

BootCon Project: Password Strength and Complexity



By: Melai, Charity, Myia, Lauren, Nabta





Introduction

In the digital age, the importance of robust password security cannot be overstated. With cyber threats on the rise, ensuring that passwords are both complex and secure is a critical task for individuals and organizations alike. This project involves the development of a Python script designed to evaluate the complexity and security of encrypted passwords.



About the project

Objectives:

- **Assess Password Complexity:** The script will analyze passwords to determine their strength based on various criteria such as length, use of uppercase and lowercase letters, numbers, and special characters.
- **Evaluate Security:** It will also check for common weaknesses, such as the use of dictionary words, common patterns, and previously breached passwords.
- **Encryption Handling:** By working with encrypted passwords, the script ensures that the actual passwords are never exposed, maintaining confidentiality while performing the necessary checks.



FEATURES

Complexity Scoring: Provides a numerical score or grade representing the complexity of each password.

Security Recommendations: Offers suggestions for improving weak passwords.

Compatibility: Supports various encryption algorithms to ensure wide applicability.

Efficiency: Optimized for performance, allowing it to handle large datasets of passwords



Project goals



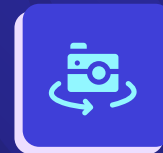
Assess Password Complexity

The script will analyze passwords to determine their strength based on various criteria such as length, use of uppercase and lowercase letters, numbers, and special characters.



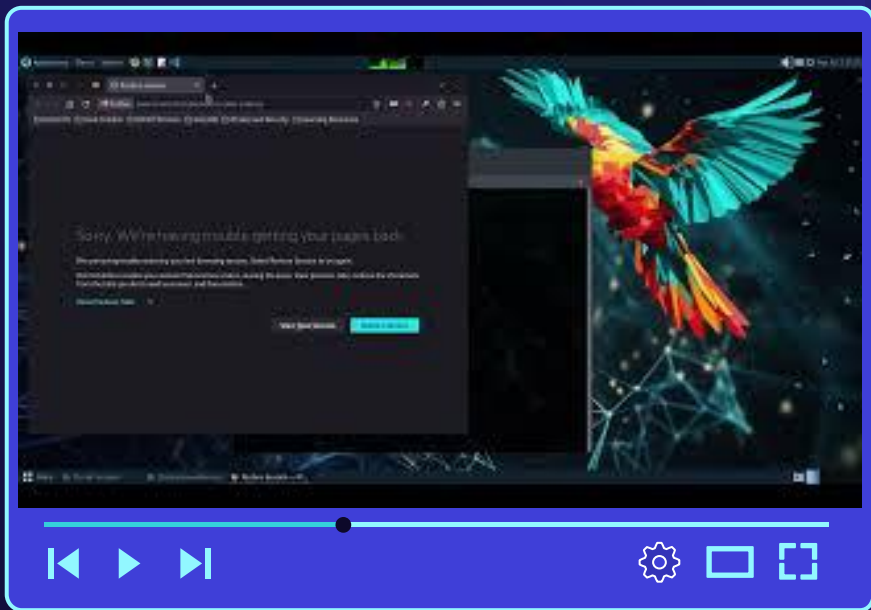
Evaluate Security

It will also check for common weaknesses, such as the use of dictionary words, common patterns, and previously breached passwords.



Encryption Handling

By working with encrypted passwords, the script ensures that the actual passwords are never exposed, maintaining confidentiality while performing the necessary checks.



DEMO

PYTHON SCRIPTS IN ACTION



EXPLANATION

- The “***index***” route displays a simple registration form.
- The “***register***” route handles the form submission, checks the password complexity, and either provides feedback or displays the salt and hashed password.
- The “***check_password_complexity***” function checks the complexity of the password based on length, presence of uppercase and lowercase letters, digits, special characters, and entropy.
- The “***hash_password***” function hashes the password using SHA-256 with a randomly generated salt.

Best Practices for Password Security

Tips for Creating Strong Passwords

Length and Complexity:

- Use at least 12 characters.
- Combine uppercase and lowercase letters, numbers, and special characters.
- Avoid using easily obtainable information, such as your name, birthday, or common phrases.

Recommendations for Storing and Managing Passwords

Password Managers:

- Use a reputable password manager to store and manage your passwords securely. They can generate strong passwords and fill them in automatically when needed.
- Examples include LastPass, 1Password, and Bitwarden.

Tools

2FA Apps:

- Google Authenticator: A free app that generates time-based one-time passwords (TOTP).
- Authy: Similar to Google Authenticator but includes cloud backup and multi-device syncing.