Engineer

OpenID Connect

шар ідентифікації поверх OAuth для входу користувачів

Що це таке

OpenID Connect — це стандарт, за яким застосунок дізнається, хто перед ним, від сервісу, що щойно виконав вхід цієї людини. Специфікація сама називає його простим шаром ідентифікації поверх OAuth 2.0, і влаштований він саме так: ті самі перенаправлення, ендпоінти й токени, що й в OAuth, плюс ще один токен, ще один ендпоінт і правила, за якими їх перевіряють. Скорочено його часто пишуть OIDC. Основна специфікація стала остаточною в лютому 2014 року. Чинний текст, до якого внесено другий набір виправлень, датовано 15 грудня 2023 року, і його супроводжує робоча група OpenID Foundation.

Сам по собі OAuth відповідає на запитання «що цьому застосунку дозволено?». Застосунок отримує токен доступу, призначений для API, і сам читати його не повинен. OpenID Connect ставить ще й запитання «хто увійшов?» і відповідає на нього ID-токеном: JSON Web Token, адресованим самому застосунку, де сказано, хто ця людина, хто за це ручається, коли токен видано і для якого застосунку його призначено. ID-токени мають бути підписані. Специфікація дозволяє `alg: none` лише тоді, коли тип відповіді не повертає жодного ID-токена з ендпоінта авторизації і клієнт явно попросив `none` під час реєстрації.

Для цього завдання специфікація перейменовує ролі OAuth. Постачальник OpenID (OpenID Provider, OP) — це сервер авторизації OAuth, який автентифікує людей і видає ID-токени. Залежна сторона (Relying Party, RP) — клієнт OAuth, якому треба знати, хто ця людина. Саму людину названо кінцевим користувачем (End-User). Запит стає запитом OpenID Connect, коли в його scope є значення `openid`, і це значення обов’язкове: без нього, як прямо сказано в специфікації, поведінка зовсім не визначена.

Інші значення scope запитують дані профілю. `profile` охоплює ім’я, псевдонім, фото, мовні налаштування та подібні твердження (claims), `email` додає `email` і `email_verified`, а `address` і `phone` дають те, що й обіцяють їхні назви. Ці твердження застосунок може отримати з ендпоінта UserInfo в постачальника, звичайного API під захистом OAuth, який викликають із токеном доступу.

Працювати з ним можна трьома способами. Потік authorization code передає через браузер лише короткоживучий код, а токени отримує через HTTPS-запит до ендпоінта токенів. Неявний потік (implicit) повертає токени просто в перенаправленні, а гібридний поєднує обидва. Супровідна специфікація, OpenID Connect Discovery 1.0, дає застосунку змогу дізнатися всі ендпоінти й ключі підпису з однієї адреси.

Застосуйте поняття до рішення

Застосуйте «Що таке OpenID Connect?» 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

Головна причина — єдиний вхід (single sign-on). Коли перед багатьма застосунками стоїть один постачальник ідентичності, люди мають один набір облікових даних, застосунки їх не зберігають узагалі, а правила входу, від вимог до паролів до другого фактора, задаються в одному місці, у постачальника. Обліковий запис, заблокований у постачальника, вже не відкриє нового сеансу в застосунках, що на нього покладаються, тож офбординг зводиться до одного кроку замість цілого списку.

Стандарт закриває й діру, в яку потрапляють команди, коли будують вхід на голому OAuth. RFC 6749 попереджає, що в неявному потоці публічний клієнт не може визначити, якому клієнтові видано токен доступу. Клієнт, який вважає, що дійсний токен йому може дати лише власник ресурсу, відкритий для підміни особи: зловмисник із токеном, виданим іншому застосунку,

пред'являє його й сходить за людину, якій той токен належить. ID-токен у твердженні `aud` називає застосунок, для якого його видано, і застосунок це перевіряє.

Оскільки це стандарт, зроблене один раз переноситься далі. Специфікація обіцяє віддавати дані профілю в сумісний (interoperable) спосіб, тож код, що впускає людей через одного постачальника, впустить їх і через іншого. А з Discovery застосунок дізнається ендпоінти й ключі підпису постачальника з його URL видавця (issuer), а не з налаштувань, які хтось переписав уручну.

