Engineer

OpenID Connect

Eine Identitätsschicht auf OAuth zur Benutzeranmeldung

Was es ist

OpenID Connect ist ein standardisiertes Verfahren, mit dem eine Anwendung von einem Dienst erfährt, wer eine gerade dort angemeldete Person ist. Die Spezifikation nennt es eine einfache Identitätsschicht auf OAuth 2.0. Genau so funktioniert es: dieselben Weiterleitungen, Endpunkte und Token wie bei OAuth, ergänzt um ein weiteres Token, einen weiteren Endpunkt und die Regeln für deren Prüfung. Meist wird es OIDC abgekürzt. Die Kernspezifikation wurde im Februar 2014 verabschiedet. Der aktuelle Text enthält den zweiten Korrektursatz, ist auf den 15. Dezember 2023 datiert und wird von einer Arbeitsgruppe der OpenID Foundation gepflegt.

OAuth allein beantwortet die Frage „Was darf diese Anwendung tun?“. Es übergibt der Anwendung ein für eine API bestimmtes Zugriffstoken, das die Anwendung nicht selbst auslesen soll. OpenID Connect ergänzt die Frage „Wer hat sich angemeldet?“ und beantwortet sie mit einem ID-Token: einem an die Anwendung selbst gerichteten JSON Web Token. Es gibt an, wer die Person ist, wer dafür einsteht, wann das Token ausgestellt wurde und für welche Anwendung es bestimmt ist. ID-Token müssen signiert sein. Als Ausnahme erlaubt die Spezifikation `alg: none` nur dann, wenn der Antworttyp kein ID-Token vom Autorisierungsendpunkt zurückgibt und der Client bei der Registrierung ausdrücklich `none` angefordert hat.

Die Spezifikation benennt die OAuth-Rollen für diese Aufgabe um. Der OpenID Provider (OP) ist ein OAuth-Autorisierungsserver, der Personen authentifiziert und ID-Token ausstellt. Die Relying Party (RP) ist der OAuth-Client, der wissen muss, wer die Person ist. Die Person heißt End-User. Eine Anfrage wird zur OpenID-Connect-Anfrage, indem sie `openid` in ihrem Scope angibt. Dieser Wert ist verpflichtend: „Fehlt der Scope-Wert openid, ist das Verhalten vollständig unspezifiziert.“

Weitere Scope-Werte fordern Profildaten an. `profile` umfasst Name, Spitzname, Bild, Gebietsschema und ähnliche Claims. `email` ergänzt `email` und `email_verified`, und `address` und `phone` tun, was ihre Namen sagen. Die Anwendung kann diese Claims vom UserInfo-Endpunkt des Anbieters abrufen. Das ist eine gewöhnliche OAuth-geschützte API, die mit dem Zugriffstoken aufgerufen wird.

Es gibt drei Abläufe. Der Authorization Code Flow schickt nur einen kurzlebigen Code durch den Browser und ruft die Token mit einer HTTPS-Anfrage am Token-Endpunkt ab. Der Implicit Flow liefert Token direkt in der Weiterleitung zurück; der Hybrid Flow kombiniert beide. Eine ergänzende Spezifikation, OpenID Connect Discovery 1.0, ermöglicht es der Anwendung, über eine einzige Adresse sämtliche Endpunkte und Signierschlüssel zu ermitteln.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist OpenID Connect?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Konten, Passwörter und Berechtigungen werden so geordnet, dass alle genau den beruflich erforderlichen Zugriff haben und ihn am Tag ihres Ausscheidens verlieren.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 1.000 RBAC-Regeln für sichere Linux-Server — 1,000 Zugriffsregeln zum Schließen einer Sicherheitslücke

1.000 RBAC-Regeln für sichere Linux-Server Kontakt aufnehmen →

Warum es wichtig ist

Der Grund für die Einführung ist Single Sign-on. Mit einem Identitätsanbieter vor vielen Anwendungen behält jede Person einen einzigen Satz Zugangsdaten, während die Anwendungen selbst keine speichern. Die Regeln für die Anmeldung, von Passwortvorgaben bis zum zweiten Faktor, werden einmal beim Anbieter festgelegt. Wird ein Konto dort deaktiviert, kann es in keiner darauf vertrauenden Anwendung eine neue Sitzung beginnen. Dadurch wird Offboarding zu einem einzelnen Schritt statt zu einer Liste.

Es schließt auch eine Lücke, in die Teams geraten, wenn sie eine Anmeldung allein auf OAuth aufbauen. RFC 6749 warnt davor, dass ein öffentlicher Client beim Implicit Flow nicht feststellen kann, für welchen Client ein Zugriffstoken ausgestellt wurde. Nimmt er an, nur der Ressourcenbesitzer könne ihm ein gültiges Token vorlegen, ist eine Identitätsvortäuschung möglich: Ein Angreifer

kann ein für eine andere Anwendung ausgestelltes Token vorlegen und als die dahinterstehende Person behandelt werden. Ein ID-Token nennt seine Zielanwendung im Claim `aud`, und die Anwendung prüft diesen Wert.

