



# RECENT EVOLUTIONS IN THE OAUTH 2.0 AND OPENID CONNECT LANDSCAPE

---

DR. PHILIPPE DE RYCK

<https://PragmaticWebSecurity.com>

# DR. PHILIPPE DE RYCK

- Deep understanding of the web security landscape
- Google Developer Expert (not employed by Google)
- Course curator of the  **SecAppDev** course  
(<https://secappdev.org>)



## Pragmatic Web Security

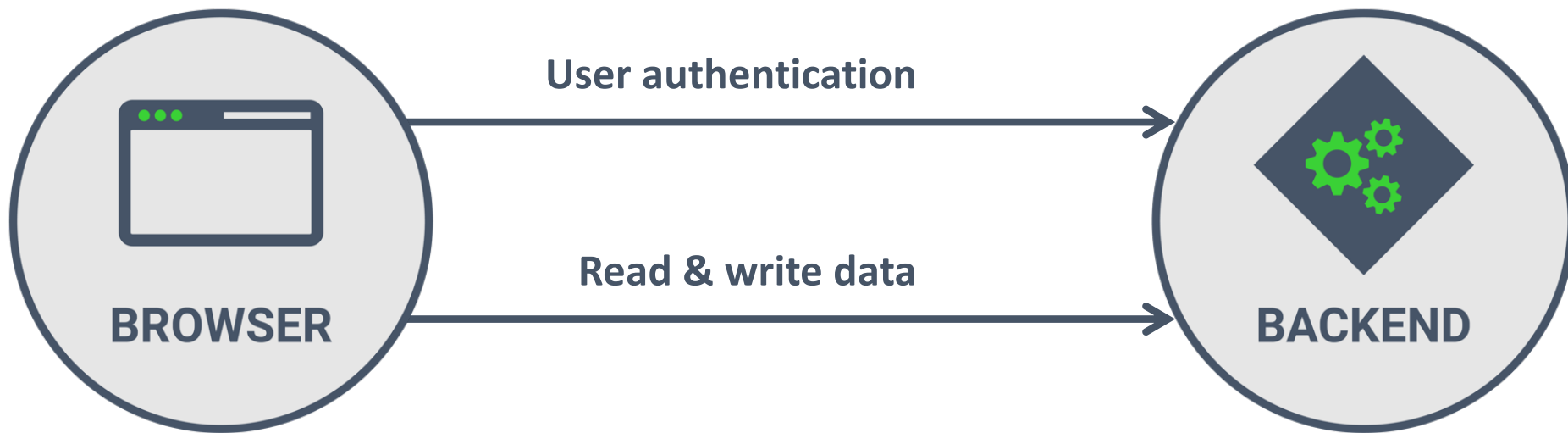
High-quality security training for developers and managers

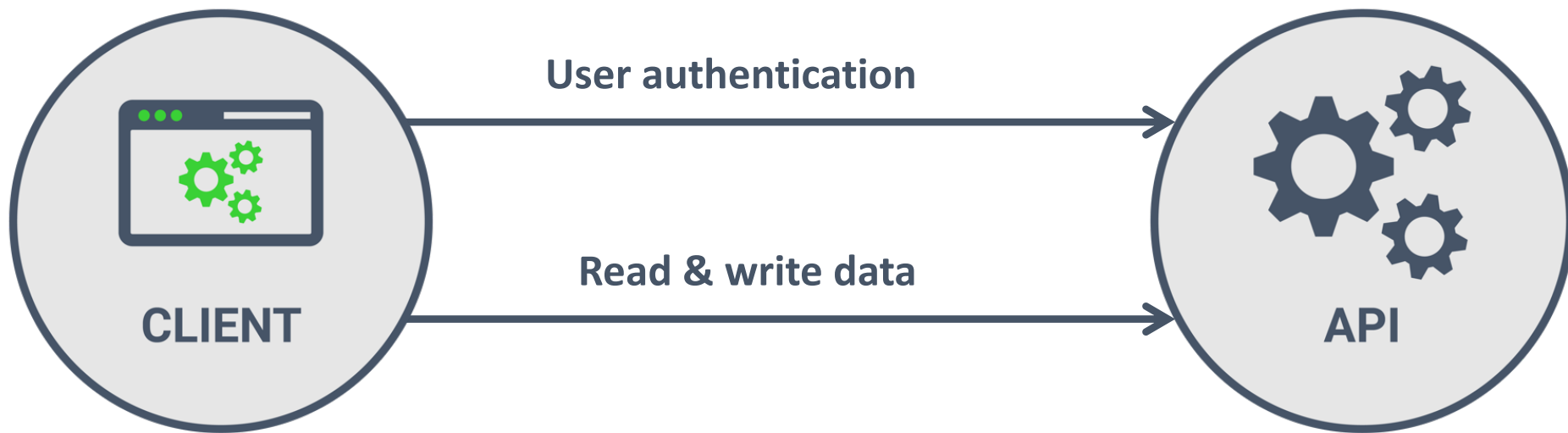
Custom courses covering web security, API security, Angular security, ...

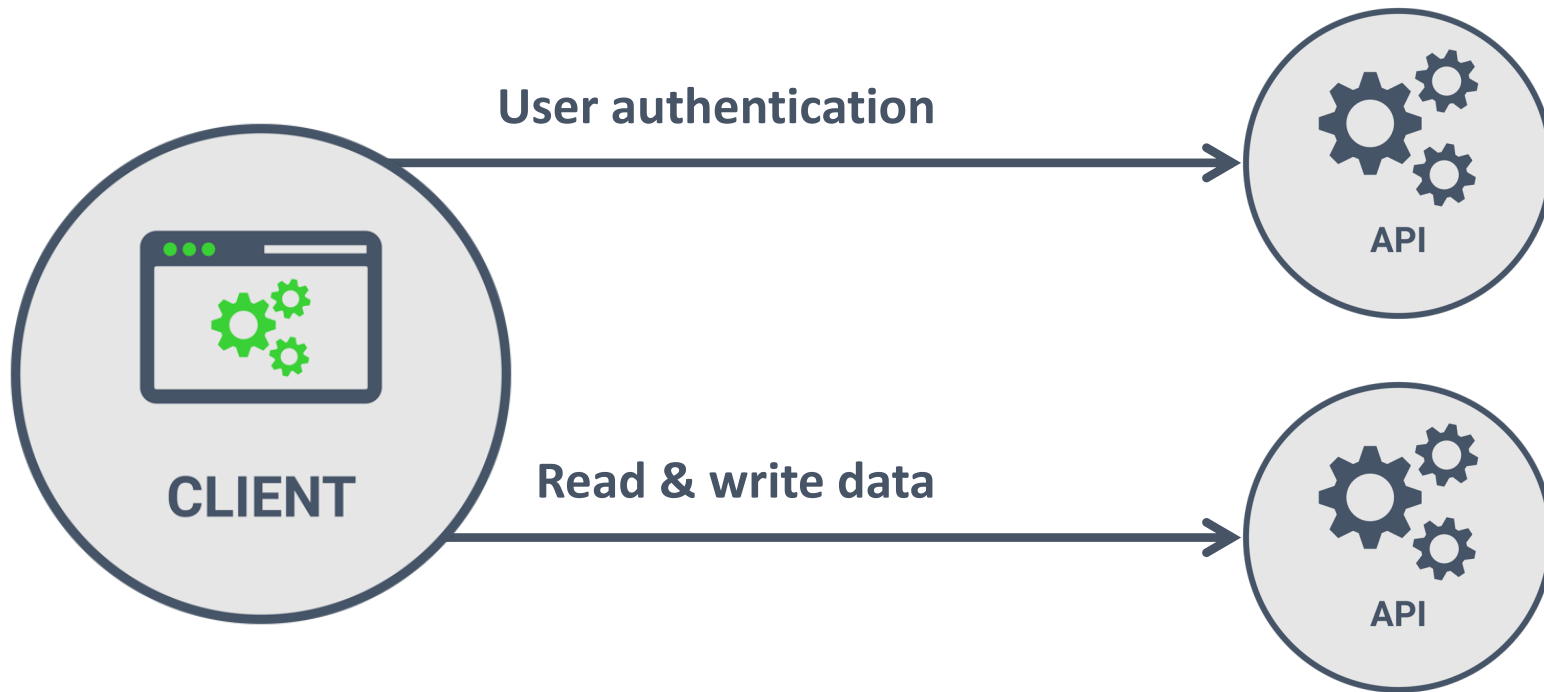
Consulting services on security, OAuth 2.0, OpenID Connect, ...

