CS-069 · API Security

OpenID Connect

An identity layer on top of OAuth 2.0 that proves who the user is, not just what an app may access.

Executive Summary

OpenID Connect, or OIDC, is an identity layer built on top of OAuth 2.0 that lets an application confirm who a user is. Where OAuth grants delegated access, OIDC adds a signed ID token that carries verified identity claims from a trusted identity provider. It is the foundation of most modern sign-in and single sign-on, and its security depends on validating the ID token and the flow correctly.

What It Is

OpenID Connect extends OAuth 2.0 with a standard way to authenticate users and share basic profile information. OAuth alone answers what an application is allowed to do; OIDC answers who the user is. When a user signs in through an identity provider, the application, called the relying party, receives an ID token in addition to any OAuth access token. The ID token is a signed set of claims about the authentication event and the user, such as the subject identifier, the issuer, the audience, and the time the user authenticated. OIDC also defines a standard profile endpoint for additional user information and standard scopes such as one for basic profile data. Because it is built on OAuth, it reuses the same flows, most importantly the authorization code flow, while adding identity on top.

Why It Matters

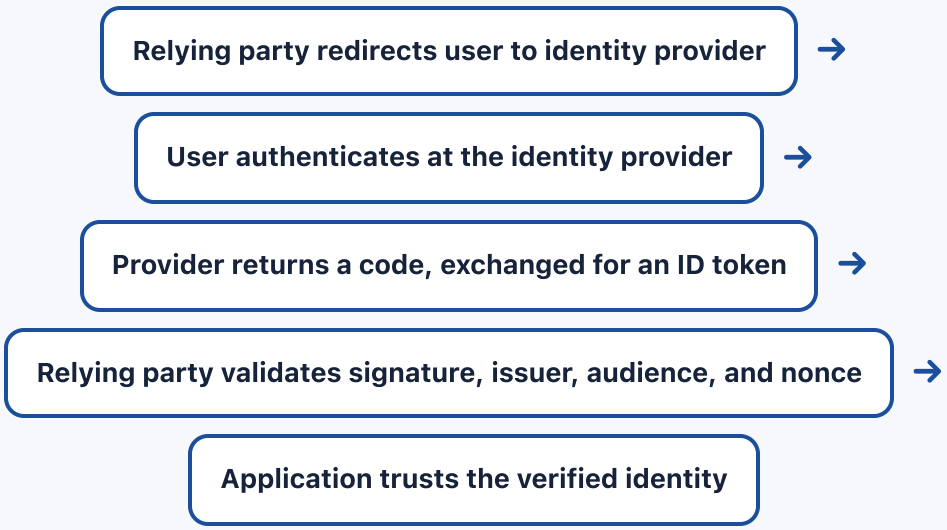
OpenID Connect powers sign-in-with buttons and enterprise single sign-on across a huge portion of the web, so it decides how billions of logins are trusted. When implemented correctly, it lets organizations centralize authentication, apply strong controls like multi-factor authentication in one place, and stop every application from storing its own passwords. When

implemented poorly, the consequences are severe: an application that fails to validate the ID token's signature, issuer, or audience can be tricked into accepting a token meant for another service or forged by an attacker, which leads to account takeover. For engineers and security professionals, OIDC is a high-value skill because it concentrates identity, which means a single validation flaw can compromise many users at once.

