Oauth

OAuth 2.0 Study Guide

What is OAuth?

OAuth 2.0 is:

an authorization framework

used to securely allow:

- applications
- APIs
- users
- systems

to access resources WITHOUT sharing passwords directly.

Simple Explanation

Instead of giving your password to every application:

```
Application → requests authorization
```

OAuth provides:

temporary access tokens

for secure access.

Real-World Example

Example:

```
You log into an app using Google or Microsoft
```

The app:

- never sees your password
- receives a secure token instead

That process often uses OAuth.

OAuth Main Purpose

OAuth solves:

- secure API authentication
- delegated access
- token-based security
- controlled permissions

Simple OAuth Flow

```
User logs in
      ↓
OAuth server validates identity
      ↓
```

```
Access token issued
      ↓
Application uses token for API calls
```

Important OAuth Components

Component	Purpose
User	Person authenticating
Client Application	App requesting access
Authorization Server	Validates identity and issues tokens
Resource Server	API/backend service
Access Token	Temporary credential used for API access
Refresh Token	Used to obtain new access token

OAuth Tokens

Access Token

Temporary credential used in API requests.

Example:

```
Authorization: Bearer eyJhbGc...
```

Usually:

- short-lived
- expires after some time

Refresh Token

Used to:

request new access token

without forcing user to log in again.

Bearer Token

Bearer token =

token used in Authorization header

Example:

```
Authorization: Bearer abc123xyz
```

Meaning:

> “I already authenticated.”

OAuth vs Bearer Token

OAuth	Bearer Token

Security framework	Actual token
Handles authorization process	Used for API access
Issues tokens	Credential sent in requests

Common OAuth Flow Example

Step 1 — User Authentication

User logs into application.

Step 2 — Authorization Server Validates User

Example:

- Microsoft
- Okta
- Google
- Auth0

Step 3 — Access Token Generated

Example response:

```
{
  "access_token": "abc123xyz",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

Step 4 — API Request Uses Token

```
GET /api/customer
Authorization: Bearer abc123xyz
```

Why OAuth Is Important

OAuth improves:

- security
- scalability
- session control
- API protection
- user management

VERY important in:

- banking

- cloud platforms
- CCaaS
- enterprise APIs

OAuth Benefits

Benefit	Description
No password sharing	Apps never see user password
Secure API access	Token-based authentication
Temporary access	Tokens expire
Permission control	Scoped access
Centralized authentication	SSO/identity provider support

Common OAuth Troubleshooting

Problem 1 — Expired Token

Example response:

```
401 Unauthorized
```

Cause:

- access token expired

Troubleshooting:

- refresh token
- reauthenticate user

Problem 2 — Missing Authorization Header

Missing:

```
Authorization: Bearer token
```

Result:

```
401 Unauthorized
```

Problem 3 — Invalid Token

Possible causes:

- malformed token
- copied incorrectly
- revoked token

Result:

```
401 Unauthorized
```

Problem 4 — Insufficient Permissions

Result:

```
403 Forbidden
```

Meaning:

- authenticated successfully
- lacks required permissions

Problem 5 — Wrong OAuth Scope

OAuth scopes define:

what API access is allowed

Example:

```
read:customers
write:customers
admin
```

If token lacks required scope: API rejects request.

Common OAuth Troubleshooting Flow

Step 1 — Validate Token

Check:

- token present?

- expired?
- malformed?

Step 2 — Validate Authorization Header

Correct format:

```
Authorization: Bearer token
```

Step 3 — Validate Permissions/Scopes

Check:

- API access allowed?
- correct user role?
- proper OAuth scopes?

Step 4 — Validate HTTPS/TLS

OAuth tokens should ONLY travel over:

HTTPS/TLS encrypted connections

Step 5 — Review Logs

Check:

- auth logs
- API logs
- timestamps
- token expiration

OAuth vs API Key

OAuth	API Key
More secure	Less secure
User/session-based	Usually app-based
Temporary tokens	Often static
Permission scopes	Limited control
Enterprise-grade	Simpler authentication

Common Interview Questions

“What is OAuth?”

Good Answer:

> “OAuth 2.0 is an authorization framework that enables secure API access through token-based authentication without exposing user credentials directly.”

“What is a Bearer Token?”

Good Answer:

> “A bearer token is the access token issued during OAuth authentication and used in API requests for authorization.”

“Difference between 401 and 403?”

Code	Meaning
401	Authentication failed
403	Authenticated but not authorized

“How would you troubleshoot OAuth issues?”

Good Answer:

> “I would validate the access token, confirm the Authorization header format, verify token expiration and scopes, review authentication logs, and confirm

> HTTPS connectivity and permissions.”

Important Security Concepts

NEVER expose:

- tokens
- secrets
- credentials

Tokens should always be:

- protected
- encrypted in transit
- short-lived

Easy Memory Trick

OAuth = Security Process

Bearer Token = Access Badge

Example:

```
OAuth authenticates user

Bearer token grants access
```

Important Terms To Know

Term	Meaning
OAuth	Authorization framework
Access Token	Temporary API credential
Bearer Token	Token used in requests
Refresh Token	Generates new access token
Authorization Server	Issues tokens
Scope	Permission level
HTTPS/TLS	Secure encrypted communication
401	Authentication failure
403	Permission denied

Revision #1
Created 21 May 2026 00:45:25 by Cesar Gzz
Updated 21 May 2026 00:45:44 by Cesar Gzz