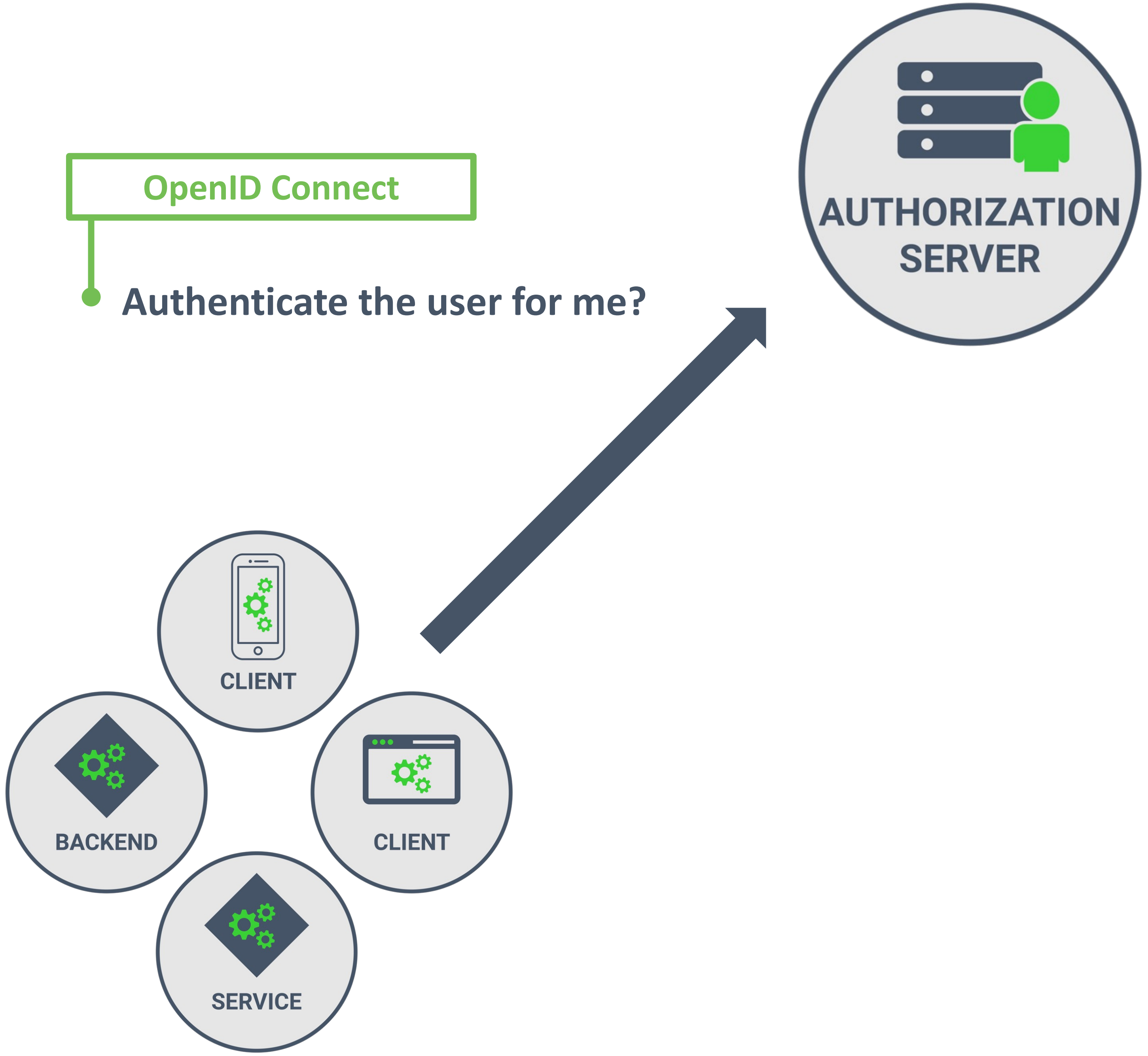
Securing an API ecosystem with OAuth 2.0

