MS in CyberSecurity

Unit-12

Python Project

Implementation of Multi-factor Authentication and the use of encryption technologies to store user data using Python

Project Readme File

Introduction:

In the preceding assignment, we constructed a web-based appointment and scheduling management information system (ASMIS) for Queens medical centre and discussed potential cyberattacks and mitigation strategies against this setup. In this project module, we will cover the creation of a solution for cyberattacks including password hacking in ASMIS. This solution was implemented using Python code, therefore let's discuss it in greater depth.

Project description:

It is essential to consider password-based authentication when setting up ASMIS. For any online booking or appointment scheduling, requiring a username and password to log in is one of the most secure security measures. Considering this, we've utilized Python programming to construct the password authentication procedure and avoid cyberattacks. Multi-Factor Authentication is the most crucial need for preventing password-based cyberattacks.

Password Protection:

The stronger the password, the more secure the computer will be from hackers and dangerous software; all accounts on the computer should have strong passwords. Using the above-mentioned password criterion, I've built the following code to ensure that the ASMIS system in the Queens healthcare infrastructure configuration is secure. Users, physicians, patients, and administrators must adhere to these requirements while creating a password for the ASMIS website login.

Multifactor authentication (MFA):

It is a security system that requires multiple ways of authentication from distinct credential categories to authenticate a user's identity for a login or other transactions. Using one-time passwords is one of the ways technology businesses have combated password theft and other sorts of assaults (OTPs). OTP is a sort of multi-factor authentication (MFA) that makes it significantly more difficult for hackers to access protected data.

Requirements:

- To put this approach into action, you'll need an algorithm.
- Python 3.7 and higher
- To run Python code, use the Codio tool.
- In the execution tool, install the necessary python script libraries.

Algorithm:

Step-1: Start

Step-2: Install the necessary packages and libraries

Step 3: Define the primary class to evaluate password attributes

Step 4: Print password is acceptable to continue or false

Step-5: Declare code to encrypt and decrypt the password; save it to the database.

Step-6: Declare class to carry out multi-factor authentication (OTP)

Step 7: Enter User ID and password

Step 8: Perform password validation using schema

Step 9: Stop

Installation:

Execute the following commands and implement these packages as mentioned in the below screenshot.

- ✓ pip install bcrypt
- ✓ pip install pyotp
- ✓ pip install schema
- ✓ pip install password-validator

```
codio@rainbowsubject-siliconpixel:~/workspace$ pip install bcrypt
Collecting bcrypt
  Using cached https://files.pythonhosted.org/packages/ad/36/9a0227d048e98409f012570f7bef8a8c2373b9c9c5dfbf82963cbae05ede/bcrypt-3.1.7-cp27-cp27mu-manylinux1_x86_64.whl
Collecting cffi>=1.1 (from bcrypt)
  Using cached https://files.pythonhosted.org/packages/26/28/fb01ff898aa7c93e4774799b3dc0a3693cfee48c5ea4e524ce30e6b10e7e/cffi-1.15.0-cp27-cp27mu-manylinux1_x86_64.whl
Collecting six>=1.4.1 (from bcrypt)
  Using cached https://files.pythonhosted.org/packages/d9/5a/e7c31adbe875f2abbb91bd84cf2dc52d792b5a01506781dbcf25c91daf11/six-1.16.0-py2.py3-none-any.whl
Collecting pycparser (from cffi>=1.1->bcrypt)
  Using cached https://files.pythonhosted.org/packages/62/d5/5f610ebe421e85889f2e55e33b7f9a6795bd982198517d912eb1c76e1a53/pycparser-2.21-py2.py3-none-any.whl
Installing collected packages: pycparser, cffi, six, bcrypt
Successfully installed bcrypt-3.1.7 cffi-1.15.0 pycparser-2.21 six-1.16.0
Segmentation fault (core dumped)
codio@rainbowsubject-siliconpixel:~/workspace$ pip install pyotp
Collecting pyotp
  Using cached https://files.pythonhosted.org/packages/a5/cd/9114b5116a9bfbf7133e79f2f12c31a8611a689849a1a990fe5586c08526/pyotp-2.6.0-py2.py3-none-any.whl
Installing collected packages: pyotp
Successfully installed pyotp-2.6.0
```

```
codio@rainbowsubject-siliconpixel:~/workspace$ pip install schema
Collecting schema
  Using cached https://files.pythonhosted.org/packages/0d/93/ca8aa5a772efd69043d0a745172d92bee027caa7565c7f774a2f44b91207/schema-0.7.5-py2.py3-none-any.whl
Collecting contextlib2>=0.5.5 (from schema)
  Using cached https://files.pythonhosted.org/packages/85/60/370352f7ef6aa96c52fb001831622f50f923c1d575427d021b8ab3311236/contextlib2-0.6.0.post1-py2.py3-none-any.whl
Installing collected packages: contextlib2, schema
Successfully installed contextlib2-0.6.0.post1 schema-0.7.5
codio@rainbowsubject-siliconpixel:~/workspace$ pip install password-validator
Collecting password-validator
Installing collected packages: password-validator
Successfully installed password-validator-1.0
```

Configuration with sample execution:

##Declare the main function to start this programming

```
if·__name__·==·'__main__':
········main()
```

##Now, please verify that all required packages and libraries have been imported and installed.

