recipient.
Email spoofing is a tactic used in phishing attacks and spams. The email header is forged by the attacker in such a way that the email looks like it has originated from somewhere else other than its actual source. The sender tricks the recipient into opening the email and responding, and then causing intended harm to the recipient security or stealing some information from them.
In this programming assignment we will learn about how to do email spoofing. We as client will establish a TCP connection with mail server. After this the client will have a dialogue with the mail server using SMTP protocol, send an email to the recipient via mail server, and finally close the TCP connection with mail server.

Required information

- Textbook companion website's skeleton code is used for client.
- Client code is written in python.
- The server I am using is local host with standard port number 25.
- I am sending email from mallroy@notexisting.org (It was supposed to be mallory@notexisting.org but I didn't realize my mistake earlier.)
- On receiver side I am using my gmail id: ramanchumber1@gamil.com

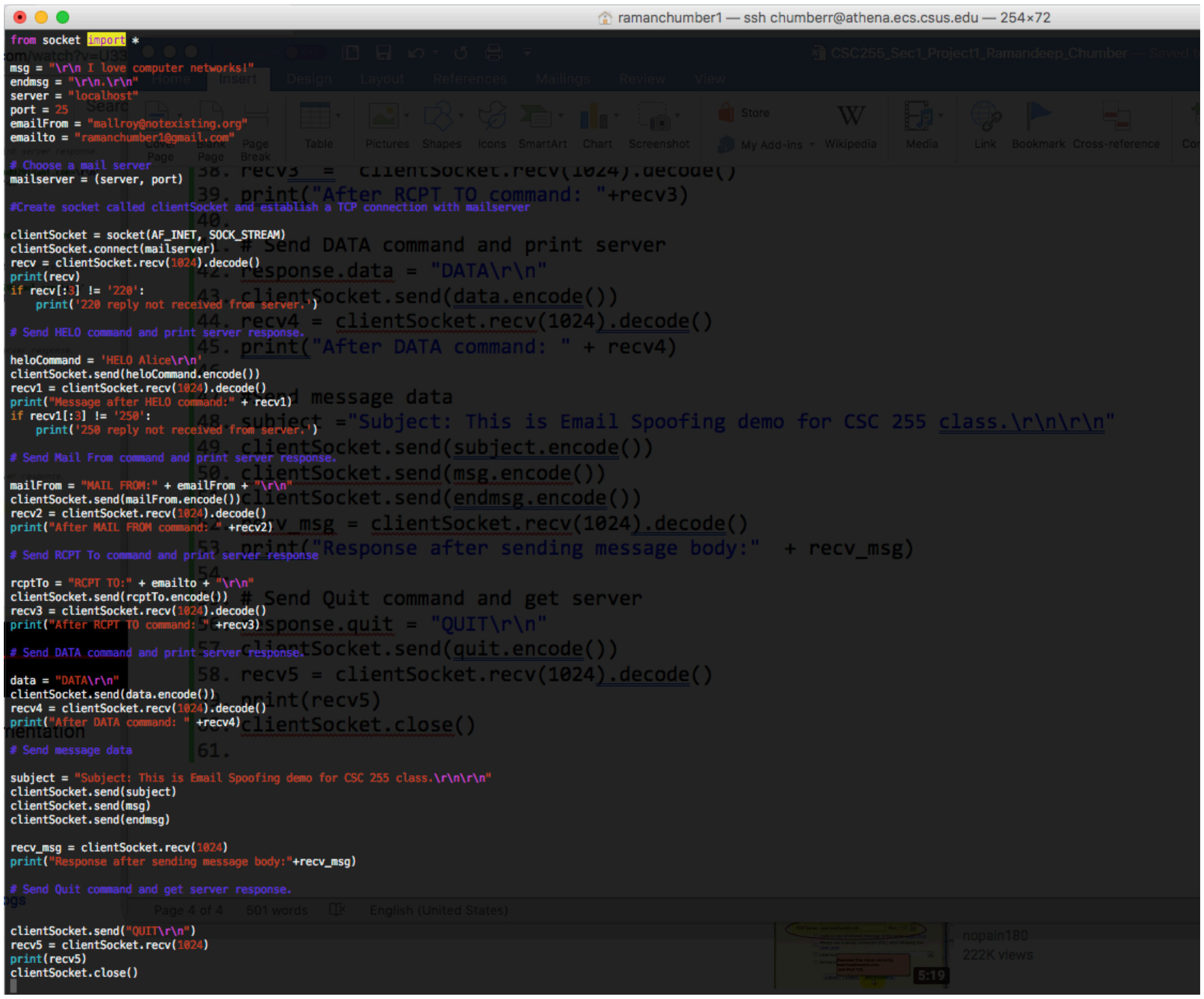
Code with Explanation

I completed the skeleton code shown below. The code is explained after # symbol.

```
From socket import *                              #import socket

msg = "\r\n I love computer networks!"            #message to be send in email
endmsg = "\r\n.\r\n"
server = "localhost"
port = 25                                          #standard port number is used
emailFrom = "mallroy@notexisting.org"              # sender of email
emailto = ramanchumber1@gmail.com                  # receiver email

#Choose a mail server
Mailserver = (server, port)

#Create socket called clientSocket and establish a TCP connection with mailserver
clientSocket=socket(AF_INET,SOCK_STREAM)  #AF_INET used for IPv4 protocols sock_stream for TCP
clientSocket.connect(mailserver)          #to connect with server
recv = clientSocket.recv(1024).decode()   #receiving message from server
