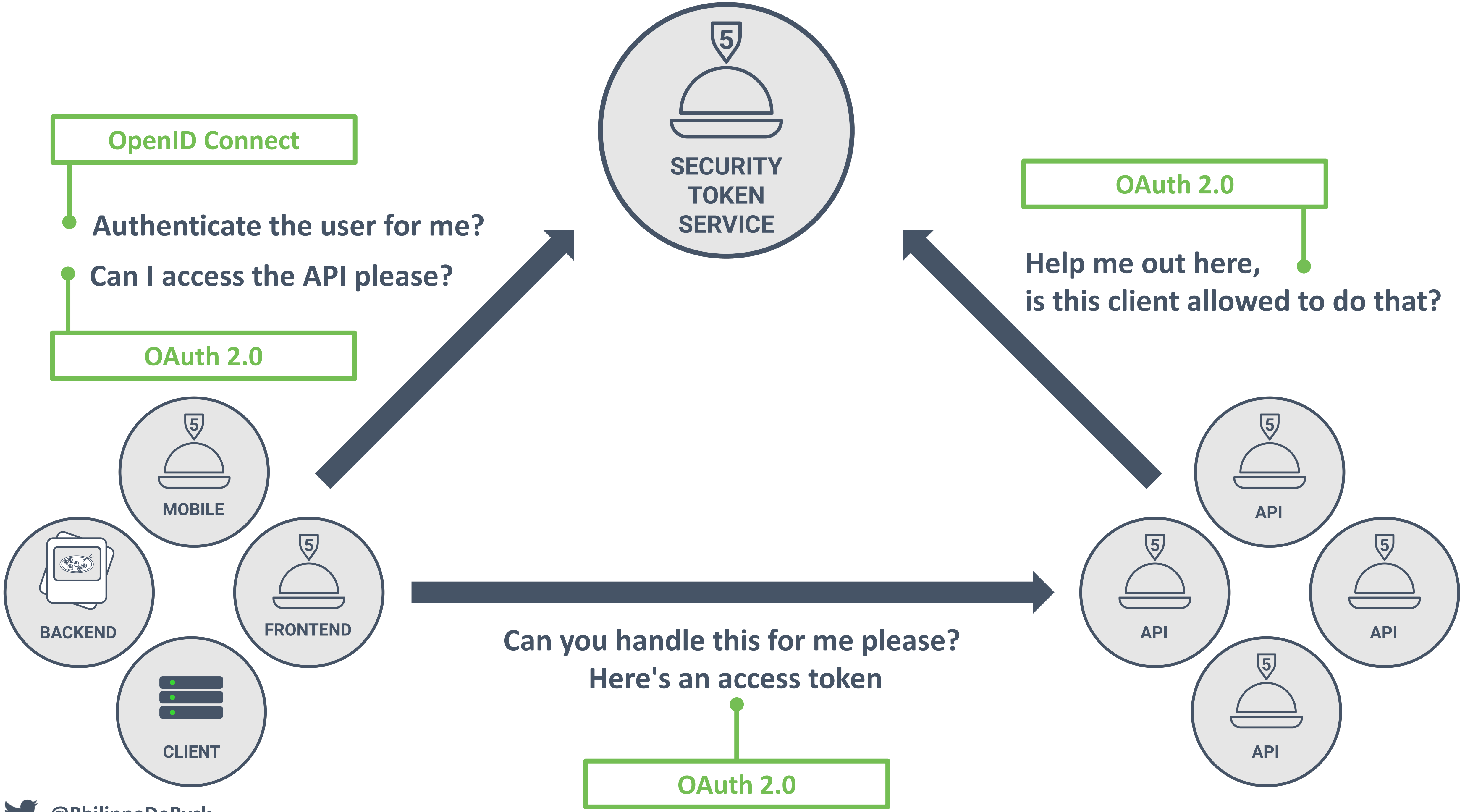
OAuth 2.0 and OpenID Connect for Single Page Applications

Dr. Philippe De Ryck

https://Pragmatic Web Security.com


SECURITY TOKEN SERVICE
OpenID Connect
Authenticate the user for me?
Can I access the API please?
OAuth 2.0
OAuth 2.0
Help me out here, is this client allowed to do that?
MOBILE
BACKEND
FRONTEND
CLIENT
Can you handle this for me please? Here's an access token
OAuth 2.0
API
API
API
API

@PhilippeDeRyck

TERMINOLOGY

This session	OAuth 2.0	OpenID Connect
User	Resource Owner	End-User
API	Resource Server	
Security Token Service (STS)	Authorization Server	OpenID Provider
Client	Client	Relying Party

@PhilippeDeRyck

I am *Dr. Philippe De Ryck*

Founder of Pragmatic Web Security

Google Developer Expert

Auth0 Ambassador

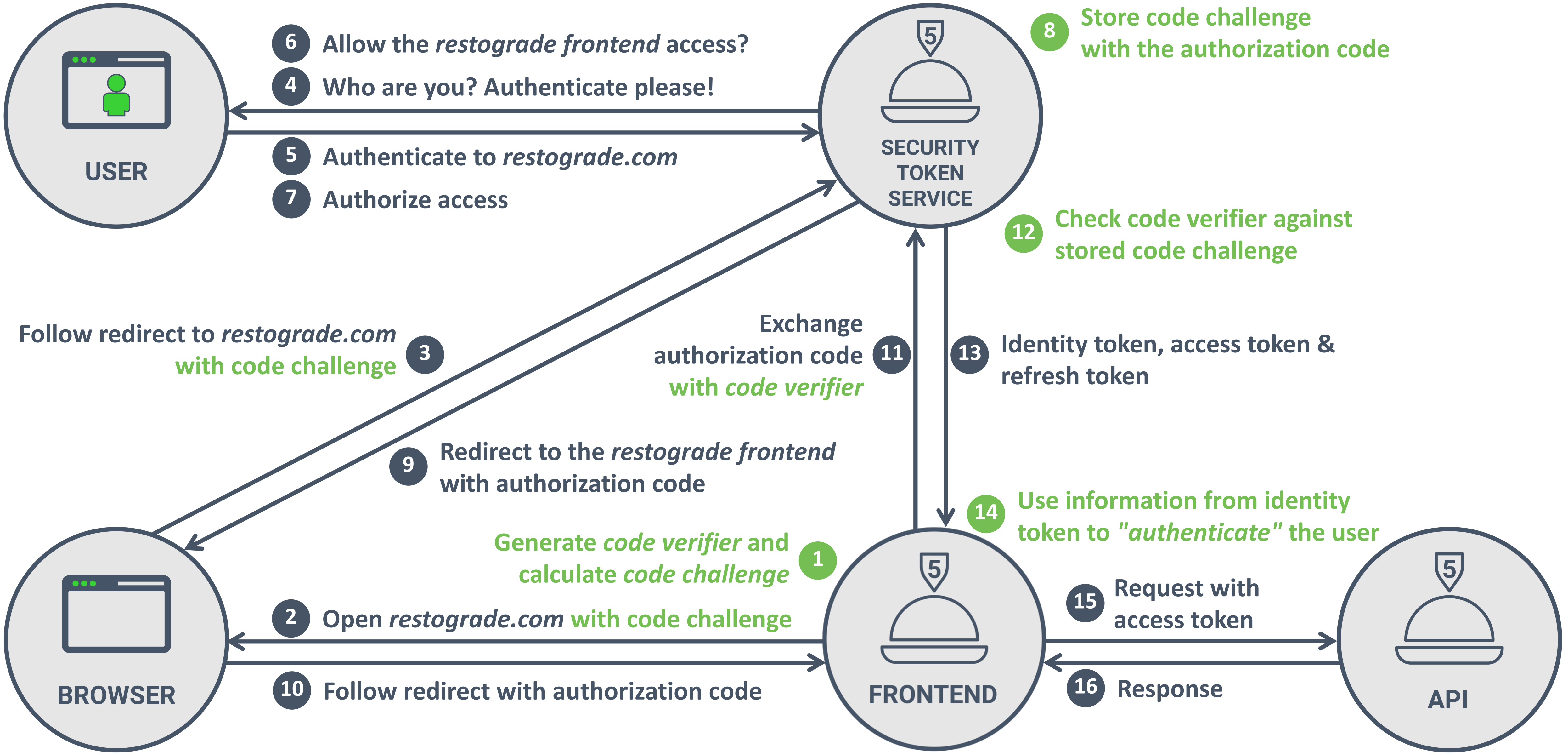
SecAppDev organizer

I help developers with security

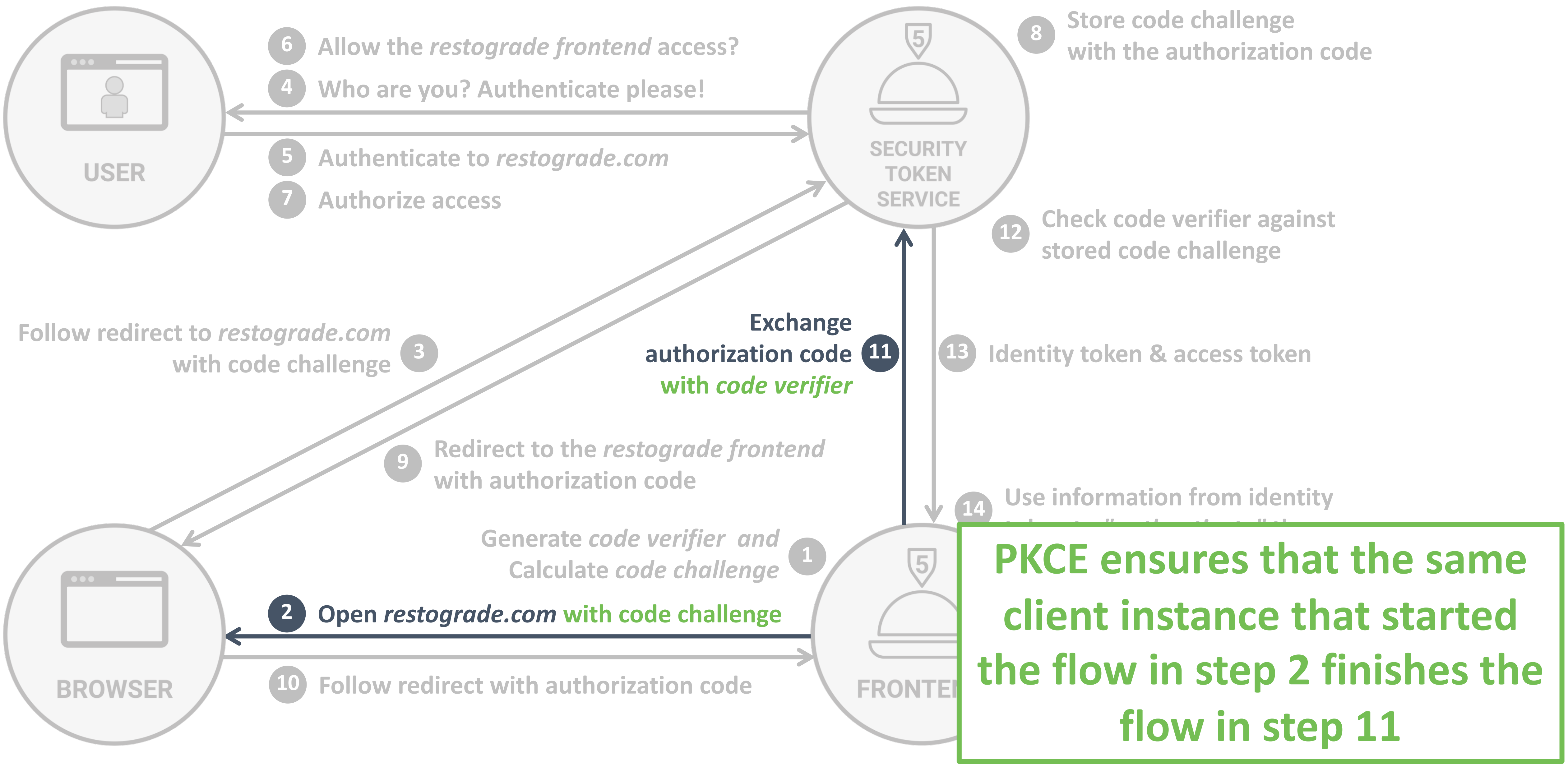
- ✓ Hands-on in-depth security training
- ✓ Advanced online security courses
- ✓ Security advisory services