##Define the primary class used for password condition validation.The following characteristics are examined when validating passwords:

- Password length must be between 6 and 30 characters.
- Combinations of capital and lowercase letters along with digits are required for a password.
- Special characters are required to guarantee strong password security.
- Moreover, the password should not be the same as the username
- Previously, passwords were not permitted.

```
def checkPassword(password):
    global gErrFlag
    ps_len = len(password)

    if not (ps_len >= 8 and ps_len <= 25):
        gErrFlag = True
        return "Password length should be in min 8 to max 25"

    if not any(ch.isupper() for ch in password):
        gErrFlag = True
        return('password should have at least one alphabetic Upper case character')

    if not any(ch.islower() for ch in password):
        gErrFlag = True
        return('password should have at least one alphabetic Lower case character')

    if not any(ch.isdigit() for ch in password):
        gErrFlag = True
        return('password should have at least one numeral')

    specialSymb = ['~','!','@','#','$','%','^','&','*','(',')','_','-','+','=']

    if not any(ch in specialSymb for ch in password):
        gErrFlag = True
        return('password should have at least one of the symbols ~!@#$%^&*()_-+=')

    # If every is ok
    return ('Password ok!')
```

##Define a class to request the user to enter a username and password

```
def main():

    global gErrFlag
    gErrFlag = False

    username = input ("Enter your username :")
    password = input ("Enter your password :")

    if username == password:
        return print("Username and Password are same!")

    while(True):
        # input("Enter you password :")
        print(checkPassword(password))
        break

    #If any error exit the programm
    if gErrFlag:
        return
```

##Password should be encrypted and saved to SQL database using an encryption algorithm like SHA256 or MD5. This helps to prevent the password from cyber-attacks.

```
    ##Encryption process
    #string need to conver byte before hashing
    password = bytes(password,'utf-8')

    salt = bcrypt.gensalt()
    hashed = bcrypt.hashpw(password,salt)
```

##Using decryption code, you can validate the password.

```
    ##Decryption process
    if bcrypt.checkpw(password,hashed):
        print("You have a match on encrypt-decrypt process")
    else:
        print("Password does not match")
```

##The Python OTP package enables multi-factor authentication to produce OTP.

##If all of the above conditions are satisfied, the password is successfully input and encrypted using multifactor authentication.

```
····##·Multi-factor·authentication
····totp·=·pyotp.TOTP('base32secret3232')
····print("Your·OTP·is:",totp.now())

····if·totp.verify(totp.now())==True:
····  ··print("Successful·login")
····else:
····  ··print("Unsuccessfull·login")
```

##Execute below schema to validate the password:

```
···#·Create·a·schema
····schema·=·PasswordValidator()

····#·Add·properties·to·it
····schema\
····.min(8)\
····.max(15)\
····.has().uppercase()\
····.has().lowercase()\
····.has().digits()\
····.has().no().spaces()\

····#·Validate·against·a·password·string
····print('Schema·validation·:·')
····print(schema.validate('validPASS123'));
····#·=>·True
····print(schema.validate('invalidPASS'));
····#·=>·False
```

Execution and Validation:

##Enter Username and password as shown below and if it matches, will show the output.

```
codio@rainbowsubject-siliconpixel:~/workspace$ python3 f1
Enter your username : gokulessex12
Enter your password : CyberSecurity@3
Password ok!
```

##If the username and password are the same it throws an error, and if the password doesn't match any of these criteria it will show an error.

```
codio@rainbowsubject-siliconpixel:~/workspace$ python3 f1
Enter your username : gokulessex12
Enter your password : gokulessex
password should have at least one alphabetic Upper case character
codio@rainbowsubject-siliconpixel:~/workspace$ python3 f1
Enter your username : gokulessex12
Enter your password : gokulessex12
Username and Password are same!
```

##All outputs are validated, and it is displayed as below successful.

```
Enter your username : gokulessex12
Enter your password : CyberSecuirty#3
Password ok!
You have a match on encrypt-decrypt process
Your OTP is: 481431
Successful login
Schema validation :
True
```

References:

https://pypi.org/project [Accessed on 28 May 2022]

https://www.codio.com/resources/intro-python [Accessed on 20 May 2022]

Bowne, Samuel. *Hands-On Cryptography with Python: Leverage the power of Python to encrypt and decrypt data*. Packt Publishing Ltd, 2018.

M. D. Leonhard and V. N. Venkatakrishnan, "A comparative study of three random password generators," *2007 IEEE International Conference on Electro/Information Technology*, 2007, pp. 227-232, doi: 10.1109/EIT.2007.4374533.

F. Z. Glory, A. Ul Aftab, O. Tremblay-Savard and N. Mohammed, "Strong Password Generation Based On User Inputs," *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2019, pp. 0416-0423, doi: 10.1109/IEMCON.2019.8936178.
