DOI 10.31891/2307-5732-2024-331-6
УДК 004.056

БАГРІЙ РУСЛАН
Хмельницький національний університет
https://orcid.org/0000-0001-5219-1185
e-mail: bahriiro@khmnu.edu.ua
БАРМАК ОЛЕКСАНДР
Хмельницький національний університет
https://orcid.org/0000-0003-0739-9678
e-mail: alexander.barmak@gmail.com
МАНЗЮК ЕДУАРД
Хмельницький національний університет
https://orcid.org/0000-0002-7310-2126
e-mail: eduard.em.km@gmail.com

ПІДВИЩЕННЯ СТІЙКОСТІ ПАРОЛІВ У ВЕБ-СИСТЕМАХ ЗА ДОПОМОГОЮ ВДОСКОНАЛЕНИХ СХЕМ ХЕШУВАННЯ

Підвищення стійкості паролів є важливим аспектом безпеки веб-систем. Це особливо важливо, оскільки багато користувачів використовують недостатньо надійні паролі, що створює ризик несанкціонованого доступу до їх облікових записів. Одним із підходів до підвищення стійких до зламу паролів є використання технологій хешування. При створенні облікового запису користувача пароль перетворюється на хеш-код за допомогою обраної хеш-функції. Розвиток паралельних обчислень дозволяє здійснювати багато атак при зламі хешів паролів. Для протидії таким атакам необхідно постійно розробляти нові схеми хешування паролів, які будуть ефективними та забезпечуватимуть вищий рівень безпеки паролів у веб-системах.

В роботі проведено дослідження схем хешування паролів для підвищення стійкості паролів до різних типів атак для веб-систем. Також висвітлено важливість оновлення та вдосконалення методів хешування та застосування найновіших стандартів безпеки.

Ключові слова: безпека веб-систем, схеми хешування паролів, стійкість паролів.

BAHRII RUSLAN, BARMAK OLEXANDER, MANZIUK EDUARD
Khmelnytskyi National University

IMPROVING THE RESISTANCE OF PASSWORDS IN WEB SYSTEMS USING ADVANCED HASHING SCHEMES

Researching the security of web systems is a relevant and integral component in the process of developing and operating Internet projects. Ensuring the security of user passwords is a key aspect in this context, as compromised passwords can lead to undesirable consequences, including loss of sensitive information, unauthorized access and website compromise.

One approach to making passwords more resistant to cracking is the use of hashing techniques. When a user account is created, the password is hashed using the selected hash function. The development of parallel computing allows for many attacks when cracking password hashes. To counter such attacks, it is necessary to constantly develop new password hashing schemes that will be effective and provide a higher level of password security in web systems.

The results of the study confirm that most of the popular frameworks used for the development of web systems do not provide a sufficiently high level of password protection. Many of them simply hash user data using fast and less stable algorithms such as SHA2 or MD5. This makes passwords vulnerable to hash table and dictionary attacks.

According to OWASP guidelines and accepted security practices, highly robust salted hashing algorithms are one of the most effective methods for keeping passwords secure. Such algorithms provide much greater resistance to various attacks, including attacks using specialized hardware and parallel computing.

Argon2id is one of the most robust password hashing algorithms that has won a password hashing contest and is considered one of the most reliable options for password security. Using Argon2id allows you to configure configuration parameters such as the minimum memory size, the number of iterations and the degree of parallelism, which allows you to achieve an optimal balance between security and performance.

Keywords: security of web systems, password-hashing schemes, resistance of passwords

Постановка проблеми

Зловмисники використовують недостатній рівень знань користувачів веб-систем щодо управління паролями та отримають доступ до великих обсягів даних, пов'язаних з цими користувачами. Такі атаки можуть призвести до серйозних фінансових втрат для власників веб-ресурсів.

Зростаючий прогрес у сфері спеціалізованих апаратних пристроїв та паралельних обчислень створює нові можливості для обчислювальних атак [1]. Використання апаратних пристроїв ASIC (англ. Application-Specific Integrated Circuit, «Інтегральна схема для специфічного застосування»), технології FPGA (Field-Programmable Gate Array, Програмована логічна матриця) та графічних процесорів (GPU) сприяє значному прискоренню розкриття інформації користувача за допомогою паралельного перебору багатьох можливих паролів [2].

