

ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ АУТЕНТИФІКАЦІЇ У ВЕБ-ЗАСТОСУНКАХ

Вінницький національний технічний університет

Анотація. У роботі розглянуто основні методи аутентифікації у веб-застосунках, зокрема сесійно-орієнтована аутентифікація на основі Cookies, токен-орієнтована аутентифікація JWT та OAuth. Проаналізовано особливості реалізації, переваги та недоліки кожного з методів з огляду на безпеку, продуктивність та зручність інтеграції. Визначено чинники, які впливають на вибір конкретного методу під час проектування веб-систем, орієнтованих на різні сценарії використання та навантаження.

Ключові слова: аутентифікація, Cookies, JWT, OAuth, веб-безпека, токени, сесії.

Abstract. The paper presents the main authentication methods in web applications, including session-based Cookies, token-based JWT, and OAuth. The study analyzes the implementation features, advantages, and disadvantages of each approach in terms of security, performance, and ease of integration. It also identifies factors affecting the choice of a particular method when designing web systems with diverse use cases and workloads.

Keywords: authentication, Cookies, JWT, OAuth, web security, tokens, sessions.

Вступ

Сучасні веб-застосунки стали повсякденною частиною життя людей, адже вони забезпечують доступ до різноманітних сервісів – від соціальних мереж і онлайн-банкінгу до хмарних сховищ та інструментів командної роботи. У цих умовах надійна аутентифікація користувачів є надзвичайно важливою для захисту конфіденційних даних, запобігання несанкціонованим діям і дотримання політик безпеки. Серед основних підходів, які активно застосовуються у веб-середовищі, можна виокремити сесійно-орієнтовану аутентифікацію за допомогою Cookies, токен-орієнтовану з використанням JSON Web Token (JWT), а також OAuth, що дає змогу надавати стороннім сервісам делегований доступ без розкриття справжніх облікових даних користувача.

Вибір конкретного методу аутентифікації впливає не лише на технічні аспекти роботи веб-застосунку, а й на зручність для кінцевого користувача, безпеку системи та легкість масштабування в майбутньому. Так, належним чином сконфігурована аутентифікація здатна підвищити надійність та позитивний користувацький досвід, тоді як неправильна імплементація може стати джерелом численних вразливостей і порушенням конфіденційності даних. Саме тому важливо порівняти особливості різних методів та оцінити їх з огляду на технічну складність, продуктивність, зручність інтеграції та рівень захисту від поширених атак.

Мета дослідження – визначити, як кожен із ключових методів аутентифікації (Cookies, JWT і OAuth) впливає на архітектурні характеристики веб-застосунків та у яких умовах кожен метод може бути застосований найбільш ефективно.

Сесійно-орієнтована аутентифікація на основі Cookies

Cookies широко застосовуються для збереження інформації про користувача та створення сесії. Після успішного входу на сервері формується сеансовий ідентифікатор, який зберігається у Cookie браузера та використовується для перевірки прав доступу при кожному запиті. Завдяки цьому сервер знає, який користувач робить запит, і може відповідно надавати доступ до певних ресурсів. Такий підхід є доволі зручним у впровадженні, адже більшість фреймворків мають уніфіковані інструменти керування сесіями: автоматичне створення, оновлення та видалення Cookies, а також механізми перевірки поточних прав доступу.

Однак для забезпечення безпеки сесійних Cookies треба налаштовувати додаткові атрибути, такі як HttpOnly, Secure та SameSite для того, щоб захиститися від XSS-атак та CSRF-атак. Ще один виклик полягає у масштабуванні. Коли система починає обробляти багато користувачів одночасно або розгортається на декількох серверах, з'являється потреба у розподіленому зберіганні сесій. У традиційній реалізації всі дані сесії зберігаються у внутрішній пам'яті сервера або в базі даних, а ідентифікатор сесії лише прив'язує користувача до запису на сервері. В умовах кластеризації чи використання мікросервісної архітектури доводиться розгортати додаткові сервіси, наприклад, Redis або Memcached, які виступають централізованим сховищем сесій. Це підвищує складність інфраструктури.

Важливим аспектом є захист від CSRF (Cross-Site Request Forgery). Оскільки браузери за замовчуванням додають Cookies до кожного запиту до домену, користувач може бути обманутим, щоб надіслати шкідливий запит зі своїми Cookies де, наприклад, є сеансовий ідентифікатор. Якщо у системі не впроваджені механізми

перевірки CSRF-токена або інші захисні заходи, цей запит може виконати небажану дію: змінити налаштування профілю, надіслати кошти чи видалити важливі дані. Тому для більшості фреймворків обов'язковим кроком є активація та налаштування вбудованих інструментів проти CSRF-атак.

У підсумку, сесійно-орієнтований підхід на основі Cookies залишається одним із найпопулярніших завдяки простоті та звичності для розробників. Він добре підходить для невеликих чи середніх проєктів, де немає підвищених вимог до масштабування. Але у великих системах необхідно передбачати додаткові рішення для розподіленого зберігання сесій і забезпечувати багаторівневий захист Cookies та запитів від атак CSRF і XSS.