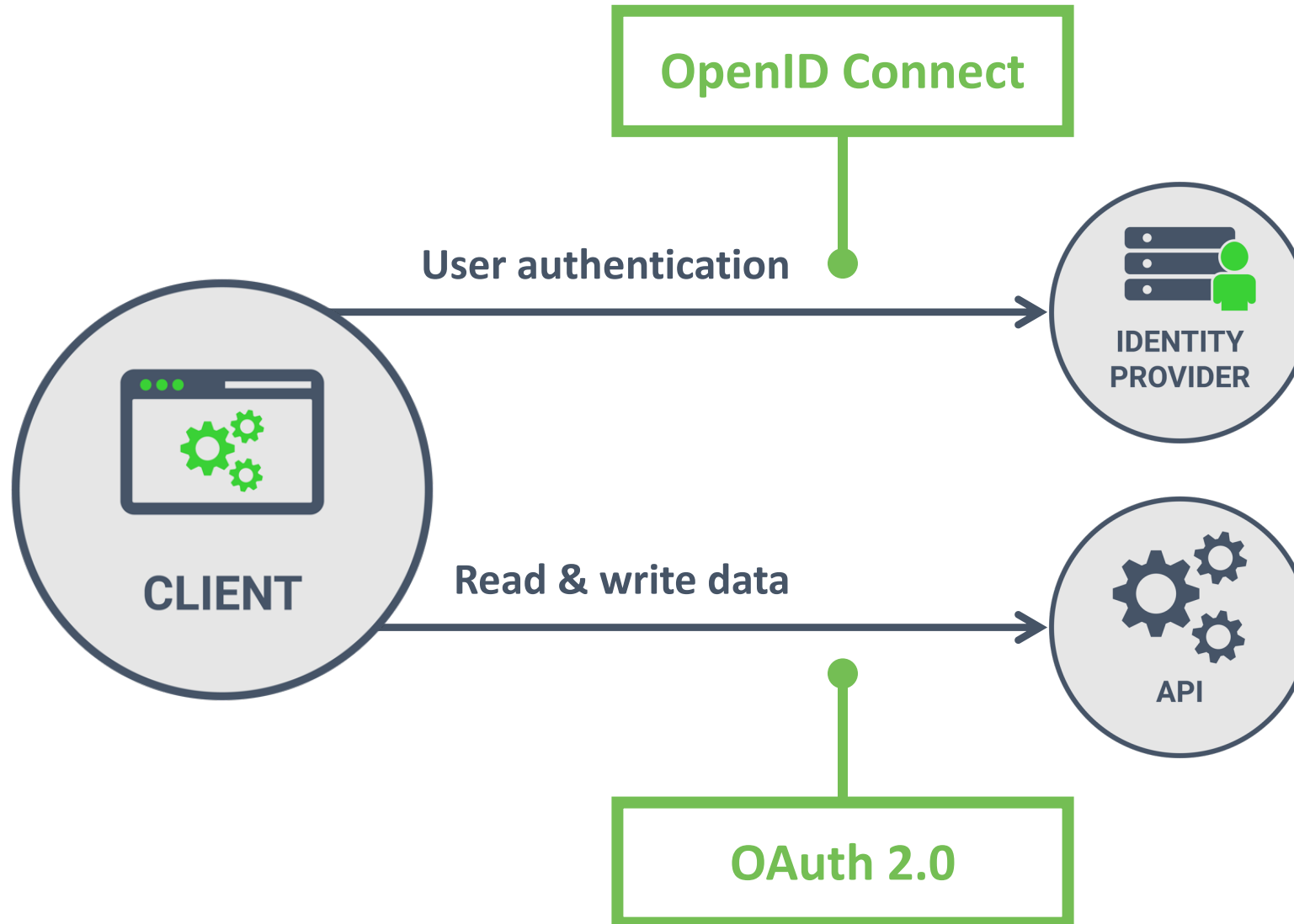
@PHILIPPEDERYCK

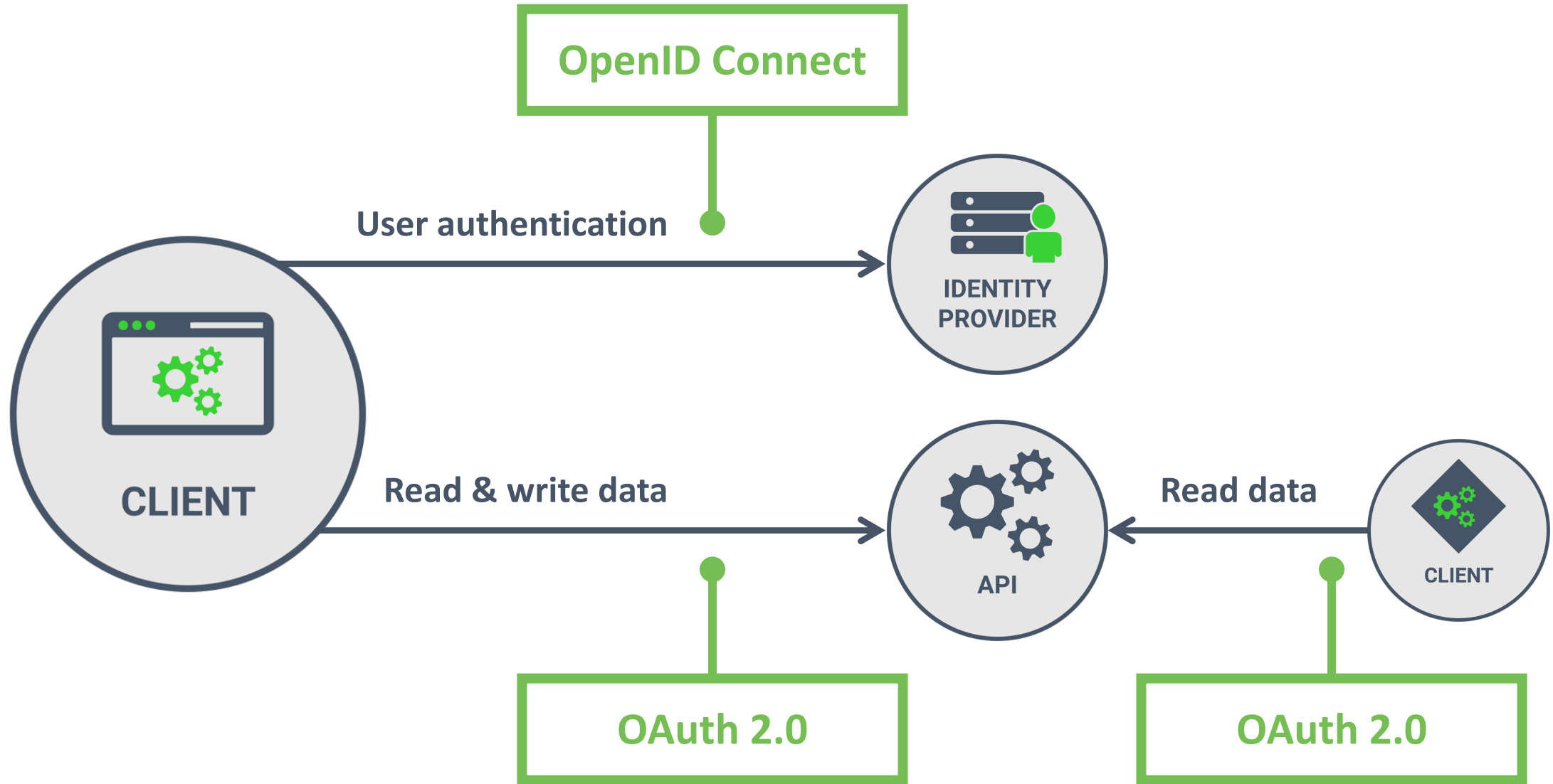
[HTTPS://PRAGMATICWEBSECURITY.COM](https://pragmaticwebsecurity.com)