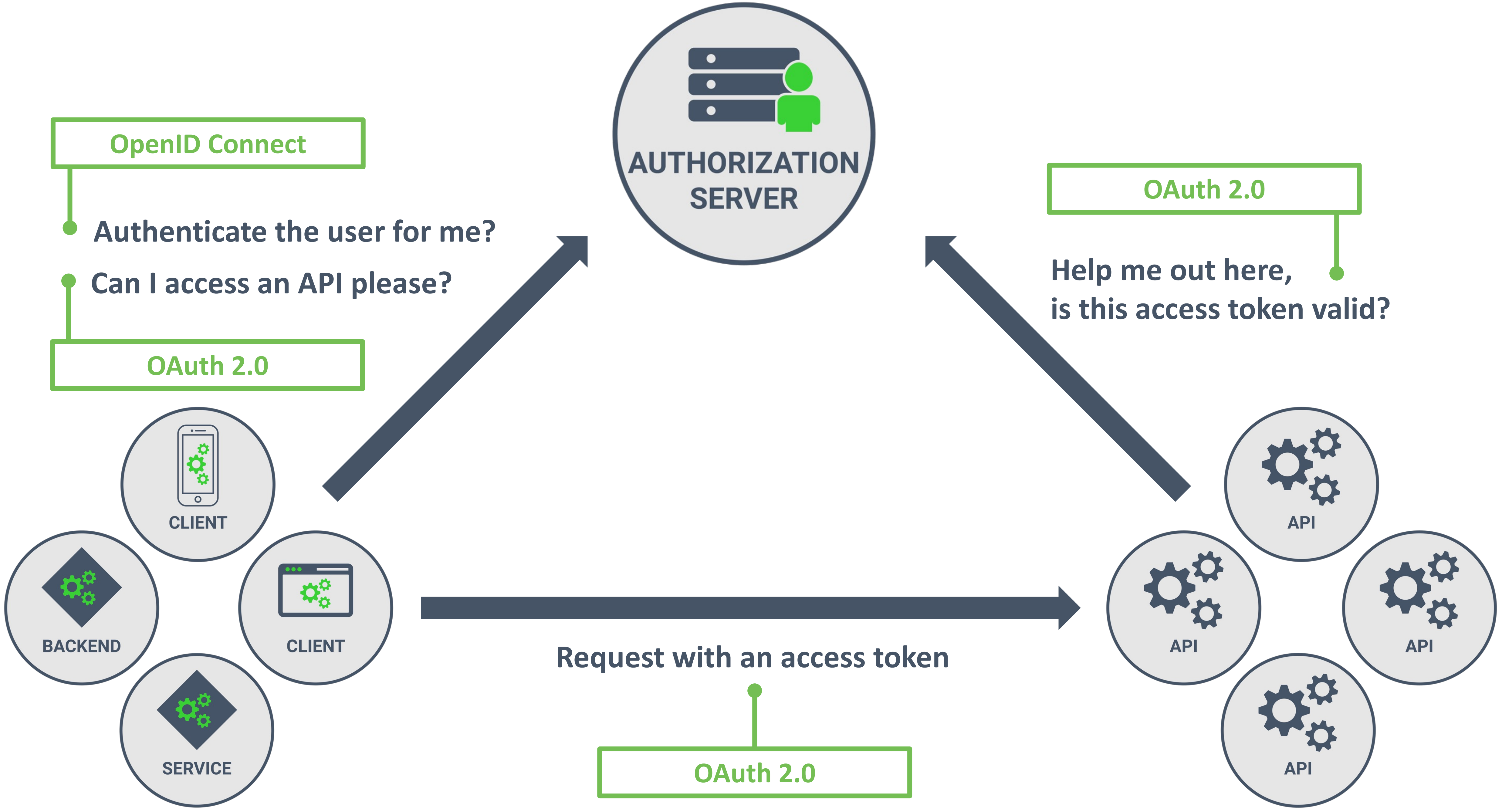
Dr. Philippe De Ryck

https://PragmaticWebSecurity.com

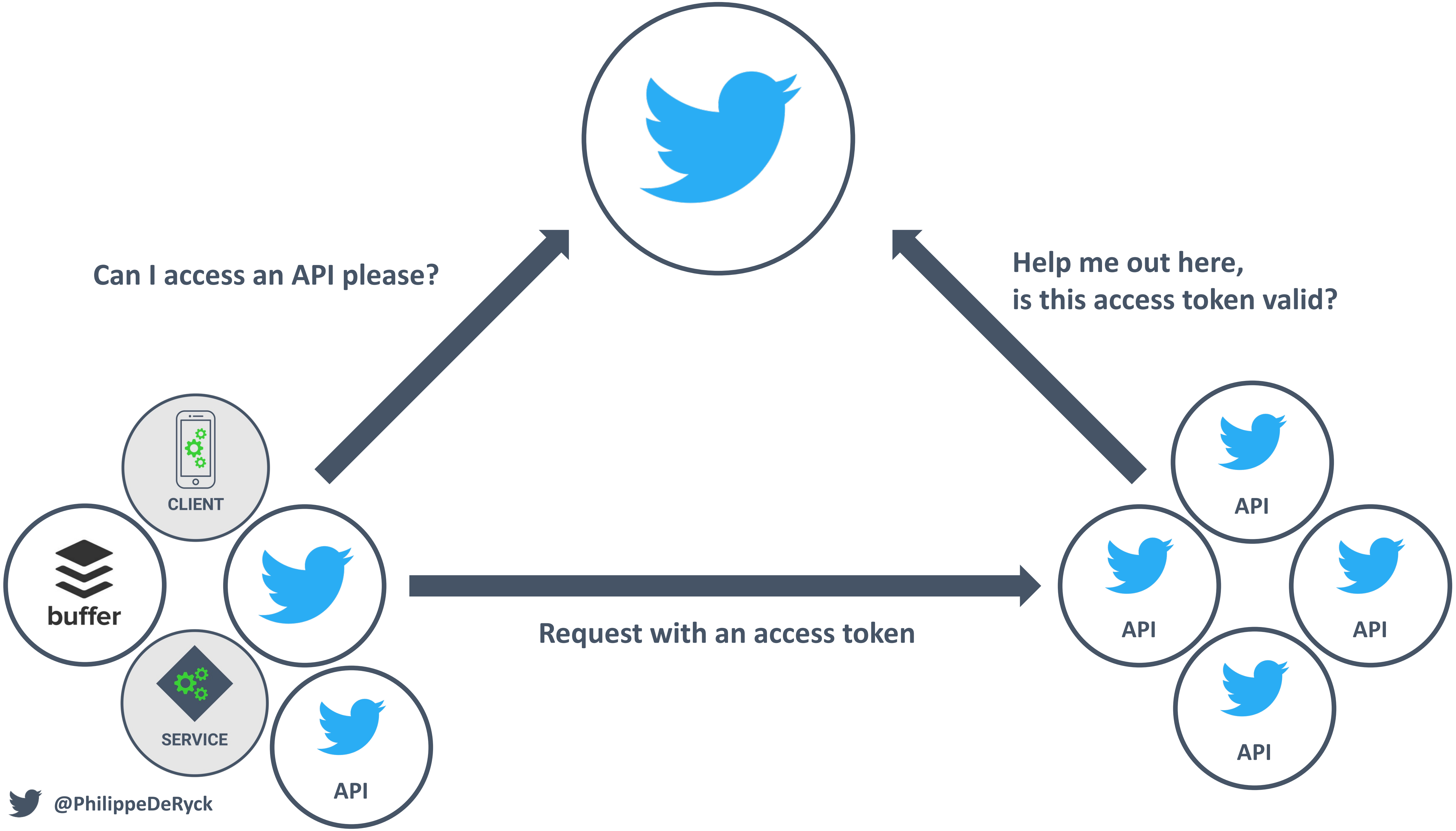
OpenID Connect

Authenticate the user for me?





@PhilippeDeRyck


Can I access an API please?
Help me out here,
is this access token valid?
CLIENT
buffer
SERVICE
API
Request with an access token
API
API
API
API

@PhilippeDeRyck

