Engineer

JWT

компактний токен із твердженнями в JSON, придатний для URL

Що це таке

JWT — це короткий рядок, який переносить набір тверджень від однієї сторони до іншої в такому вигляді, що одержувач може їх перевірити. Абревіатура розшифровується як JSON Web Token. Формат визначено в RFC 7519 у травні 2015 року як компактний спосіб подати твердження (claims), який можна без змін передати в URL, а назву там пропонують вимовляти як англійське слово «jot». Твердження — це назва і значення, які стверджують про суб'єкт: `"sub": "24400320"` каже, про кого токен, а `"exp": 1300819380` — коли він перестає бути чинним.

Під час передавання JWT — це кілька рядків base64url, з'єднаних крапками. Підписаний JWT має три частини: заголовок, що називає алгоритм, корисне навантаження (payload) із твердженнями і підпис, обчислений над першими двома. Зашифрований має п'ять частин, і його твердження неможливо прочитати без ключа. Сам RFC 7519 криптографією не займається. Підписування або захист кодом автентифікації повідомлення (MAC) він покладає на JSON Web Signature (RFC 7515), а шифрування — на JSON Web Encryption (RFC 7516). Алгоритми перелічено в RFC 7518, а ключі можна публікувати у форматі JSON Web Key з RFC 7517.

`HS256` — це HMAC із SHA-256: один спільний секрет і створює токен, і перевіряє його, тож кожен сервіс, здатний його перевірити, може й випустити власний. `RS256` і `ES256` використовують пару ключів: видавець підписує закритим ключем, а всі інші перевіряють відкритим, який можна вільно роздавати. RFC 7519 вимагає реалізувати лише `HS256` і непідписаний `none`, а `RS256` і `ES256` рекомендує.

Розділ 4.1 реєструє сім назв тверджень: `iss` (видавець), `sub` (суб'єкт), `aud` (аудиторія, тобто одержувач), `exp` (час завершення дії), `nbf` (не раніше ніж), `iat` (час видачі) і `jti` (унікальний ідентифікатор, за яким можна не допустити повторного використання токена). Назви мають по три літери, бо формат задумано компактним, і всі вони необов'язкові. Які твердження має містити токен, вирішує застосунок, що його використовує. Крім зареєстрованих, будь-хто може визначати публічні твердження, зареєстровані в IANA або з назвою, що не може конфліктувати з чужими (наприклад, URL), а також приватні, про які домовилися дві сторони.

Застосуйте поняття до рішення

Застосуйте «Що таке JWT?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Облікові записи, паролі й права налаштовані так, щоб доступ відповідав роботі, а звільнені втрачали його того ж дня.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Написала 1 000 правил RBAC для Linux-серверів — 1,000 правил доступу написано, щоб усунути вразливість

Написала 1 000 правил RBAC для Linux-серверів Зв'язатися з нами →

Чому це важливо

Суть JWT у тому, що одержувач може перевірити його сам. Маючи ключ видавця і годинник, сервіс визначає, чи токен справжній, для кого він і чи ще чинний, без запиту до бази даних і без звертання до видавця. RFC 9068 (жовтень 2021 року) констатує, що багато комерційних реалізацій OAuth видають токени доступу, які сервери ресурсів розбирають і перевіряють самостійно, що такий підхід особливо поширений там, де сервер, який видає токени, і сервер, який їх приймає, розділені певною межею, і що багато з цих реалізацій використовують JWT. Далі цей RFC стандартизує структуру таких токенів, щоб вони працювали в рішеннях різних виробників.

JWT — це також формат, на якому побудовано OpenID Connect: кожен ID-токен є JWT, адресованим застосунку, який його запросив. Постачальник ідентичності має підписати його, за винятком випадку, коли одночасно виконано обидві умови використання `alg` зі значенням `none` : тип відповіді не повертає ID-токена з Authorization Endpoint, як у Authorization Code Flow, і клієнт під час реєстрації явно запросив `none` . Коли 2015 року реєстрували медіатип `application/jwt` , у реєстрації серед користувачів уже значилися OpenID Connect, Salesforce, Google, Android, Windows Azure та Amazon Web Services.

RFC 7519 розраховує формат на місця з обмеженим простором, як-от HTTP-заголовок Authorization і параметри запиту в URI. Тому назви тверджень короткі, а кодування використовує лише символи, які проходять в URL без екранування.

