Engineer

OAuth

✒ спосіб дати застосунку обмежений доступ без пароля

Що це таке

OAuth — це протокол, який дає змогу одному застосунку користуватися даними людини в іншому сервісі, з її згоди й у межах, які вона сама бачить, причому застосунок так і не дізнається її пароля. Чинна версія — OAuth 2.0: IETF опублікувала її в жовтні 2012 року як RFC 6749, і її так само часто пишуть OAuth 2.0 чи OAuth2, як і просто OAuth. Вона замінила OAuth 1.0, і ці дві версії між собою несумісні.

Специфікація визначає чотири ролі. Власник ресурсу — той, хто може надати доступ, зазвичай людина. Сервер ресурсів зберігає захищені дані й відповідає на запити, до яких додано токен. Клієнт — застосунок, якому потрібен доступ: вебзастосунок, мобільний застосунок, скрипт, ШІ-агент. Сервер авторизації перевіряє власника, просить його згоди й видає токени. У невеликому продукті ролі сервера ресурсів і сервера авторизації може виконувати одна система; у компанії частіше один постачальник ідентичності стоїть перед багатьма API.

Клієнт отримує токен доступу (access token) — рядок, що означає певний обсяг доступу на певний час. Цей токен клієнт додає до кожного запиту до сервера ресурсів. RFC називає такий рядок зазвичай непрозорим для клієнта: клієнт його передає, але читати не мусить. Сервер авторизації може також видати токен оновлення (refresh token), який клієнт обмінює на новий токен доступу, коли строк дії старого спливає, і людину вдруге не питають.

Перший токен клієнт отримує через грант, тобто спосіб надання дозволу. RFC 6749 визначає чотири: authorization code, implicit, resource owner password credentials і client credentials. Client credentials — для програми, що діє від власного імені, без участі людини. Для всього іншого важить грант authorization code, бо чинні рекомендації з безпеки вивели з ужитку implicit і

password. В IETF ці рекомендації разом із Proof Key for Code Exchange (PKCE) і правилами для bearer-токенів зводять в один документ, OAuth 2.1. У шістнадцятій редакції, датованій 3 вересня 2026 року, це досі Internet-Draft, тобто робочий проєкт, а не стандарт.

OAuth стосується авторизації, а не автентифікації. Токен повідомляє API, що клієнтові дозволено, а про те, хто ця людина, нічого певного клієнтові не говорить. Вхід користувачів поверх OAuth — завдання OpenID Connect, який додає токен, призначений саме для того, щоб його читав клієнт.

Застосуйте поняття до рішення

Застосуйте «Що таке OAuth?» 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

RFC 6749 починається з проблеми, яку мав розв’язати. Щоб сторонній застосунок отримав доступ до ваших даних, ви давали йому свій пароль. Застосунок мусив цей пароль зберігати, зазвичай відкритим текстом. Він діставав усе, до чого діставав пароль, і стільки, скільки хотів. Відкликати доступ в одного застосунку можна було, лише змінивши пароль, а це відрізало й усі інші. А коли зламували будь-який із них, разом із ним витікали пароль і все, що за ним стояло. OAuth прибирає кожну з цих проблем: застосунок ніколи не тримає пароля, токен має обсяг і строк дії, а доступ можна відкликати в одного клієнта, не зачіпаючи решти.

Команда стикається з OAuth з обох боків. Якщо API належить вам, OAuth дає змогу партнерам, інтеграціям і власним інструментам ваших клієнтів викликати його від імені користувача, і ніхто не зберігає чужих паролів. Якщо ви користуєтеся чужим API, то OAuth — це те, що пропонують календарі, поштові скриньки, бухгалтерські системи й файлові сховища, тож клієнтська половина — код, який ваша команда пише й підтримує. ШІ-агенти, що діють від імені людини, займають те саме місце, що й будь-який інший клієнт: їм належить мати токен, обмежений тим, що людина дозволила, а не її облікові дані.

Обсяги доступу (scopes) перетворюють принцип найменших привілеїв на правило, яке сервер може забезпечити. Токен, що дозволяє сервісу друку читати фото, не повинен дозволяти їх видаляти, а токен, виданий для одного API, має відхилити інший. Рекомендації IETF з безпеки 2025 року, RFC 9700, вимагають саме цього: привілеї токена доступу мають бути мінімальними для конкретного випадку, а самі токени — обмеженими сервером ресурсів, для якого їх видано.

Як це виглядає на практиці

Приклад із самого RFC — сервіс друку, якому потрібні фото із сайту для обміну фотографіями. Ось цей обмін у вигляді гранту authorization code з PKCE — схеми, яку чинні рекомендації просять використовувати всіх клієнтів. Насамперед клієнт генерує випадковий code verifier завдовжки від 43 до 128 символів і тримає його в себе. Він хешує verifier алгоритмом SHA-256, кодує результат у base64url і переспрямовує браузер людини на сервер авторизації, передаючи цей результат як code challenge:

```
GET /authorize?response_type=code&client_id=s6BhdRkqt3&state=xyz
    &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
    &code_challenge=E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
    &code_challenge_method=S256 HTTP/1.1
Host: server.example.com
```

Сервер авторизації перевіряє людину, показує, чого просить сервіс друку, і, коли людина погоджується, повертає браузер назад із короткоживучим кодом авторизації. Клієнт перевіряє, що `state` — те саме значення, яке він надсилав:

```
HTTP/1.1 302 Found
Location: https://client.example.com/cb?code=SplxlOBeZQQYbYS6WxSbIA
          &state=xyz
```

