
omega
point.

OAuth2 and OpenID Connect

Best practices and the BFF pattern

Agenda

- Grant types
- PKCE
- BFF
- Cookies
- Tokens
- Client authentication

Authentication

Who are you?

Authorization

What do you have access to?

what is oauth2?

All Videos Images Books Shopping More Tools

About 16,400,000 results (0.58 seconds)

OAuth 2.0, which stands for “Open Authorization”, is **a standard designed to allow a website or application to access resources hosted by other web apps on behalf of a user**. It replaced OAuth 1.0 in 2012 and is now the de facto industry standard for online authorization.

what is openid connect?

All Videos Images News Shopping More Tools

About 12,900,000 results (0.64 seconds)

OpenID Connect (OIDC) is **an open authentication protocol that works on top of the OAuth 2.0 framework**. Targeted toward consumers, OIDC allows individuals to use single sign-on (SSO) to access relying party sites using OpenID Providers (OPs), such as an email provider or social network, to authenticate their identities.

Best practices

Workgroup: Web Authorization Protocol
Internet-Draft:
draft-ietf-oauth-security-topics-22
Published: 13 March 2023
Intended Status: Best Current Practice
Expires: 14 September 2023

T. Lodderstedt
yes.com
J. Bradley
Yubico
A. Labunets
Independent Researcher
D. Fett
yes.com

OAuth 2.0 Security Best Current Practice

Abstract

This document describes best current security practice for OAuth 2.0. It updates and extends the OAuth 2.0 Security Threat Model to incorporate practical experiences gathered since OAuth 2.0 was published and covers new threats relevant due to the broader application of OAuth 2.0.

https://datatracker.ietf.org/doc/html/draft-ietf-oauth-security-topics

Financial-grade API Security Profile 1.0 - Part 1: Baseline

Foreword

The OpenID Foundation (OIDF) promotes, protects and nurtures the OpenID community and technologies. As a non-profit international standardizing body, it is comprised by over 160 participating entities (workgroup participants). The work of preparing implementer drafts and final international standards is carried out through OIDF workgroups in accordance with the OpenID Process. Participants interested in a subject for which a workgroup has been established have the right to be represented in that workgroup. International organizations, governmental and non-governmental, in liaison with OIDF, also take part in the work. OIDF collaborates closely with other standardizing bodies in the related fields.

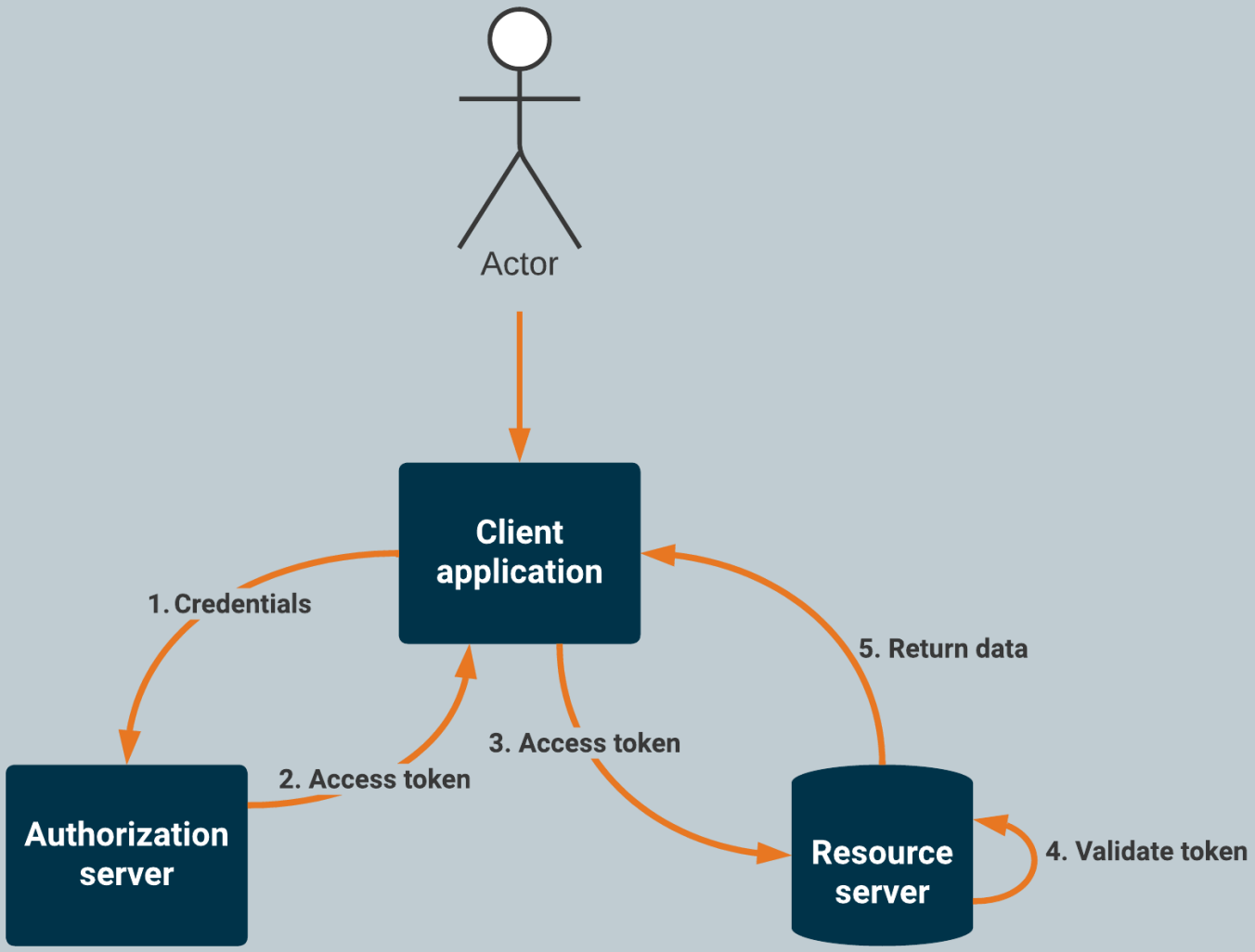
Final drafts adopted by the Workgroup through consensus are circulated publicly for the public review for 60 days and for the OIDF members for voting. Publication as an OIDF Standard requires approval by at least 50 % of the members casting a vote. There is a possibility that some of the elements of this document may be the subject to patent rights. OIDF shall not be held responsible for identifying any or all such patent rights.