```

```
print(recv)                              #print the received message
if recv[: 3]!='220':                     #in case of not receiving any reply from server
   print('220 reply not received from server.’)

# Send HELO command and print server
response.heloCommand ='HELO Alice\r\n'
clientSocket.send(heloCommand.encode())              #helo command sent
recv1 = clientSocket.recv(1024).decode()             #receiving message after helo command
 print("Message after HELO command:"+recv1)          #printing message after helo
if recv1[:3]!='250':                           # in case no message is received from server
   print('250 reply not received from server.')

# Send Mail From command and print serve response.
mailFrom ="MAIL FROM: " +emailFrom+ "\r\n"                   #email sent from
clientSocket.send(mailFrom.encode())
recv2= clientSocket.recv(1024).decode()
print("After MAIL FROM command: "  +recv2)

# Send RCPT To command and prin server response
rcptTo = "RCPT TO:"  +emailto+ "\r\n"                          #email received by
clientSocket.send(rcptTo.encode())
recv3  =  clientSocket.recv(1024).decode()
print("After RCPT TO command: "+recv3)

# Send DATA command and print server
response.data = "DATA\r\n"                                   #data received by recipient
clientSocket.send(data.encode())
recv4 = clientSocket.recv(1024).decode()
print("After DATA command: " + recv4)

#Send message data
subject ="Subject: This is Email Spoofing demo for CSC 255 class.\r\n\r\n"
clientSocket.send(subject)                          # email subject to be sent
clientSocket.send(msg)                              # send message
clientSocket.send(endmsg)                       #to tell the server that message has ended.
recv_msg = clientSocket.recv(1024)
print("Response after sending message body:"  + recv_msg)

# Send Quit command and get server
clientSocket.send("QUIT\r\n")
recv5 = clientSocket.recv(1024)
print(recv5)
clientSocket.close()
```

client.py is client program in python shown below.

