11. 11. Password-Based Authentication Program

Introduction:

In modern computing, secure authentication is essential to protect user data and ensure privacy. This lab demonstrates a basic password-based authentication system using Python. The program verifies a user's credentials against a predefined database (implemented as a dictionary) and grants access only if both the username and password match.

Objectives:

- Implement a secure password input system using the getpass module.
- Validate user credentials against a stored user-password database.
- Handle incorrect login attempts gracefully through repeated prompts.

Algorithm:

1. Start the program.
2. Import the getpass module to securely input the password without echoing it on the screen.
3. Define a dictionary called database containing valid usernames as keys and their corresponding passwords as values.
4. Prompt the user to input their username.
5. Check if the entered username exists in the database.
6. If the username exists:
    - Prompt the user to enter their password securely using getpass.
    - While the entered password does not match the stored password:
        - Display a message indicating the password is incorrect.
        - Prompt the user to re-enter the password.
    - Once the correct password is entered, display "Access Granted. Verified User."
7. If the username does not exist:
    - Display "User not valid."
8. End the program.

Program:

Reference:

PlainEnglish. (2023). Password Authentication with Python: A Step-by-Step Guide. Retrieved from: https://python.plainenglish.io/password-authentication-with-python-a-step-by-step-guide-d1a853886e2d

12. Dictionary Attack Password Checker

Introduction:

A dictionary attack is a common cybersecurity technique used to crack passwords by systematically trying a list of likely passwords. This program demonstrates how to check if a specific password, such as "oxford," exists in a known password wordlist. If the password is found, it indicates the password is weak and vulnerable to attacks.

Algorithm:

1. Import the required libraries: hashlib for hashing and urlopen from urllib to fetch the wordlist.
2. Define a function to hash passwords using the SHA-256 algorithm.
3. Set the target password (e.g., "oxford") and compute its hash.
4. Load a wordlist of common passwords from a publicly accessible URL.
5. Iterate through each word in the wordlist:
   - Hash the word using the same hashing function.
   - Compare the hashed word with the target password's hash.
6. If a matching hash is found, print that the password is vulnerable.
7. If no match is found after checking all words, print that the password is not in the wordlist.

Program:

Reference:

PlainEnglish. (2023). Dictionary Attack Password Checker in Python.
Retrieved from:
https://trinket.io/python3/c47ff05883

13.

Introduction:

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is a security measure used to distinguish humans from automated bots. This program generates a random CAPTCHA consisting of letters and digits and asks the user to enter it correctly. Successful entry verifies that the user is human, helping prevent automated attacks and spam.

Algorithm:

1. Import random and string modules.
2. Define generate_captcha(length) to:
   - Combine letters and digits into a character set.
   - Randomly select length characters to form the CAPTCHA.
   - Return the CAPTCHA string.
3. Define verify_human() to:
   - Generate a CAPTCHA using generate_captcha().
   - Display the CAPTCHA and prompt user input.
   - Compare input with the generated CAPTCHA.
   - Print success if matched, else print failure.
4. Call verify_human() to run the verification process.

Program code:

Reference:

GeeksforGeeks. (2023). Program to Generate CAPTCHA and Verify User.

Retrieved from:

https://www.geeksforgeeks.org/program-generate-captcha-verify-user/

14. Logical Bomb to Display Christmas Tree on Christmas Day

Introduction:

A logic bomb is a piece of code that triggers an action when a specific condition is met. This program demonstrates a harmless logic bomb that checks if the system date is Christmas Day (December 25). If it is, the program displays a festive Christmas tree and greeting. Otherwise, it runs normally without any effect. This example illustrates how logic bombs can be used for triggered actions based on date or events.

Algorithm:

1. Import the date class from the datetime module to access the current date.
2. Get the current system date using date.today().
3. Define the target date (Christmas Day) as December 25 of the current year.
4. Define a function show_message() that prints a Christmas tree pattern made of stars (*) along with a festive greeting.
5. Define a function bomb() to check if today's date matches the target date.
6. If the date matches, call show_message() to display the Christmas tree and exit the program.
7. If the date does not match, continue the normal program execution and print a generic message.

Reference:

Craig88. (2023). Christmas Logic Bomb. GitHub repository.
Retrieved from:
https://github.com/Craig88/christmas-logic-bomb
Trinket.io Python example:
https://trinket.io/python3/c47ff05883