Financial-grade API Security Profile 1.0 consists of the following parts:

- Financial-grade API Security Profile 1.0 - Part 1: Baseline
- **Financial-grade API Security Profile 1.0 - Part 2: Advanced**

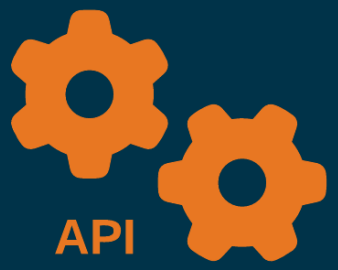
These parts are intended to be used with **RFC6749**, **RFC6750**, **RFC7636**, and **OIDC**.

https://openid.net/specs/openid-financial-api-part-1-1_0.html


Actor
Client application
1. Credentials
2. Access token
3. Access token
4. Validate token
5. Return data
Authorization server
Resource server

Public clients

Confidential clients



Grant types

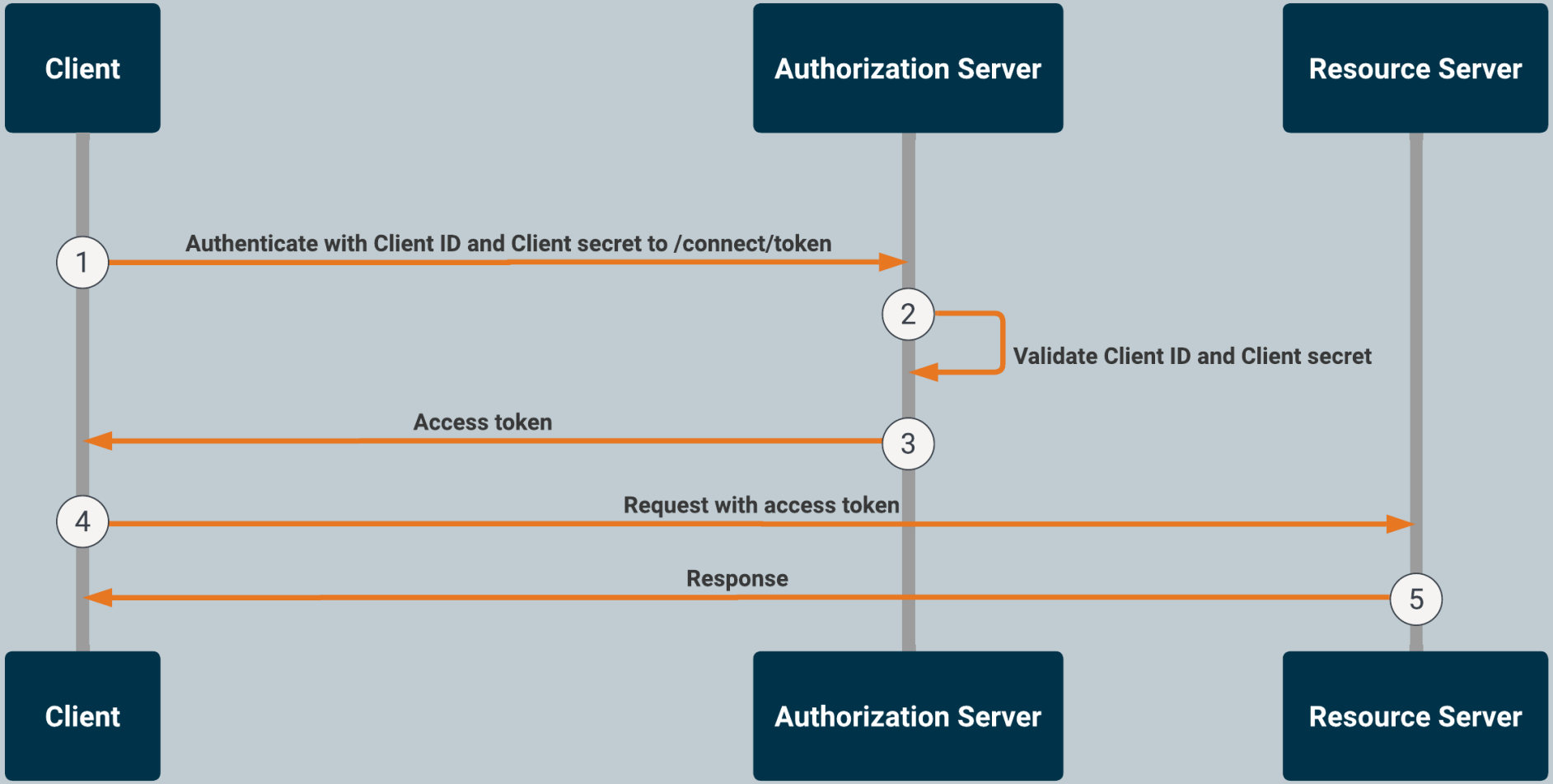
- Authorization code grant
- Client credentials grant
- Device grant
- Implicit grant
- Resource owner password credentials grant

When to use what grant?

Client credentials grant	Authorization code grant	Device grant
• No user context • Machine to machine	• User context	• User context • Cross device • Hardware without keyboard