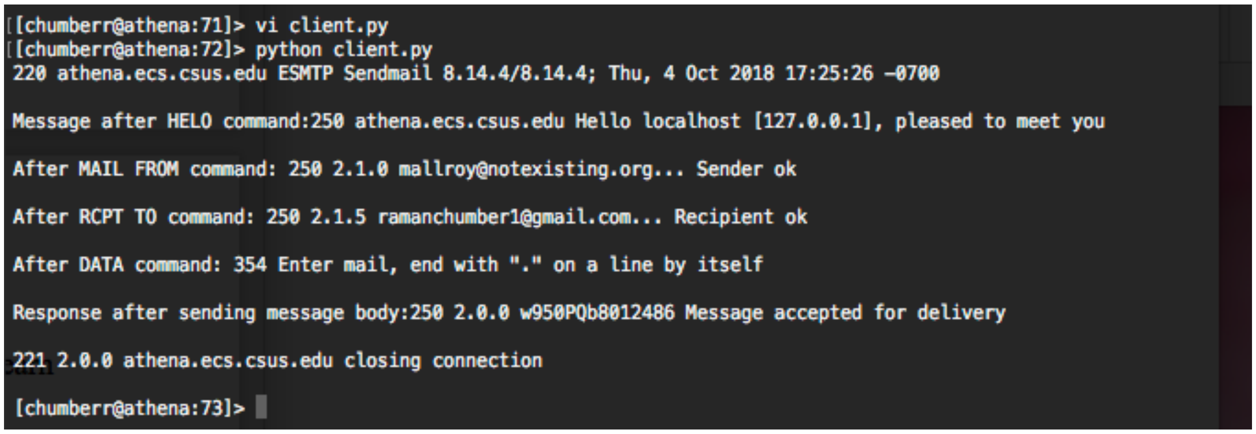


ramanchumber1 — ssh chumberr@athena.ecs.csus.edu — 254×72

```
from socket import *
msg = "\r\n I love computer networks!"
endmsg = "\r\n.\r\n"
server = "localhost"
port = 25
emailFrom = "mallroy@notexisting.org"
emailto = "ramanchumber1@gmail.com"

# Choose a mail server
mailserver = (server, port)

#Create socket called clientSocket and establish a TCP connection with mailserver

clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect(mailserver)
recv = clientSocket.recv(1024).decode()
print(recv)
if recv[:3] != '220':
    print('220 reply not received from server.')

# Send HELO command and print server response.

heloCommand = 'HELO Alice\r\n'
clientSocket.send(heloCommand.encode())
recv1 = clientSocket.recv(1024).decode()
print("Message after HELO command:" + recv1)
if recv1[:3] != '250':
    print('250 reply not received from server.')

# Send Mail From command and print server response.

mailFrom = "MAIL FROM:" + emailFrom + "\r\n"
clientSocket.send(mailFrom.encode())
recv2 = clientSocket.recv(1024).decode()
print("After MAIL FROM command: " +recv2)

# Send RCPT To command and print server response

rcptTo = "RCPT TO:" + emailto + "\r\n"
clientSocket.send(rcptTo.encode())
recv3 = clientSocket.recv(1024).decode()
print("After RCPT TO command: " +recv3)

# Send DATA command and print server response.

data = "DATA\r\n"
clientSocket.send(data.encode())
recv4 = clientSocket.recv(1024).decode()
print("After DATA command: " +recv4)

# Send message data

subject = "Subject: This is Email Spoofing demo for CSC 255 class.\r\n\r\n"
clientSocket.send(subject)
clientSocket.send(msg)
clientSocket.send(endmsg)

recv_msg = clientSocket.recv(1024)
print("Response after sending message body:"+recv_msg)

# Send Quit command and get server response.

clientSocket.send("QUIT\r\n")
recv5 = clientSocket.recv(1024)
print(recv5)
clientSocket.close()
```

Figure 1: Client.py program

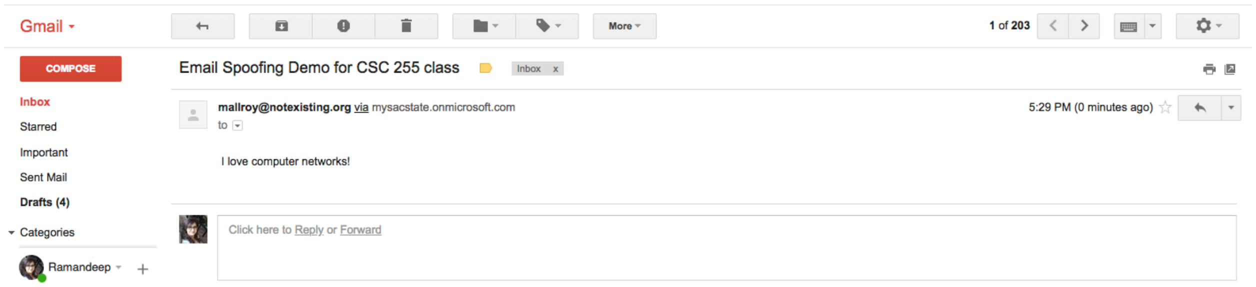
The result below were seen after client.py is run.



```
[chumberr@athena:71]> vi client.py
[chumberr@athena:72]> python client.py
220 athena.ecs.csus.edu ESMTP Sendmail 8.14.4/8.14.4; Thu, 4 Oct 2018 17:25:26 -0700

Message after HELO command:250 athena.ecs.csus.edu Hello localhost [127.0.0.1], pleased to meet you

After MAIL FROM command: 250 2.1.0 mallroy@notexisting.org... Sender ok

After RCPT TO command: 250 2.1.5 ramanchumber1@gmail.com... Recipient ok

After DATA command: 354 Enter mail, end with "." on a line by itself

Response after sending message body:250 2.0.0 w950PQb8012486 Message accepted for delivery

221 2.0.0 athena.ecs.csus.edu closing connection

[chumberr@athena:73]>
```

Figure 2: result after python client.py command is run

Spoofed email received at ramanchumber1@gmail.com. Email sender is mallroy@nonexisting.org. The subject is: Email Spoofing Demo for CSC 255 class. The message says "I love computer networks!".



Email Spoofing Demo for CSC 255 class

mallroy@notexisting.org via mysacstate.onmicrosoft.com
to

5:29 PM (0 minutes ago)

I love computer networks!

Click here to Reply or Forward

Figure 3: Spoofed email received

Conclusion

This programming assignment was very informative. I learned about SMTP protocol and how it works. I studied about email spoofing by security perspective in my computer security CSC 250 class last semester. I always wonder how it can be done. Now I know the process, it feels good to know how it works. I never realized that TCP and SMTP has so much to do with going back and forth in order to establish a connection. And it helped me a lot in understanding the communication between server and the client. It was a fun programming assignment and it helped me to build up on my basic knowledge of email spoofing. It also helped me in understanding the email sending and receiving process in general using SMTP.