Дослідження безпеки веб-систем вказують на недостатній рівень захисту паролів у більшості популярних фреймворків, які використовуються для розробки [3]. Більшість з них, задля зручності та швидкості реалізації, застосовують прості методи хешування, які не є безпечними з точки зору сучасних стандартів і можуть бути легко зламані зловмисниками [4].

Один із найпоширеніших недоліків полягає в використанні застарілих алгоритмів хешування, таких

як MD5 і SHA1. Ці алгоритми є вразливими до атак, оскільки можуть бути легко розкриті шляхом перебору або використанням попередньо згенерованих хеш-таблиць (rainbow tables) [5].

Деякі фреймворки можуть використовувати базові схеми хешування, такі як SHA256 або SHA512, що покращує безпеку в порівнянні з MD5 і SHA1, проте вони все ще можуть бути вразливі до атак з використанням спеціалізованих апаратних засобів або паралельних обчислень [6].

Ще однією проблемою є ситуація коли багато користувачів мають однаковий пароль і результати хешування також будуть однаковими. Коли один із цих паролів розкривається, облікові дані всіх користувачів можуть бути розкриті. Також, оскільки один користувач може застосовувати однаковий пароль у багатьох облікових записах або використовувати єдину платформу входу (наприклад, соціальну платформу входу Facebook) для багатьох веб-систем, розкриття одного з цих облікових записів викликає загрозу щодо безпеки решти веб-систем.

Метою роботи є дослідження схем хешування паролів для підвищення стійкості паролів до атак та злому веб-систем. Для досягнення мети пропонується використання додаткових технік, таких як солі (модифікатора), ітеративне хешування та використання інших сучасних алгоритмів хешування.

Виклад основного матеріалу

Пароль є секретом, який запам'ятовується користувачем і складається з послідовності друкованих символів.

При створенні облікового запису користувача пароль перетворюється на хеш-код за допомогою обраної хеш-функції. Хеш-код зберігається у системі замість самого пароля. Під час процесу автентифікації система порівнює хеш-код, збережений у базі даних, з хеш-кодом, обчисленим з введеного користувачем пароля. Якщо хеш-коди співпадають, пароль вважається правильним (рис.1).

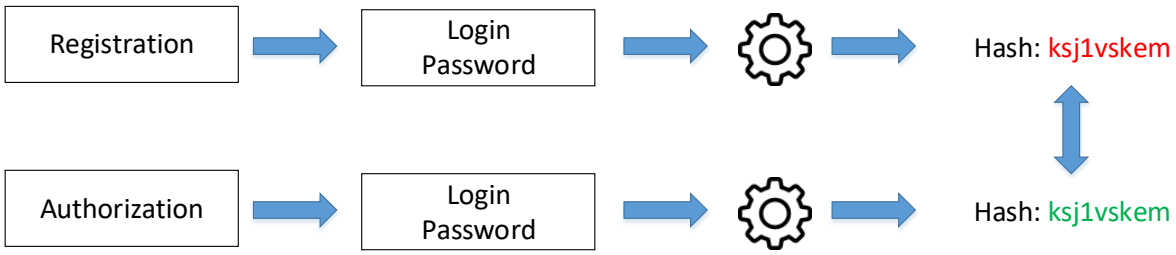

Рис. 1. Схема реєстрації та автентифікації облікових записів з хешуванням

Хеш-функції є математичними алгоритмами, які перетворюють вхідні дані будь-якого розміру в вихідний хеш-код фіксованої довжини. Важливою властивістю хеш-функцій є те, що навіть незначні зміни у вхідних даних повинні призводити до значних змін у вихідному хеш-коді. Виконання хешування має бути швидким і ефективним, два різних вхідних значення мають призводити до різних хеш-кодів. Також хеш-функції мають властивість незворотності, тобто неможливо відновити вхідні дані з хеш-коду.

Однією з рекомендацій OWASP (Open Web Application Security Project) є використання хешування паролів з сіллю, що дозволяє забезпечити більший рівень захисту паролів користувачів [7]. Сіль – це випадковий набір символів, який додається до паролю перед його хешуванням (рис. 2).