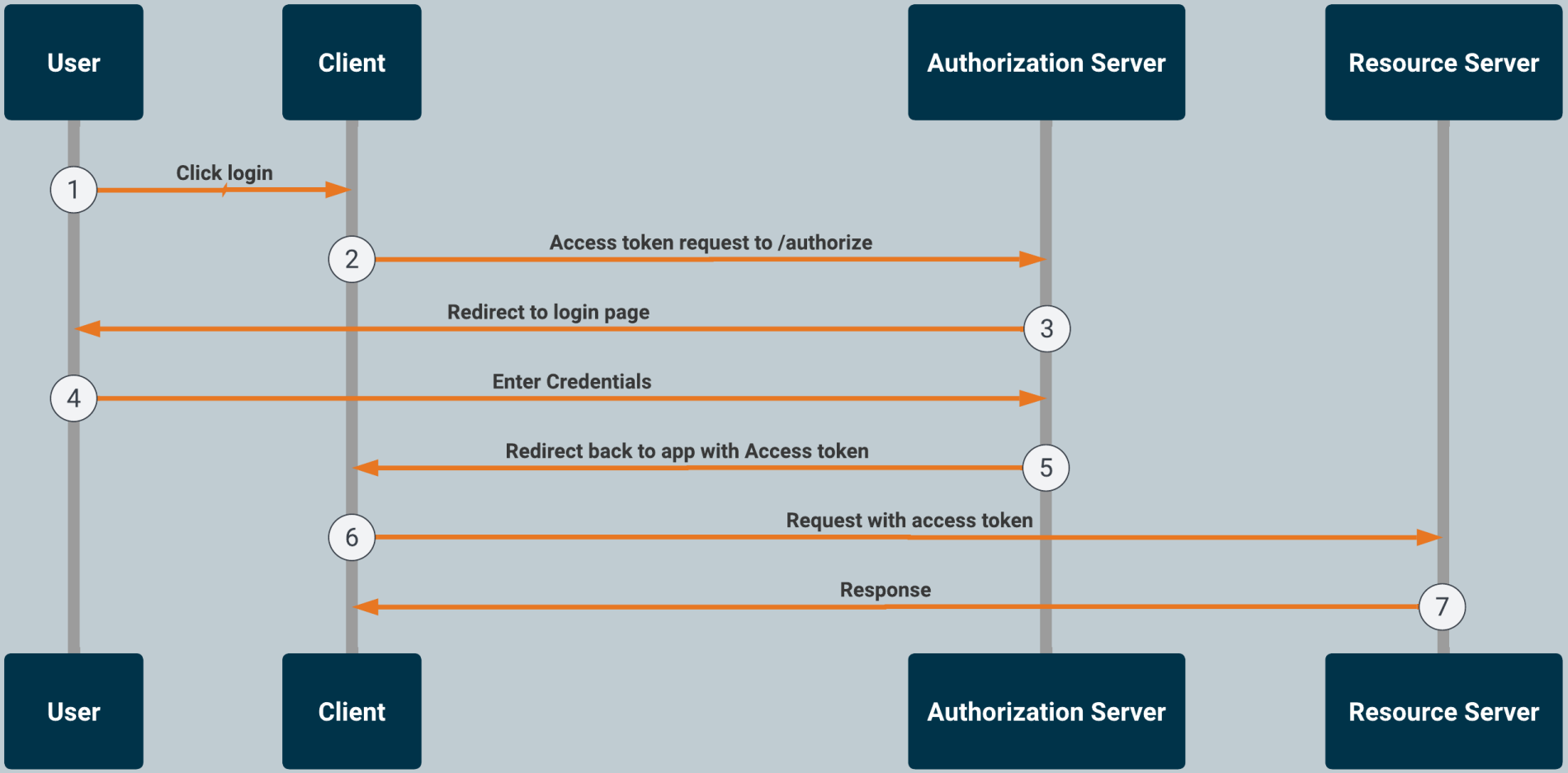
Client credential grant

- Machine to machine
- No user context



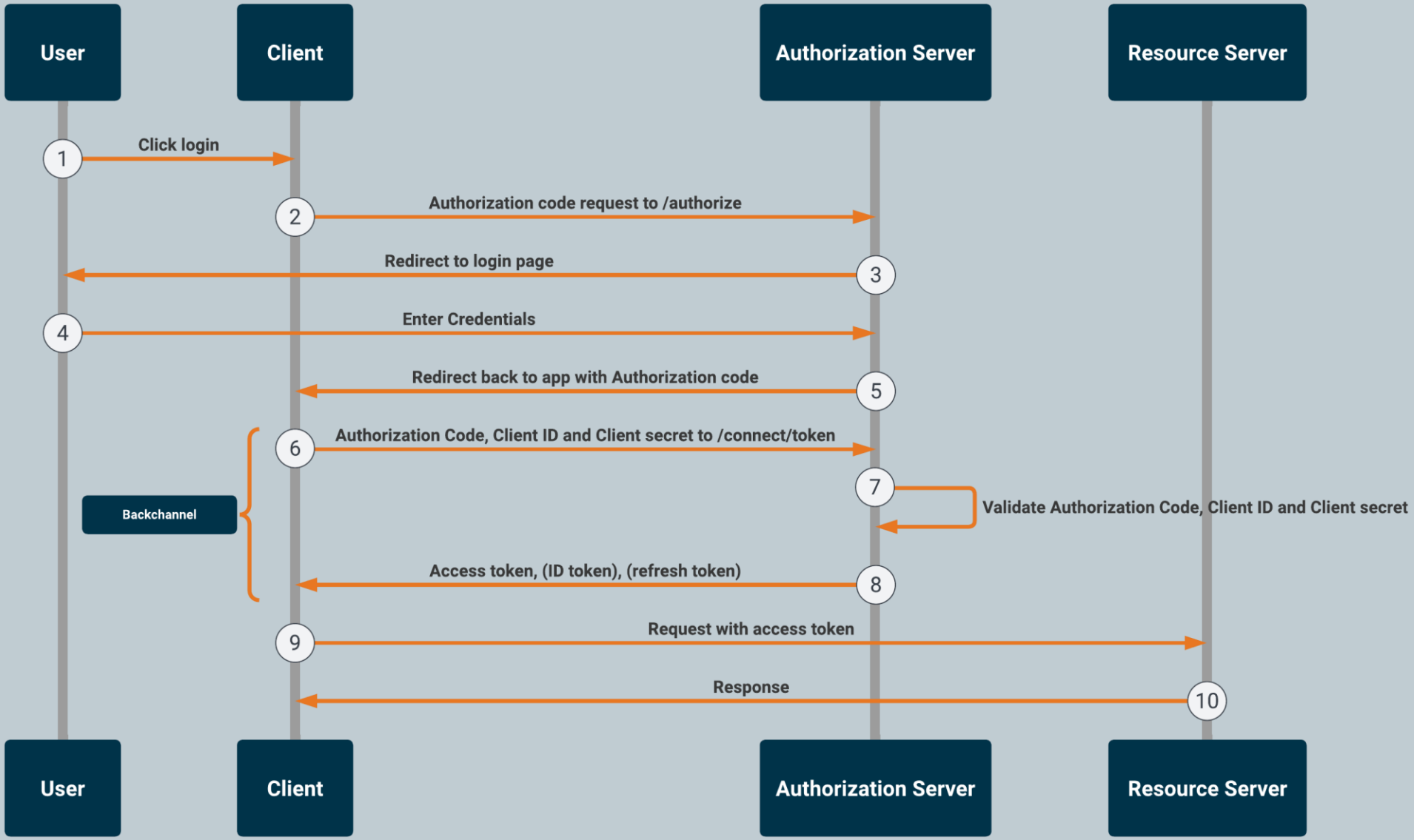
GRANT TYPES

Implicit grant

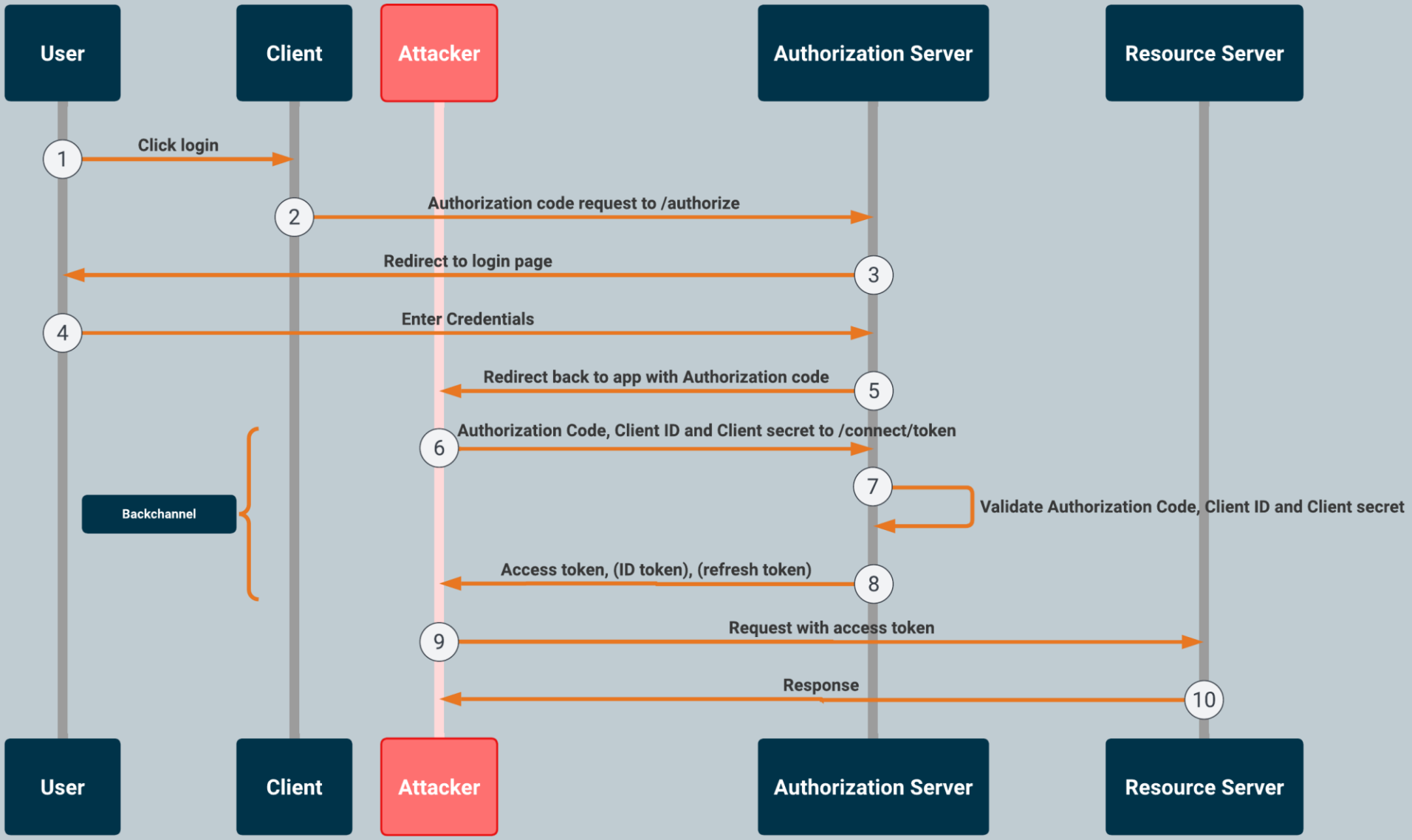


Authorization code grant

- User context

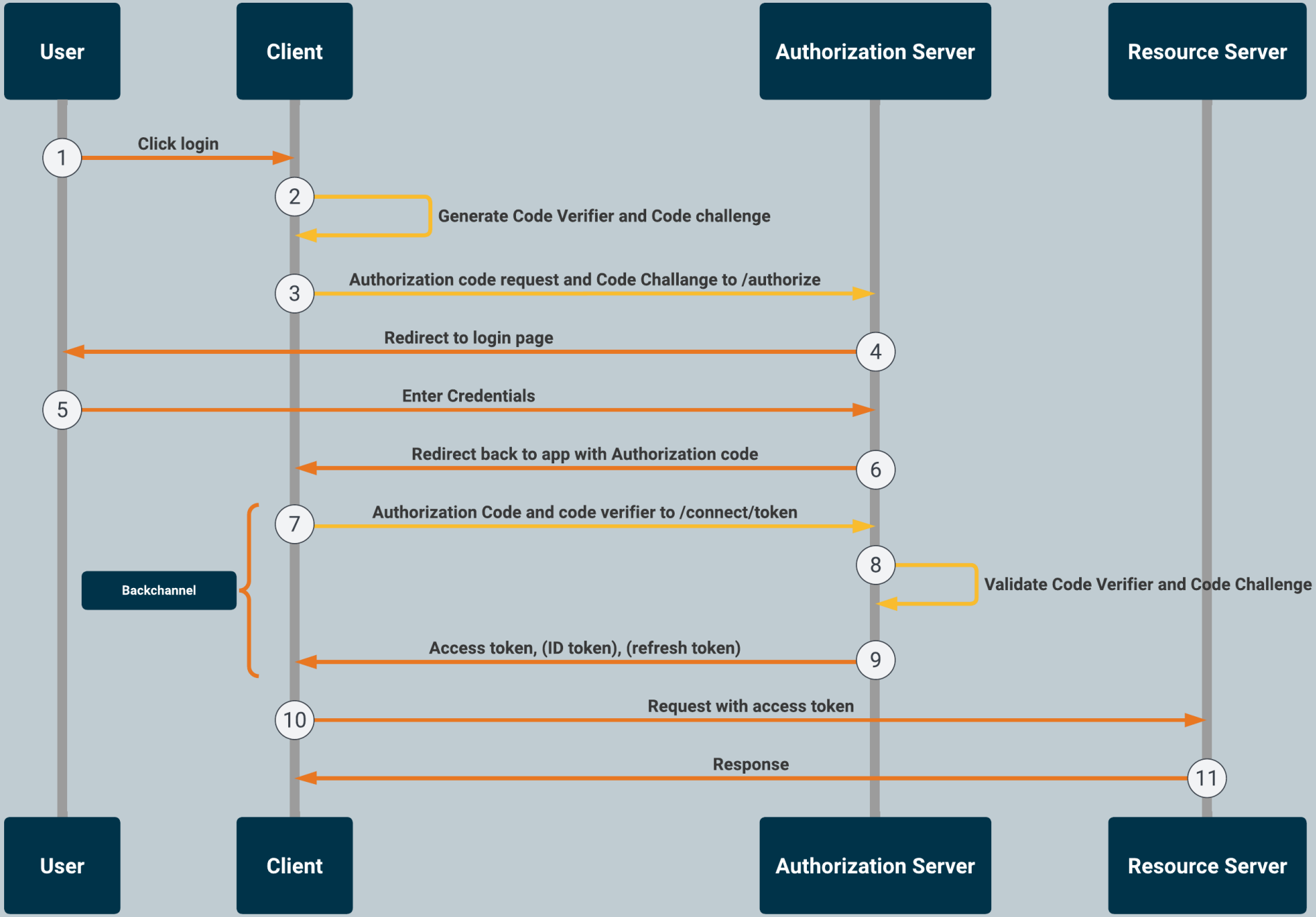


Authorization code injection attack



