УДК 004.62

Гавліцький В.Ф.,
начальник Навчального Курсу
факультету інформаційних технології
Військового інституту телекомунікацій
та інформатизації імені Героїв Крут
Цегольник В. В., здобувач 2 курсу
спеціальності Комп'ютерні науки

JWT ТА ЙОГО РОЛЬ У WEB-ПРОГРАМУВАННІ

Донецький національний університет імені Василя Стуса м. Вінниця

JWT (JSON Web Tokens) - це компактний та самодостатній формат веб-токена, який зазвичай використовується для передачі інформації між двома сторонами у безпечному вигляді. Загалом це автентифікація користувачів у веб-додатках, хоча подекуди їх також можна використовувати для авторизації (надання доступу до певних ресурсів або операцій у програмі). Він складається з трьох частин: заголовка (header), корисного навантаження (payload) та підпису (signature) [3].

Заголовок JWT є JSON-об'єктом, який містить інформацію про тип токена та алгоритм підпису, що використовується для генерації підпису. Його вигляд показаний на рис. 1, де alg це алгоритм підпису HMAC-SHA256, позначений як "HS256", а typ – тип токена. У випадку JWT це поле завжди матиме значення "JWT" для можливості відрізнити від інших токенів, які можуть існувати.

```
{
    "alg": "HS256",
    "typ": "JWT"
}
```

Рис. 10. Приклад заголовка JWT

Корисне навантаження JWT містить дані, які ми хочемо передати. Це може бути будь-який JSON-об'єкт, який містить корисну інформацію, наприклад, ідентифікатор користувача чи термін дії токена.

Підпис JWT використовується для перевірки цілісності токена та підтвердження його автентичності. Він складається з заголовка та корисного навантаження, які конкатенуються разом з секретним ключем або приватним ключем (залежно від використаного алгоритму підпису), і потім підписується за алгоритмом, вказаним у заголовку [4].

Кожна з трьох частин представлена у форматі Base64Url і розділена крапкою. Після отримання токена, його можна розкодувати, перевірити цілісність підпису і отримати доступ до корисного навантаження.

Що стосується авторизації бази даних за допомогою JWT, ось огляд того, як цього можна досягти:

1. Автентифікація користувача: користувач надає серверу свої облікові дані (наприклад, ім'я користувача та пароль).

2. Процес автентифікації: сервер перевіряє облікові дані та, якщо вони дійсні, генерує JWT.

3. Генерація JWT: сервер створює JWT, що містить ідентифікаційні дані користувача та будь-яку додаткову відповідну інформацію, як-от ролі або дозволи.

4. Відповідь JWT: сервер надсилає JWT назад клієнту (зазвичай у тілі відповіді або як заголовок HTTP).

5. Клієнтське сховище: клієнтська програма безпечно зберігає JWT. Зазвичай він зберігається в локальному сховищі або як файл cookie лише HTTP.

6. Подальші запити: клієнт включає JWT у заголовки наступних запитів до сервера, зазвичай використовуючи заголовок «Авторизація» зі схемою «Носія» (наприклад, Авторизація: носій <jwt_token>).

7. Перевірка JWT: після отримання запиту сервер перевіряє підпис JWT, забезпечуючи його цілісність і автентичність.

8. Контроль доступу: сервер витягує особу користувача та будь-які відповідні дані з JWT. На основі цієї інформації він виконує перевірку авторизації, щоб визначити, чи має користувач необхідні дозволи для доступу до запитуваних ресурсів бази даних.

9. Доступ до бази даних: якщо перевірки авторизації пройшли, сервер дозволяє запитані операції з базою даних [5].

Важливо зауважити, що фактичні деталі реалізації можуть відрізнятися залежно від мови програмування, фреймворку та конкретних вимог вашої програми. Бібліотеки та фреймворки часто забезпечують вбудовану підтримку для обробки JWT, включаючи генерацію маркерів, перевірку та витяг запитів користувачів.

Коли справа доходить до авторизації бази даних, дуже важливо правильно налаштувати сервер для обробки автентифікації та авторизації. Це може включати інтеграцію перевірки JWT і впровадження відповідних механізмів контролю доступу у серверний код для захисту ресурсів бази даних від несанкціонованого доступу. Завжди варто переконуватися, що JWT належним чином захищений, використовуючи надійні криптографічні алгоритми, і дотримуватись найкращих практик щодо зберігання та передачі токенів, щоб мінімізувати ризик уразливості безпеки [2].

Список літератури:

1. *Половенко Л. П. , Мерінова С.В. Технології продукування знань на основі веб-сервісів. Наукові перспективи, № 5(22). 2022.*
2. *Barybin O., Zaitseva E., Brazhnyi V. "Testing the Security ESP32 Internet of Things Devices". IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), 2019. URL: https://ieeexplore.ieee.org/abstract/document/9061269*
3. *"JSON Web Tokens Introduction". JWT Documentation. URL: https://jwt.io/introduction/*
4. *"The Anatomy of a JSON Web Token". Sevilleja, C. URL: https://www.digitalocean.com/community/tutorials/the-anatomy-of-a-json-web-token*
5. *"Understanding JWT". Atlassian Connect Documentation. URL: https://web.archive.org/web/20150518082002/https://developer.atlassian.com/static/connect/docs/latest/concepts/understanding-jwt.html*