Сам собою код нічого не вартий. Клієнт обмінює його, надсилаючи HTTPS-запит до кінцевої точки токенів сервера авторизації, і цього разу передає сам verifier:

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
&client_id=s6BhdRkqt3
&code_verifier=dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
```

Сервер хешує verifier, порівнює результат із challenge, який зберіг поряд із кодом, і лише коли вони збіглися, повертає токени з позначкою, що забороняє кешам їх зберігати:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "access_token":"2YotnFZFEjr1zCsicMWpAA",
  "token_type":"Bearer",
  "expires_in":3600,
  "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA"
}
```

Далі клієнт надсилає `Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA` з кожним викликом до API фотографій. За годину строк дії токена спливає, API відповідає `401 Unauthorized` з `error="invalid_token"` у заголовку `WWW-Authenticate`, і клієнт витрачає токен оновлення, щоб отримати новий. Запит до чогось поза наданим обсягом доступу натомість отримує `403 Forbidden` і `insufficient_scope`. Сам verifier захищає від використання перехопленого коду. Код, скопійований з пристрою під час передавання, не зможе обміняти ніхто, хто не має verifier: саме проти такої атаки спочатку розробили PKCE. Сам verifier не надсилали ні в запиті на авторизацію, ні під час переспрямування браузера.

Те саме делегування дає програмам змогу працювати з агентами. На сторінці сайту про те, як перетворити програми компанії на інструменти для ШІ-агентів, сказано, що OAuth забезпечує делегований доступ: агент діє від імені користувача без передавання пароля і лише в межах прав цього користувача.

Де виникають проблеми

Нестрога перевірка адреси переспрямування віддає код чужому. Сервер авторизації, що приймає будь-який redirect URI, який починається із зареєстрованого префікса чи підпадає під шаблон із символом підстановки, можна змусити доставити код на сторінку, яку контролює зловмисник. RFC 9700 вимагає, щоб сервери авторизації порівнювали URI переспрямування із зареєстрованими за точним збігом рядків, з одним винятком для номерів портів на localhost у нативних застосунках, і забороняє як клієнтам, так і серверам мати відкриті редиректори, що пересилають браузер на довільну адресу з параметра запиту.

Клієнт приймає відповідь, якої не запитував. Якщо клієнт бере будь-який код, що прийшов на його redirect URI, зловмисник може протиснути власний код через браузер жертви, і жертва працюватиме в обліковому записі зловмисника. Це підробка міжсайтових запитів (cross-site request forgery), націлена на переспрямування. Клієнт може покластися на захист PKCE, якщо переконався, що сервер підтримує PKCE; інакше він надсилає одноразове значення `state`, прив'язане до сесії браузера людини, і відхиляє відповідь, яка його не повертає.

Токени подорожують в URL. Грант implicit кладе токен доступу просто в переспрямування, звідки історія браузера й заголовки `Referer` можуть рознести його туди, куди він не мав потрапити, і ніщо не прив'язує його до клієнта, який його запитував. RFC 9700 каже, що клієнтам не слід ним користуватися, а проєкт OAuth 2.1 його взагалі не містить. Те саме стосується API, що приймає токен у параметрі запиту: за RFC 6750 bearer-токени не слід передавати в URL сторінок. Використовуйте грант authorization code і надсилайте токени в заголовку `Authorization`.

Застосунок однаково просить пароль. Грант resource owner password credentials дає клієнтові змогу зібрати ім'я користувача й пароль людини та обміняти їх на токен, що зводить нанівець саму причину існування OAuth. RFC 9700 прямо каже, що його не можна використовувати: він відкриває облікові дані клієнтові, привчає людей вводити пароль деінде, крім сервера авторизації, і ніколи не був розрахований на двофакторну автентифікацію чи вхід у кілька кроків. Відправляйте людей на сервер авторизації через грант authorization code, а коли людини немає взагалі, використовуйте client credentials.

Викрадений токен працює на того, хто його тримає. RFC 6750 визначає bearer-токен як такий, яким може скористатися будь-яка сторона, що ним володіє, без додаткового доказу. Тримайте токени доступу короткоживучими, обмежуйте кожен тим API, для якого його видано, а там, де ставки це виправдовують, прив'язуйте токени до клієнта через взаємний TLS (RFC 8705) або DPoP (RFC 9449). Токени оновлення для публічних клієнтів, як-от мобільних і браузерних застосунків, мають бути або прив'язані до клієнта, або ротуватися: кожне оновлення повертає новий токен і анулює старий. Якщо викрадену копію теж пустили в хід, одна зі сторін рано чи пізно пред'явить анульований токен, сервер відкличе чинний, і атака припиниться ціною того, що справжньому клієнтові доведеться знову просити в людини дозвіл.

Що почитати далі

- RFC 6749: The OAuth 2.0 Authorization Framework — чотири ролі, чотири типи грантів і кінцеві точки в тому вигляді, в якому їх опубліковано в жовтні 2012 року.
- RFC 7636: Proof Key for Code Exchange by OAuth Public Clients — code verifier і code challenge, з прикладом S256, використаним вище.
- RFC 9700: Best Current Practice for OAuth 2.0 Security — рекомендації січня 2025 року: точне порівняння адрес переспрямування, PKCE для кожного клієнта і відмова від грантів implicit і password.
- RFC 6750: Bearer Token Usage — як клієнт пред'являє токен доступу і як API повідомляє, що токена немає, що він прострочений або що його обсяг завузький.
- The OAuth 2.1 Authorization Framework — проєкт робочої групи, що зводить ці документи в один, досі Internet-Draft у жовтні 2026 року.

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 OAuth?» 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/oauth/