Authorization code can be obtained by

- Redirection attack
- Mobile OS issues
- Mix-up attack
- Logs



GRANT TYPES

Authorization code grant with PKCE (Proof key for code exchange)

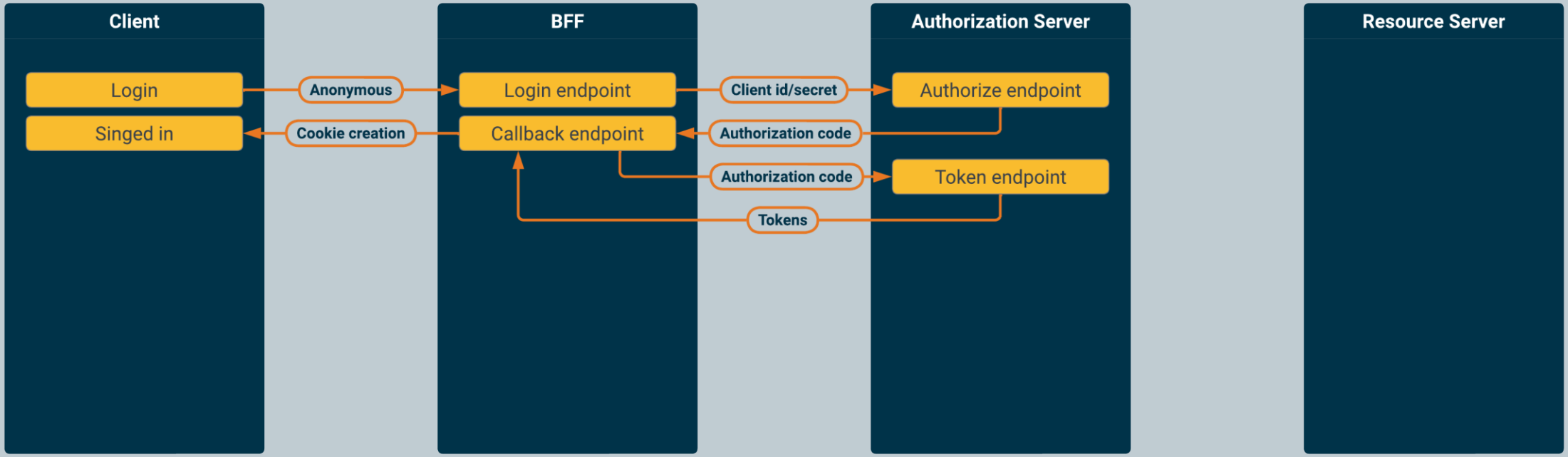
- Code Verifier:
  - random string
- Code challenge:
  - base64(sha256(code Verifier))