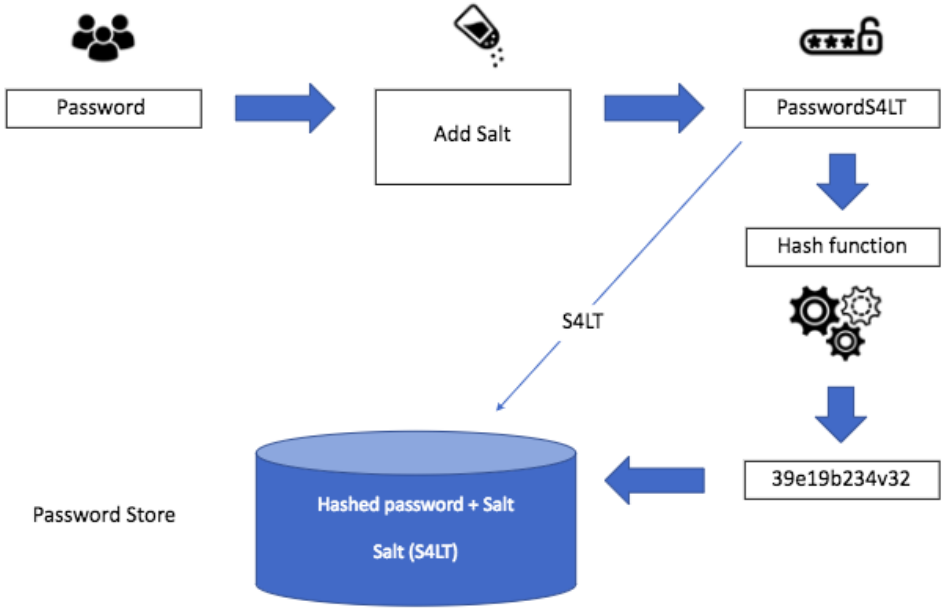

Рис. 2. Схема хешування пароля з додаванням солі [8]

Додавання солі дозволяє отримати унікальні хеш-коди для кожного вводу, незалежно від того, чи є введення унікальним. Це дозволяє захистити веб-систему від злому на основі хеш-таблиць, а також значно підвищує складність для методів злому на основі словників та з використанням повного перебору [9].

Ще однією рекомендацією OWASP для покращення стійкості паролів є збільшенні часу за допомогою ітеративного хешування [7]. Кількість ітерацій впливає на кількість повторень алгоритму хешування, що виконується для створення хешу пароля. Чим більше ітерацій, тим більше часу займає процес хешування, і тим більш ресурсозатратний він стає для зловмисників, які намагаються зламати хеш методом перебору.

Розглянемо схему хешування паролів на прикладі сучасного фреймворку Laravel.

Laravel використовує алгоритми хешування Bcrypt та Argon2id для зберігання паролів користувачів [4]. При хешуванні пароля алгоритмом Bcrypt (за замовчуванням), Laravel автоматично генерує випадкову сіль, що додається до паролю перед хешуванням. Зазвичай, сіль розміщується безпосередньо у хеші паролю:

```
$2y$10$CxVTUcDeou7.Q7kmILOlR.5xOH6SQfeg.12nXK8REmVhIexH8jv/6
```

У цьому прикладі, **2y** позначає версію Bcrypt, **10** вказує на кількість раундів хешування (складність хешування), а всі решта символів після другого доларового знаку – це поєднання солі та хешу паролю.

При автентифікації, Laravel використовує цей хеш, розділяє його на складові частини, витягує сіль і застосовує її разом із введеним паролем, який потім порівнює зі збереженим хешем в базі даних. Це дозволяє Laravel правильно автентифікувати користувачів і забезпечити безпеку паролів.

Алгоритми хешування Argon2 є переможцем конкурсу хешування паролів та вважається одним із найбільш стійких алгоритмів хешування паролів [10].

У версії Argon2id є можливість налаштування параметрів, таких як мінімальний розмір пам'яті, мінімальна кількість ітерацій та ступінь паралелізму, що дозволяє збалансувати використання ресурсів та безпеку залежно від конкретних потреб веб-застосунки.