Weil es ein Standard ist, lässt sich die Arbeit übertragen. Die Spezifikation verspricht interoperabel bereitgestellte Profilinformationen. Code, der Personen über einen Anbieter anmeldet, kann sie deshalb auch über einen anderen anmelden. Mit Discovery ermittelt die Anwendung die Endpunkte und Signierschlüssel des Anbieters anhand seiner Issuer-URL statt aus von Hand kopierten Einstellungen.

Schließlich gibt OpenID Connect jeder Person eine stabile Kennung. Die Spezifikation sagt ausdrücklich, dass allein das Paar aus `iss` (dem Aussteller des Tokens) und `sub` (der Kennung des Subjekts) als Claim-Paar eine verlässliche stabile Personenkennung bildet. Der Wert von `sub` ist innerhalb des Ausstellers eindeutig, wird niemals neu vergeben und umfasst höchstens 255 ASCII-Zeichen. Ein Anbieter kann sogar jeder Anwendung eine andere `sub` für dieselbe Person ausstellen, eine paarweise Kennung. Dadurch können Anwendungen die Aktivitäten einer Person nicht ohne Erlaubnis miteinander verknüpfen.

Wie es in der Praxis aussieht

Nehmen wir eine Webanwendung, die bei einem OpenID Provider unter `https://server.example.com` mit der Client-ID `s6BhdRkqt3` registriert ist. Die folgenden Werte stammen aus den Beispielen der Spezifikation. Zunächst liest die Anwendung die Konfiguration des Anbieters an der vorgesehenen Well-known-Adresse unterhalb der Issuer-URL:

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

Die Antwort ist ein JSON-Dokument. Eine gekürzte Fassung:

```
{
  "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 verlangt, dass der `issuer` in diesem Dokument mit der Issuer-URL identisch ist, die die Anwendung dem Well-known-Pfad vorangestellt hat. Bei einer Abweichung bricht die Anwendung ab. Danach schickt sie den Browser der Person mit dem Code Flow, dem Scope `openid` und zwei Zufallswerten zum Autorisierungsendpunkt. Diese Werte hat sie der Browsersitzung zugeordnet gespeichert: `state` für die Weiterleitung und `nonce` für das ID-Token. Ein öffentlicher Client würde außerdem die PKCE-Parameter hinzufügen, die RFC 9700 für ihn verpflichtend macht. Sie fehlen hier nur, um die Zeilen kurz zu halten.

```
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
```

Der Anbieter meldet die Person an und schickt den Browser mit einem Code zurück. Die Anwendung löst den Code genau wie bei reinem OAuth am Token-Endpunkt ein. Der Unterschied liegt in der Antwort: Sie enthält jetzt neben dem Zugriffstoken ein `id_token` (hier gekürzt):

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

Das ID-Token besteht aus drei durch Punkte getrennten base64url-Teilen. Der erste wird zu `{"alg":"RS256","kid":"1e9gdk7"}` dekodiert: dem Signaturalgorithmus und der Kennung des Schlüssels im veröffentlichten Schlüsselsatz des Anbieters. Der mittlere Teil enthält die Claims. Das Beispiel der Spezifikation lautet:

```
{
  "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"
}
```

Bevor die Anwendung irgendeiner Angabe daraus vertraut, validiert sie das Token nach den in der Spezifikation aufgeführten Schritten. `iss` muss exakt dem Aussteller des Anbieters entsprechen. `aud` muss die eigene Client-ID der Anwendung enthalten und darf keine Zielgruppe nennen, der sie nicht vertraut. Die Signatur muss mit den Schlüsseln des Anbieters aus `jwks_uri` und dem Algorithmus im Header erfolgreich geprüft werden. Kam das Token wie hier unmittelbar über TLS vom Token-Endpunkt, darf die Prüfung des TLS-Zertifikats des Servers die Signaturprüfung ersetzen. Die aktuelle Zeit muss vor `exp` liegen. `nonce` muss dem von der Anwendung gesendeten Wert entsprechen. Erst dann meldet sie die Person unter dem Schlüssel aus `https://server.example.com` und `24400320` an und verwendet bei Bedarf das Zugriffstoken, um E-Mail- und Profil-Claims vom UserInfo-Endpunkt abzurufen.

Ein einziger Anbieter für viele Systeme ist ebenso eine organisatorische wie eine technische Veränderung. Eines der Projekte auf dieser Website, die Zugriffsverwaltung über M365 und SSO hinweg, vereinheitlichte Identitäten und Zugriffsrechte in M365, 1Password, Red Hat SSO und OKTA SSO, setzte das Prinzip der geringsten Rechte durch und standardisierte Onboarding und Offboarding. So wurden Zugriffsrechte zeitnah entzogen, wenn jemand ausschied.

Wo es schiefgeht

Die E-Mail-Adresse dient als Kontoschlüssel. Sie ist die naheliegende und die falsche Wahl. Laut Spezifikation darf ein Anbieter eine E-Mail-Adresse später einer anderen Person zuweisen. `email`, `phone_number`, `preferred_username` und `name` dürfen nicht als eindeutige Kennungen verwendet werden. Eine Anwendung, die Personen anhand ihrer E-Mail-Adresse nachschlägt, kann das Konto einer Person an den nächsten Besitzer dieser Adresse übergeben.

Speichern Sie `iss` und `sub` gemeinsam als Schlüssel und die E-Mail-Adresse als Attribut. Prüfen Sie außerdem, ob `email_verified` wahr ist, bevor Sie sich für irgendeinen Zweck auf die Adresse verlassen.

Ein Zugriffstoken gilt als Nachweis dafür, wer sich angemeldet hat. Manche Anwendungen erhalten ein Zugriffstoken, rufen damit eine API auf und werten die Antwort als Anmeldung. Das Token kann jedoch für eine ganz andere Anwendung ausgestellt worden sein. Das ist die Identitätsvortäuschung, vor der RFC 6749 warnt. Melden Sie Personen anhand des ID-Tokens an, und zwar erst, nachdem Sie bestätigt haben, dass dessen `aud` die eigene Client-ID nennt.

Das ID-Token wird dekodiert, aber nie geprüft. Um seine Claims zu lesen, genügt eine base64url-Dekodierung ganz ohne Schlüssel. Gerade diese Einfachheit ist die Falle. Ein Token, dessen Signatur, `iss`, `aud` oder `exp` ungeprüft bleibt, kann gefälscht, für eine andere Anwendung bestimmt oder abgelaufen sein. Verwenden Sie eine gepflegte Bibliothek, geben Sie ihr Aussteller und Client-ID und lassen Sie sie jede Prüfung der Spezifikation ausführen, statt nur die bequemen auszuwählen.

Die Nonce fehlt, ist fest vorgegeben oder wird nie verglichen. Ohne sie lässt sich ein bei einer Anmeldung abgefangenes ID-Token bei einer anderen wiederverwenden. RFC 9700 ergänzt, dass die Nonce für genau eine Anmeldung gelten und an den Browser gebunden sein muss, der sie begonnen hat. Die Spezifikation schlägt dafür ein Verfahren vor: Einen kryptografisch zufälligen Wert in einem HttpOnly-Sitzungscookie speichern, seinen Hash als `nonce` senden und beide beim Eintreffen des ID-Tokens miteinander vergleichen.

Signierschlüssel bleiben auf Dauer im Cache. Anbieter wechseln ihre Schlüssel. Eine Anwendung mit dem Schlüsselsatz von gestern lehnt dann jedes mit dem neuen Schlüssel signierte Token ab. Die Spezifikation erwartet, dass der Anbieter die Schlüssel unter `jwks_uri` veröffentlicht und jedes Token mit einer `kid` kennzeichnet. Die Anwendung soll den Schlüsselsatz erneut abrufen, sobald ihr eine unbekannte `kid` begegnet. Halten Sie Schlüssel im Cache, aber behandeln Sie eine unbekannte `kid` als Aufforderung zum Aktualisieren statt als Grund, die Anmeldung scheitern zu lassen.

Zwei Mandanten auf einem Host werden als ein Aussteller behandelt. Ein Anbieter kann auf einem Host mehrere Aussteller betreiben. Der Pfad ist Teil der Ausstellerkennung: Das Subjekt `1234` unter `https://example.com` und das Subjekt

`1234` unter `https://example.com/sales` sind unterschiedliche Personen. Speichern Sie die vollständige Ausstellerzeichenfolge, vergleichen Sie sie exakt und kürzen Sie sie niemals auf einen Hostnamen.

Weiterführende Lektüre

- OpenID Connect Core 1.0 — das ID-Token, die drei Abläufe, die standardisierten Claims und die Validierungsschritte in der Fassung mit Korrektursatz 2 vom Dezember 2023.
- OpenID Connect Discovery 1.0 — das Well-known-Konfigurationsdokument und die Regel, dass dessen Aussteller mit der Issuer-URL übereinstimmen muss, unter der es abgerufen wurde.
- RFC 6749: The OAuth 2.0 Authorization Framework — das Framework unter OpenID Connect, mit der Warnung in Abschnitt 10.16 vor der Verwendung von Zugriffstoken zur Identifizierung von Personen.
- RFC 7519: JSON Web Token (JWT) — das Tokenformat, das jedes ID-Token verwendet.
- RFC 9700: Best Current Practice for OAuth 2.0 Security — aktuelle Empfehlungen zu PKCE, Nonce und Weiterleitung, die für OpenID Connect genauso gelten wie für OAuth.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist OpenID Connect?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/openid-connect/