Токен-орієнтована аутентифікація на основі JWT

JSON Web Token, відомий як JWT, зберігає дані у стиснутому та підписаному вигляді, що дає змогу перевіряти його справжність на сервері без звернення до бази чи стану сесії. Зазвичай токен передають у заголовку Authorization або розміщують у локальному сховищі браузера. Оскільки відомості про користувача та час дії записані безпосередньо в самому токені, для збільшення кількості серверів не потрібно синхронізувати сеанси, що полегшує розгортання та масштабування.

Структура JWT складається з двох основних частин і підпису. Заголовок визначає тип токена та алгоритм підпису, а корисне навантаження містить дані про користувача, які називають claims. Завдяки цьому сервер може швидко перевірити автентичність, оскільки вся потрібна інформація є в самому токені. Можна також додавати власні поля, наприклад відомості про роль користувача чи час створення, що робить JWT гнучким у середовищі з кількома сервісами або мобільними клієнтами.

Існують і ризики. Якщо токен перехопили, злоумисник матиме доступ до завершення його дії. Тому краще задавати короткий час життя, застосовувати токени для оновлення та організовувати механізм відкликання у разі підозри на витік. Важливо також подбати про безпечне зберігання, щоб запобігти викраденню токенів через вразливості типу cross site scripting. Один із шляхів зберігати JWT у спеціальних HTTP only «куках», але тоді потрібно впроваджувати захист від CSRF і правильно використовувати атрибути SameSite та Secure. Завдяки зручності перевірки і відсутності залежності від зовнішніх сховищ JWT добре підходить для високонавантажених систем з потребою в ефективному масштабуванні, однак він вимагає ретельного налаштування безпеки.

OAuth (Open Authorization)

Open Authorization, або OAuth, дає можливість стороннім застосункам отримати обмежений доступ до ресурсів користувача без розкриття його особистих даних для входу. Такий підхід поділяє ролі між серверами авторизації та ресурсів, а клієнтський сервіс отримує тимчасовий ключ доступу після дозволу від користувача. Це означає, що один сервіс може виконувати аутентифікацію, а інший керувати ресурсами, не вимагаючи в користувача зайвих паролів.

Існує кілька потоків авторизації, зокрема сценарій, коли користувач перенаправляється на сторінку авторизації, надає згоду, отримує код і повертається до вихідного додатка. Потім система видає короткотривалий токен, який клієнт використовує для звернення до захищених ресурсів. Типовим прикладом є соціальний вхід через Google, Facebook чи GitHub, де користувач дозволяє конкретному сервісу доступ лише до певної частини свого профілю.

Попри зручність та підвищену безпеку, бо паролі не передаються стороннім додаткам, реалізація OAuth є складнішою. Розробникам потрібно коректно налаштовувати сервер авторизації, продумувати сценарії дії токена і забезпечувати процедуру відкликання. Також важливо обмежувати права і визначати термін дії виданих токенів. Завдяки здатності делегувати права та зберігати конфіденційність справжніх облікових даних OAuth став популярною основою соціальних входів і залишається гнучким рішенням для систем із широкою інтеграцією між сторонніми сервісами.

Отже, завдяки гнучкості та широкій підтримці OAuth є основою для інтеграції багатьох сучасних сервісів. Його використання дозволяє покращити безпеку, зменшити навантаження на користувача та забезпечити зручну взаємодію між різними системами без потреби повторної реєстрації.

Висновки

У результаті проведеного дослідження було проаналізовано три основні методи аутентифікації у веб-застосунках: сесійно-орієнтовану на основі Cookies, токен-орієнтовану на основі JWT та делеговану авторизацію за допомогою OAuth. Кожен із підходів має свої переваги, недоліки та специфіку застосування, що визначає доцільність його використання залежно від потреб конкретного проєкту.

Сесійна аутентифікація є простою у впровадженні та зручною для традиційних серверних застосунків, проте потребує додаткових заходів захисту від CSRF та ускладнює масштабування. JWT забезпечує незалежність від стану сервера, що полегшує розгортання у розподіленому середовищі, але вимагає ретельного налаштування

безпеки з огляду на ризики викрадення токена. OAuth забезпечує безпечну взаємодію між сторонніми сервісами та дозволяє реалізувати соціальні входи без передачі облікових даних, однак має складнішу структуру реалізації.

Таким чином, вибір методу аутентифікації має базуватись на особливостях архітектури застосунку, вимогах до безпеки, масштабованості та зручності для кінцевого користувача. У деяких випадках доцільно комбінувати підходи, щоб досягти максимальної гнучкості та захищеності системи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ:

1. Auth0 Documentation. URL: <https://auth0.com/docs/protocols/oauth2> (дата звернення: 21.03.2024).
2. Cookies vs Tokens: The Definitive Guide. URL: <https://auth0.com/blog/cookies-vs-tokens> (дата звернення: 21.03.2024).
3. Web Security: Authentication and Session Management. URL: https://owasp.org/www-project-top-ten/Broken_Authentication (дата звернення: 21.03.2024).

Луп'як Марія Дмитрівна – студентка групи 4ПІ-24б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: marialupiak@gmail.com

Черноволик Галина Олександрівна – доцент кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: lina2433@gmail.com

Lupiak Mariia Dmytrivna – student of group 4PI-21b, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: eugeneedge17@gmail.com

Chernovolyyk Halyna Oleksandrivna – Associate Professor of Program Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: lina2433@gmail.com