Engineer

OpenID Connect

et identitetslag oven på OAuth til at logge brugere ind

Hvad det er

OpenID Connect er en standard, der lader et program få at vide, hvem en person er, fra en tjeneste, der lige har logget personen ind. Specifikationen kalder sig selv et enkelt identitetslag oven på OAuth 2.0, og sådan er den også bygget: de samme omdirigeringer, endpoints og tokens som i OAuth, plus ét token mere, ét endpoint mere og reglerne for, hvordan de kontrolleres. I daglig tale hedder den ofte bare OIDC. Kernespecifikationen blev endelig i februar 2014. Den gældende tekst, hvor anden runde rettelser er indarbejdet, er dateret 15. december 2023 og vedligeholdes af en arbejdsgruppe under OpenID Foundation.

OAuth alene svarer på spørgsmålet »hvad må dette program?«. Programmet får et adgangstoken, der er beregnet til et API, og det er ikke meningen, at programmet selv læser det. OpenID Connect stiller også spørgsmålet »hvem loggede ind?« og besvarer det med et ID-token: et JSON Web Token, stilet til programmet selv, som fortæller, hvem personen er, hvem der står inde for det, hvornår tokenet blev udstedt, og hvilket program det er tiltænkt. ID-tokens skal signeres. Specifikationen tillader kun `alg: none`, hvis svartypen ikke returnerer noget ID-token fra autorisationsendpointet, og klienten udtrykkeligt bad om `none` ved registreringen.

Til den opgave giver specifikationen OAuth-rollerne nye navne. En OpenID Provider (OP) er en autorisationsserver, der autentificerer personer og udsteder ID-tokens. En Relying Party (RP) er den OAuth-klient, der har brug for at vide, hvem personen er, og personen selv hedder End-User. En forespørgsel bliver til en OpenID Connect-forespørgsel ved at have `openid` med i sit scope, og den værdi er obligatorisk: mangler den, er adfærden efter specifikationens egne ord fuldstændig uspecificeret.

Andre scope-værdier beder om profildata. `profile` dækker navn, kaldenavn, billede, sprogindstilling og lignende claims, `email` giver `email` og `email_verified`, og `address` og `phone` gør, hvad navnene siger. Programmet kan hente de claims fra udbyderens UserInfo-endpoint, et helt almindeligt OAuth-beskyttet API, som det kalder med adgangstokenet.

Der er tre måder at køre det på. Authorization code-flowet sender kun en kortlivet kode gennem browseren og henter selve tokens med en HTTPS-forespørgsel til token-endpointet. Implicit-flowet leverer tokens lige i omdirigeringen, og hybrid-flowet blander de to. En ledsagende specifikation, OpenID Connect Discovery 1.0, lader programmet slå alle endpoints og signeringsnøgler op ud fra én adresse.

Brug begrebet i en beslutning

Brug »Hvad er OpenID Connect?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Konti, adgangskoder og rettigheder sat op, så folk har præcis den adgang, jobbet kræver, og fratrådte mister den samme dag.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har skrevet 1.000 RBAC-regler til Linux-servere — 1,000 adgangsregler skrevet for at lukke et sikkerhedshul

Har skrevet 1.000 RBAC-regler til Linux-servere Kontakt os →

Hvorfor det betyder noget

Grunden til at tage det i brug er single sign-on. Med én identitetsudbyder foran mange programmer har folk ét sæt loginoplysninger, programmerne gemmer ingen, og reglerne for, hvordan man logger ind, fra krav til adgangskoder til en ekstra faktor, fastlægges ét sted, hos udbyderen. Spærrer man en konto hos udbyderen, kan den ikke starte en ny session i nogen af de programmer, der læner sig op ad den, og så er offboarding ét trin og ikke en tjekliste.

Standarden lukker også et hul, som teams falder i, når de bygger login på ren OAuth. RFC 6749 advarer om, at en offentlig klient i implicit-flowet ikke kan se, hvilken klient et adgangstoken blev udstedt til. En klient, der går ud fra, at kun ressourceejeren kan give den et gyldigt token, kan derfor narres: En angriber med et token, der er udstedt til et andet program, afleverer det og bliver taget for personen bag. Et ID-token angiver i sit `aud`-claim, hvilket program det er til, og programmet kontrollerer det.