PKCE downgrade attack countermeasures

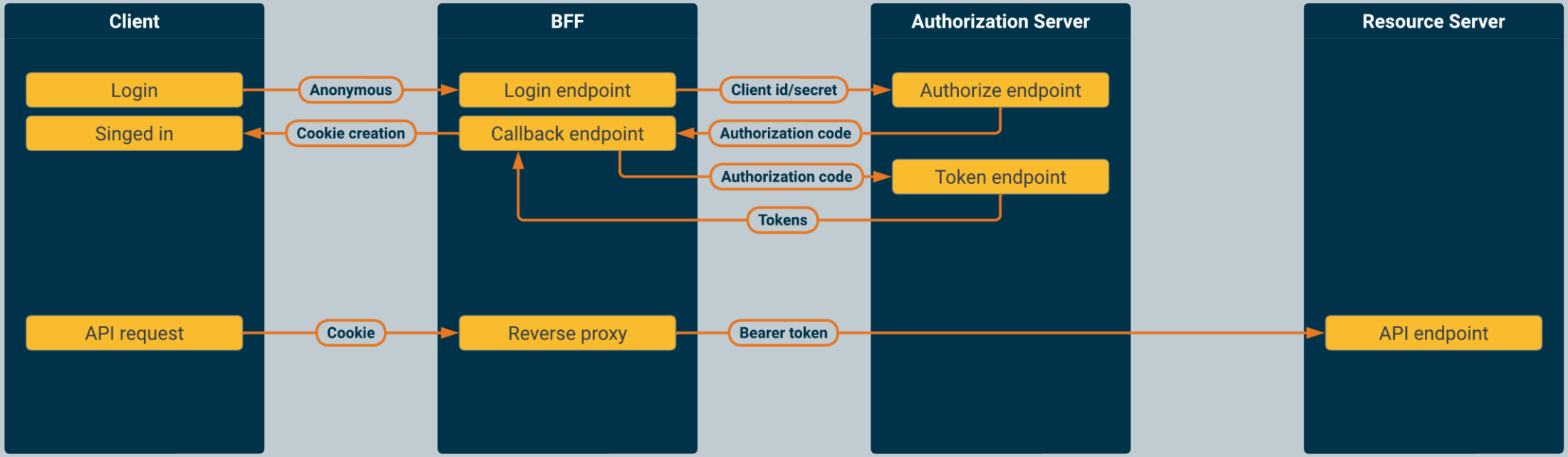
- Use state
- Authorization server validates presence of code verifier

Cross-Site Scripting (XSS)

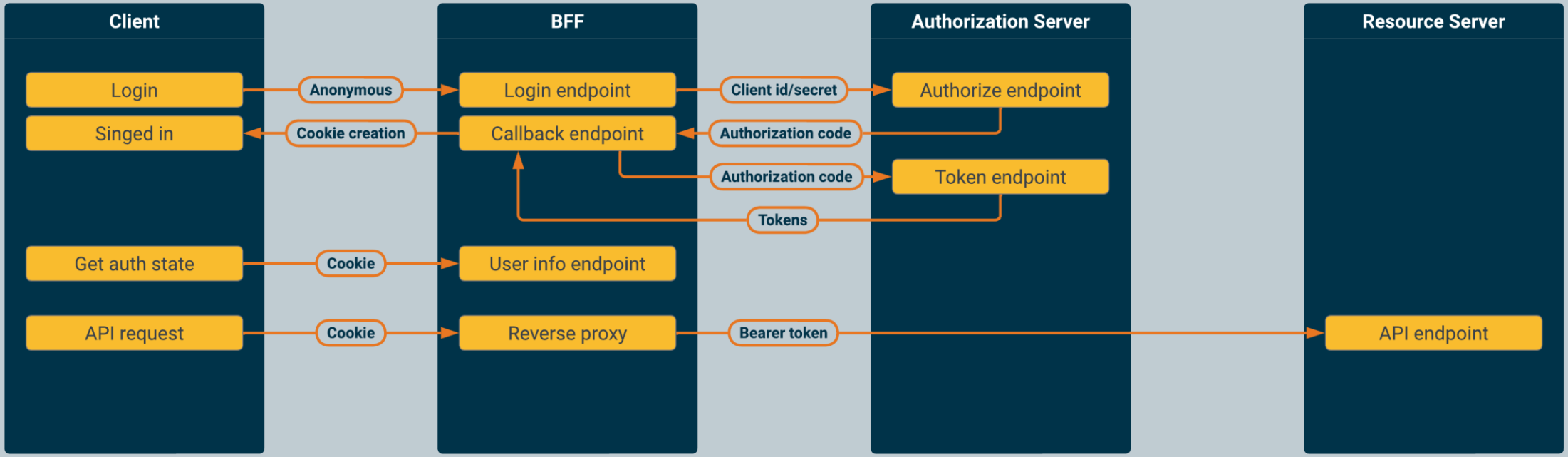
BFF – Backend For Frontend



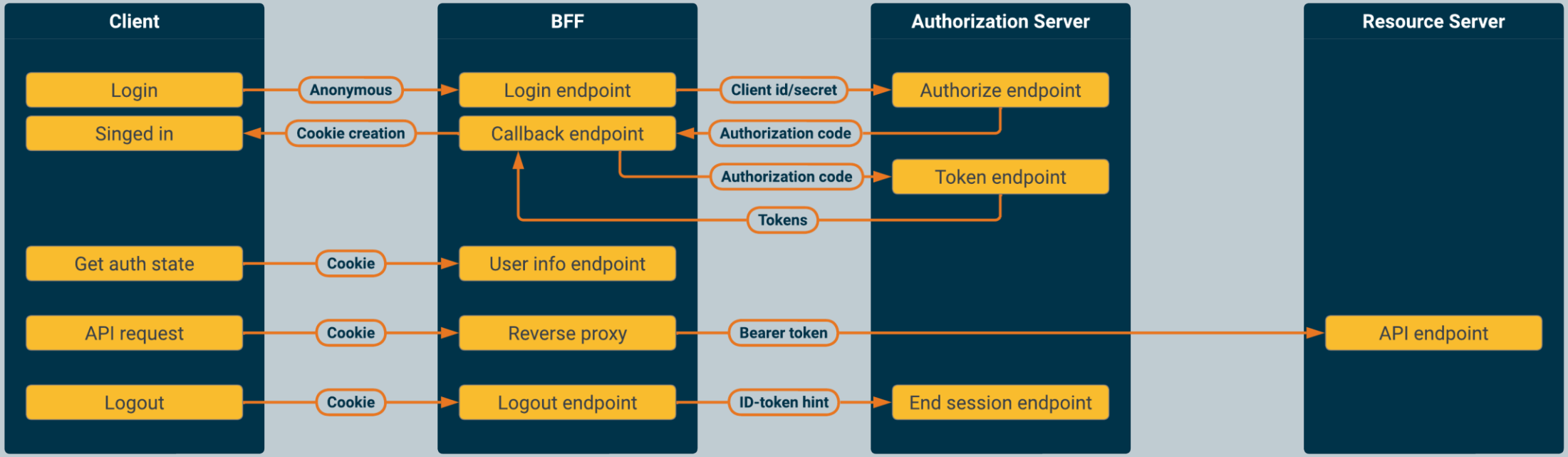
BFF – Backend For Frontend



BFF – Backend For Frontend



BFF – Backend For Frontend



Cookies

- Session bound
- Httponly
- Encrypted
- Samesite: strict

- Sliding expiration

Cross-Site Request Forgery (CSRF or XSRF)