https://pragmaticwebsecurity.com

The *Authorization Code* flow with PKCE

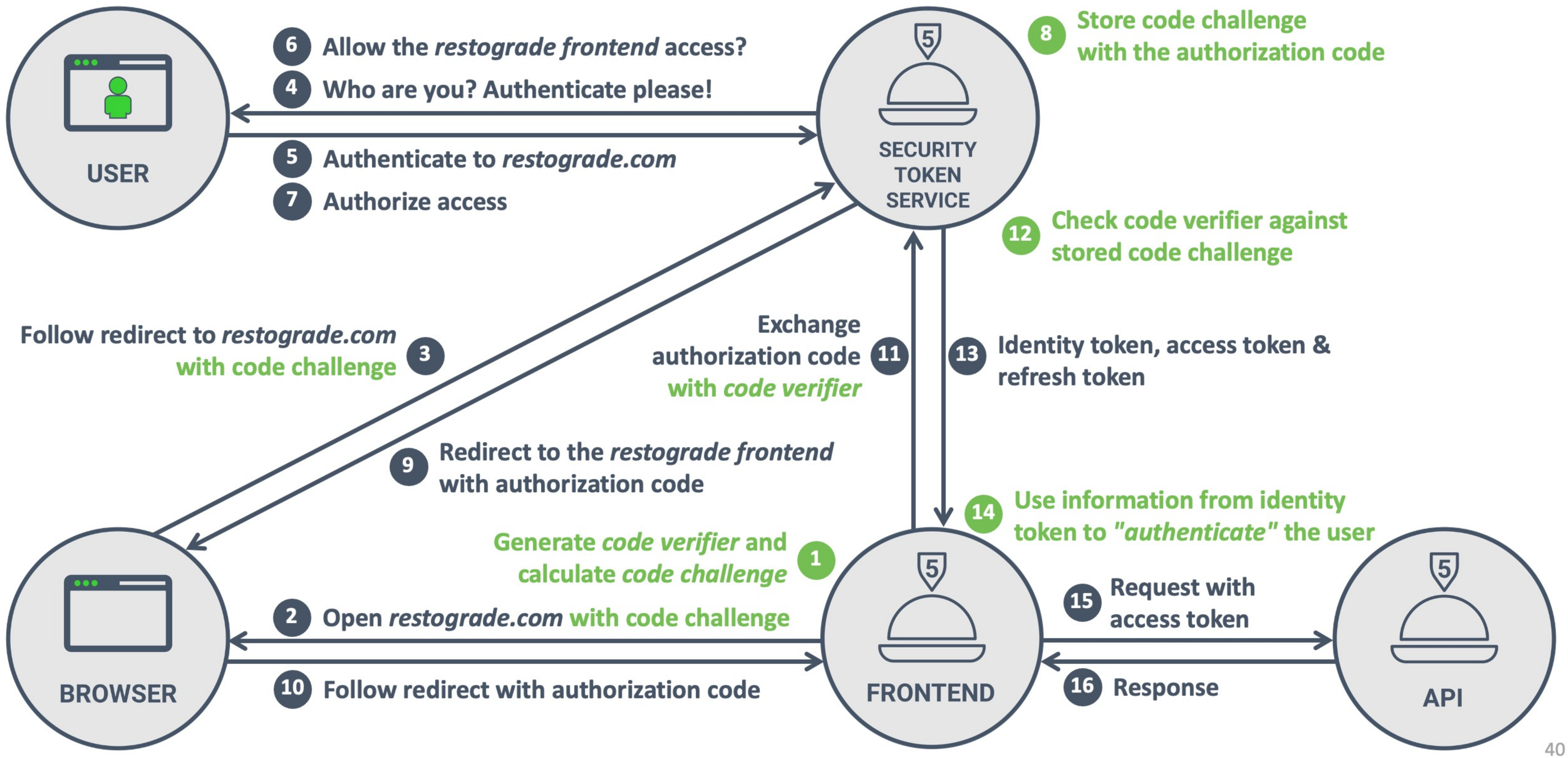

USER
SECURITY TOKEN SERVICE
BROWSER
FRONTEND
API
6 Allow the *restograde frontend* access?
4 Who are you? Authenticate please!
5 Authenticate to *restograde.com*
7 Authorize access
8 Store code challenge with the authorization code
12 Check code verifier against stored code challenge
3 Follow redirect to *restograde.com* with code challenge
11 Exchange authorization code with *code verifier*
13 Identity token, access token & refresh token
9 Redirect to the *restograde frontend* with authorization code
14 Use information from identity token to *"authenticate"* the user
1 Generate *code verifier* and calculate *code challenge*
2 Open *restograde.com* with code challenge
15 Request with access token
10 Follow redirect with authorization code
16 Response

THE *AUTHORIZATION CODE* FLOW WITH PKCE


USER
SECURITY TOKEN SERVICE
BROWSER
FRONTEN
6 Allow the *restograde frontend* access?
4 Who are you? Authenticate please!
5 Authenticate to *restograde.com*
7 Authorize access
8 Store code challenge with the authorization code
12 Check code verifier against stored code challenge
Follow redirect to *restograde.com* with code challenge 3
Exchange authorization code *with code verifier* 11
13 Identity token & access token
9 Redirect to the *restograde frontend* with authorization code
14 Use information from identity
Generate *code verifier* and Calculate *code challenge* 1
2 Open *restograde.com* with code challenge
10 Follow redirect with authorization code

PKCE ensures that the same client instance that started the flow in step 2 finishes the flow in step 11

The *Authorization Code* flow with PKCE



USER
SECURITY TOKEN SERVICE
BROWSER
FRONTEND
API
1 Generate *code verifier* and calculate *code challenge*
2 Open *restograde.com* with code challenge
3 Follow redirect to *restograde.com* with code challenge
4 Who are you? Authenticate please!
5 Authenticate to *restograde.com*
6 Allow the *restograde frontend* access?
7 Authorize access
8 Store code challenge with the authorization code
9 Redirect to the *restograde frontend* with authorization code
10 Follow redirect with authorization code
11 Exchange authorization code with *code verifier*
12 Check code verifier against stored code challenge
13 Identity token, access token & refresh token
14 Use information from identity token to *"authenticate"* the user
15 Request with access token
16 Response

2 3 *The redirect URI*

```
https://sts.restograde.com/authorize
  ?response_type=code
  &client_id=lY5g0BKB7Mow4yDlb6rdGPsO2i1g7Osv
  &scope=read
  &redirect_uri=https://app.restograde.com/callback
  &state=s0wzojm2w8c23xzprkk6
  &code_challenge=JhEN0Amnj7B…Wh5PxWitZYK1woWh5PxWitZY
  &code_challenge_method=S256
```