How It Works

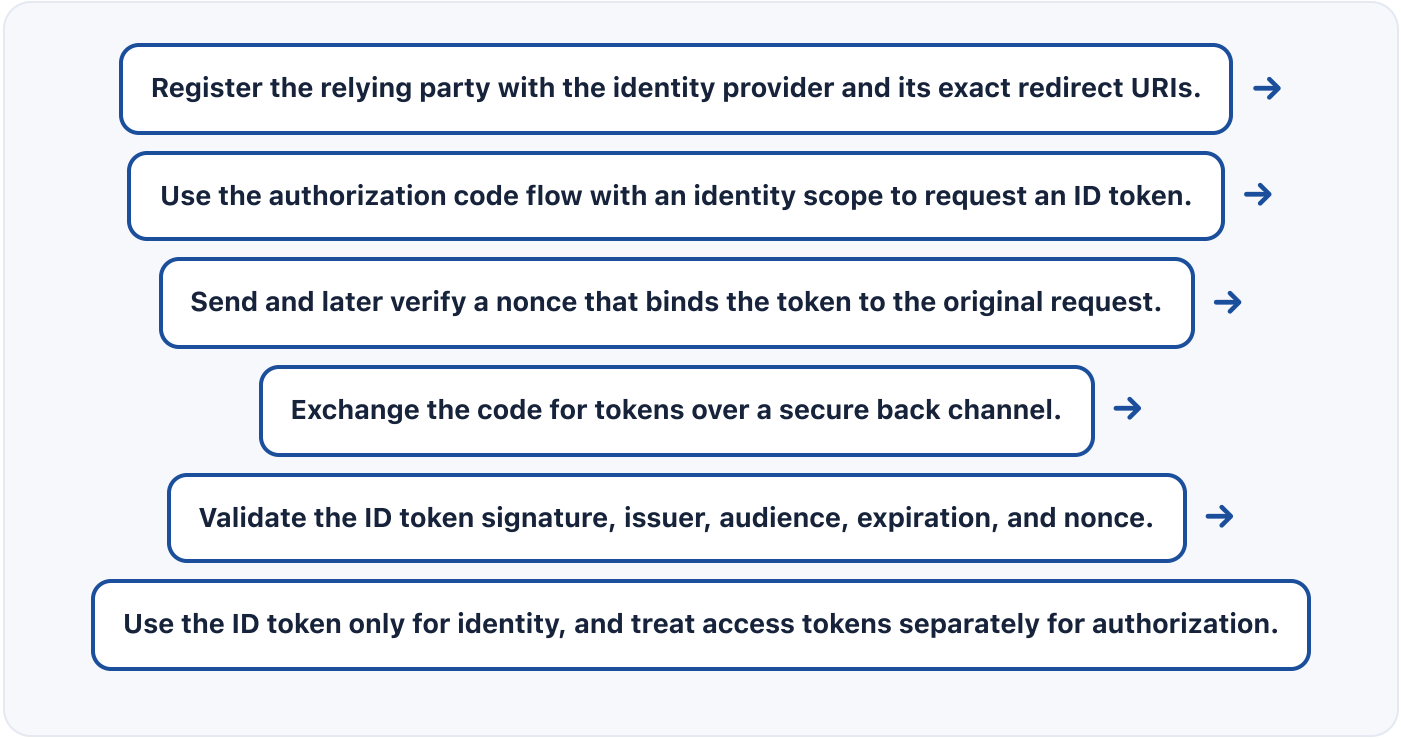
OIDC uses the OAuth authorization code flow with an identity scope. The relying party redirects the user to the identity provider, where the user authenticates. The identity provider returns an authorization code, which the relying party exchanges for tokens, including a signed ID token. The relying party must then validate that ID token: verify the signature against the identity provider's published keys, confirm the issuer matches the expected provider, confirm the audience matches this application, check that the token has not expired, and verify the nonce that ties the token to the original request. Only after these checks does the application trust the identity. The identity provider publishes its configuration and signing keys at well-known endpoints so relying parties can discover and verify them automatically. Access tokens from the same flow are for authorization; the ID token is specifically for authentication and should not be used as an access token.

Architecture Diagram



The relying party redirects the user to the identity provider, then validates the returned ID token before trusting the login.

