Project 2: Email, Web Scraping with Python

Due Nov. 15th, 2024

Hand in: Electronic submission of entire project (source files, report, etc.) using 'Assignment' on Canvas. Please zip up your whole project directory and submit the zip file. In your submission, please write down a comprehensive **report**, which explains your code and demos the functions of the project with screen **snapshots**. Your project must be implemented by **Python**. Do **NOT** write down your email or password in your report or your Python code!!!!!

Please note the following email server part is for fun. It will NOT be graded. The part of Web Scraping with Python will be graded.

OK, let's first play with mail server, and do this part step-by-step.

- First, we want to play the mail server of TCNJ. To play with the mail servers, you need to find the mail server. Open your terminal, type

  nslookup –type=mx tcnj.edu

  In the terminal, there will be two mail servers displayed in terminal. Pick one. Without loss of generality, we assume the picked server is xxxx.tcnj.edu. Type the following command in terminal.

  telnet xxxx.tcnj.edu 25

  Then, type the commands you see on page 60 of the chapter 2 slides. Replace the sender and the receiver email address to be your own email address. Check your mailbox, see whether you get the email or not.

  Take a snapshot of your terminal when you type the commands.

  Hint: It will NOT be surprising to see you cannot send out email to the mail server on campus. These email servers reverse your IP address to a registered hostname (PTR) on DNS server. If the reversed hostname does not match your claim hostname, your request will be rejected. The sad fact is our own computers are definitely not registered in PTR record (some TCNJ computers are registered).

- Since TCNJ outsourced the email service to Gmail a few years ago, let's try the mail server of Gmail. First, we need to figure out which gmail server we want to talk with. If we type the following command in terminal

nslookup –type=mx gmail.com

we will see the following results:

```
Non-authoritative answer:
gmail.com     mail exchanger = 20 alt2.gmail-smtp-in.l.google.com.
gmail.com     mail exchanger = 30 alt3.gmail-smtp-in.l.google.com.
gmail.com     mail exchanger = 5 gmail-smtp-in.l.google.com.
gmail.com     mail exchanger = 40 alt4.gmail-smtp-in.l.google.com.
gmail.com     mail exchanger = 10 alt1.gmail-smtp-in.l.google.com.

Authoritative answers can be found from:
alt3.gmail-smtp-in.l.google.com    internet address = 74.125.140.27
alt3.gmail-smtp-in.l.google.com    has AAAA address
2a00:1450:400c:c08::1a
gmail-smtp-in.l.google.com    internet address = 173.194.66.26
gmail-smtp-in.l.google.com    has AAAA address 2607:f8b0:400d:c01::1a
alt4.gmail-smtp-in.l.google.com    internet address = 74.125.143.26
alt4.gmail-smtp-in.l.google.com    has AAAA address
2a00:1450:4013:c03::1b
alt1.gmail-smtp-in.l.google.com    internet address = 64.233.190.26
alt1.gmail-smtp-in.l.google.com    has AAAA address 2800:3f0:4003:c01::1a
alt2.gmail-smtp-in.l.google.com    internet address = 209.85.203.26
alt2.gmail-smtp-in.l.google.com    has AAAA address
2a00:1450:400b:c03::1a
```

It looks there are a bunch of mail servers in Gmail. It is a bit confusing, if we visit the gmail webpage at https://support.google.com/mail/answer/7126229?hl=en , which says the IMAP server (incoming emails) is imap.gmail.com and the SMTP (outgoing emails) server is smtp.gmail.com. As you can see imap.gmail.com and smtp.gmail.com are alias names and the names returned by nslookup are canonical names. In this project, we just use alias name. More specifically, to simplify the project and discussion, this project use smtp.gmail.com. Once you understand SMTP, you can easily figure out IMAP details.

According to the webpage at https://support.google.com/mail/answer/7126229?hl=en, smtp.gmail.com has two port numbers, 465 and 587, for SMTP service. Remember that we used port number 25 in class? Why is Gmail using 465 and 587, instead of 25? The short answer is that port 25 is for plaintext SMTP, while 465 and 587 are for encrypted SMTP transfer. As we show in the lab, we can easily use a network sniffer like Wireshark to capture the plaintext transfer. So today, it is common to use 465 and 587 for security reasons.