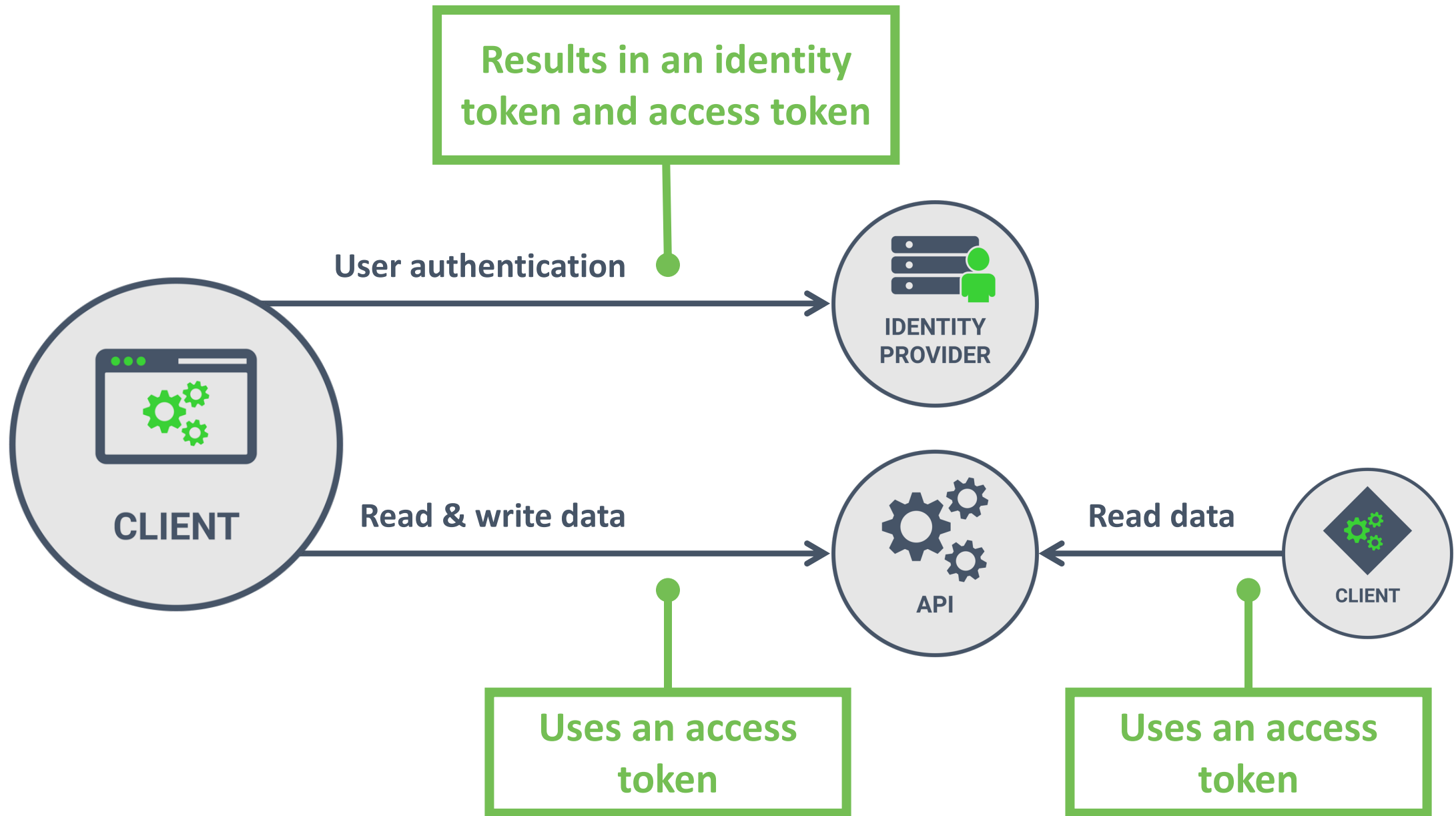














# OAuth 2.0 AND OPENID CONNECT

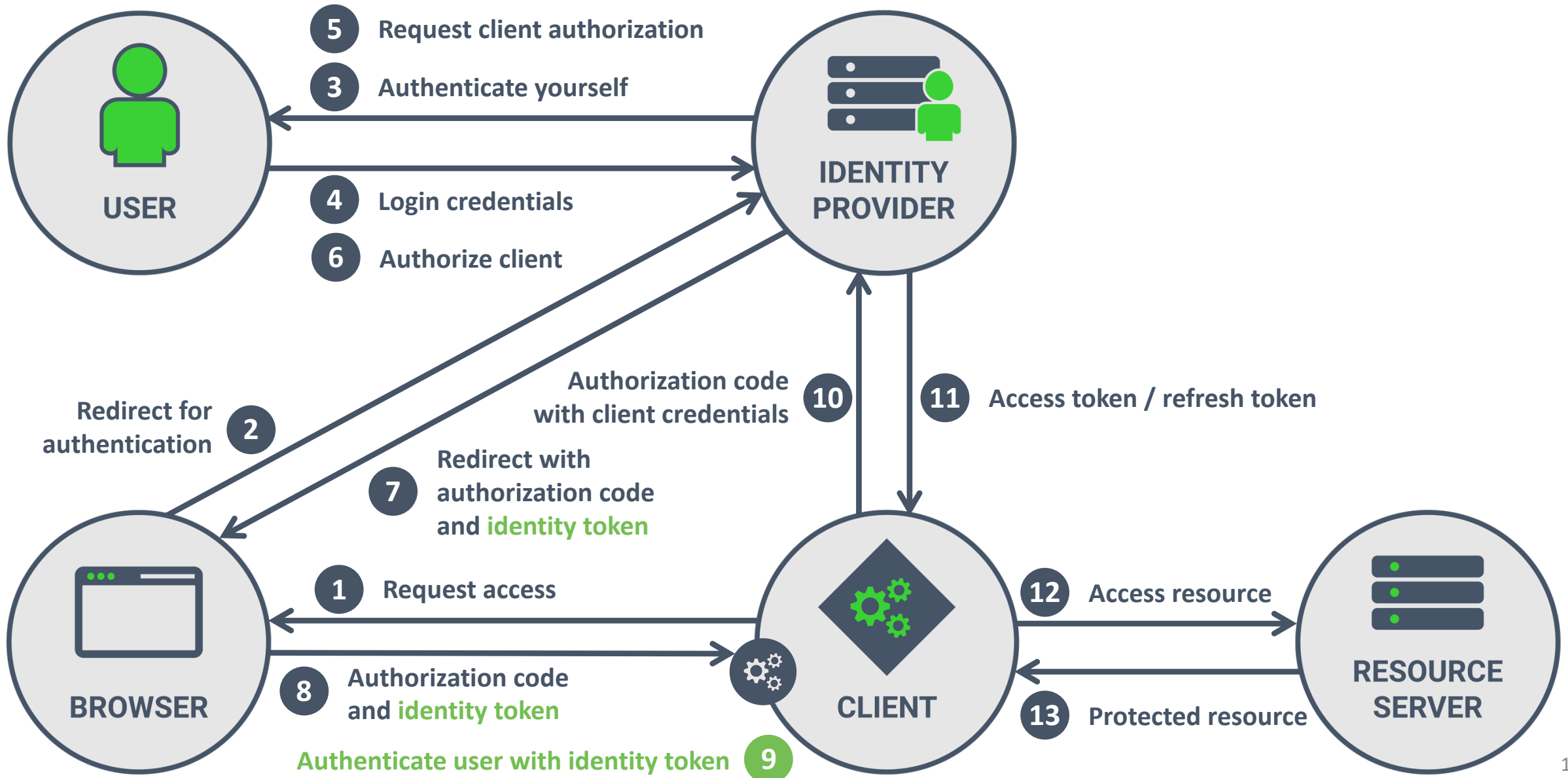


*OpenID Connect provides user authentication*

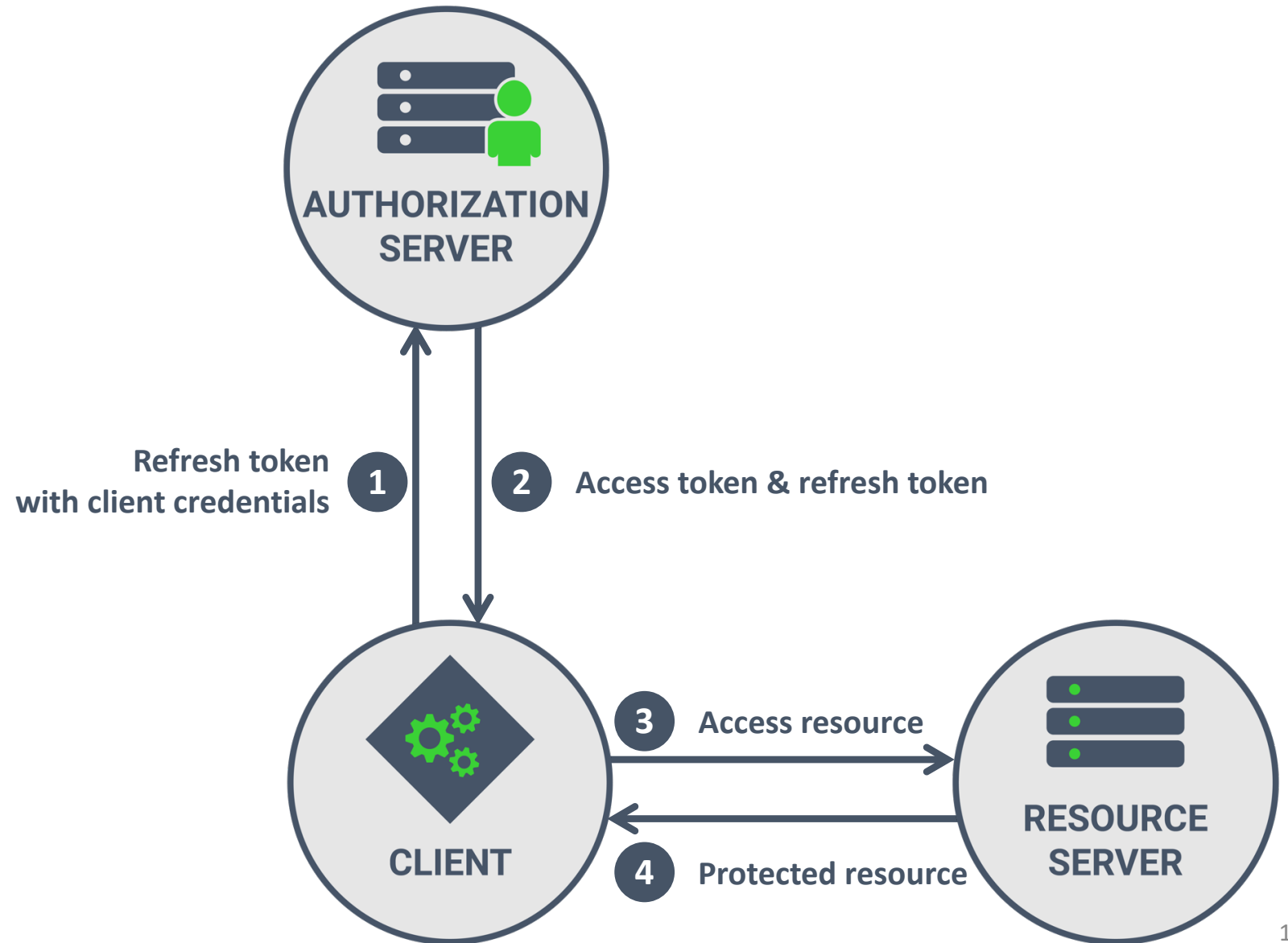
*OAuth 2.0 allows a client to access resources on behalf of the user*

*Modern applications use a combination of both protocols*

# THE OIDC HYBRID FLOW



# THE REFRESH TOKEN FLOW



# THE OIDC HYBRID FLOW

- Clients are backend applications running in a "secure" environment
- The hybrid flow returns an identity token, access token and refresh token
  - Identity tokens are issued through the frontchannel, along with an authorization code
  - The authorization code can be exchanged for an access token and refresh token
  - Using the authorization code requires client authentication
- Refresh tokens allow the client to obtain a new access token
  - Using a refresh token requires client authentication