Indicates the *authorization code flow*

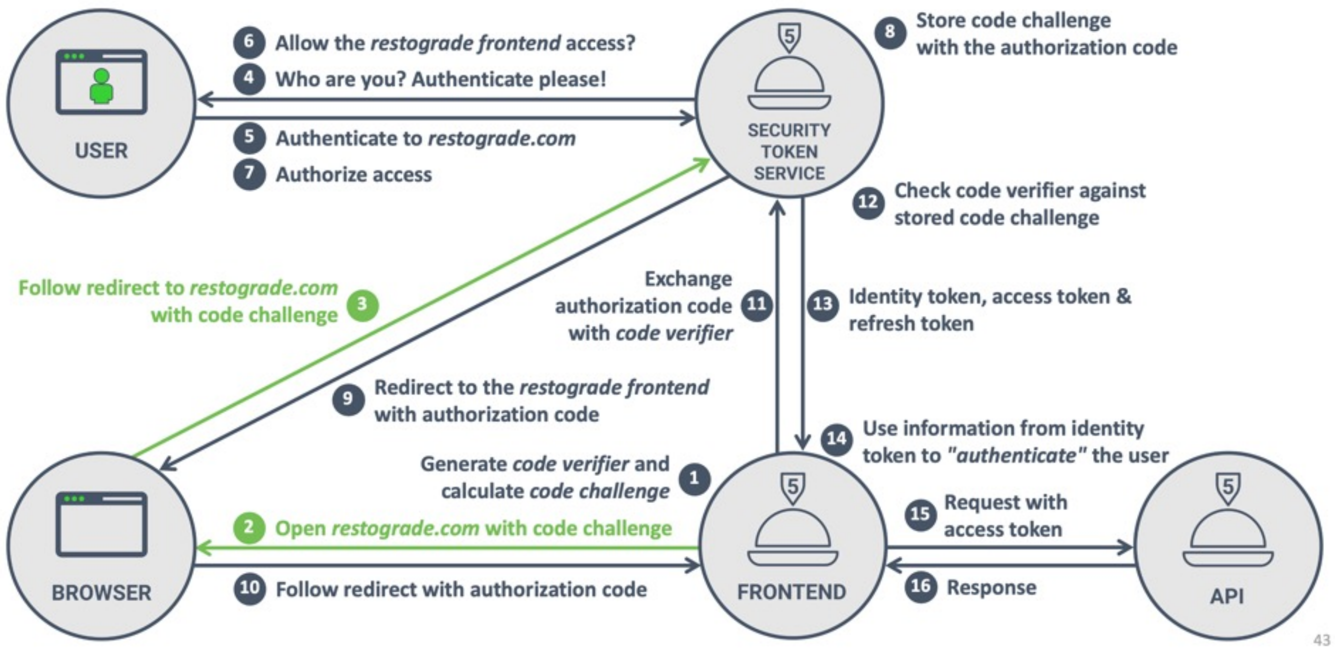
The client requesting access

Where the STS should send the code

The PKCE code challenge

The PKCE hash function

THE *AUTHORIZATION CODE* FLOW WITH PKCE

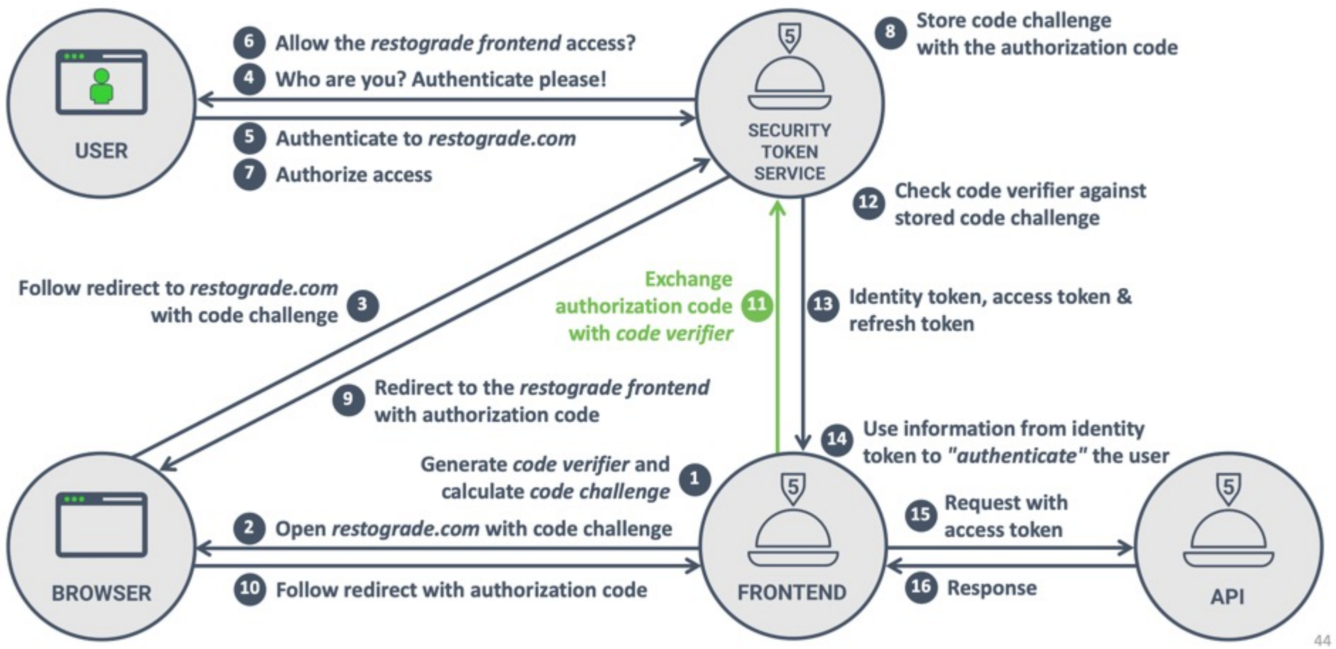

USER
SECURITY TOKEN SERVICE
BROWSER
FRONTEND
API
6 Allow the *restograde frontend* access?
4 Who are you? Authenticate please!
5 Authenticate to *restograde.com*
7 Authorize access
8 Store code challenge with the authorization code
12 Check code verifier against stored code challenge
Follow redirect to *restograde.com* with code challenge 3
Exchange authorization code with *code verifier* 11
13 Identity token, access token & refresh token
9 Redirect to the *restograde frontend* with authorization code
14 Use information from identity token to "*authenticate*" the user
Generate *code verifier* and calculate *code challenge* 1
2 Open *restograde.com* with code challenge
10 Follow redirect with authorization code
15 Request with access token
16 Response

11 *The request to exchange the authorization code*

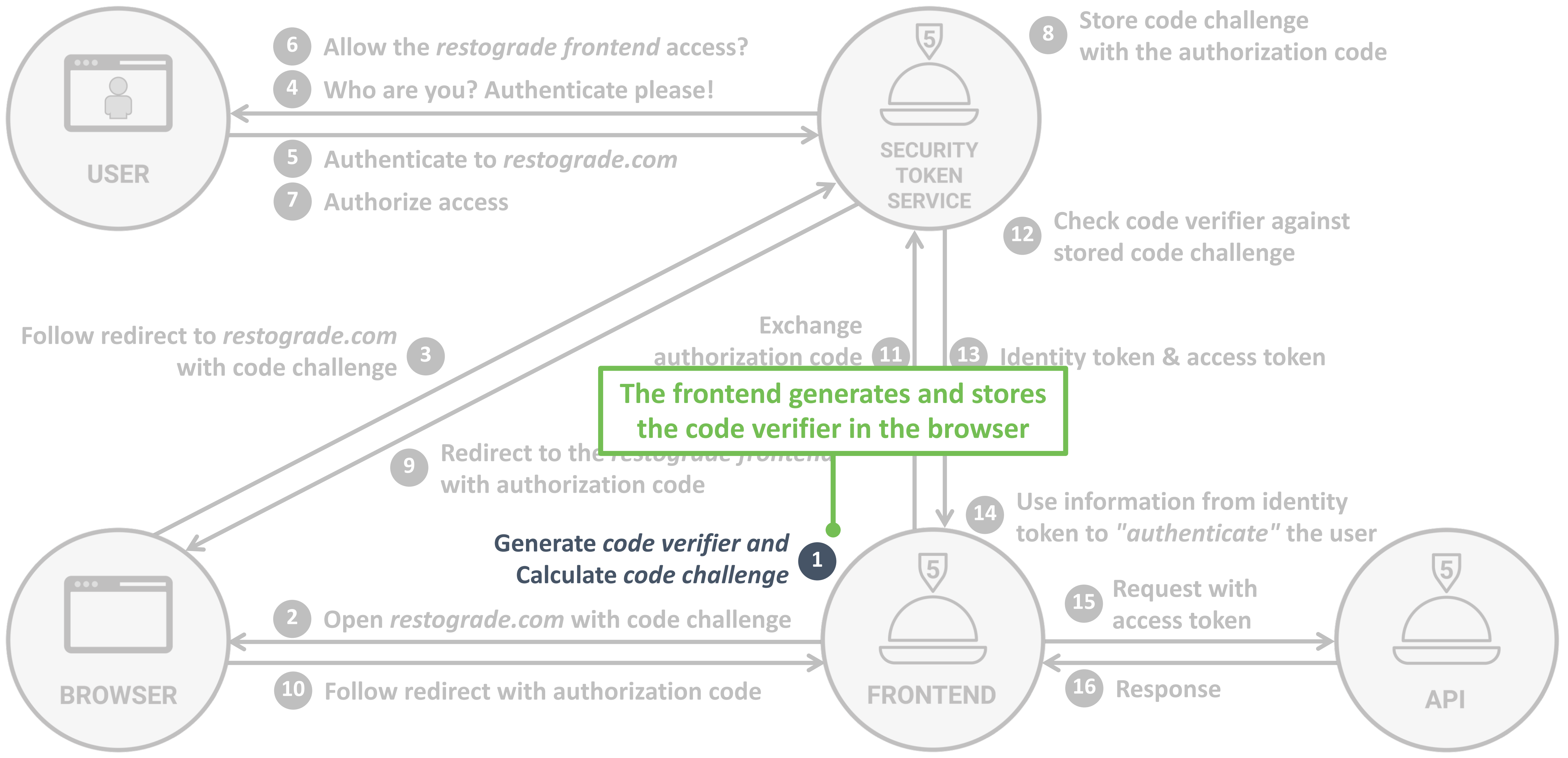
```
POST /oauth/token
Host: sts.restograde.com

 grant_type=authorization_code
&client_id=lY5g0BKB7Mow4yDlb6rdGPsO2i1g7Osv
&redirect_uri=https://app.restograde.com/callback
&code=SplxlOBeZQQYbYS6WxSbIA
&code_verifier=lT5q6nbPQRtdj…~IUdkErVDFG.fF4z7CzCxo
```

Indicates the code exchange request
The client exchanging the code
The redirect URI used before
The code received in step 10
The code verifier from step 1

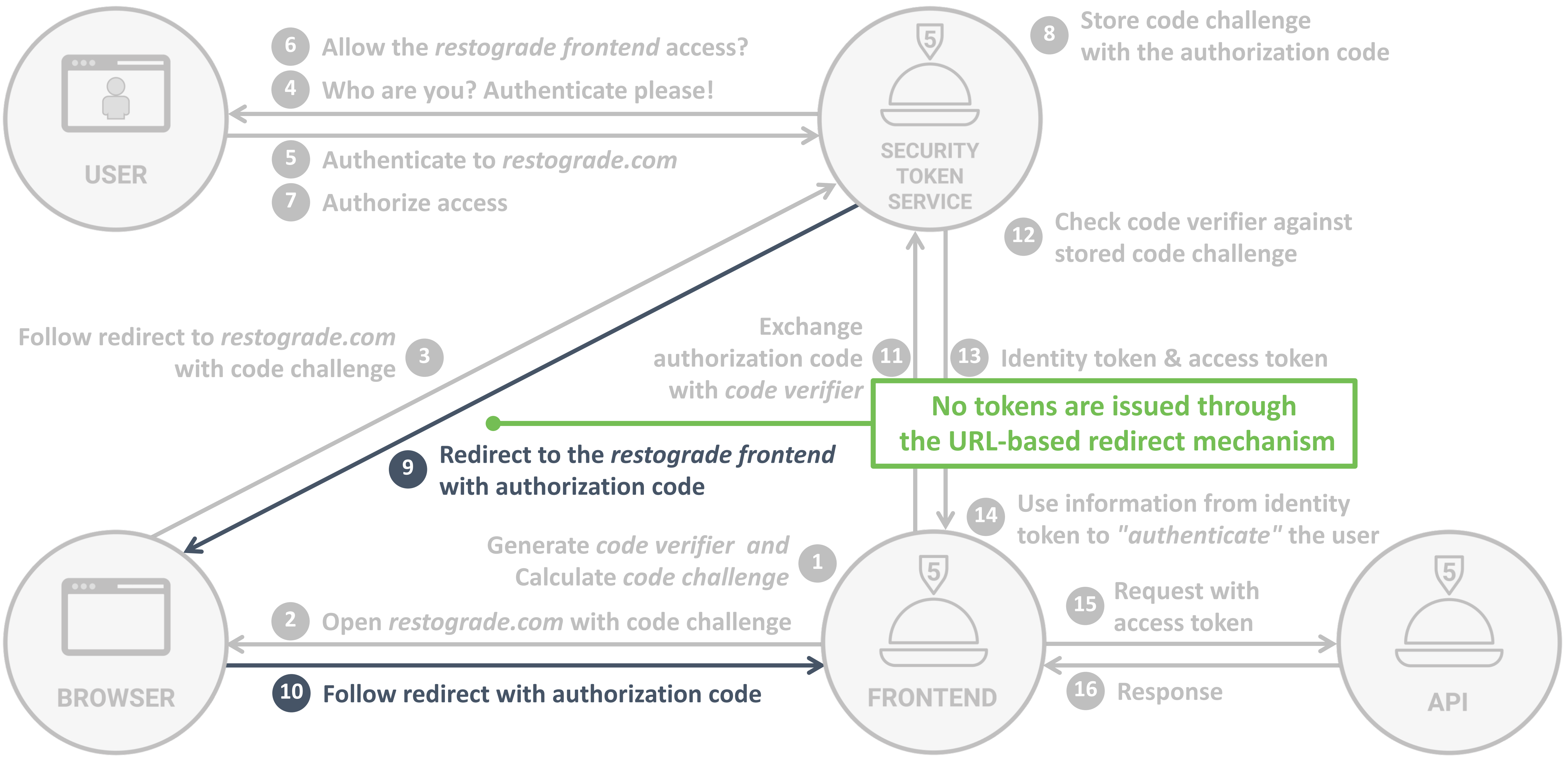
THE *AUTHORIZATION CODE* FLOW WITH PKCE


USER
SECURITY TOKEN SERVICE
BROWSER
FRONTEND
API
6 Allow the *restograde frontend* access?
4 Who are you? Authenticate please!
5 Authenticate to *restograde.com*
7 Authorize access
8 Store code challenge with the authorization code
12 Check code verifier against stored code challenge
Follow redirect to *restograde.com* with code challenge 3
Exchange authorization code with *code verifier* 11
13 Identity token, access token & refresh token
9 Redirect to the *restograde frontend* with authorization code
14 Use information from identity token to "*authenticate*" the user
Generate *code verifier* and calculate *code challenge* 1
2 Open *restograde.com* with code challenge
10 Follow redirect with authorization code
15 Request with access token
16 Response