What is the difference between 465 and 587 for Gmail? There are two popular protocols to do network encryption/decryption. One is SSL (including version 1.0, 2.0, 3.0), the other one is TLS (including version 1.0, 1.1, 1.2, and 1.3). SSL is an older protocol, it is gradually phasing out. TLS is a more recent/advanced one. If it is possible, TLS is preferred over SSL. A detailed discussion of TLS will be provided in security course. For this project, you do not have to worry about the details of TLS itself. By default, we should always use 587 (using TLS) to submit email to server. Port 465 is obsolete.

Can we use telnet to connect to the port 587 smtp.gmail.com? No! telnet itself is plaintext protocol, it does not support SSL or TLS. To use SSL or TLS, you should use software like openssl.

Do the following steps in your terminal, let's try to connect to smtp.gmail.com **manually**.

1. Assume your gmail address is myEmail@gmail.com, and your password is myPassword. Type the following command in the terminal. In your own command, myEmail@gmail.com shall be replaced by your gamil account and myPassword should be replaced by your own password.

   ```
   perl –MMIME::Base64 –e 'print
   encode_base64("\000myEmail\@gmail.com\000myPassword")'
   ```

   Please note the above command is supposed to be one line, just like the following snapshot.

   ```
   Compscijli16:~ jli$ perl –MMIME::Base64 –e 'print encode_base64("\000myEmail\@gmail.com\000myPassword")'
   AG15RW1haWxAZ21haWwuY29tAG15UGFzc3dvcmQ=
   ```

   In above, AG15RW1haWxAZ21haWwuY29tAG15UGFzc3dvcmQ= is the output of the command. We will use it as credential to login Gmail (your output is different from the example).

2. Type following command to connect to smtp.gmail.com

   openssl s_client -starttls smtp -connect smtp.gmail.com:587 -crlf -ign_eof

3. then type:

   EHLO localhost

4. then type:

   AUTH PLAIN AG15RW1haWxAZ21haWwuY29tAG15UGFzc3dvcmQ=

5. try to type:

   MAIL FROM: <myEmail@gmail.com>

   RCPT TO: <myEmail@gmail.com>

   DATA

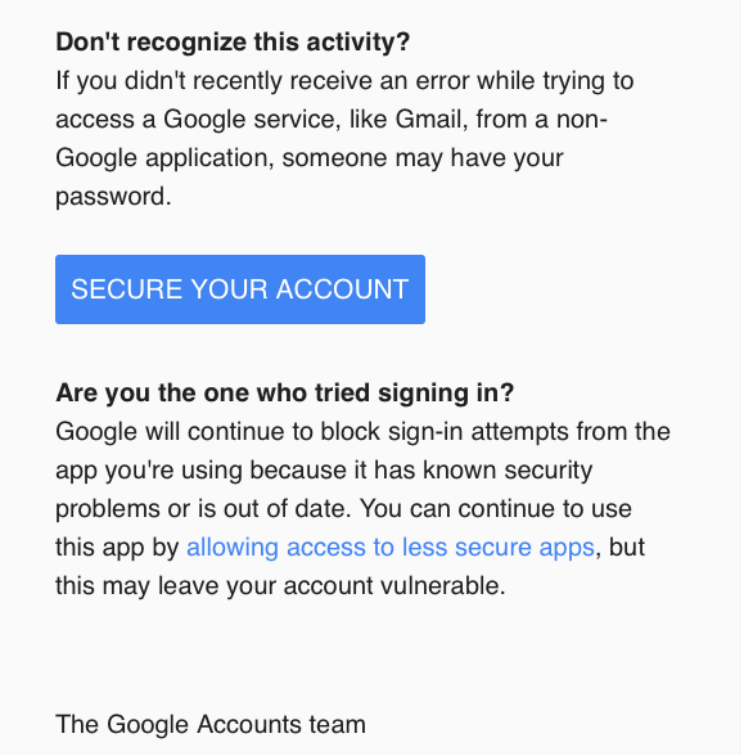
   Subject: Hello gmail!

It works!

.

quit

Does the commands work? If not, open your gmail account in a browser. Do you get a warning email about your login? Does the warning email looks similar to this snapshot?