# Buffer security breach has been resolved – here is what you need to know



by Joel Gascoigne

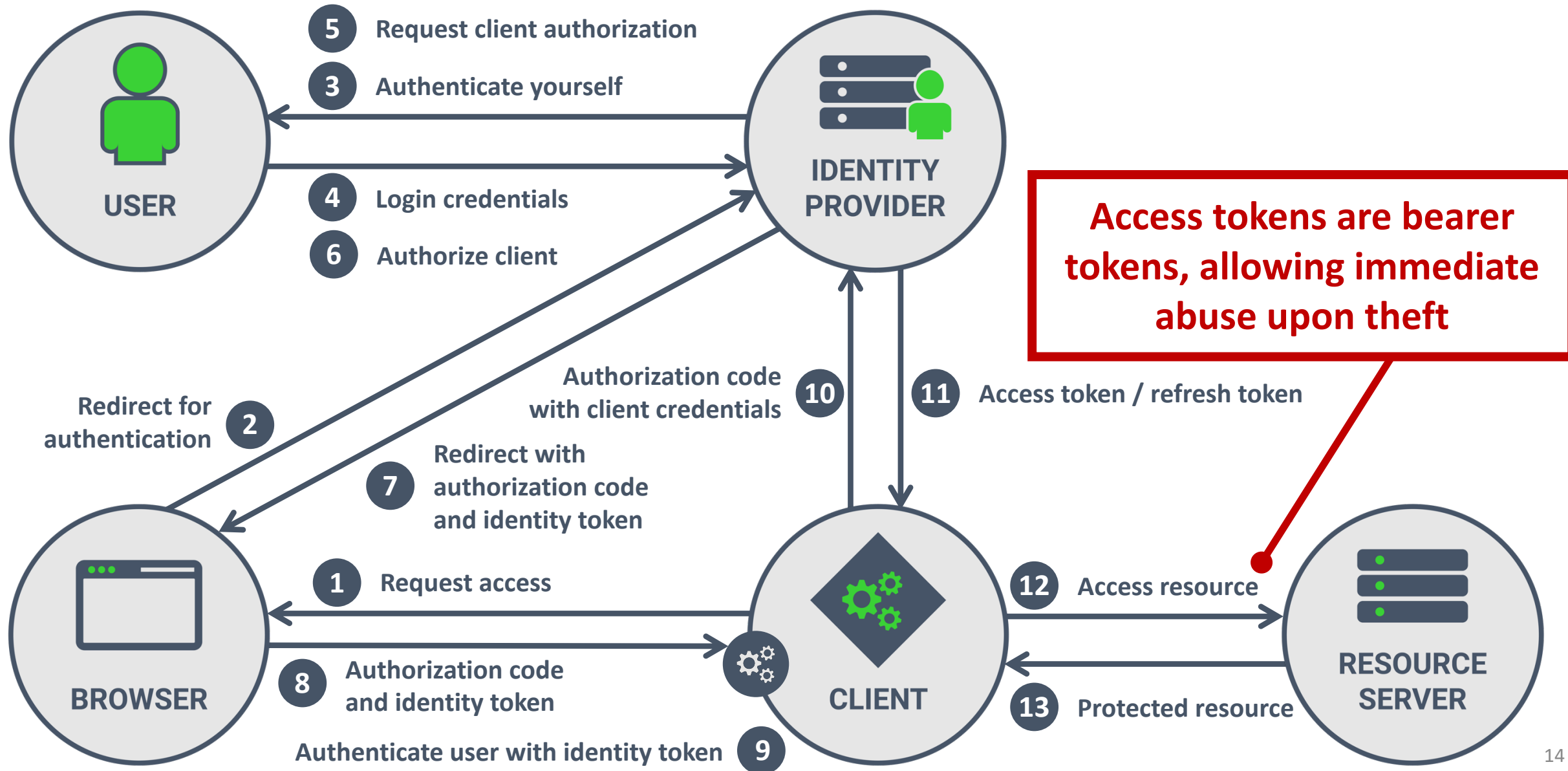


“

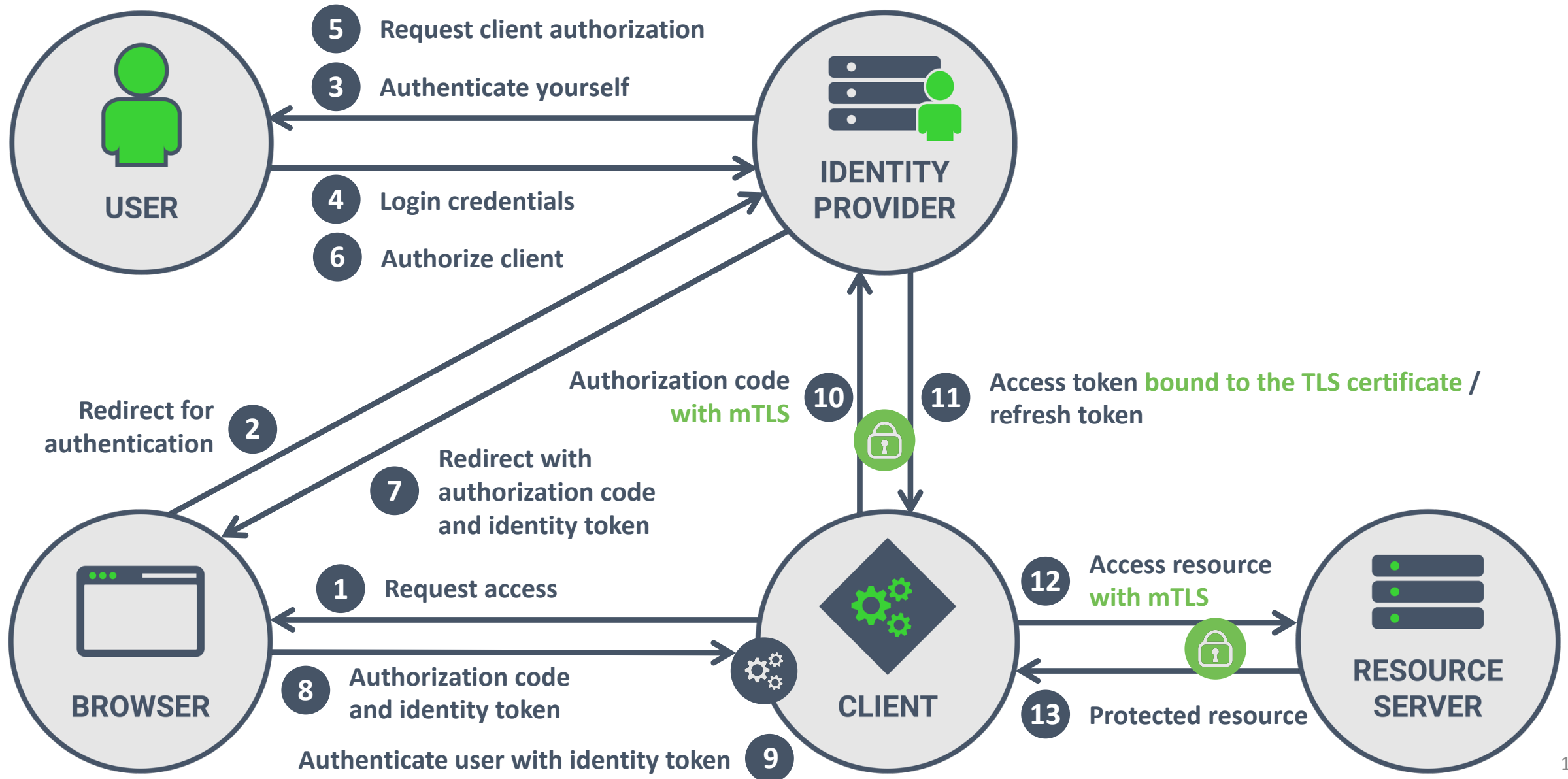
The hackers were able to steal some of our Facebook and Twitter access tokens from our users.

”

# THE DANGER OF BEARER TOKENS



# BINDING TOKENS TO TLS CERTIFICATES



```
{  
  
  "sub": "jdoe@example.com",  
  "aud": "https://api.example.com",  
  "azp": "RandomClientID",  
  "iss": "https://authorizationserver.example.com/",  
  "exp": 1419356238,  
  "iat": 1419350238,  
  "scope": "read write",  
  "jti": "405b4d4e-8501-4e1a-a138-ed8455cd1d47",  
  "cnf": {  
    "x5t#S256": "bwck0esc3ACC3DB2Y5_1ESsXE8o91tc05089jdN-dg2"  
  }  
}
```