Fordi det er en standard, kan arbejdet genbruges. Specifikationen lover profiloplysninger på en interoperabel måde, så den kode, der logger folk ind via én udbyder, også kan logge dem ind via en anden. Med Discovery lærer programmet udbyderens endpoints og signeringsnøgler at kende ud fra dens issuer-URL i stedet for fra indstillinger, som nogen har skrevet af i hånden.

Endelig giver OpenID Connect hver person en stabil identifikator. Specifikationen siger klart, at parret `iss` (udstederen) og `sub` (subjektets identifikator) er de eneste claims, et program kan regne med som stabil identifikator for en person. `sub` er entydig hos udstederen, bliver aldrig genbrugt og er højst 255 ASCII-tegn lang. En udbyder kan endda give hvert program sin egen `sub`, en såkaldt parvis identifikator, så programmerne ikke kan koble en persons aktiviteter sammen uden lov.

Sådan ser det ud i praksis

Tag en webapplikation, der er registreret hos en OpenID Provider på `https://server.example.com` med klient-ID'et `s6BhdRkqt3`; værdierne nedenfor stammer fra specifikationens egne eksempler. Programmet begynder med at hente udbyderens konfiguration fra den faste adresse under udstederen:

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

Svaret er et JSON-dokument. I forkortet form:

```
{
  "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 kræver, at `issuer` i dokumentet er identisk med den issuer-URL, programmet satte foran den faste sti, og programmet stopper, hvis de to afviger. Derefter sender det personens browser til autorisationsendpointet med code-

flowet, scopet `openid` og to tilfældige værdier, som det har gemt sammen med browsersessionen: `state` til omdirigeringen og `nonce` til ID-tokenet. En offentlig klient ville også sende PKCE-parametrene, som RFC 9700 gør obligatoriske for den; de er udeladt her for at holde linjerne korte.

```
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
```

Udbyderen logger personen ind og sender browseren tilbage med en kode, som programmet indløser ved token-endpointet nøjagtig som i almindelig OAuth. Forskellen ligger i svaret, der nu har et `id_token` ved siden af adgangstokenet (forkortet her):

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

ID-tokenet består af tre base64url-dele, adskilt af punktummer. Den første afkodes til `{"alg":"RS256","kid":"1e9gdk7"}` : signeringsalgoritmen og ID'et på nøglen i udbyderens offentliggjorte nøglesæt. Den midterste del rummer selve claims, og i specifikationens eksempel ser de sådan ud:

```
{
  "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"
}
```

Før programmet stoler på et eneste ord af det, validerer det tokenet, og specifikationen opregner trinnene. `iss` skal stemme nøjagtigt overens med udbyderens issuer. `aud` skal indeholde programmets eget klient-ID og ingen målgruppe, programmet ikke stoler på. Signaturen skal kunne verificeres med udbyderens nøgler fra `jwks_uri` og med algoritmen i headeren; kom tokenet, som her, direkte fra token-endpointet over TLS, må en kontrol af serverens TLS-certifikat træde i stedet for signaturkontrollen. Klokken skal være før `exp`, og `nonce` skal være den værdi, programmet selv sendte. Først da logger det personen ind, med `https://server.example.com` plus `24400320` som nøgle, og bruger adgangstokenet til at hente e-mail og profiloplysninger fra UserInfo-endpointet, hvis det har brug for dem.

Én udbyder til mange systemer er lige så meget en organisatorisk ændring som en teknisk. I et af projekterne her på sitet, adgangsstyring med M365 og SSO, blev identiteter og adgange samlet i M365, 1Password, Red Hat SSO og OKTA SSO. Least privilege blev håndhævet, og on- og offboarding blev standardiseret, så adgangen blev frataget hurtigt og ensartet i stedet for at blive hængende, efter at nogen var gået videre.

Hvor det går galt

E-mailadressen bruges som kontonøgle. Det er det oplagte valg og det forkerte. Specifikationen siger, at en udsteder med tiden må give den samme e-mailadresse til en anden person, og at `email`, `phone_number`, `preferred_username` og `name` ikke må bruges som entydige identifikatorer. Et program, der slår folk op på e-mail, kan give én persons konto til den næste, der får adressen. Gem `iss` og `sub` sammen som nøgle, behold e-mailen som en almindelig egenskab, og kontrollér, at `email_verified` er sand, før adressen bruges til noget som helst.

Et adgangstoken tages som bevis på, hvem der loggede ind. Nogle programmer modtager et adgangstoken, kalder et API med det og behandler svaret som et login. Tokenet kan være udstedt til et helt andet program, og så er det netop det angreb, hvor nogen udgiver sig for en anden, som RFC 6749 advarer imod. Log folk ind på ID-tokenet, og først når I har bekræftet, at dets `aud` indeholder jeres eget klient-ID.

ID-tokenet afkodes, men kontrolleres aldrig. At læse claims ud af et token kræver en base64url-afkodning og ingen nøgle overhovedet, og netop det gør det så fristende. Et token, hvis signatur, `iss`, `aud` eller `exp` ikke bliver

kontrolleret, kan være forfalsket, høre til et andet program eller være udløbet. Brug et vedligeholdt bibliotek, giv det udstederen og klient-ID'et, og lad det køre alle trinnene i specifikationens liste, ikke kun de bekvemme.

Nonce mangler, er fast eller sammenlignes aldrig. Uden den kan et ID-token, der er opsnappet fra ét login, genbruges i et andet. RFC 9700 tilføjer, at nonce skal høre til ét bestemt login og være bundet til den browser, der startede det. Specifikationen foreslår én måde at gøre det på: Gem en kryptografisk tilfældig værdi i en HttpOnly-sessionscookie, send dens hash som `nonce`, og sammenlign de to, når ID-tokenet kommer tilbage.

Signeringsnøglerne caches for evigt. Udbydere skifter nøgler, og et program, der holder fast i gårsdagens nøglesæt, afviser alle tokens, som er signeret med den nye. Specifikationen forventer, at udbyderen offentliggør nøglerne på `jwks_uri` og mærker hvert token med et `kid`, og at programmet henter nøglesættet igen, når det møder et `kid`, det ikke kender. Cache gerne nøglerne, men lad et ukendt `kid` være et signal til at hente dem på ny og ikke en grund til at afvise login.

To tenants på samme vært behandles som én udsteder. En udbyder kan køre flere udstedere på samme vært, og stien er en del af udstederens identifikator: Subjektet `1234` hos `https://example.com` og subjektet `1234` hos `https://example.com/sales` er to forskellige personer. Gem hele issuer-strengen, sammenlign den tegn for tegn, og skær den aldrig ned til et værtsnavn.

Læs mere

- OpenID Connect Core 1.0 — ID-tokenet, de tre flows, standardclaims og valideringstrinnene, i udgaven med rettelsessæt 2 fra december 2023.
- OpenID Connect Discovery 1.0 — konfigurationsdokumentet på den faste adresse og reglen om, at dets issuer skal være identisk med den issuer-URL, det blev hentet under.
- RFC 6749: The OAuth 2.0 Authorization Framework — rammen, som OpenID Connect bygger på, med advarslen i afsnit 10.16 mod at bruge adgangstokens til at identificere personer.
- RFC 7519: JSON Web Token (JWT) — tokenformatet, som alle ID-tokens bruger.
- RFC 9700: Best Current Practice for OAuth 2.0 Security — de gældende anbefalinger om PKCE, nonce og omdirigeringen, som gælder for OpenID Connect lige så vel som for OAuth.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring udrulningsforløbet, driftsrisiciene og dem, der har vagten. Vi kan afgrænse en ændring med kontroller, en vej tilbage og tydeligt ansvar.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er OpenID Connect?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/openid-connect/