**Объясни детально что такое хэширование пароля. кто его делает, зачем, что на входе, что на выходе, каков процесс? что за модуль hashlib, какие у него основные методы, атрибуты?**

Хеширование паролей – это процесс преобразования исходной строки (пароля) в фиксированную строку символов определенной длины, которая называется хешем. Этот процесс является односторонним, то есть из хеша нельзя восстановить исходный пароль. Основная цель хеширования заключается в защите паролей пользователей от несанкционированного доступа даже в случае утечки базы данных.

**Кто выполняет хеширование?**

Обычно хеширование выполняется системой аутентификации при регистрации пользователя или изменении пароля. Когда пользователь вводит пароль, система преобразует его в хеш и сохраняет именно хеш, а не сам пароль. При последующей проверке подлинности система снова хеширует предоставленный пользователем пароль и сравнивает результат с сохранённым хешем.

**Зачем нужно хешировать пароли?**

Основные причины использования хеширования:

1. **Безопасность**: Даже если злоумышленник получит доступ к базе данных, он не сможет узнать реальные пароли пользователей, поскольку хранятся только хеши.
2. **Конфиденциальность**: Хеширование защищает личные данные пользователей, предотвращая возможность восстановления исходных паролей.
3. **Стандартизация хранения**: Независимо от длины пароля, хеш всегда имеет одинаковую длину, что упрощает хранение и обработку данных.

**Что происходит на входе и выходе?**

На вход подается строка текста (пароль), а на выходе получается строка фиксированной длины (хеш). Например, если используется алгоритм SHA-256, длина хеша составит 64 символа (256 бит).

**Процесс хеширования**

Процесс хеширования включает несколько шагов:

1. **Преобразование пароля**: Пароль преобразуется в байтовый массив.
2. **Применение алгоритма хеширования**: К байтовому массиву применяется выбранный алгоритм хеширования (например, MD5, SHA-256, Bcrypt и др.).
3. **Получение хеша**: На выходе получаем хеш – строку фиксированной длины, представляющую собой уникальное представление исходного пароля.

**Пример процесса хеширования**

Допустим, у нас есть пароль "password123" и мы используем алгоритм SHA-256:

import hashlib

# Исходный пароль

password = "password123"

# Преобразуем пароль в байты

byte\_password = password.encode('utf-8')

# Применяем алгоритм хеширования SHA-256

hash\_object = hashlib.sha256(byte\_password)

# Получаем хеш

hex\_digested = hash\_object.hexdigest()

print(hex\_digested)

Результат выполнения кода:

ddeddc03e59b0d9fccbc1d53ffc6e4e10b96ce34c51ae7b8ee30f27a79e01ed7

Таким образом, вместо сохранения пароля "password123" в базу данных сохраняется хеш ddeddc03e59b0d9fccbc1d53ffc6e4e10b96ce34c51ae7b8ee30f27a79e01ed7.

**Модуль hashlib**

Модуль hashlib в Python предоставляет интерфейс для работы с различными алгоритмами хеширования. Он поддерживает множество популярных алгоритмов, таких как MD5, SHA-1, SHA-256 и другие.

**Основные методы модуля hashlib**

1. **hashlib.md5()**
   * Создает объект хеш-функции MD5. Несмотря на свою популярность, MD5 считается небезопасным и больше не рекомендуется для защиты паролей.
2. **hashlib.sha1()**
   * Создает объект хеш-функции SHA-1. Хотя SHA-1 более безопасен, чем MD5, он также считается уязвимым и не рекомендуется для современных систем.
3. **hashlib.sha224(), hashlib.sha256(), hashlib.sha384(), hashlib.sha512()**
   * Эти функции создают объекты хеш-функций семейства SHA-2. Они считаются достаточно безопасными для большинства применений.
4. **hashlib.new(name)**
   * Универсальный метод создания объекта хеш-функции по имени. Поддерживает различные алгоритмы, такие как md5, sha1, sha256 и другие.

**Основные атрибуты объектов хеш-функций**

1. **digest\_size**
   * Размер дигеста (результата хеширования) в байтах.
2. **block\_size**
   * Размер блока данных, обрабатываемого функцией хеширования.
3. **name**
   * Имя алгоритма хеширования.
4. **update(data)**
   * Обновляет текущий объект хеш-функции данными. Можно вызывать несколько раз для последовательной обработки больших объемов данных.
5. **digest()**
   * Возвращает дигест (результат хеширования) в виде байтового массива.
6. **hexdigest()**
   * Возвращает дигест в виде строки шестнадцатеричных цифр.

**Заключение**

Хеширование паролей – это важный аспект безопасности приложений. Оно позволяет защитить конфиденциальные данные пользователей, обеспечивая невозможность восстановления исходных паролей из хранимых хешей. Модуль hashlib в Python предоставляет удобный интерфейс для работы с различными алгоритмами хеширования, позволяя разработчикам легко интегрировать эту функциональность в свои проекты.

Top of Form