# PROOF-OF-POSSESSION FOR ACCESS TOKENS

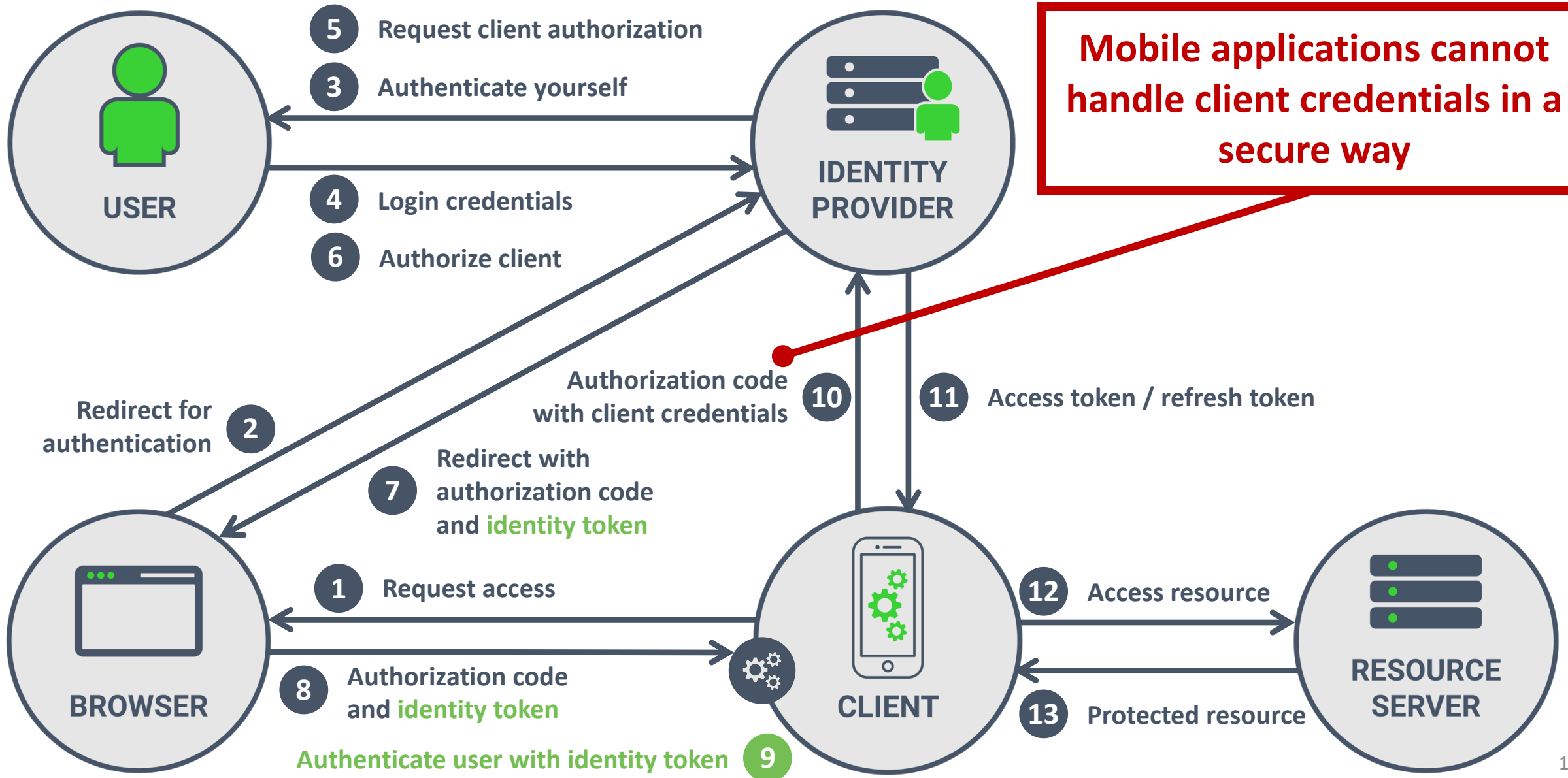


*Many confidential clients still rely on bearer access tokens*

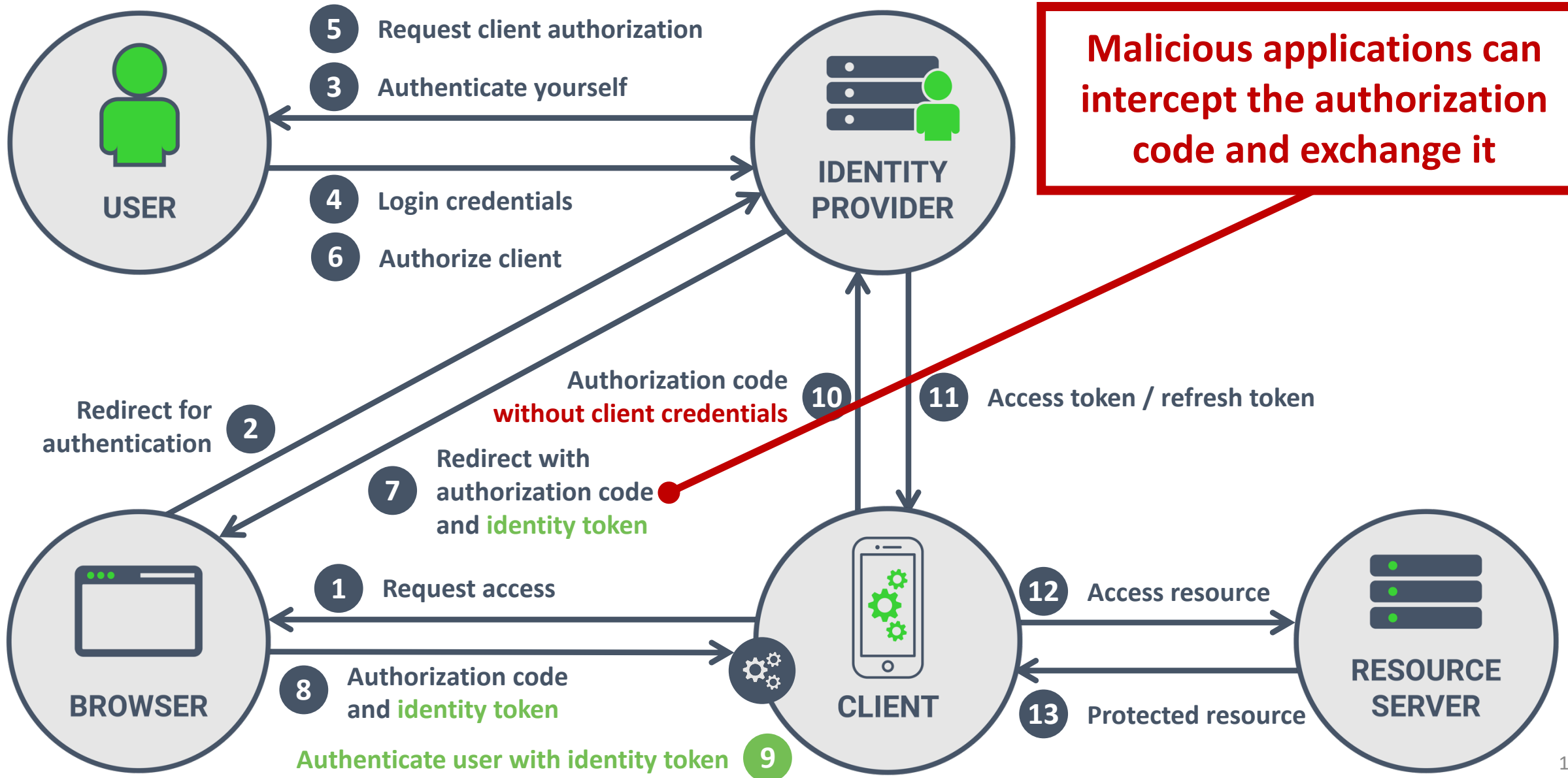
*The confidential client can authenticate with a TLS certificate*