Для переходу на цей алгоритм хешування необхідно змінити у файлі config/hashing.php проекту ключ 'driver' в масиві 'defaults' на значення 'argon2id'.

```
'defaults' => [
    'driver' => 'argon2id',
],
```

Також можна змінити основні параметри алгоритму хешування:

```
'argon2id' => [
    'memory_cost' => 1024,  // Мінімальний розмір пам'яті (в KB)
    'time_cost' => 2,       // Мінімальна кількість ітерацій
    'threads' => 2,         // Ступінь паралелізму
],
```

Наприклад, збільшення значення `memory_cost` робить хешування більш ресурсозатратним, що робить атаки методом перебору менш ефективними.

При виборі оптимальних параметрів хешування Argon2id слід робити баланс між безпекою і продуктивністю. Занадто високі значення параметрів можуть збільшити витрати обчислювальних ресурсів і зробити реєстрацію і аутентифікацію повільнішими для користувачів. Занадто низькі значення можуть знизити рівень захисту.

Згідно з рекомендаціями OWASP та результатами конкурсу хешування паролів [11] на даний момент прийнятний рівень безпеки досягається при значення:

- використання оперативної паняті: від 64 до 128Мб;
- кількість ітерацій: від 2 до 4;
- кількість потоків: рівне кількості ядер процесора, які має сервер.

Висновки

Дослідження безпеки веб-систем показують, що збереження паролів у безпечних форматах є важливою складовою забезпечення високого рівня захисту для користувачів і веб-сайтів загалом. Використання хешування паролів з додаванням солі та інших технік дозволяє захистити веб-системи від різних атак, зокрема використання хеш-таблиць та атак за словником.

Рекомендації OWASP та інших організацій з безпеки дозволяють досягти оптимального балансу між безпекою та продуктивністю. Застосування високостійких алгоритмів хешування, таких як Argon2id, спільно з додатковими техніками безпеки, допоможе забезпечити надійний рівень захисту паролів та підвищити безпеку веб-систем

Однак, варто пам'ятати, що безпека – це постійний процес, і рекомендації можуть змінюватися з часом залежно від еволюції загроз та технологій. Підтримка безпеки веб-систем вимагає постійного оновлення і вдосконалення методів хешування та застосування найновіших стандартів безпеки.

Література

1. Alkhwaja, I.; Albugami, M.; Alkhwaja, A.; Alghamdi, M.; Abahussain, H.; Alfawaz, F.; Almurayh, A.; Min-Allah, N. Password Cracking with Brute Force Algorithm and Dictionary Attack Using Parallel Programming. Appl. Sci. 2023, 13, 5979. https://doi.org/10.3390/app13105979.

2. Gillela, M.; Prenosil, V.; Ginjala, V.R. Parallelization of Brute-Force Attack on MD5 Hash Algorithm on FPGA. In Proceedings of the 2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID), Delhi, India, 5–9 January 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 88–93.

3. How Secure are the Top Frameworks for Development? Available at: https://hackernoon.com/how-secure-are-the-top-frameworks-for-development

4. Goyal, Divyam and Jain, Pulkit and Bhushan, Bharat, Enhancement of Security using Various Web Development Frameworks. Proceedings of the International Conference on Innovative Computing & Communications (ICICC) 2020.

5. Hatzivasilis, G. Password-Hashing Status. Cryptography, 2017. https://doi.org/10.3390 /cryptography1020010

6. Chang, D.; Jati, A.; Mishra, S.; Sanadhya, S.K. Cryptanalytic time-memory trade-off for password hashing schemes. Int. J. Inf. Secur. 2019, 18, 163–180.

7. Password Storage Cheat Sheet. Available at: https://cheatsheetseries.owasp.org/cheatsheets/ Password_Storage_Cheat_Sheet.html

8. What are Salted Passwords and Password Hashing? Available at: https://www.okta.com/blog/2019/03/what-are-salted-passwords-and-password-hashing/

9. Adding Salt to Hashing: A Better Way to Store Passwords. Available at: https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/

10. Password Hashing Competition. Available at: https://en.wikipedia.org/wiki/Password_Hashing_Competition

11. Biryukov, A., Dinu, D., & Khovratovich, D. Argon 2. The memory-hard function for password hashing and other applications, 2015.