Та сама властивість, що робить JWT дешевим у перевірці, ускладнює його відкликання. Одержувач, який перевіряє підпис і час, прийматиме токен до `exp`, хоч би що сталося з обліковим записом за цей час.

Як це виглядає на практиці

RFC 7519 розбирає повний приклад. Заголовок оголошує, що об'єкт є JWT і що його захищено HMAC SHA-256:

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

Твердження називають видавця, час завершення дії та одне публічне твердження, назва якого є URL, щоб воно не конфліктувало з твердженнями, які визначає хтось інший:

```
{"iss":"joe",
 "exp":1300819380,
 "http://example.com/is_root":true}
```

Кожну частину кодують у base64url, HMAC обчислюють над заголовком і корисним навантаженням, з'єднаними крапкою, а три результати з'єднують в один рядок. RFC наводить його з розривами рядків, доданими лише для зручності читання:

```
eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9
.
eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly9leGFt
cGxlLmNvbS9pc19yb290Ijp0cnVlfQ
.
dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
```

З цього прикладу випливають три речі. По-перше, твердження може прочитати кожен, у кого є токен: base64url — це кодування, тож середня частина після декодування знову стає JSON, і ключ для цього не потрібен. По-друге, підпис охоплює закодовані рядки точно в тому вигляді, у якому їх надіслано. Заголовок у RFC містить розрив рядка CRLF і пробіл на початку другого рядка, і RFC наводить точні байти, щоб уникнути неоднозначності. Якщо охайніше відформатувати JSON і закодувати його знову, вийде інший рядок, і підпис перестане сходитися. Ключ для цього прикладу наведено в додатку A.1 до RFC 7515, і з ним підпис успішно перевіряється. По-третє, строк дії токена давно минув: `1300819380` секунд після 1 січня 1970 року — це 22 березня 2011 року, 18:43 UTC, тож коректний валідатор сьогодні його відхилить.

Одержувач, який отримав такий токен, виконує фіксований набір перевірок, перш ніж прочитати бодай одне твердження. Він розділяє рядок за крапками, декодує заголовок і переконується, що це коректний JSON, у якому лише ті параметри й значення, синтаксис і зміст яких він розуміє та підтримує, або які прямо визначено як такі, що їх слід ігнорувати, якщо вони невідомі. Він порівнює `alg` з алгоритмами, які вирішив приймати саме для цього видавця і ключа: RFC 7519 прямо каже, що це рішення застосунку і що токен слід відхилити, якщо алгоритм неприйнятний, навіть коли підпис дійсний. Він перевіряє підпис ключем, який належить видавцеві. Потім перевіряє твердження: `iss` має бути видавцем, якому він довіряє, `aud` має називати самого одержувача, поточний час має бути раніше за `exp` (із запасом щонайбільше кілька хвилин на розбіжність годинників) і пізніше за `nbf`, якщо таке твердження є. Лише після цього він діє на підставі тверджень.

Токен доступу у форматі JWT за RFC 9068 показує, що додає конкретний профіль. Його заголовок у цьому RFC такий: `{"typ":"at+JWT","alg":"RS256","kid":"RjEwOwOA"}`, тобто явний тип, асиметричний алгоритм та ідентифікатор ключа в опублікованому наборі ключів видавця. Профіль робить обов'язковими `iss`, `exp`, `aud`, `sub`, `client_id`, `iat` і `jti`, а сервер ресурсів має перевірити, що `typ` дорівнює `at+jwt` або `application/at+jwt`, і відхилити інші типи. Ці назви медіатипів і підтипів нечутливі до регістру; `at+JWT` у прикладі — рівнозначний запис.

Твердження в токені — це знімок стану на момент підписання, тож усе, що може змінитися до `exp`, краще вирішувати на сервері. В одному з проєктів цього сайту, посиленні захисту застосунку на кожному рівні, сервер

повторно перевіряє права доступу (тариф, роль, організацію, прив'язку до судна) під час кожного запиту, а обмеження на фронтенді вважаються суто питанням UX.