*The TLS certificate can be used to enable token binding*

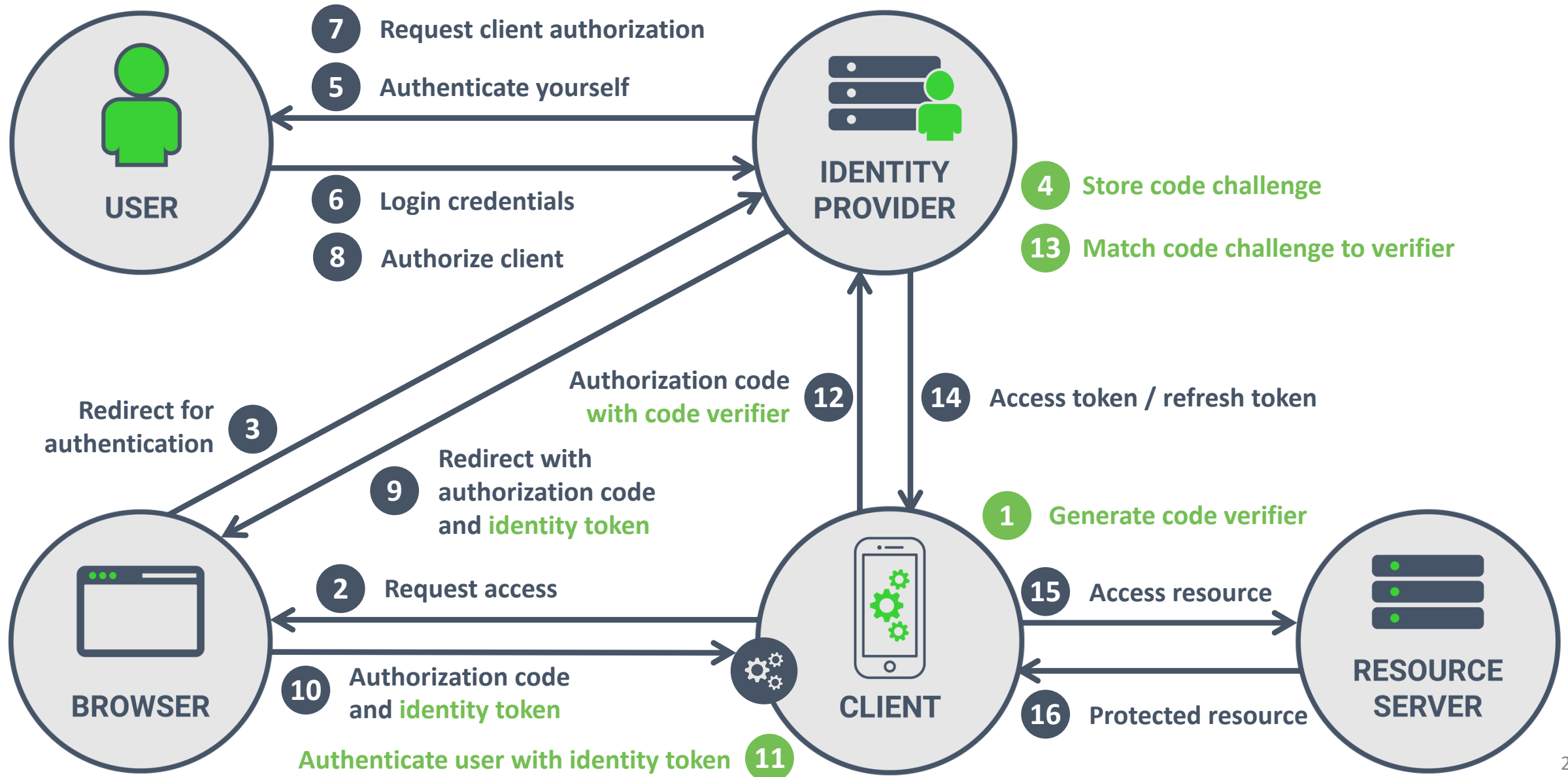
# THE OIDC HYBRID FLOW



# THE OIDC HYBRID FLOW



# THE OIDC HYBRID FLOW WITH PKCE

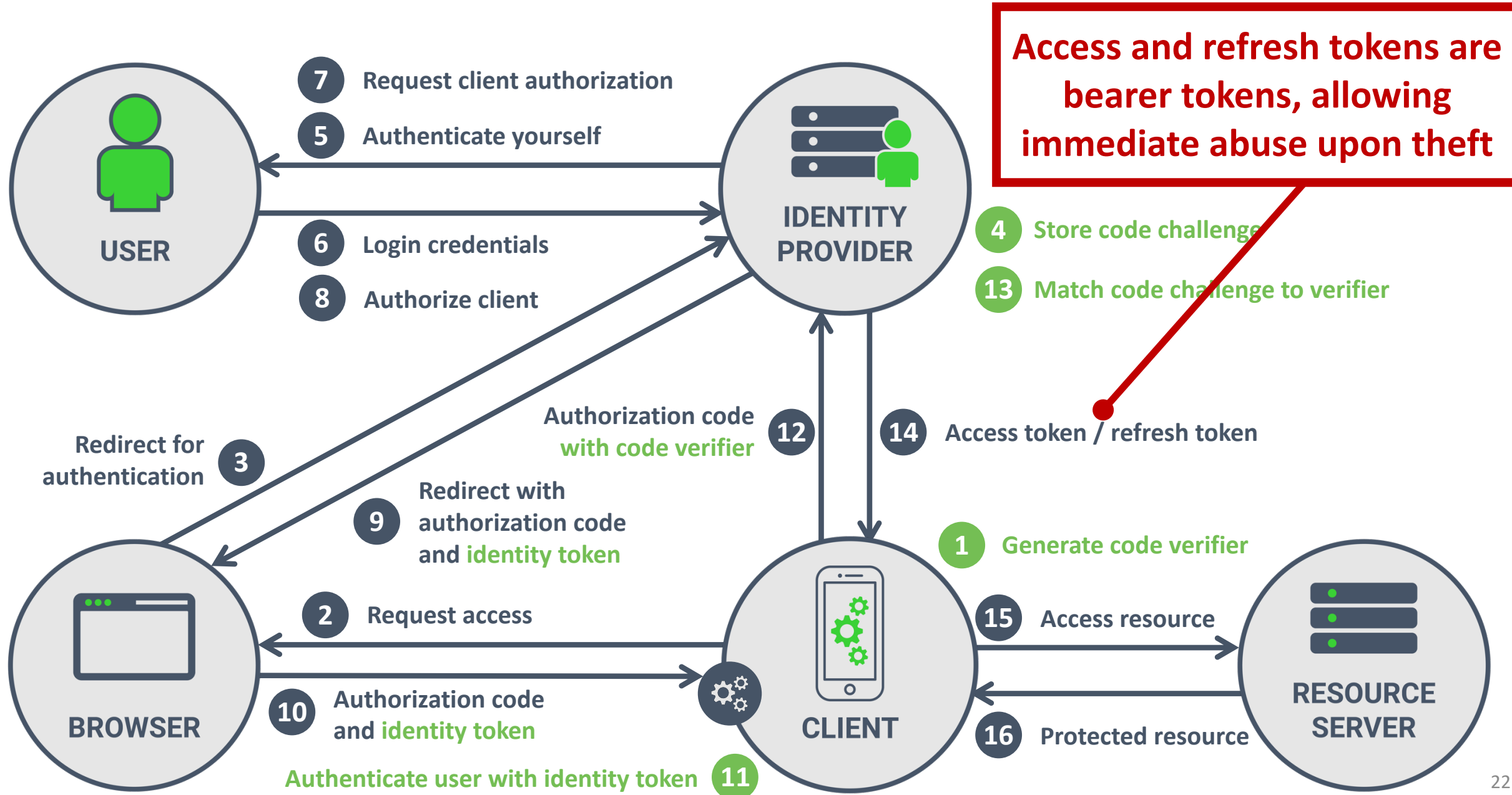


# THE OIDC HYBRID FLOW WITH PKCE

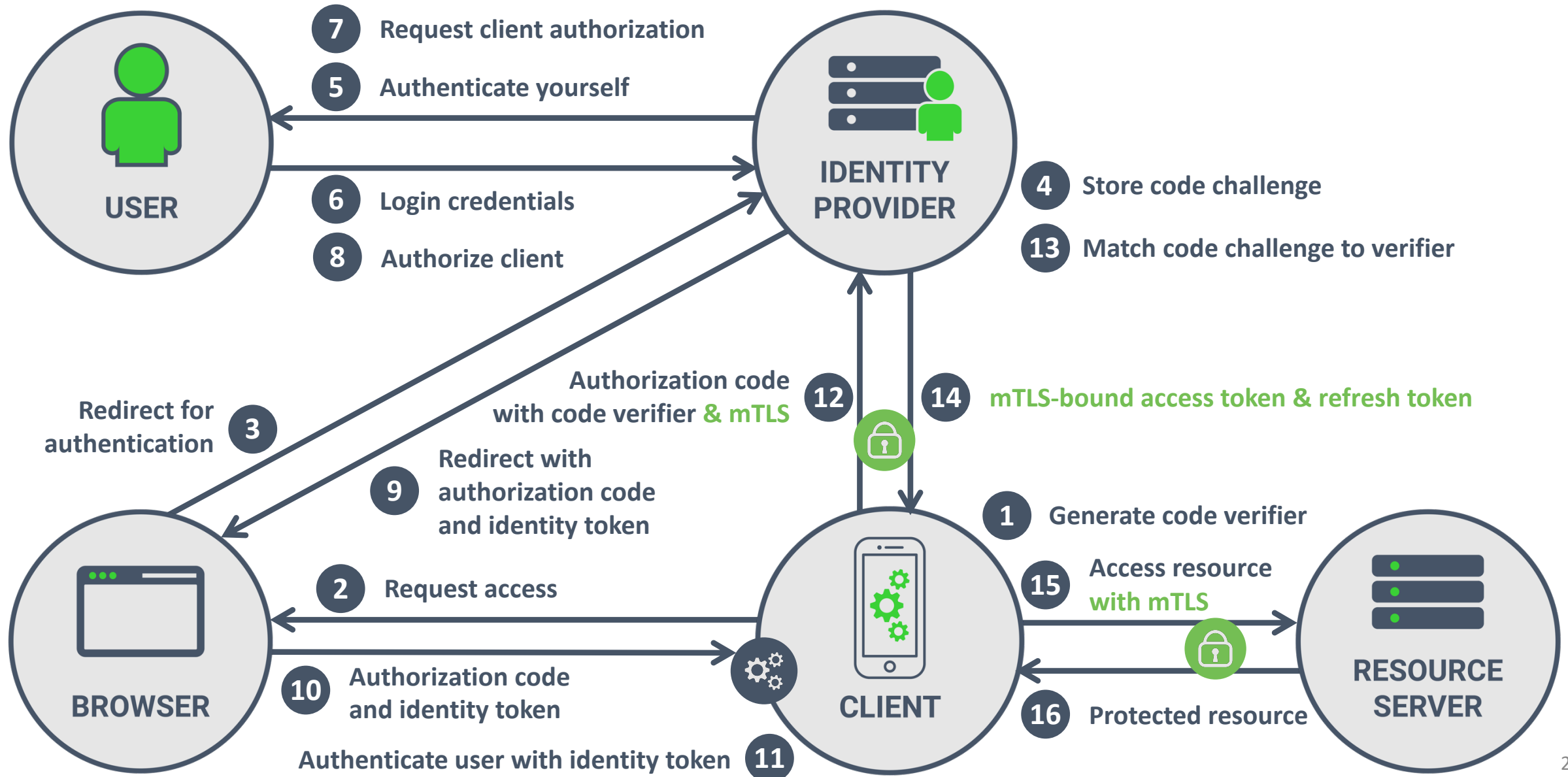
- Mobile applications are public clients
  - The lack of client authentication exposes the authorization code to attacks
- The Proof-Key-for-Code-Exchange addition keeps the authorization code secure
  - PKCE essentially acts as a one-time password for each individual client
  - Prevents the abuse of a stolen authorization code
- Mobile applications can use refresh tokens if they store them securely
  - Refresh tokens do not require authentication, so are bearer tokens
  - Only good place to store is in the OS's secure application storage



# THE DANGER OF BEARER TOKENS



# BINDING TOKENS TO TLS CERTIFICATES ON PUBLIC CLIENTS



# PROOF-OF-POSSESSION IN MOBILE CLIENTS



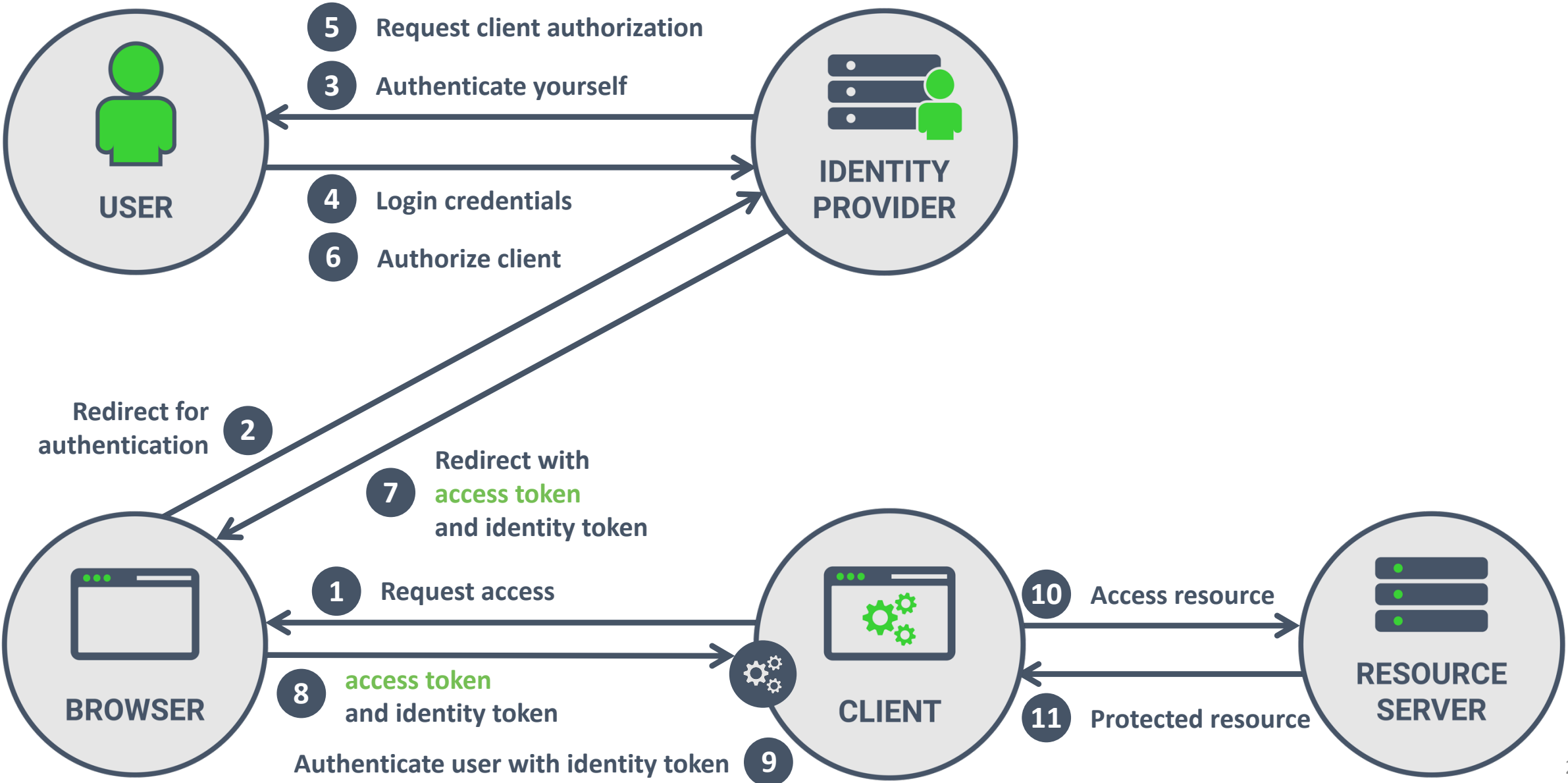
*Each client instance generates its own certificate*