Visual Workflow



Common Attacks

- Accepting an ID token without verifying its signature against the provider's keys
- Token substitution, where a token issued for one application is accepted by another because the audience is not checked
- Replay of a captured token when the nonce is missing or not validated
- Redirect URI manipulation that diverts the sign-in response to an attacker
- Using the ID token as an access token, exposing identity claims to resource servers

Common Mistakes

- Trusting an ID token without validating signature, issuer, audience, and expiration
- Skipping the nonce check, leaving the flow open to replay
- Confusing the ID token with the access token and using each for the wrong purpose
- Allowing loose redirect URIs instead of exact registered values
- Fetching signing keys insecurely instead of from the provider's published endpoint
- Building a custom identity layer on OAuth instead of using standard OIDC

Best Practices

- Use the authorization code flow and validate every ID token claim before trusting it
- Verify the signature using keys from the provider's published, well-known endpoint
- Confirm issuer and audience match, check expiration, and validate the nonce
- Register exact redirect URIs and reject mismatches
- Keep the ID token for authentication and use access tokens for authorization
- Centralize multi-factor authentication and session policy at the identity provider

Quick Checklist

- [x] Authorization code flow with an identity scope in use
- [x] ID token signature verified against the provider's published keys
- [x] Issuer, audience, and expiration validated on every login
- [x] Nonce sent and verified to prevent replay
- [x] Redirect URIs registered exactly and matched strictly
- [x] ID token and access token used only for their intended purposes

Recommended Tools

Identity provider
Authenticates users and issues signed ID tokens and access tokens

OIDC client library
Implements discovery, token exchange, and ID token validation correctly

Discovery and key endpoints
Publishes provider configuration and signing keys for automatic verification

Single sign-on platform
Centralizes federated login and session policy across applications

Industry Standards

OpenID Connect Core
Defines the identity layer, ID token, and

OAuth 2.0 Authorization Framework (RFC 6749)

validation rules on top of OAuth 2.0

The authorization framework OIDC extends

JSON Web Token (RFC 7519)
The token format used for the ID token

Career Relevance

OpenID Connect is central for backend engineers, application security engineers, and identity-focused API security engineers who implement and secure login and single sign-on. Security engineers and penetration testers assess OIDC deployments for token validation flaws, and identity and AI governance professionals depend on OIDC concepts because verified identity is the foundation of access control across applications and AI systems.

Interview Questions

- How does OpenID Connect differ from OAuth 2.0?
- What is an ID token, and which claims must you validate before trusting it?
- Why should you not use an ID token as an access token, or the reverse?
- What is the purpose of the nonce in an OIDC flow?
- How does a relying party obtain and verify the identity provider's signing keys?

Related Certifications

OWASP-aligned application security training certifications (vendor-neutral concepts)

Identity and access management vendor

CompTIA Security+ (foundational)

Further Reading

- OpenID Connect (official site)
- OpenID Connect Core specification
- RFC 6749: OAuth 2.0 Authorization Framework

Key Takeaways

- OpenID Connect adds verified identity on top of OAuth 2.0's delegated authorization.
- The signed ID token carries who the user is and must be fully validated before trust.
- Validate signature, issuer, audience, expiration, and nonce on every login.
- Keep the ID token for authentication and access tokens for authorization.
- Centralizing identity means one validation flaw can affect many users, so get it right.

FAQ

What is the difference between OpenID Connect and OAuth 2.0?

What is an ID token and how is it different from an access token?

Why is the nonce important in OIDC?

Related Careers

RELATED ROLES

Application Security Engineer

Api Security Engineer

Backend Engineer

Security Engineer

RELATED CERTIFICATIONS

OWASP-aligned application security training

Identity and access management vendor certifications (vendor-neutral concepts)

CompTIA Security+ (foundational)

CURRENT OPENINGS

Live roles are on the job board.

Browse all jobs

GUIDES & SALARY

Career guides

Role roadmaps

Salary data

Career tools

SUGGESTED LEARNING PATH

1. Ground the basics with CS-001 Cybersecurity
2. Study this sheet: **OpenID Connect**
3. Go deeper: OAuth 2.0
4. Go deeper: JSON Web Tokens (JWT)
5. Validate it: work toward **OWASP-aligned application security training**
6. Find the role: browse current openings

RELATED SHEETS

CS-068 OAuth 2.0 | CS-067 JSON Web Tokens (JWT) | CS-066 REST API Security | CS-062 Authentication | CS-063 Authorization