THE *AUTHORIZATION CODE* FLOW WITH PKCE

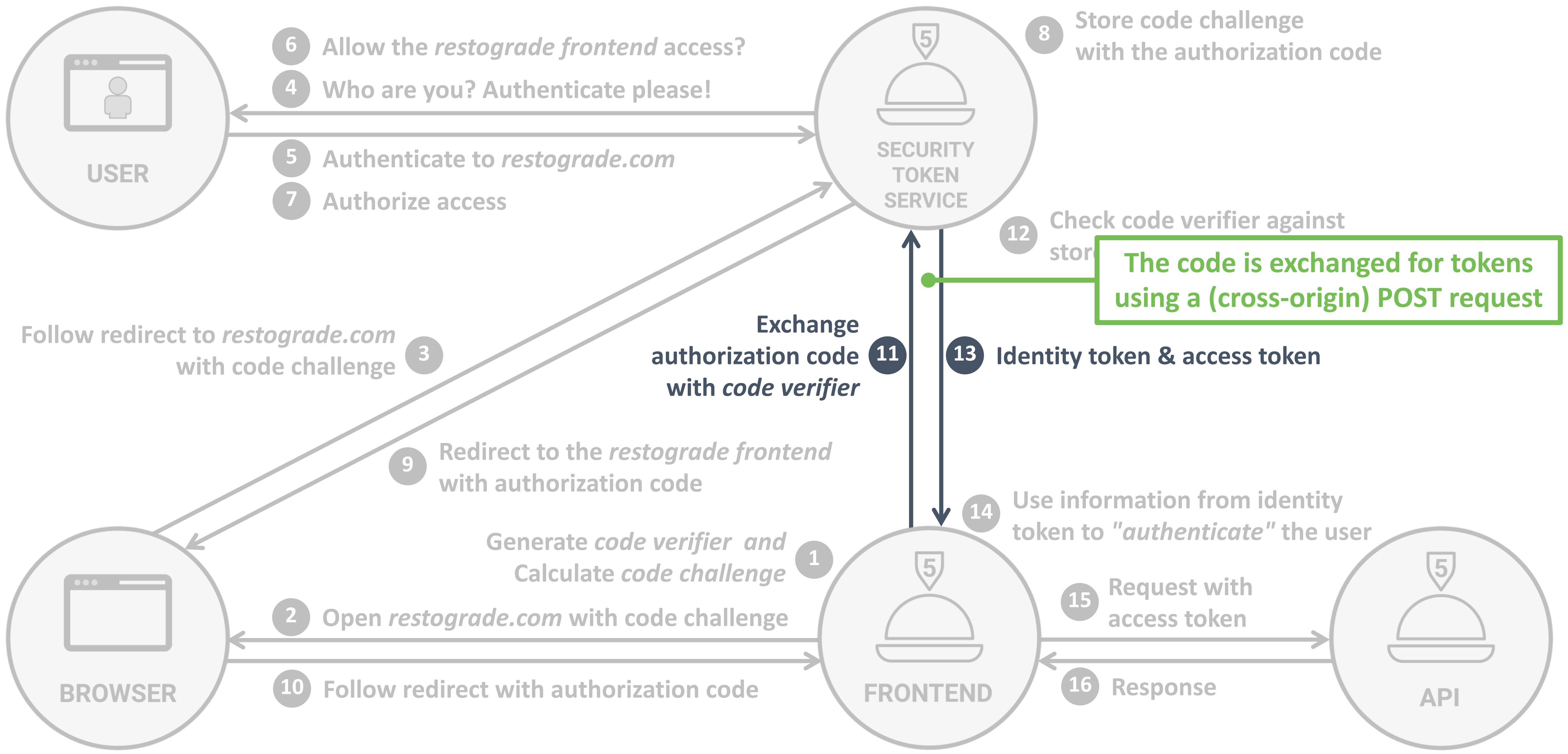

USER
SECURITY TOKEN SERVICE
BROWSER
FRONTEND
API
6 Allow the *restograde frontend* access?
4 Who are you? Authenticate please!
5 Authenticate to *restograde.com*
7 Authorize access
8 Store code challenge with the authorization code
12 Check code verifier against stored code challenge
Follow redirect to *restograde.com* with code challenge 3
Exchange authorization code 11
13 Identity token & access token
The frontend generates and stores the code verifier in the browser
9 Redirect to the *restograde frontend* with authorization code
14 Use information from identity token to *"authenticate"* the user
Generate *code verifier and* Calculate *code challenge* 1
15 Request with access token
2 Open *restograde.com* with code challenge
10 Follow redirect with authorization code
16 Response

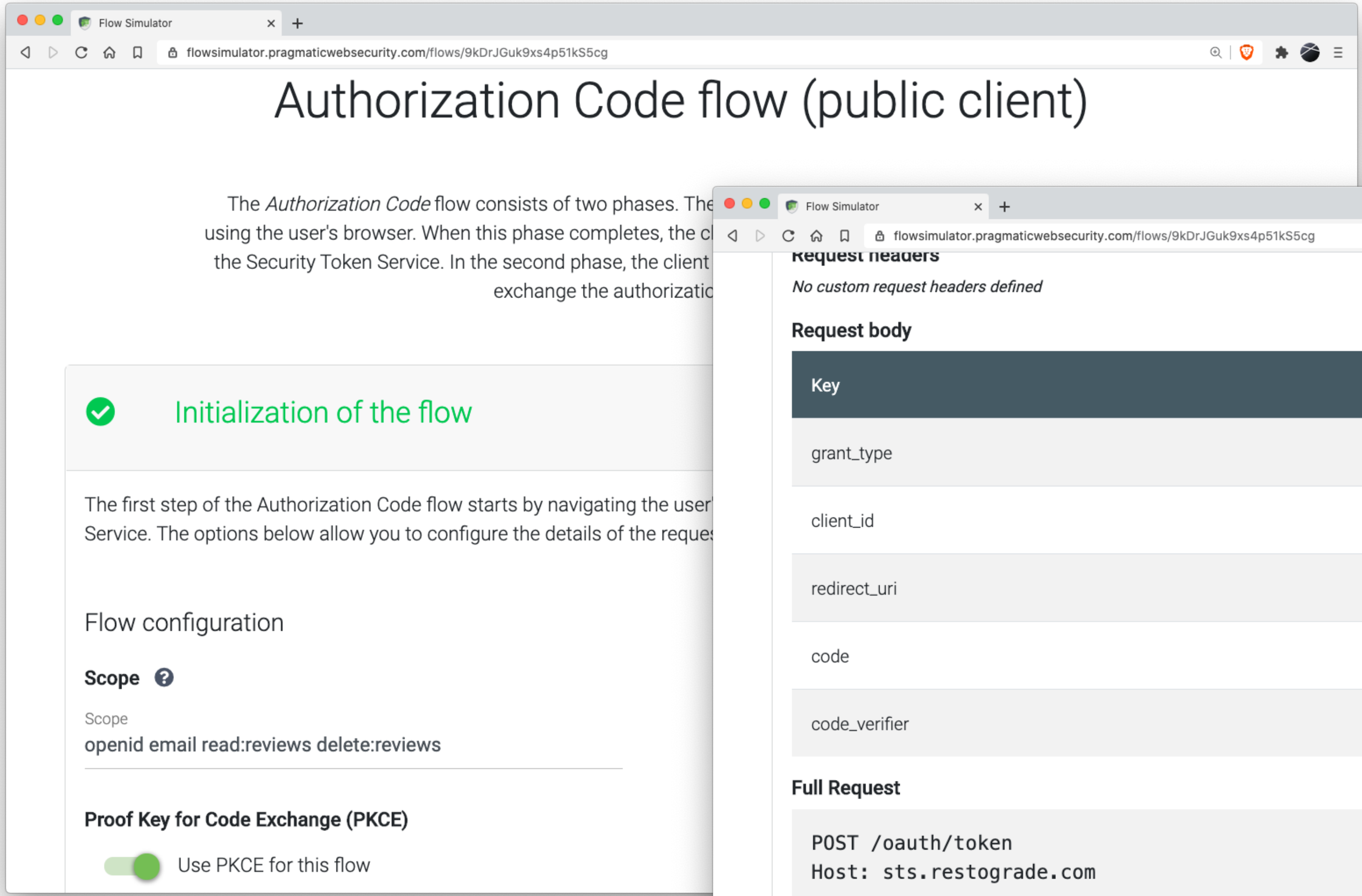
The *Authorization Code* flow with PKCE



USER
SECURITY TOKEN SERVICE
BROWSER
FRONTEND
API
6 Allow the *restograde frontend* access?
4 Who are you? Authenticate please!
5 Authenticate to *restograde.com*
7 Authorize access
8 Store code challenge with the authorization code
12 Check code verifier against stored code challenge
Follow redirect to *restograde.com* with code challenge 3
Exchange authorization code with *code verifier* 11
13 Identity token & access token
No tokens are issued through the URL-based redirect mechanism
9 Redirect to the *restograde frontend* with authorization code
14 Use information from identity token to *"authenticate"* the user
Generate *code verifier* and Calculate *code challenge* 1
15 Request with access token
2 Open *restograde.com* with code challenge
10 Follow redirect with authorization code
16 Response

THE *AUTHORIZATION CODE* FLOW WITH PKCE


USER
6 Allow the *restograde frontend* access?
4 Who are you? Authenticate please!
5 Authenticate to *restograde.com*
7 Authorize access
SECURITY TOKEN SERVICE
8 Store code challenge with the authorization code
12 Check code verifier against stor
The code is exchanged for tokens using a (cross-origin) POST request
Follow redirect to *restograde.com* with code challenge 3
Exchange authorization code with *code verifier* 11
13 Identity token & access token
9 Redirect to the *restograde frontend* with authorization code
14 Use information from identity token to "*authenticate*" the user
Generate *code verifier* and Calculate *code challenge* 1
2 Open *restograde.com* with code challenge
10 Follow redirect with authorization code
BROWSER
FRONTEND
15 Request with access token
16 Response
API



Flow Simulator

flowsimulator.pragmaticwebsecurity.com/flows/9kDrJGuk9xs4p51kS5cg

Authorization Code flow (public client)

The *Authorization Code* flow consists of two phases. The
using the user's browser. When this phase completes, the cl
the Security Token Service. In the second phase, the client
exchange the authorizatio

Initialization of the flow

The first step of the Authorization Code flow starts by navigating the user'
Service. The options below allow you to configure the details of the reques