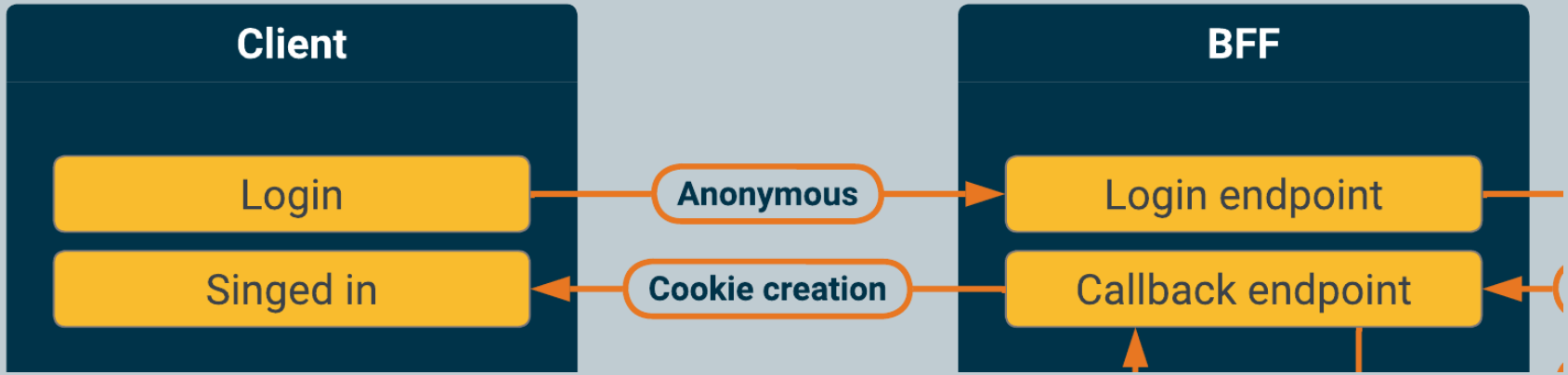
- Same site session cookie
- Anti-forgery token for CSRF protection from sub-domains

Open redirector attack

- Redirect to product page:

https://myapp.com/login?returnUrl=%2Fproducts

https://myapp.com/login?returnUrl=http%3A%2F%2Fevilapp.com



Token handling

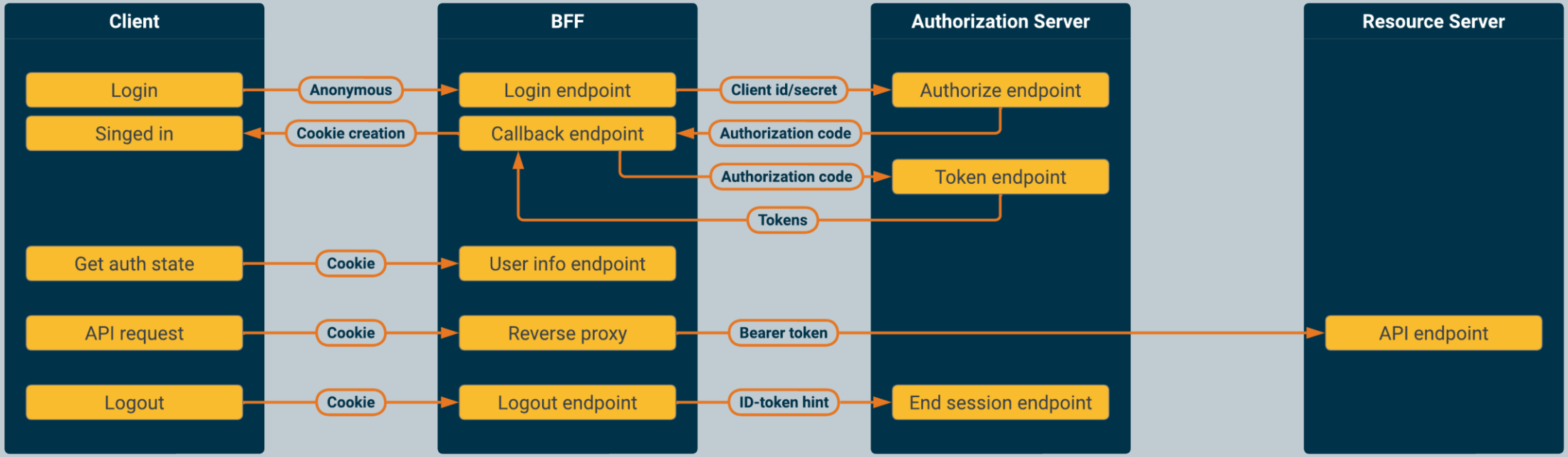
- Id token: for the client
- Access token: for the recipient

- Not available outside the application

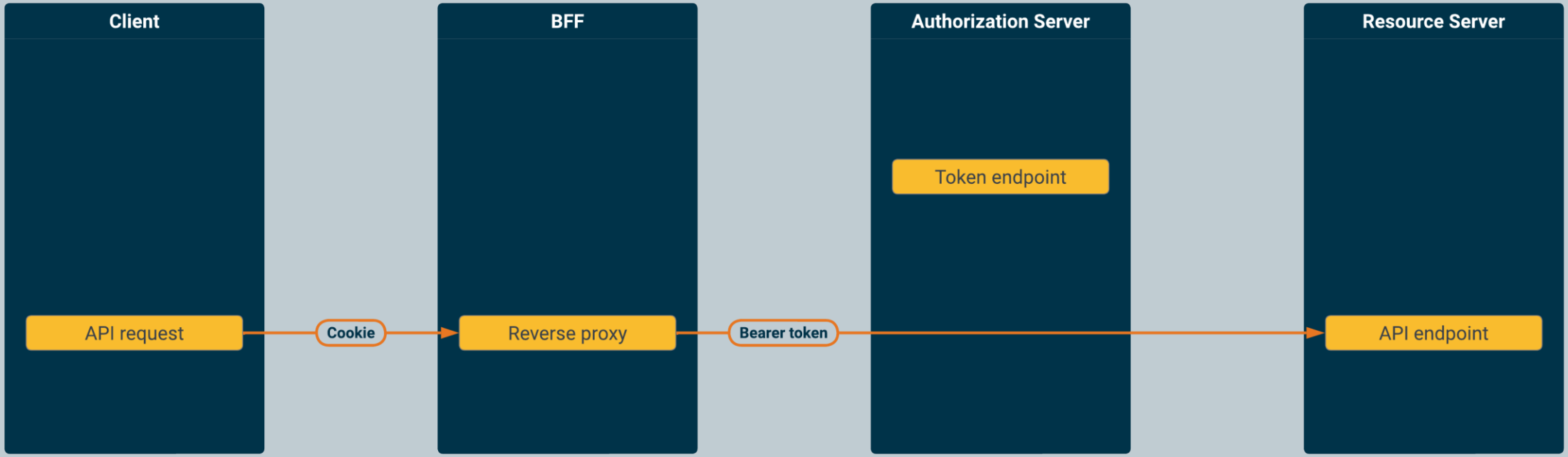
Refresh tokens

- scope: offline_access
- Single use
- Refresh token rotation
- Absolute expiration