І нарешті, OpenID Connect дає кожній людині стабільний ідентифікатор. Специфікація прямо каже, що пара `iss` (видавець) і `sub` (ідентифікатор суб'єкта) — єдині твердження, на які застосунок може покластися як на стабільний ідентифікатор людини. Значення `sub` унікальне в межах видавця, ніколи не призначається повторно і має не більше 255 символів ASCII. Постачальник може навіть видавати кожному застосункові окремий `sub`, так званий попарний ідентифікатор, щоб застосунки не могли без дозволу зіставляти між собою дії людини.

Як це виглядає на практиці

Візьмімо вебзастосунок, зареєстрований у постачальника OpenID за адресою `https://server.example.com` з ідентифікатором клієнта `s6BhdRkqt3`; значення нижче взято з прикладів самої специфікації. Спершу застосунок читає конфігурацію постачальника за стандартною адресою під URL видавця:

```
GET /.well-known/openid-configuration HTTP/1.1
Host: server.example.com
```

Відповідь — документ JSON. У скороченому вигляді:

```
{
  "issuer": "https://server.example.com",
  "authorization_endpoint": "https://server.example.com/connect/authorize",
  "token_endpoint": "https://server.example.com/connect/token",
  "userinfo_endpoint": "https://server.example.com/connect/userinfo",
  "jwks_uri": "https://server.example.com/jwks.json",
```

```
  "scopes_supported": ["openid", "profile", "email", "address", "phone"],
  "id_token_signing_alg_values_supported": ["RS256", "ES256", "HS256"]
}
```

Discovery вимагає, щоб `issuer` у цьому документі був ідентичний URL видавця, який застосунок поставив перед стандартним шляхом; якщо вони різняться, застосунок зупиняється. Далі він відправляє браузер людини до ендпоінта авторизації з потоком code, значенням `openid` у scope і двома випадковими значеннями, які зберіг разом із сеансом браузера: `state` для перенаправлення і `nonce` для ID-токена. Публічний клієнт додав би ще й параметри PKCE, які RFC 9700 для нього робить обов'язковими; тут їх пропущено, щоб рядки не виходили задовгими.

```
GET /connect/authorize?response_type=code&client_id=s6BhdRkqt3
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
    &scope=openid%20profile%20email
    &state=af0ifjsldkj&nonce=n-0S6_WzA2Mj HTTP/1.1
Host: server.example.com
```

Постачальник виконує вхід людини й повертає браузер із кодом, а застосунок обмінює цей код на ендпоінті токенів так само, як у звичайному OAuth. Різниця — у відповіді: тепер поруч із токеном доступу в ній є `id_token` (тут скорочено):

```
{
  "access_token": "SlAV32hkKG",
  "token_type": "Bearer",
  "refresh_token": "8xLOxBtZp8",
  "expires_in": 3600,
  "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzc..."
}
```

ID-токен складається з трьох частин у base64url, розділених крапками. Перша декодується в `{"alg":"RS256","kid":"1e9gdk7"}`: це алгоритм підпису та ідентифікатор ключа в опублікованому наборі ключів постачальника. Середня частина містить твердження, і в прикладі специфікації вони такі:

```
{
  "iss": "https://server.example.com",
  "sub": "24400320",
```

```
  "aud": "s6BhdRkqt3",
  "nonce": "n-0S6_WzA2Mj",
  "exp": 1311281970,
  "iat": 1311280970,
  "auth_time": 1311280969,
  "acr": "urn:mace:incommon:iap:silver"
}
```

Перш ніж повірити бодай слову, застосунок перевіряє токен, і специфікація перелічує кроки. `iss` має точно збігатися з ідентифікатором видавця. `aud` має містити власний ідентифікатор клієнта застосунку і не містити аудиторій, яким він не довіряє. Підпис має перевірятися ключами постачальника з `jwks_uri` за алгоритмом, указаним у заголовку; якщо токен, як тут, прийшов просто з ендпоінта токенів через TLS, перевірка TLS-сертифіката сервера може замінити перевірку підпису. Поточний час має бути раніше за `exp`, а `nonce` має дорівнювати значенню, яке застосунок надіслав. Лише тоді він виконує вхід людини з ключем `https://server.example.com` плюс `24400320` і, якщо потрібно, забирає токеном доступу адресу пошти й дані профілю з ендпоінта UserInfo.