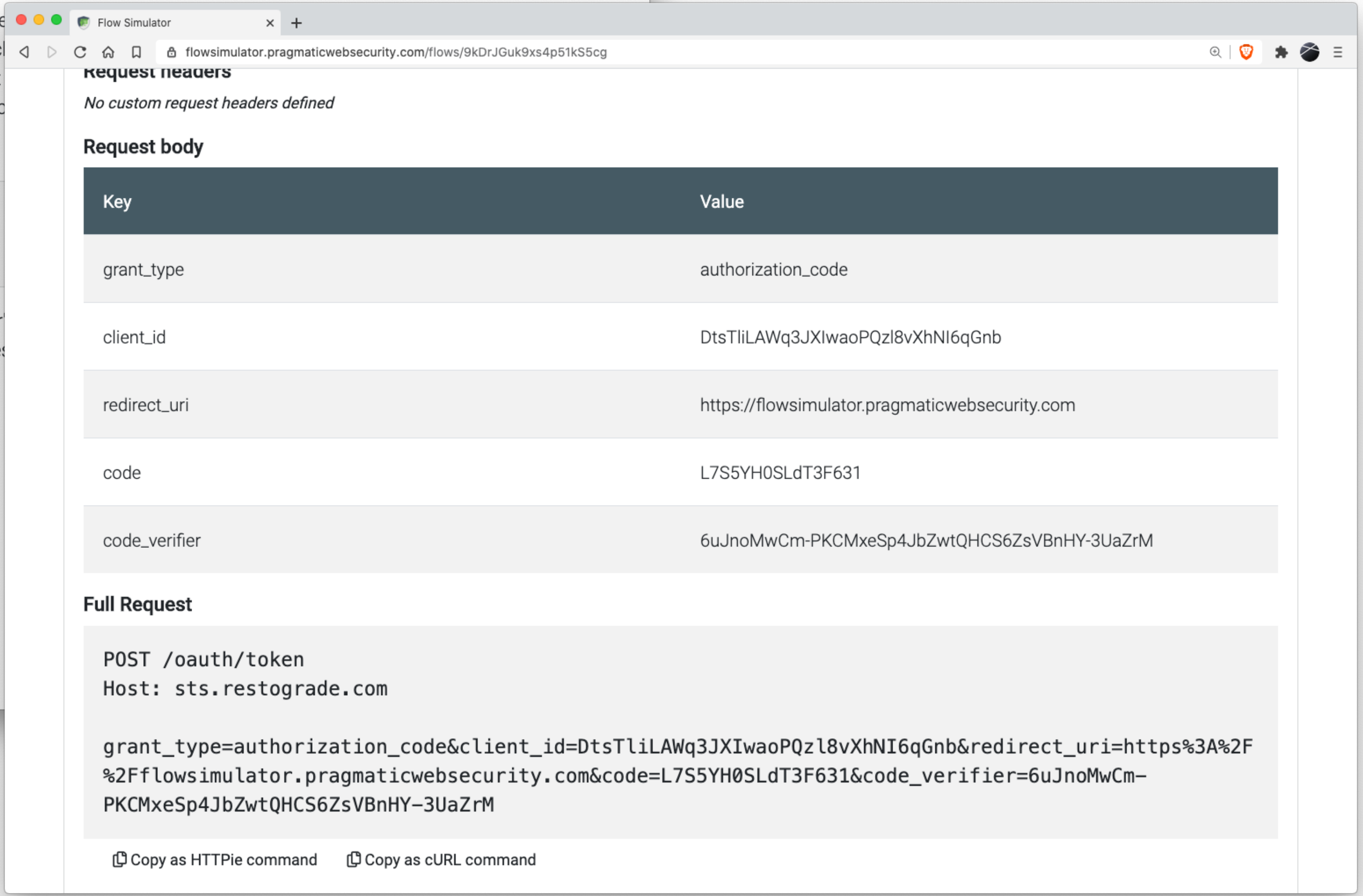
Flow configuration

Scope

Scope
openid email read:reviews delete:reviews

Proof Key for Code Exchange (PKCE)

Use PKCE for this flow



Flow Simulator

flowsimulator.pragmaticwebsecurity.com/flows/9kDrJGuk9xs4p51kS5cg

Request headers

No custom request headers defined

Request body

Key	Value
grant_type	authorization_code
client_id	DtsTliLAWq3JXIwaoPQzl8vXhNI6qGnb
redirect_uri	https://flowsimulator.pragmaticwebsecurity.com
code	L7S5YH0SLdT3F631
code_verifier	6uJnoMwCm-PKCMxeSp4JbZwtQHCS6ZsVBnHY-3UaZrM

Full Request

```
POST /oauth/token
Host: sts.restograde.com

grant_type=authorization_code&client_id=DtsTliLAWq3JXIwaoPQzl8vXhNI6qGnb&redirect_uri=https%3A%2F
%2Fflowsimulator.pragmaticwebsecurity.com&code=L7S5YH0SLdT3F631&code_verifier=6uJnoMwCm-
PKCMxeSp4JbZwtQHCS6ZsVBnHY-3UaZrM
```

Copy as HTTPie command   Copy as cURL command

@PhilippeDeRyck

https://flowsimulator.pragmaticwebsecurity.com

Loading the Auth0 service in an Angular application

```
@NgModule({
  imports: [
    BrowserModule,
    AuthModule.forRoot({
      domain: 'sts.restograde.com',
      clientId: 'lY5g0BKB7Mow4yDlb6rdGPsO2i1g7Osv',
    }),
  ],
  …
})
```

Configure the SDK with the domain of your tenant and the clientID of the SPA application

Service methods for relevant OAuth 2.0 / OIDC features

```
constructor(public auth: AuthService) {}

login() {
  this.auth.loginWithRedirect();
}
logout() {
  this.auth.logout({ returnTo: window.location.origin });
}
```

The Auth0 AuthService exposes all relevant features to use in components and services

@PhilippeDeRyck

Configuring a generic Angular OAuth 2.0 / OIDC library

```
import { AuthConfig } from 'angular-oauth2-oidc';

export const authCodeFlowConfig: AuthConfig = {
  issuer: 'https://sts.restograde.com',
  redirectUri: window.location.origin + '/index.html',
  clientId: 'lY5g0BKB7Mow4yDlb6rdGPsO2i1g7Osv',
  responseType: 'code',
  scope: 'openid profile email offline_access api',
  customQueryParams: {
    audience: 'https://api.restograde.com',
  },
};
```

Configure the library with the domain of your tenant and the clientID of the SPA application

Loading angular-oauth2-oidc *and discovering the STS settings*

```
this.oauthService.configure(authCodeFlowConfig);
this.oauthService.loadDiscoveryDocumentAndTryLogin();
```

Running an Authorization Code flow with angular-oauth2-oidc

```
this.oauthService.initCodeFlow();
```

Wrapping the Auth0 React SDK around the application

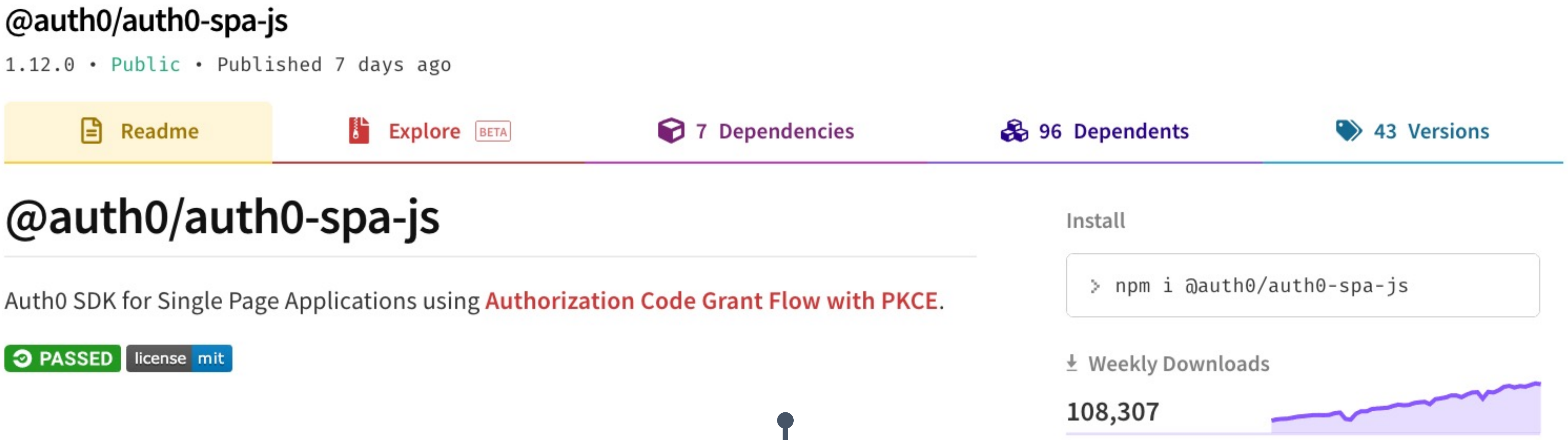
```
ReactDOM.render(
  <Auth0Provider
    domain="sts.restograde.com"
    clientId="lY5g0BKB7Mow4yDlb6rdGPsO2i1g7Osv"
    redirectUri={window.location.origin}
  >
    <App />
  </Auth0Provider>,
  document.getElementById('app')
);
```

Configure the SDK with the domain of your tenant and the clientID of the SPA application

Hooks for relevant OAuth 2.0 / OIDC features

```
const {
  isLoading,
  isAuthenticated,
  error,
  user,
  loginWithRedirect,
  logout,
  getAccessTokenSilently,
} = useAuth0();
```

Feature-specific hooks expose all relevant information and operations for use in components


@auth0/auth0-spa-js

1.12.0 • Public • Published 7 days ago

Readme | Explore BETA | 7 Dependencies | 96 Dependents | 43 Versions

@auth0/auth0-spa-js

Auth0 SDK for Single Page Applications using **Authorization Code Grant Flow with PKCE**.

PASSED | license mit

Install

```
> npm i @auth0/auth0-spa-js
```

Weekly Downloads

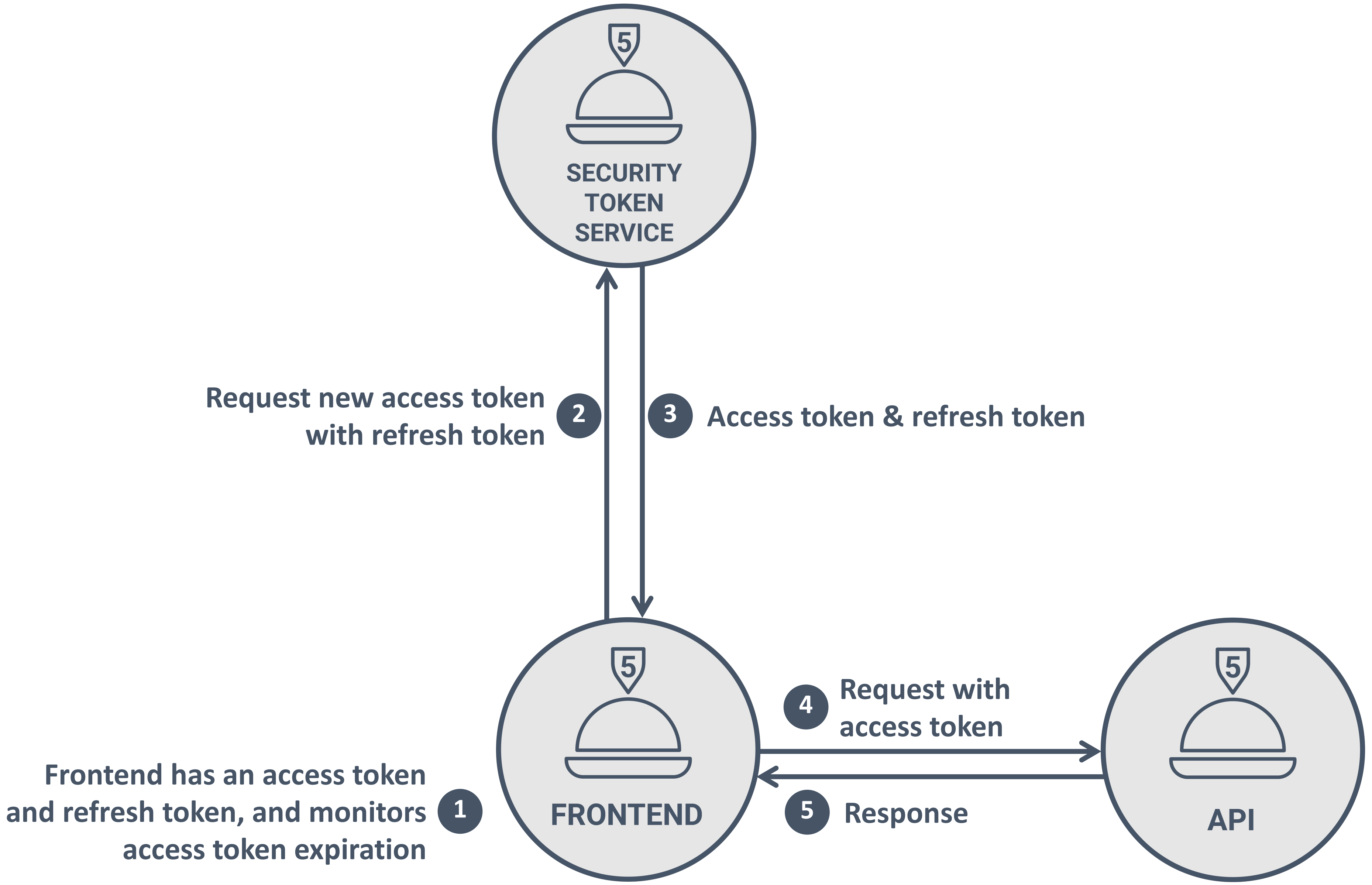
108,307

Auth0's generic JS SDK, which can be used in any JS-based framework or frontend application