Де виникають проблеми

Токен сам обирає свій алгоритм. Заголовок `alg` існує заради гнучкості, і RFC 8725 описує, як деякі бібліотеки надто йому довіряли. Зловмисник міг змінити його на `none`, і бібліотека приймала токен зовсім без підпису. Або змінити `RS256` на `HS256`, і бібліотека перевіряла HMAC, використовуючи відкритий ключ RSA видавця як спільний секрет, хоча цей ключ зловмисник може просто завантажити. Налаштуйте бібліотеку на алгоритми, які ви приймаєте, використовуйте кожен ключ рівно з одним алгоритмом і відхиляйте `none` без винятків.

Секрет HMAC — це пароль. Токен, підписаний `HS256` коротким секретом, який легко запам'ятати, можна атакувати офлайн: той, у кого є хоч один токен, може перебирати варіанти у власному темпі, не звертаючись до вашого сервера. RFC 8725 забороняє використовувати пароль, який людина здатна запам'ятати, безпосередньо як ключ HMAC, а RFC 7518 вимагає для `HS256` ключа завдовжки щонайменше 256 біт. Генеруйте ключ криптографічно стійким генератором випадкових чисел, зберігайте його у сховищі секретів і обирайте пару ключів, коли багатьом сервісам потрібно лише перевіряти.

Підпис правильний, тож усьому іншому вірять. Дійсний підпис доводить, хто видав токен, але не те, для кого його призначено. Якщо один видавець обслуговує кілька сервісів, а одержувач пропускає перевірку `aud`, токен, законно надісланий одному сервісу, можна повторно використати проти іншого. RFC 8725 називає це підміною і вимагає, щоб одержувачі таких токенів перевіряли аудиторію та відхиляли токен без неї. Звіряйте `aud` із власним ідентифікатором і перевіряйте, що ключ підпису належить видавцеві, зазначеному в `iss`.

Один вид токена видають за інший. ID-токен, токен доступу і токен події безпеки (security event token) від того самого видавця можуть бути дуже схожими, і одержувач, який перевіряє лише підпис і видавця, може прийняти не той вид. RFC 8725 називає це плутаниною між різними JWT (cross-JWT confusion) і рекомендує явно зазначати тип у заголовку `typ`; RFC 9068

застосовує це через `at+jwt`. Дайте кожному виду токенів власний тип або аудиторію і пишіть правила перевірки так, щоб кожен вид відсіювали перевірки, призначені для інших.

Ключ обирає заголовок. Заголовок `kid` вказує, яким ключем скористатися, а `jku` і `x5u` можуть указувати на URL, звідки ключ завантажити. Усі вони надходять від відправника. `kid`, переданий без перевірки в запит до бази даних або в пошук у файловому каталозі, стає шляхом усередину цієї системи, а сервер, який іде за будь-якою URL-адресою з `jku` чи `x5u`, можна змусити надсилати запити куди завгодно. Шукайте `kid` лише серед ключів, які у вас уже є, і завантажуйте ключі тільки з дозволеного переліку адрес, наприклад з опублікованого видавцем `jwks_uri`.

Корисне навантаження може прочитати будь-хто. Підписаний JWT захищає свої твердження від змін, але не від читання, а проходить він часто через браузери, журнали і проксі. RFC 7519 зазначає, що JWT може містити дані, чутливі з погляду приватності, і що найпростіший захист — не додавати їх. Інакше токен слід шифрувати або принаймні передавати лише через TLS. RFC 9068 додає, що клієнт узагалі не повинен зазирати в токен доступу, бо видавець може будь-коли змінити його формат. Кладіть у токен ідентифікатори, а не персональні дані, і шифруйте, коли сам вміст має лишатися приватним.

Що почитати далі

- RFC 7519: JSON Web Token (JWT) — сам формат: зареєстровані твердження, розібраний приклад і кроки перевірки.
- RFC 7515: JSON Web Signature (JWS) — як кодують і з’єднують заголовок, корисне навантаження та підпис, а також ключ до прикладу вище.
- RFC 8725: JSON Web Token Best Current Practices — відомі атаки на JWT і правила для бібліотек і застосунків, які їм запобігають.
- RFC 7518: JSON Web Algorithms (JWA) — назви алгоритмів для `alg` і мінімальний розмір ключа для кожного HMAC.
- RFC 9068: JWT Profile for OAuth 2.0 Access Tokens — стандартна структура токенів доступу у форматі JWT і перевірки, які має виконати сервер ресурсів.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке JWT?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/jwt/