Драчевський Н. В., здобувач,
Покотило О. А., ст. викладач,
Щур Н.О., ст. викладач
Державний університет «Житомирська політехніка»

ЗАСТОСУВАННЯ АЛГОРИТМІВ ХЕШУВАННЯ ДЛЯ ЗАХИСТУ ПАРОЛІВ У ПРОГРАМАХ НА МОВІ PYTHON

У сучасному цифровому середовищі, де зберігаються великі обсяги конфіденційної інформації, їх захист стає однією з найбільш актуальних проблем для багатьох компаній, розробників програмного забезпечення і користувачів. Використання надійних алгоритмів хешування для захисту паролів дозволяє забезпечити безпеку інформації, зокрема у веб-розробці, де дані зберігаються в базах даних. Тому питання їх застосування є актуальним.

Метою даного дослідження є аналіз різних алгоритмів хешування, виявлення їх вразливостей та методів підвищення безпеки за допомогою хешування паролів у програмах, написаних на Python.

Python дозволяє широко використовувати різноманітні бібліотеки та модулі, що сприяють високому рівню безпеки при збереженні паролів. Зазвичай, на практиці застосовуються наступні: passlib, hashlib та bcrypt.

Passlib – це бібліотека, спеціалізована на роботі з паролями, яка надає багато функцій для їх безпечного зберігання та перевірки. Вона підтримує різні алгоритми хешування, зокрема SHA-1, SHA-2, SHA-3, bcrypt, scrypt і Argon2 та має можливість працювати з різними системами збереження паролів. Passlib може застосовуватися для роботи з системами аутентифікації веб-фреймворків, таких як Flask або Django, а також для безпечного зберігання та перевірки паролів користувачів в системах аутентифікації. Проте використання passlib може бути складним через широкий спектр можливостей, а інтеграція у деякі існуючі системи вимагатиме додаткових зусиль.

Модуль hashlib входить до стандартної бібліотеки Python, та надає можливість взаємодіяти з хеш-функціями MD5, SHA-1, SHA-2, SHA-3 і BLAKE2. Крім того, в ньому є можливість використання функції pbkdf2_hmac(), яка реалізує алгоритм PBKDF2 (Password-Based Key Derivation Function 2), який кілька разів проводить хешування пароля та солі (salt – випадкове унікальне значення, яке додається до пароля перед тим, як його хешувати). Даний модуль не надає спеціалізованих інструментів для безпечного зберігання паролів, а деякі з підтримуваних хеш-функцій (MD5, SHA-1) є вразливі до атак колізій. Його доцільно використовувати для зберігання хешів паролів та при перевірці цілісності файлів шляхом обчислення хеш-суми.

Бібліотека bcrypt спеціалізується на алгоритмі хешування з аналогічною назвою на основі шифру Blowfish. Він вважається одним з найбезпечніших для збереження паролів. Завдяки тому, що модуль bcrypt дозволяє керувати параметрами алгоритму, це перетворює його в ефективний засіб захисту паролів. Але при високому рівні складності алгоритм може бути повільнішим в порівняні з іншими, та потребує більших обчислювальних ресурсів, що може впливати на продуктивність системи. Bcrypt найкраще застосовувати при захисті паролів веб-додатків та в системах аутентифікації Python для забезпечення міцності паролів [1].

Серед методів підвищення безпеки при збереженні паролів у програмах, розроблених на Python, можна виділити наступні:

– додавання випадкової унікальної інформації до паролю перед процесом хешування для ускладнення атак типу «rainbow table»;

– використання сучасних безпечних алгоритмів, що спеціально розроблювались для захисту паролів і здатні протистояти атакам;

– забезпечення обробки помилок аутентифікації при невдалих спробах входу, щоб уникнути надмірного розкриття інформації;

– використання функції хешування до паролю більше одного разу для ускладнення зловмисних атак [2].

Таким чином, застосування відповідних методів хешування паролів у програмах на Python є надзвичайно важливим для забезпечення захисту конфіденційної інформації користувачів. Сучасні алгоритми bcrypt або Argon2 – найкращий варіант для зберігання паролів, оскільки вони володіють високою стійкістю до атак.

Важливо дотримуватися розглянутих методів підвищення безпеки та постійно слідкувати за оновленнями та рекомендаціями для використання найбільш сучасних та найбезпечніших шляхів зберігання паролів.

Список використаної літератури

1. Python: 3 Ways to Hash a Password. URL: https://www.slingacademy.com/article/python-ways-to-hash-a-password/

Password Hashing and Salting: Performance vs Security. URL: https://www.linkedin.com/advice/0/how-do-you-balance-performance-security-when