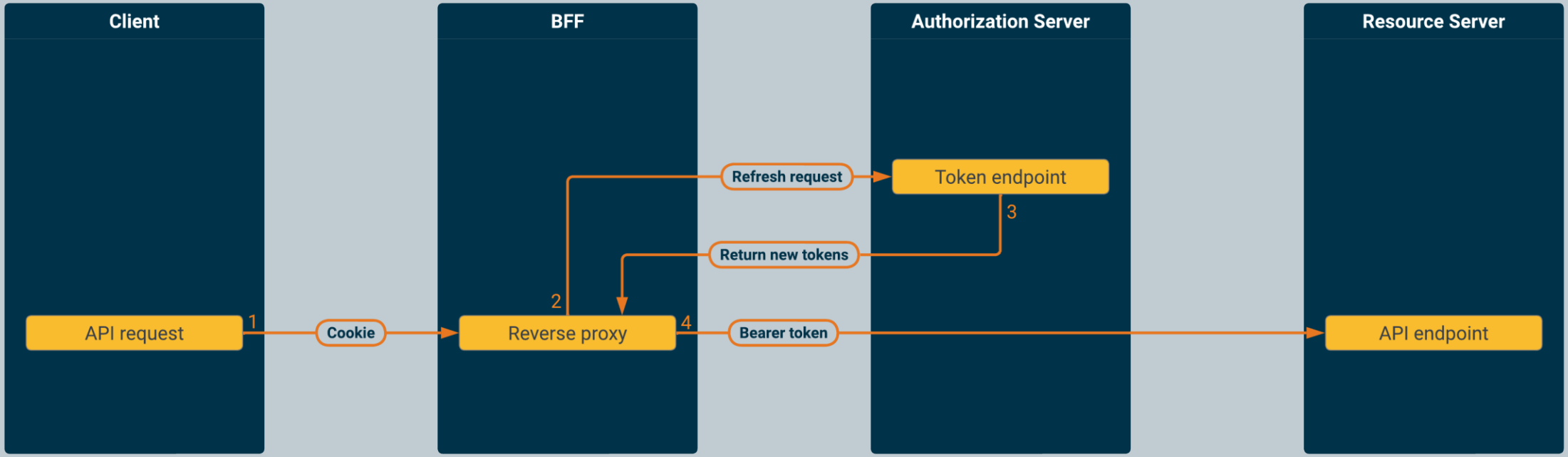
BFF – Token refresh



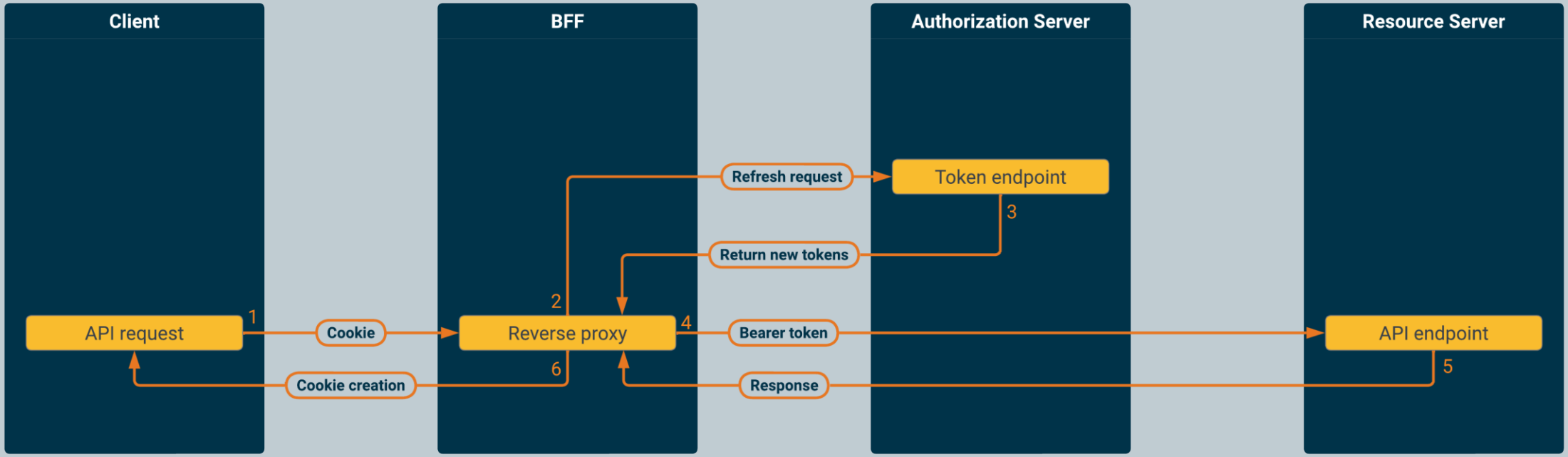
BFF – Token refresh



BFF – Token refresh



BFF – Token refresh



Client Authentication

Shared secret

Private key JWT

mTLS

JSON Web Tokens - jwt.io

jwt.io

JWT

Debugger Libraries Introduction

Encoded

PASTE A TOKEN HERE

eyJhbGciOiJSUzI1NiIsImtpZCI6Ijk4NkNGRkJBMDBDMDg1OEFGRTI5MEI1RUNGMTdDMTg0REY5Qjc3NzQiLCJ4NXQiOiJtR3pfdWdEQWhZci1LUXRlenhmQmhOLWJkM1EiLCJ0eXAiOiJKV1QifQ.eyJzdWIiOiIwMTFiZDNiMi1mNjcwLTQ1OTktODNhYy0zYTkyOTcyMzZhOWEiLCJpYXQiOiIxNjgzNTUxOTUyIiwianRpIjoiNjE5ZWNiZDctMGNlMC00Mzc1LThlNWYtZmZlYWU5YjhlMzQ5IiwiY2xpZW50X2lkIjoiMDExYmQzYjItZjY3MC00NTk5LTgzYWMtM2E5Mjk3MjM2YTlhIiwibmJmIjoxNjgzNTUxOTUyLCJleHAiOjE2ODM1NTE5NjIsImlzcyI6IjAxMWJkM2IyLWY2NzAtNDU5OS04M2FjLTNhOTI5NzIzNmE5YSIsImF1ZCI6Imh0dHBzOi8vbXktZXhhbXBsZS1zdHMuY29tIn0.QoSTQD3_-WZwrRS_5ISbuJoQ07l9tfd9M1wXGRIEGutvVntsFSu4Z_3wqSi8VJfwwO0j29rR3k39krDC6AHR5EhQoDmEQj7A_1GUNgnelxFkl-jiBIUFK2PFxTTDN7VEqNa62lel1NdNLHyX1enQcgZMUvTrM4Bt0jLAFlkcsPtJilmgRERoQ9eTeEMYCXgGaZ0lwOPxqFQq9-L8OtIn0KfXTxH_KKHOCYdbeZgoExXDnVtWXQ-6MJM9kyZ7RM8yGUn2l9tFZ_OOm7l4ByUsYq3Mdb4BY6MxgMh1f1qXOi7KuIpiaV5EUMgKEqFp7DxuiZeCUQ0vrSVpTT-7LAQUZg