The Authorization Code Flow with PKCE is the only relevant flow for SPAs today

What if an access token expires?

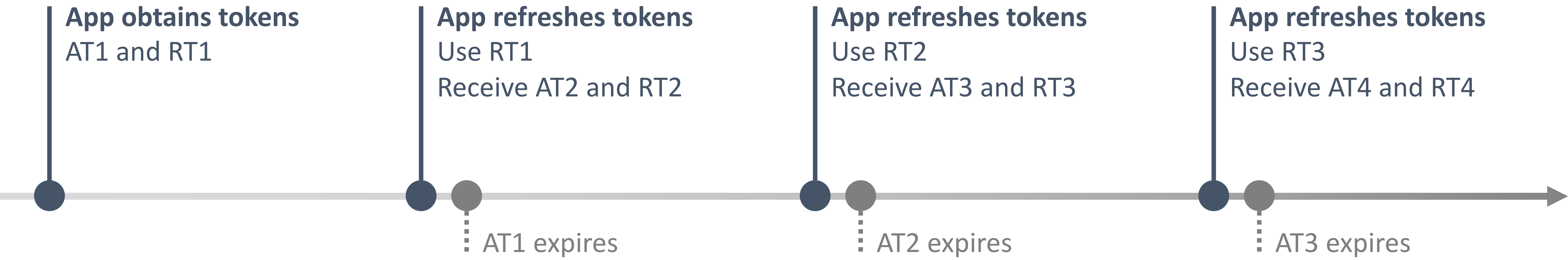
THE *REFRESH TOKEN* FLOW


SECURITY TOKEN SERVICE
Request new access token with refresh token
2
3
Access token & refresh token
Frontend has an access token and refresh token, and monitors access token expiration
1
FRONTEND
4
Request with access token
5
Response
API

What if an attacker injects malicious code to steal the tokens from the SPA?

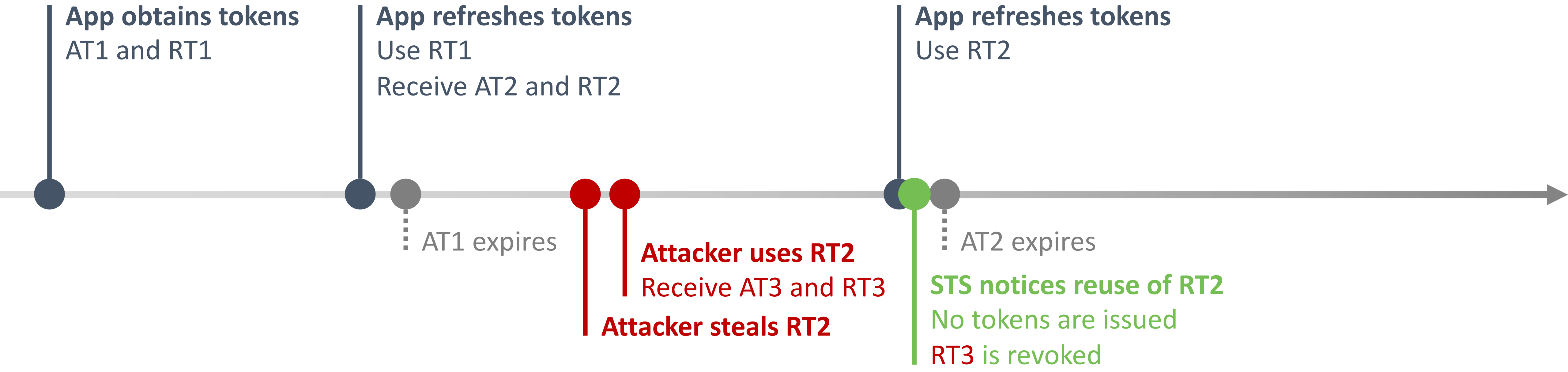
Refresh token rotation

- Refresh token rotation is required for using refresh tokens in the browser
  - Part of the ***OAuth 2.0 for Browser-Based Apps*** proposal
  - Refresh tokens are used once to obtain a new access token and new refresh token
  - Previously used refresh tokens become invalid



Detecting refresh token abuse

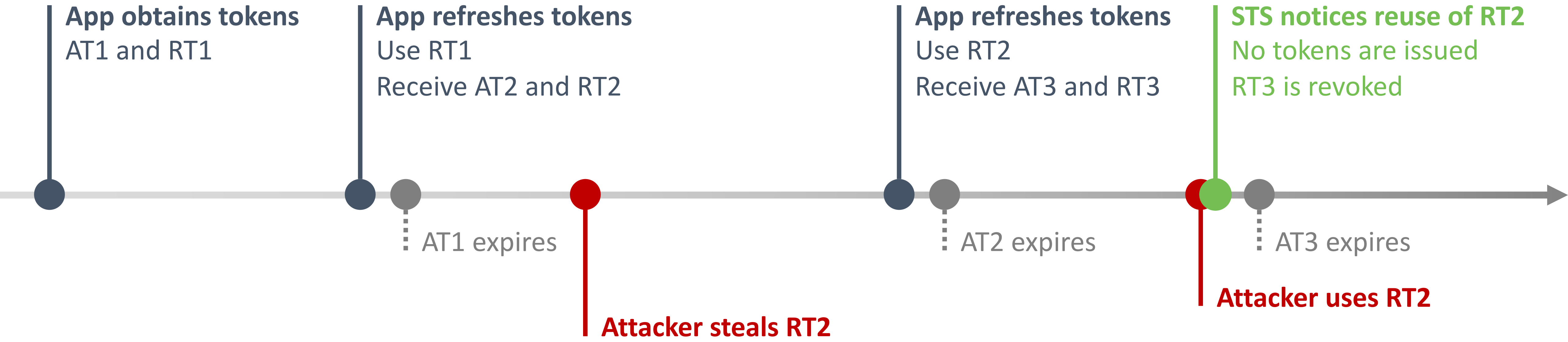
- When the STS detects the re-use of a refresh token, something is wrong
  - The refresh token is immediately revoked, preventing abuse
- To ensure security, the STS revokes the entire token chain of this refresh token
  - The abuse of *RT2* leads to the revocation of *RT3*, *RT4*, …



DETECTING REFRESH TOKEN ABUSE

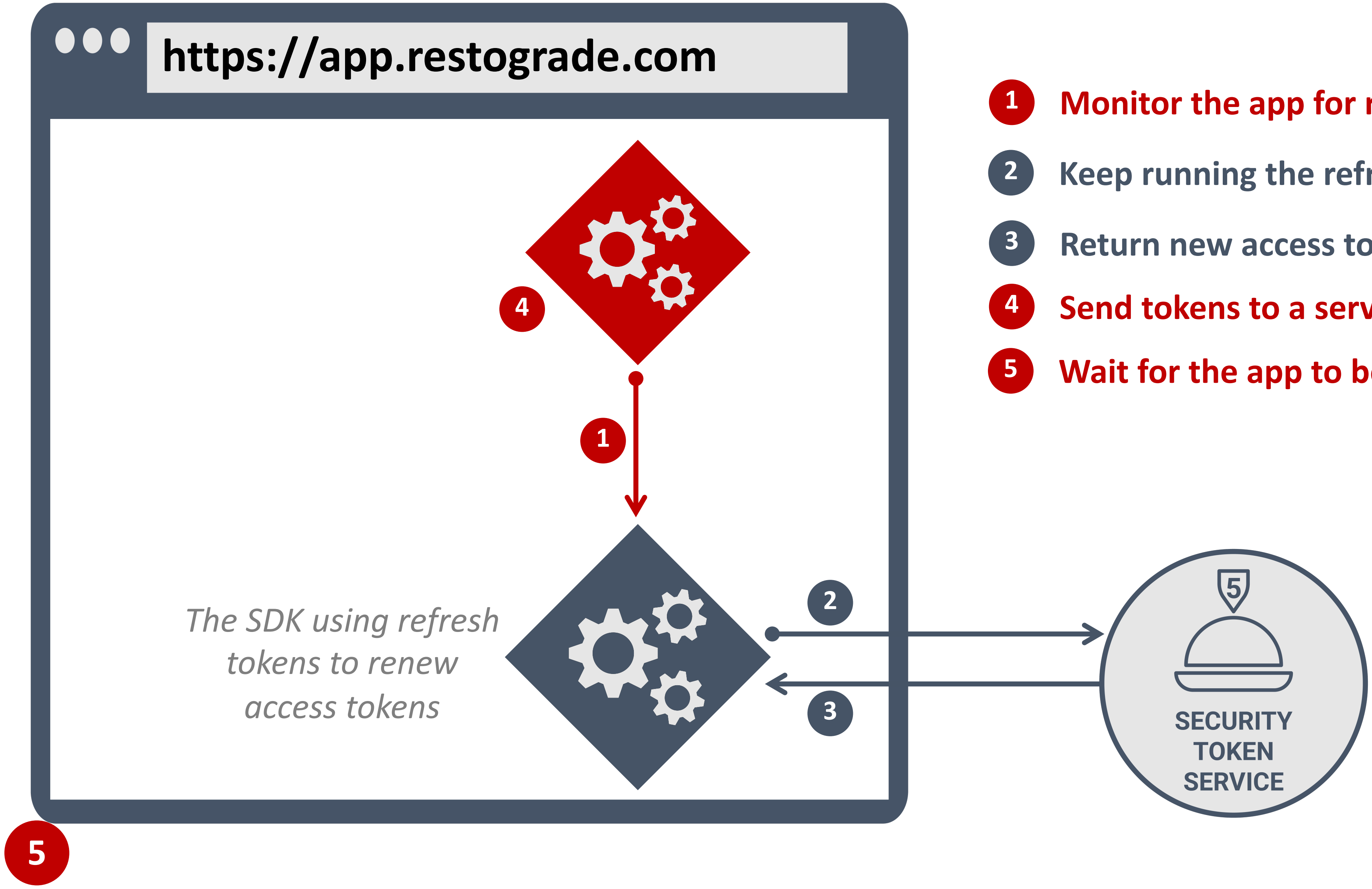
- When the STS detects the re-use of a refresh token, something is wrong
  - The refresh token is immediately revoked, preventing immediate abuse

- To ensure security, the STS revokes the entire token chain of this refresh token
  - The abuse of *RT2* leads to the revocation of *RT3*, *RT4*, …



Problem solved, right?

