Ожго Ю. А., здобувач
Дацюк Д. В., ст. викладач
Державний університет «Житомирська політехніка»

БЕЗПЕКА JSON WEB TOKEN (JWT): ВРАЗЛИВОСТІ ТА РЕКОМЕНДАЦІЇ ЩОДО ЗАХИСТУ

У сучасних інформаційних системах захист даних є критично важливим аспектом, особливо у веб-додатках, де автентифікація та авторизація забезпечують контроль доступу до ресурсів. Одним із поширених методів автентифікації є використання JSON Web Token (JWT) — відкритого стандарту для створення токенів доступу у форматі JSON, що зберігаються на стороні клієнта та забезпечують масштабовану авторизацію без необхідності підтримки сесійного стану на сервері. Незважаючи на ефективність і гнучкість цього механізму, його некоректне використання може призводити до численних загроз безпеці.

JWT складається з трьох основних компонентів: заголовка (header), корисного навантаження (payload) та підпису (signature). Заголовок містить алгоритм підпису (alg) і тип токена (typ). У корисному навантаженні зберігається інформація про користувача та метадані токена, зокрема iat (час створення), exp (час закінчення дії) [1], тощо. Підпис формується шляхом хешування заголовка і корисного навантаження за допомогою обраного алгоритму та секретного ключа. Оскільки криптографічний підпис є основним механізмом захисту JWT, його правильне використання має критичне значення, а помилки в реалізації можуть призвести до серйозних загроз безпеці.

Досить поширеною вразливістю є недостатня перевірка підпису. Деякі бібліотеки для роботи з токенами містять окремі функції для розкодування (decode()) та перевірки підпису (verify()). Якщо розробник помилково використовує лише функцію розкодування, система може прийняти змінений токен без перевірки його автентичності. Це дає змогу зловмиснику підробити токен, змінюючи, наприклад, ідентифікатор користувача або рівень доступу. У деяких випадках така вразливість виникає через тестові налаштування, коли перевірка підпису вимикається під час розробки, але після розгортання системи не активується знову. Щоб уникнути цього, необхідно гарантувати перевірку підпису для кожного отриманого токена перед його використанням.

Серйозну загрозу також становить можливість використання алгоритму None, що фактично означає відсутність підпису [2]. Якщо система дозволяє приймати токени з таким алгоритмом, зловмисник може змінити значення alg на None, видалити підпис і отримати несанкціонований доступ. Це повністю обходить механізм перевірки автентичності токена, що може призвести до компрометації облікових записів або доступу до конфіденційної інформації. Щоб уникнути цієї вразливості, додатки повинні явно відхиляти JWT із алгоритмом None, незалежно від варіацій регістру (none, NONE, nOnE тощо). Додатково слід забезпечити використання надійних криптографічних алгоритмів і гарантувати, що всі токени підписуються та проходять перевірку перед використанням.

Ще одним критичним ризиком є можливість перехоплення JWT. Якщо токен передається через незахищене з'єднання або зберігається у локальному сховищі браузера (localStorage), зловмисник може викрасти його та повторно використати. Це створює загрозу replay-атаки, коли викрадений токен застосовується повторно навіть після виходу користувача з системи [3]. Для зменшення цього ризику слід використовувати лише захищені канали передачі (HTTPS), встановлювати обмежений термін дії токенів, застосовувати механізми оновлення (refresh tokens) і зберігати JWT у HttpOnly cookie. Останній підхід унеможливлює доступ до токена через JavaScript, що значно знижує ймовірність його викрадення через XSS-атаки.

JWT є ефективним механізмом автентифікації та авторизації у веб-додатках та розподілених системах, проте його некоректна реалізація може спричинити значні загрози безпеці, зокрема недостатню перевірку підпису, атаки із використанням алгоритму None та перехоплення токенів. Аналіз вразливостей цього механізму підтверджує необхідність дотримання принципів безпечної реалізації, оскільки навіть незначні помилки можуть призвести до компрометації даних і несанкціонованого доступу. Мінімізація цих ризиків вимагає використання криптографічно стійких алгоритмів, належної перевірки підпису та захищених методів зберігання і передачі токенів.

Список використаних джерел:

1. Introduction to JSON Web Tokens. URL: https://jwt.io/introduction (дата звернення: 14.03.2025).
2. Critical vulnerabilities in JSON Web Token libraries. URL: https://auth0.com/blog/critical-vulnerabilities-in-json-web-token-libraries (дата звернення: 14.03.2025).
3. Session hijacking attack. URL: https://owasp.org/www-community/attacks/Session_hijacking_attack (дата звернення: 14.03.2025).