Один постачальник для багатьох систем — зміна не лише технічна, а й організаційна. В одному з проєктів на цьому сайті, керування доступом у M365 і SSO, облікові записи й доступ консолідували в M365, 1Password, Red Hat SSO та OKTA SSO. Запровадили принцип найменших привілеїв, а онбординг і офбординг стандартизували, тож доступ відкликали вчасно й послідовно, а не залишали після того, як людина пішла.

Де виникають проблеми

Адреса пошти слугує ключем облікового запису. Вибір очевидний і хибний. Специфікація дозволяє видавцеві з часом віддати ту саму адресу іншій людині й забороняє використовувати `email`, `phone_number`, `preferred_username` і `name` як унікальні ідентифікатори. Застосунок, що шукає людей за поштою, може передати обліковий запис однієї людини наступному власникові адреси. Зберігайте як ключ `iss` і `sub` разом, пошту тримайте звичайним атрибутом, а перш ніж на адресу в чомусь покладатися, перевірте, що `email_verified` дорівнює true.

Токен доступу вважають доказом того, хто увійшов. Деякі застосунки отримують токен доступу, викликають із ним API і вважають відповідь входом. Але токен могли видати зовсім іншому застосункові, і це та сама

підміна особи, від якої застерігає RFC 6749. Виконуйте вхід за ID-токеном і лише після того, як переконалися, що його `aud` містить ваш власний ідентифікатор клієнта.

ID-токен декодують, але не перевіряють. Щоб прочитати твердження з токена, досить декодувати base64url, ключ для цього взагалі не потрібен, і саме в цій легкості пастка. Токен, у якого не перевірено підпис, `iss`, `aud` чи `exp`, може виявитися підробленим, чужим або простроченим. Візьміть бібліотеку, яку активно підтримують, задайте їй видавця й ідентифікатор клієнта і дайте їй пройти всі кроки зі списку специфікації, а не лише зручні.

Nonce відсутній, сталий або його не порівнюють. Без нього ID-токен, перехоплений під час одного входу, можна підсунути в інший. RFC 9700 додає, що nonce має стосуватися одного конкретного входу й бути прив'язаним до браузера, який його почав. Специфікація пропонує один спосіб це зробити: зберегти криптографічно випадкове значення в сеансовому cookie з атрибутом HttpOnly, надіслати його хеш як `nonce` і звірити одне з одним, коли ID-токен повернеться.

Ключі підпису кешують назавжди. Постачальники змінюють ключі, і застосунок, що тримається за вчорашній набір ключів, відхилятиме кожен токен, підписаний новим. Специфікація очікує, що постачальник публікує ключі за адресою `jwks_uri` і позначає кожен токен ідентифікатором `kid`, а застосунок завантажує набір ключів заново, коли натрапляє на незнайомий `kid`. Кешуйте ключі, але сприймайте невідомий `kid` як сигнал оновити їх, а не як привід відмовити у вході.

Два тенанти на одному хості вважають одним видавцем. Постачальник може тримати кілька видавців на одному хості, і шлях є частиною ідентифікатора видавця: суб'єкт `1234` у `https://example.com` і суб'єкт `1234` у `https://example.com/sales` — це різні люди. Зберігайте рядок видавця повністю, порівнюйте його символ у символ і ніколи не обрізайте до імені хоста.

Що почитати далі

- OpenID Connect Core 1.0 — ID-токен, три потоки, стандартні твердження й кроки перевірки, у редакції з другим набором виправлень від грудня 2023 року.

- OpenID Connect Discovery 1.0 — документ конфігурації за стандартною адресою і правило, за яким його `issuer` має бути ідентичний URL видавця, під яким документ отримано.
- RFC 6749: The OAuth 2.0 Authorization Framework — фреймворк, на якому побудовано OpenID Connect, із попередженням у розділі 10.16 про те, чим небезпечна ідентифікація людей за токенами доступу.
- RFC 7519: JSON Web Token (JWT) — формат, у якому випускають кожен ID-токен.
- RFC 9700: Best Current Practice for OAuth 2.0 Security — чинні рекомендації щодо PKCE, nonce і перенаправлень, які стосуються OpenID Connect так само, як і OAuth.

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 OpenID Connect?» 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/openid-connect/