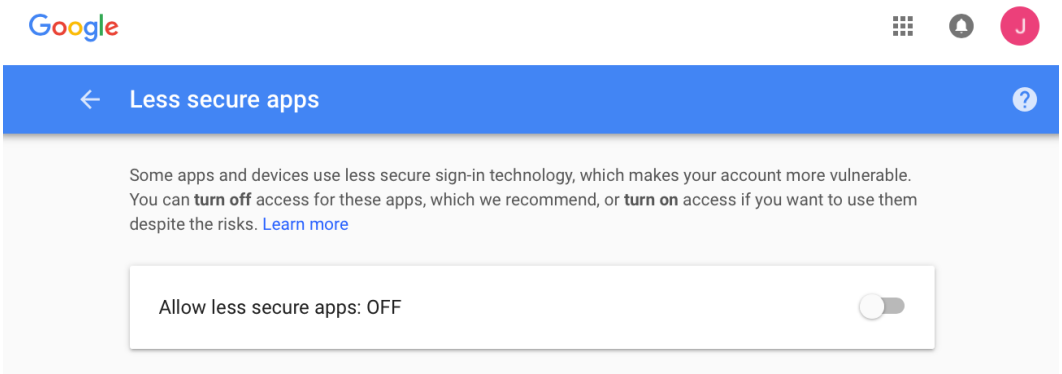
Don't recognize this activity?
If you didn't recently receive an error while trying to access a Google service, like Gmail, from a non-Google application, someone may have your password.

SECURE YOUR ACCOUNT

Are you the one who tried signing in?
Google will continue to block sign-in attempts from the app you're using because it has known security problems or is out of date. You can continue to use this app by allowing access to less secure apps, but this may leave your account vulnerable.

The Google Accounts team

If you do receive this email from Gmail, click "allowing access to less secure apps" in the email, **turn on** the switch on the following snapshot.



After turning on the switch, redo the steps from 1 to 5. Check Gmail in browser again. Do you receive the email from yourself? Take a snapshot of your terminal, **sanitize/blur** the credential part. Take a snapshot of the email in browser.

After all this is done, **turn OFF** the "Allow less secure app" switch.

PROJECT

WARNING:

1. Do not abuse the web server, your script should pace the requests generously. Because a large number of requests can be generated by your Python scripts by mistake. The web server may block the IP address that sends too many requests within a short period. To avoid possible blocking, your script should space the requests by random waiting time, such as 5 seconds, 3 seconds, 2 seconds, 6 seconds. Use a random number generator to create the waiting time.

2. Whenever you run your Python code, double check the code to make sure it will NOT generate excessive requests to the server.

The second part of this project is to analyze the reviews on Amazon. The online reviews on Amazon are used by customers as a guideline before they click the button to buy. However, dishonest sellers/competitors abused the reviewing mechanisms on Amazon. The dishonest sellers/competitors posted vast amount of biased reviews on the website, and deeply damaged the credibility of the reviewing mechanism on Amazon.

In this project, we want to use Python script to analyze the overall credibility of a given product, such as a headphone https://www.amazon.com/Bose-QuietComfort-Wireless-Headphones-Cancelling . To analyze the credibility of this product's reviews, we will use Python script to scrape the reviews of this product. Then your Python script to scrape the profile of each reviewer of this product. After analyzing the reviews made by a given reviewer, your Python script should be able to give an estimate how likely this reviewer is biased. Repeat this process for each reviewer of the product. After that, the Python script should be able to give an adjusted average score of the reviews.

Selenium (https://selenium-python.readthedocs.io) is a powerful and user-friendly tool to automate web browsing. In other words, you can write a Python script that automatically browse certain website. Automated web browsing is widely used by industry to test website, collect data from website. Many companies developed their own web browsing software. There are quite a lot of tools supporting web browsing (https://github.com/dhamaniasad/HeadlessBrowsers), Selenium is one of them. In

the second part of this project, we use Selenium to search customer reviews on Amazon.

To finish this part of the project:

1. Install selenium on your computer.

2. Read the sample code at page https://www.simplilearn.com/tutorials/python-tutorial/selenium-with-python and https://medium.com/analytics-vidhya/scraping-amazon-results-with-selenium-and-python-547fc6be8bfa .

3. While working with part of the project, you will use xpath to search certain contents in the HTML files. To get an idea of xpath, please look at the introduction at https://www.guru99.com/xpath-selenium.html .

4. To process the results returned from web server, beautifulsoup (https://pypi.org/project/beautifulsoup4/) can be extremely helpful.

5. The result of this project is open. Please use your logical thinking to justify the strategies in the project. Your results must be based on DATA. “I think…” or “I believe…” are weak expressions in a computer science report.

6. Discuss your discoveries in the report.