*The client uses the self-signed certificate during TLS connections*

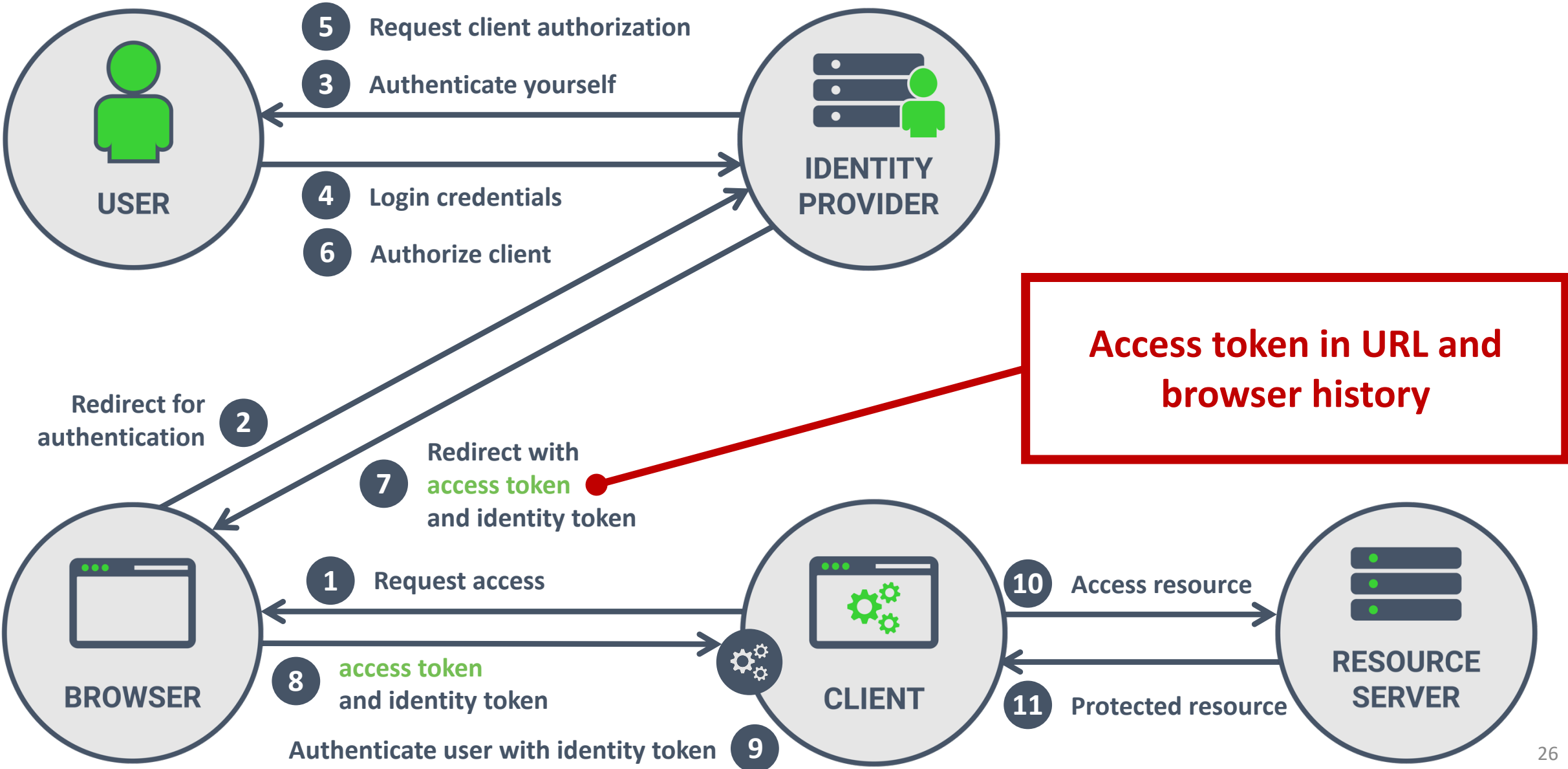
*The authorization server ties the tokens to the client certificate*