SIDESTEPPING REFRESH TOKEN ROTATION


https://app.restograde.com
4
1
The SDK using refresh tokens to renew access tokens
2
3
5
SECURITY TOKEN SERVICE
5

1. Monitor the app for refresh tokens (if available)
2. Keep running the refresh flow when needed
3. Return new access tokens and refresh tokens
4. Send tokens to a server controlled by the attacker
5. Wait for the app to become inactive to use RT

Stealing all tokens with the silent renew

1 The SDK running legitimate OAuth 2.0 flows

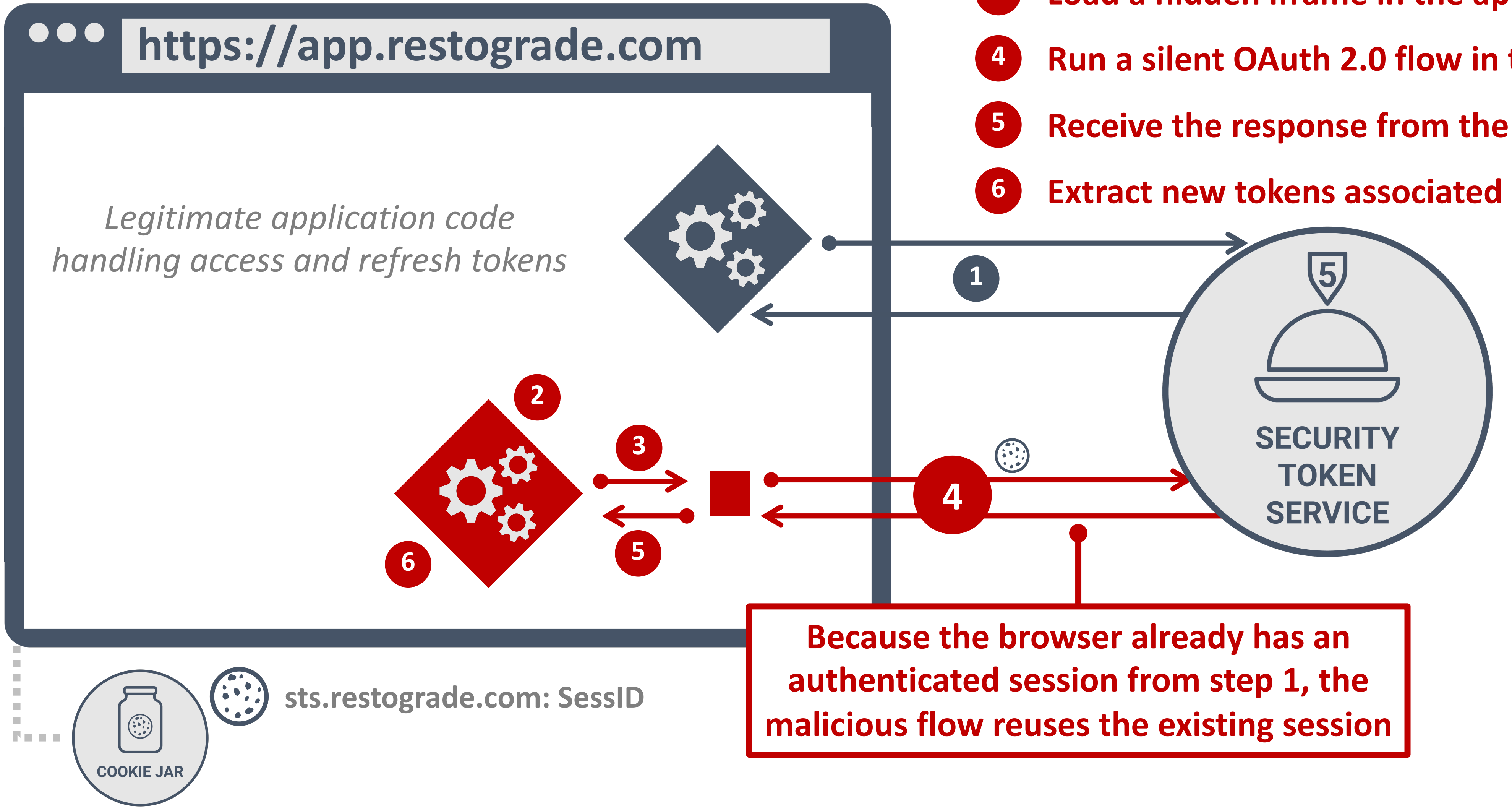
2 Setup a listener to receive messages from a frame

3 Load a hidden iframe in the application's page

4 Run a silent OAuth 2.0 flow in the hidden iframe

5 Receive the response from the iframe

6 Extract new tokens associated with the user



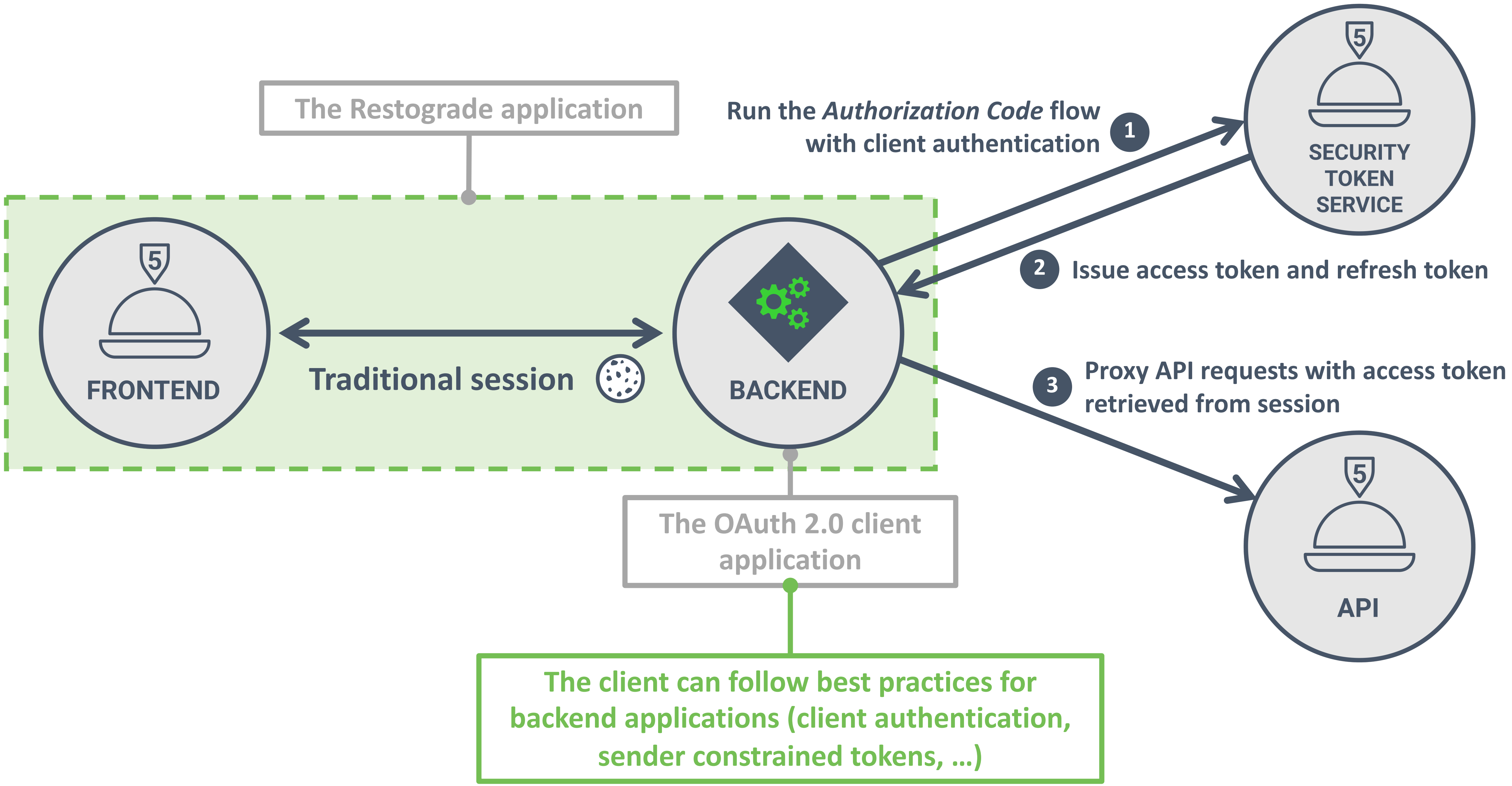
https://app.restograde.com
Legitimate application code handling access and refresh tokens
1
2
3
4
5
6
SECURITY TOKEN SERVICE
COOKIE JAR
sts.restograde.com: SessID

Because the browser already has an authenticated session from step 1, the malicious flow reuses the existing session

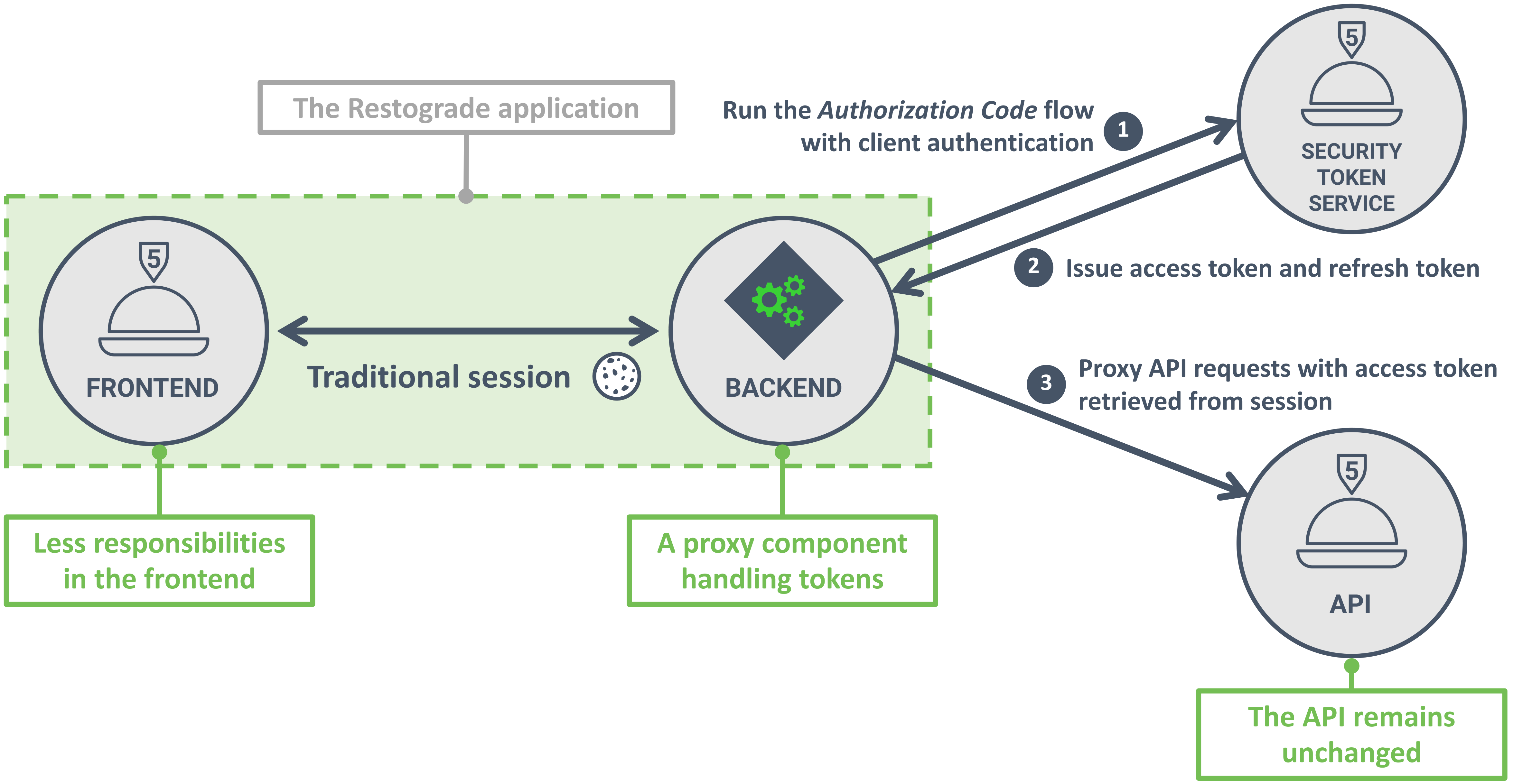
So, we're screwed?