Decoded

HEADER: ALGORIT


```
{
  "alg": "R
  "kid": "9
  "x5t": "m
  "typ": "J
}
```


PAYLOAD: DATA


```
{
  "sub": "011bd3b2-f670-4599-83ac-3a9297236a9a",
  "iat": "1683551952",
  "jti": "619ecbd7-0ce0-4375-8e5f-ffeae9b8e349",
  "client_id": "011bd3b2-f670-4599-83ac-3a9297236a9a",
  "nbf": 1683551952,
  "exp": 1683551962,
  "iss": "011bd3b2-f670-4599-83ac-3a9297236a9a",
  "aud": "https://my-example-sts.com"
}
```


VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
```

Public Key in SPKI, PKCS #1, X.509 Certificate, or JWK string format.


```
{
  "sub": "011bd3b2-f670-4599-83ac-3a9297236a9a",
  "iat": "1683551952",
  "jti": "619ecbd7-0ce0-4375-8e5f-ffeae9b8e349",
  "client_id": "011bd3b2-f670-4599-83ac-3a9297236a9a",
  "nbf": 1683551952,
  "exp": 1683551962,
  "iss": "011bd3b2-f670-4599-83ac-3a9297236a9a",
  "aud": "https://my-example-sts.com"
}
```

Private key based authentication is more secure

- Can be stored outside of application in certificate
- Authorization server does not keep secrets
- The JWT expires – limiting replay attacks
- Tokens can be bound to the private key

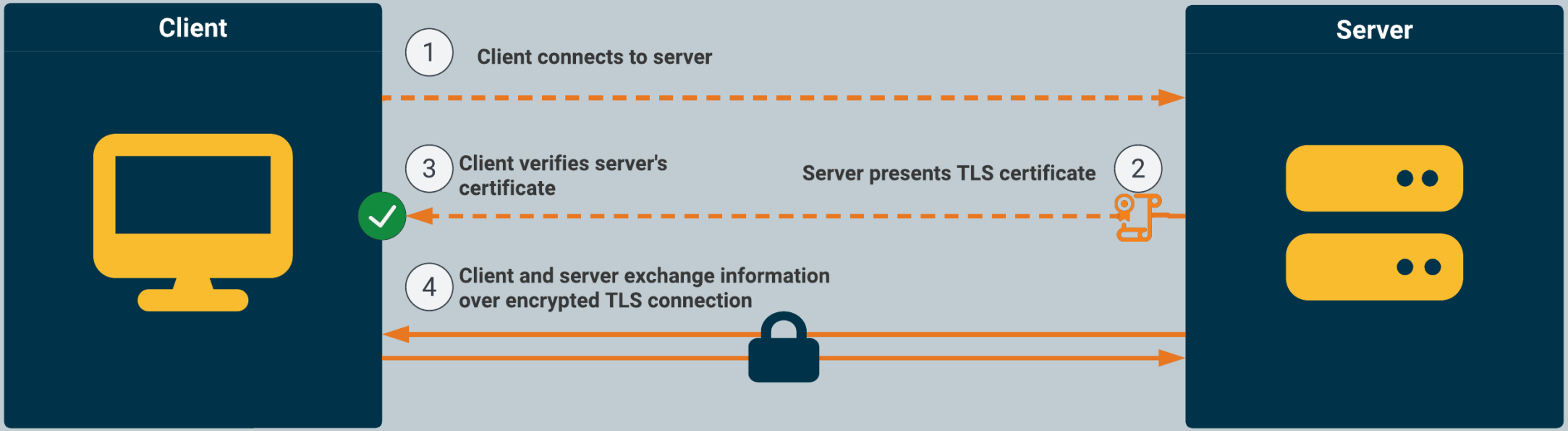
Client Authentication

Shared secret

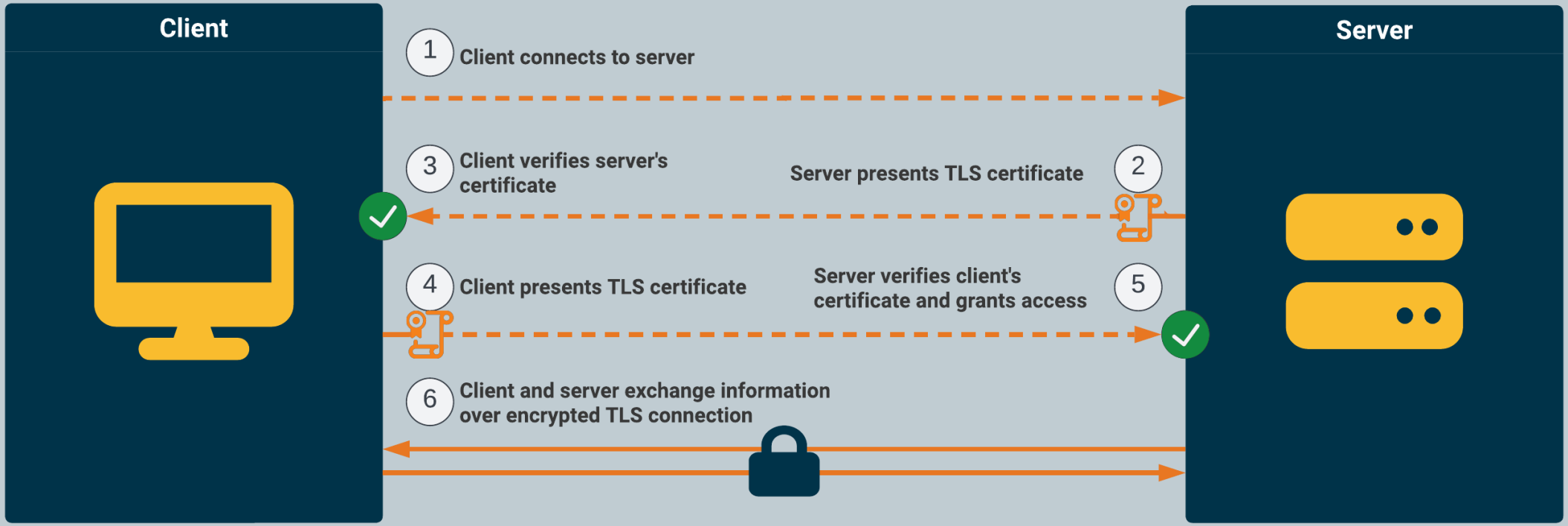
Private key JWT

mTLS

Transport Layer Security (TLS)



Mutual Transport Layer Security (mTLS)



Binding access tokens to a client

- mTLS
- Demonstration of Proof-of-Possession (DPoP)

Main takeaways

- Client credential grant – service to service
- Authorization code flow with PKCE – users
- BFF to keep tokens out of the frontend
- Protect against XSS and CSRF
- Single use refresh tokens
- Client authentication: Use private key JWT or mTLS instead of shared secret

Thank you for your time