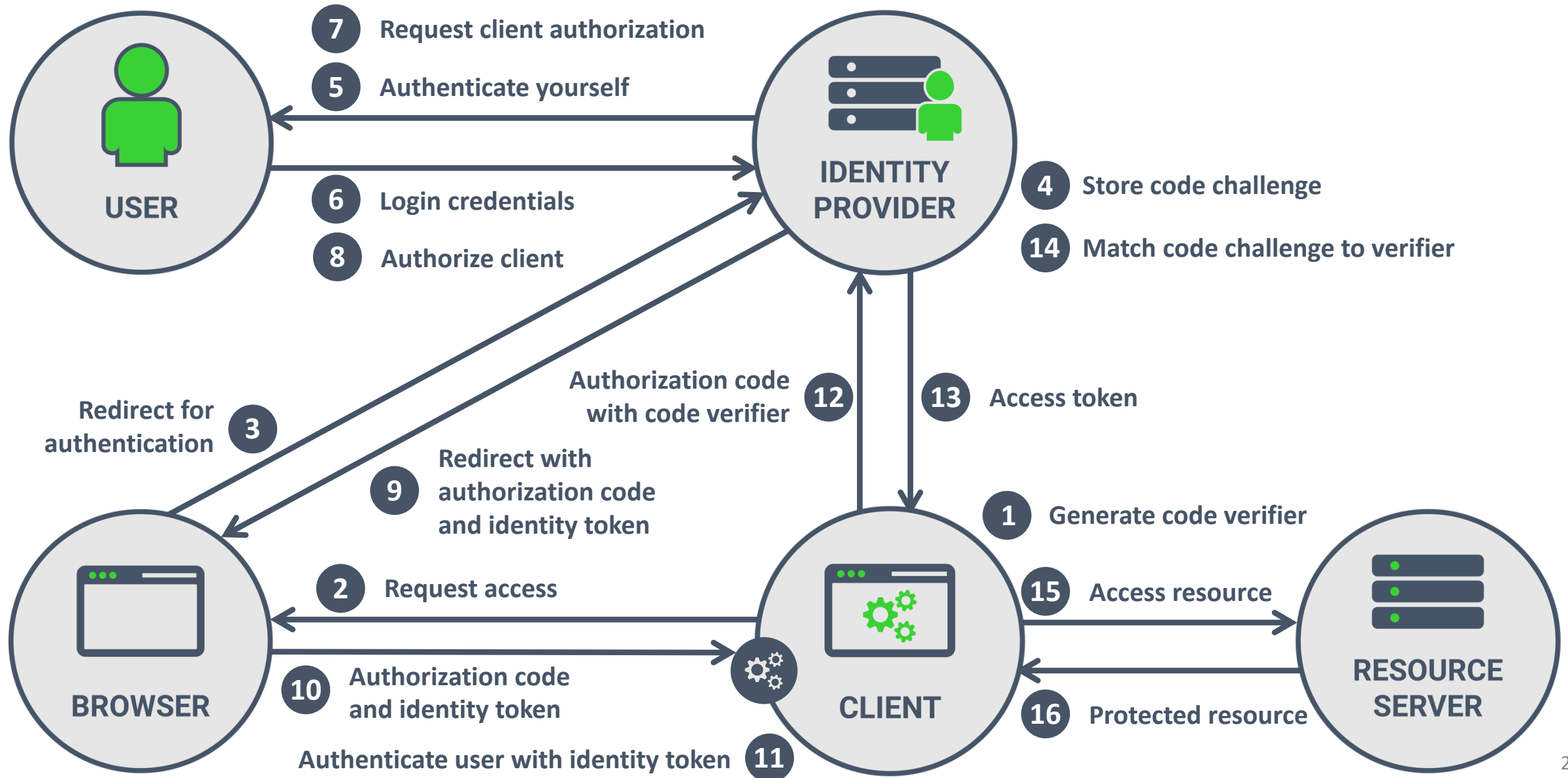
# THE OIDC IMPLICIT FLOW



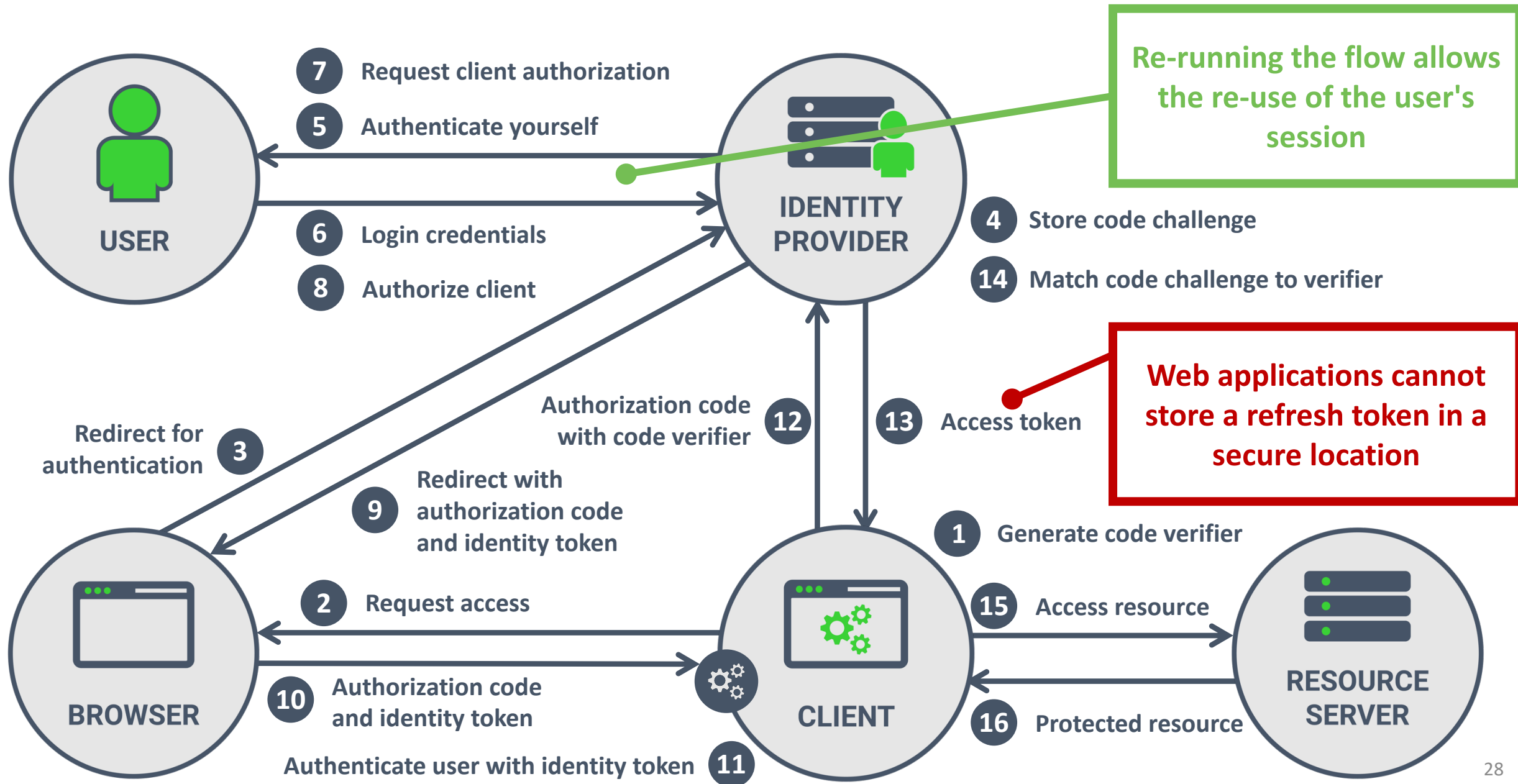
# THE OIDC IMPLICIT FLOW



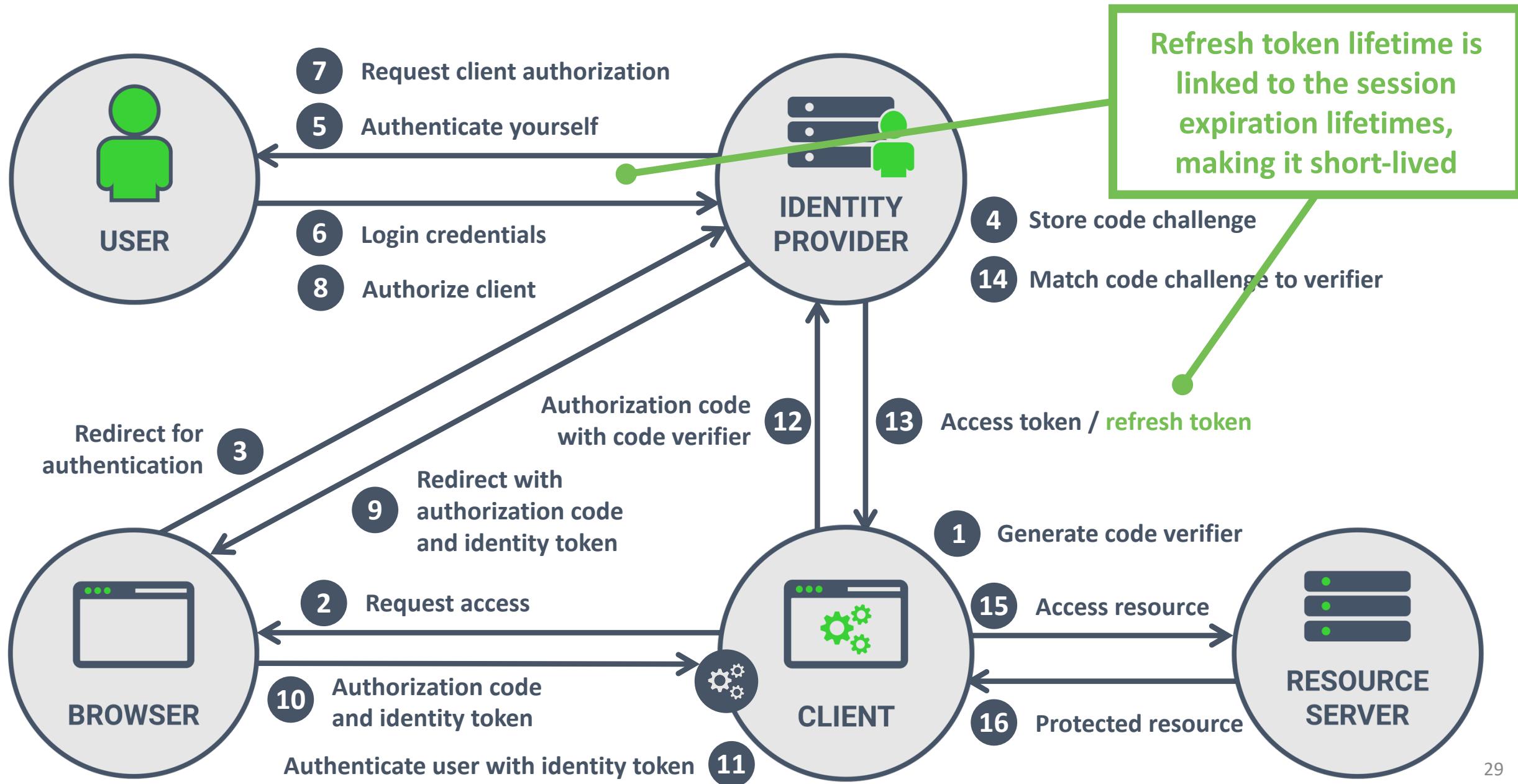
# THE OIDC HYBRID FLOW WITH PKCE



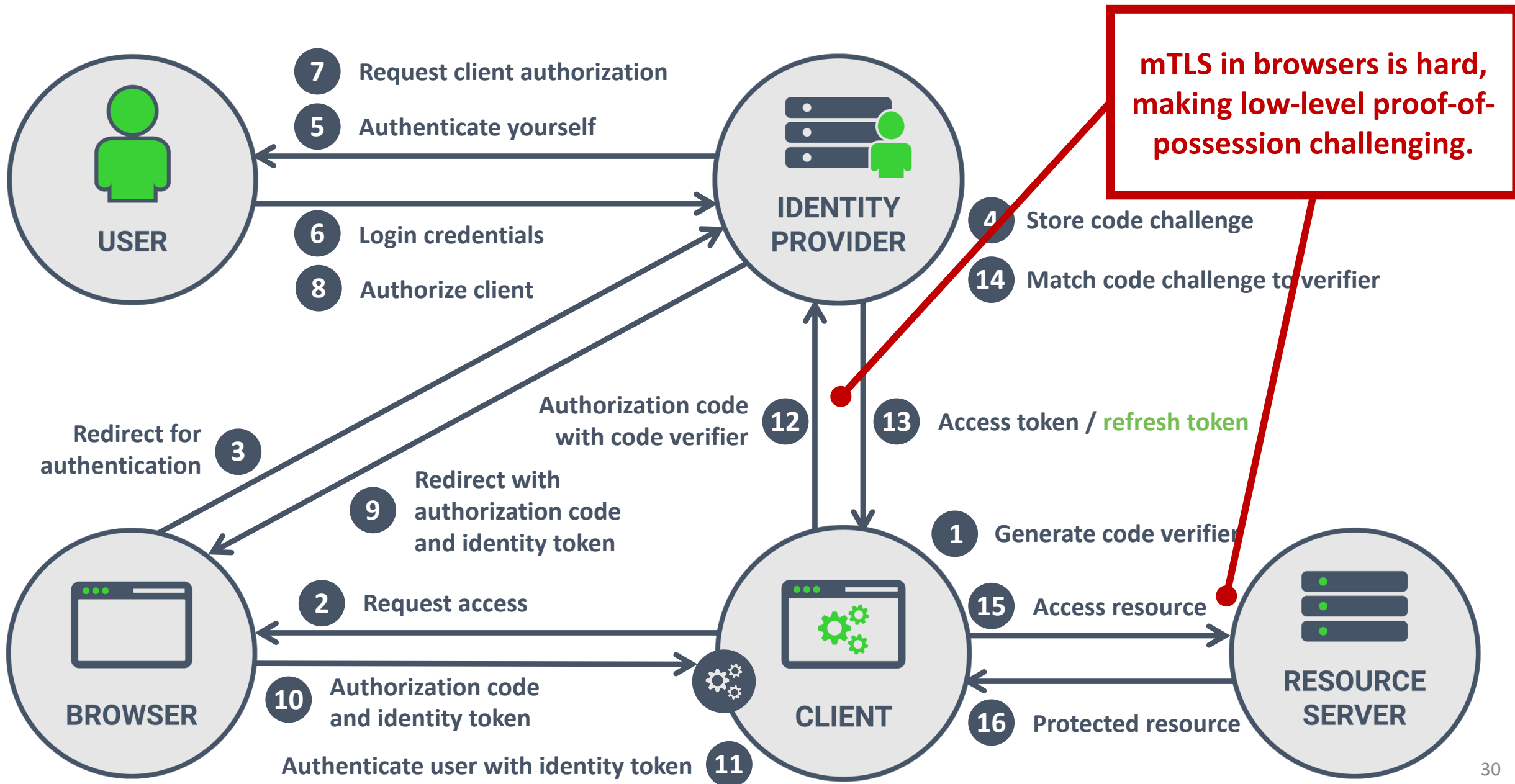
# THE OIDC HYBRID FLOW WITH PKCE



# THE OIDC HYBRID FLOW WITH PKCE



# THE OIDC HYBRID FLOW WITH PKCE



# WEB SECURITY IS HARD



*The Hybrid flow with PKCE is recommended (Implicit flow is still OK)*

*Refresh tokens cannot be used, unless they are short-lived*

*PoP tokens for web applications require application-level code*



# REFERENCES

Proof Key for Code Exchange by OAuth Public Clients

<https://tools.ietf.org/html/rfc7636>

OAuth 2.0 Security Best Current Practice

<https://tools.ietf.org/html/draft-ietf-oauth-security-topics-13>

OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens

<https://tools.ietf.org/html/draft-ietf-oauth-mtls-17>

OAuth 2.0 Demonstration of Proof-of-Possession at the Application Layer

<https://tools.ietf.org/html/draft-fett-oauth-dpop-00>





# FREE SECURITY CHEAT SHEETS FOR MODERN APPLICATIONS

this direct download link.' At the bottom, there is a section titled 'JSON Web Tokens (JWT)' with a subtext: 'JSON Web Tokens (JWTs) have become extremely popular. JWTs seem deceptively simple. However, to ensure their security properties, they depend on complex and often misunderstood concepts. This cheat sheet focuses on the underlying concepts. The cheat sheet covers essential knowledge for every developer producing or consuming JWTs.'" data-bbox="243 117 769 868"/>

Security cheat sheets

Keeping track of security best practices while building applications is challenging. These concise and to-the-point cheat sheets outline best practices for building modern and secure web applications.

## Angular and the OWASP top 10

The OWASP top 10 is one of the most influential security documents of all time. But how do these top 10 vulnerabilities resonate in an Angular application? This cheat sheet offers practical tips on five relevant items from the OWASP top 10.

[SUBSCRIBE TO THE CHEAT SHEET SERIES](#)

When subscribing, you can choose to only receive cheat sheet updates, and nothing else. If you do not want to receive updates and new cheat sheets via email, you can use [this direct download link](#).

## JSON Web Tokens (JWT)

JSON Web Tokens (JWTs) have become extremely popular. JWTs seem deceptively simple. However, to ensure their security properties, they depend on complex and often misunderstood concepts. This cheat sheet focuses on the underlying concepts. The cheat sheet covers essential knowledge for every developer producing or consuming JWTs.





March 9<sup>th</sup> – 13<sup>th</sup>, 2020  
Leuven, Belgium

A **week-long course** on Secure Application Development

Taught by **experts** from around the world

**38** in-depth lectures and **3** one-day workshops

<https://secappdev.org>

*A yearly initiative from the SecAppDev.org non-profit, since 2005*





# THANK YOU!

*Follow me on Twitter to stay up to date  
on web security best practices*



**@PhilippeDeRyck**