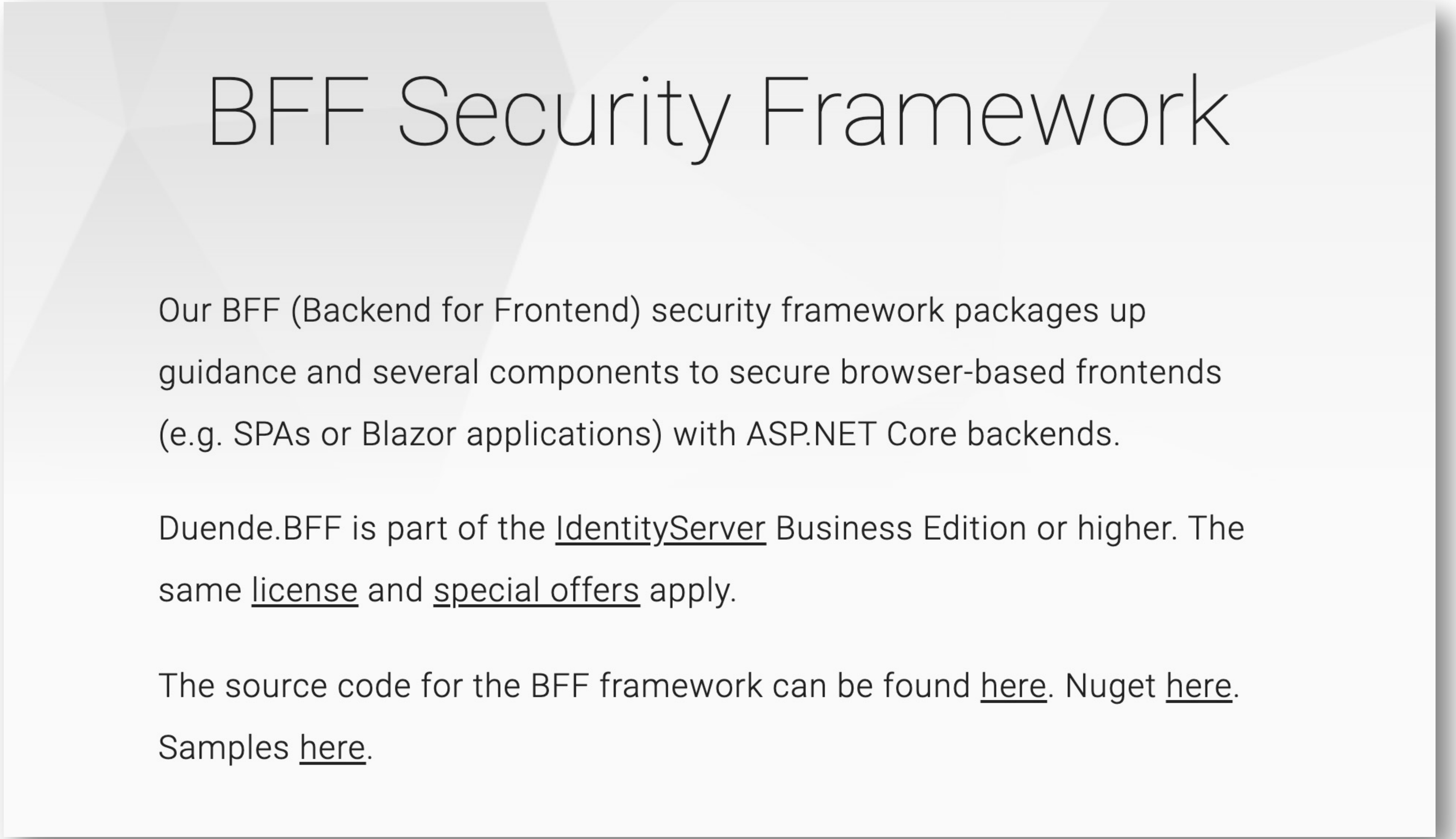
Yes.

THE CONCEPT OF A BACKEND-FOR-FRONTEND


The Restograde application
Run the *Authorization Code* flow with client authentication
1
SECURITY TOKEN SERVICE
2
Issue access token and refresh token
FRONTEND
Traditional session
BACKEND
3
Proxy API requests with access token retrieved from session
API
The OAuth 2.0 client application
The client can follow best practices for backend applications (client authentication, sender constrained tokens, …)

THE CONCEPT OF A BACKEND-FOR-FRONTEND


The Restograde application
Run the *Authorization Code* flow with client authentication
1
SECURITY TOKEN SERVICE
2
Issue access token and refresh token
FRONTEND
Traditional session
BACKEND
3
Proxy API requests with access token retrieved from session
API
Less responsibilities in the frontend
A proxy component handling tokens
The API remains unchanged



BFF Security Framework

Our BFF (Backend for Frontend) security framework packages up guidance and several components to secure browser-based frontends (e.g. SPAs or Blazor applications) with ASP.NET Core backends.

Duende.BFF is part of the IdentityServer Business Edition or higher. The same license and special offers apply.

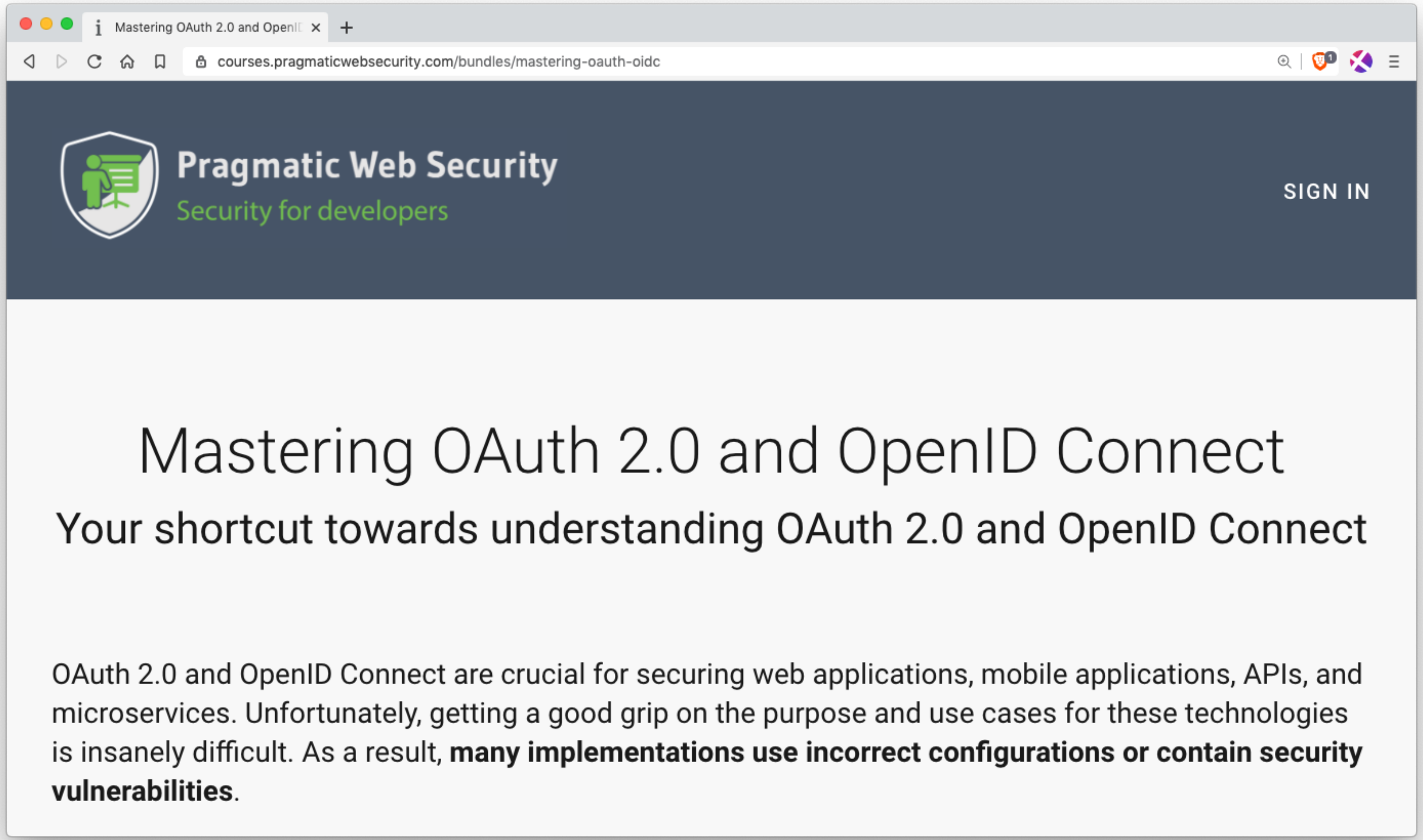
The source code for the BFF framework can be found here. Nuget here. Samples here.

Sensitive Single Page Applications should definitely consider using a BFF

Key takeaways

1. Use the *Authorization Code* flow with PKCE in SPAs
2. Use short access tokens lifetimes and refresh tokens with rotation
3. Sensitive SPAs should avoid tokens in the browser in favor of a BFF

This online course condenses dozens of confusing specs into a crystal-clear academic-level learning experience


Mastering OAuth 2.0 and OpenI
courses.pragmaticwebsecurity.com/bundles/mastering-oauth-oidc
Pragmatic Web Security
Security for developers
SIGN IN

Mastering OAuth 2.0 and OpenID Connect

Your shortcut towards understanding OAuth 2.0 and OpenID Connect

OAuth 2.0 and OpenID Connect are crucial for securing web applications, mobile applications, APIs, and microservices. Unfortunately, getting a good grip on the purpose and use cases for these technologies is insanely difficult. As a result, **many implementations use incorrect configurations or contain security vulnerabilities**.

https://courses.pragmaticwebsecurity.com

Thank you!

Connect on social media for more in-depth security content

@PhilippeDeRyck

/in/PhilippeDeRyck