Use OAuth 2.0 and OIDC as intended

OIDC enables offloading authentication to an identity provider.

OAuth 2.0 enables a uniform authorization framework to support client access to protected resources.

I am *Dr. Philippe De Ryck*

Founder of Pragmatic Web Security

Google Developer Expert

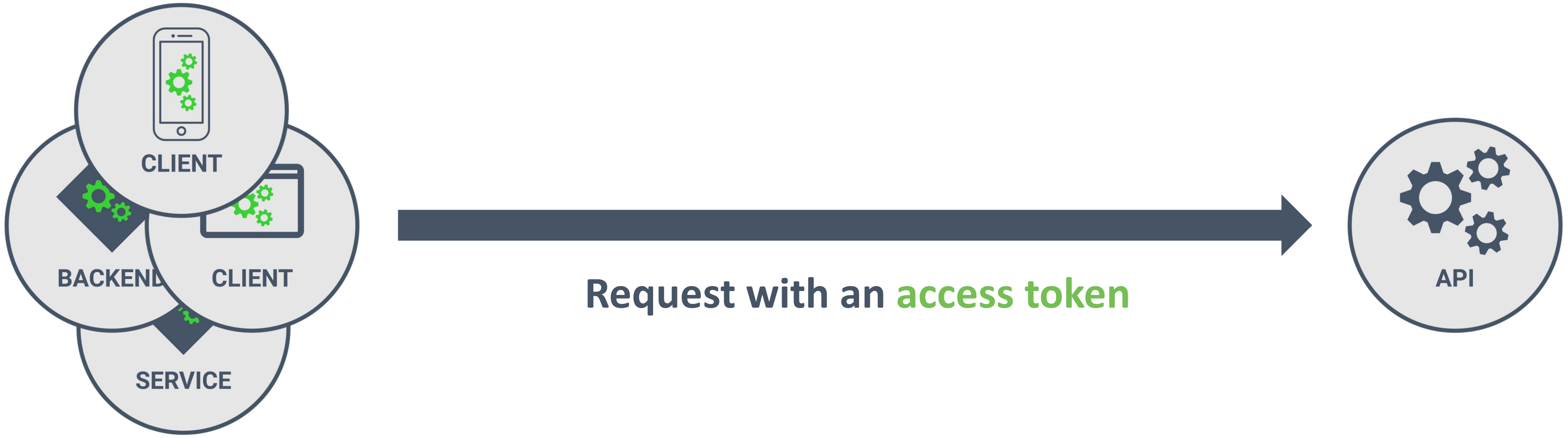
Auth0 Ambassador

SecAppDev organizer

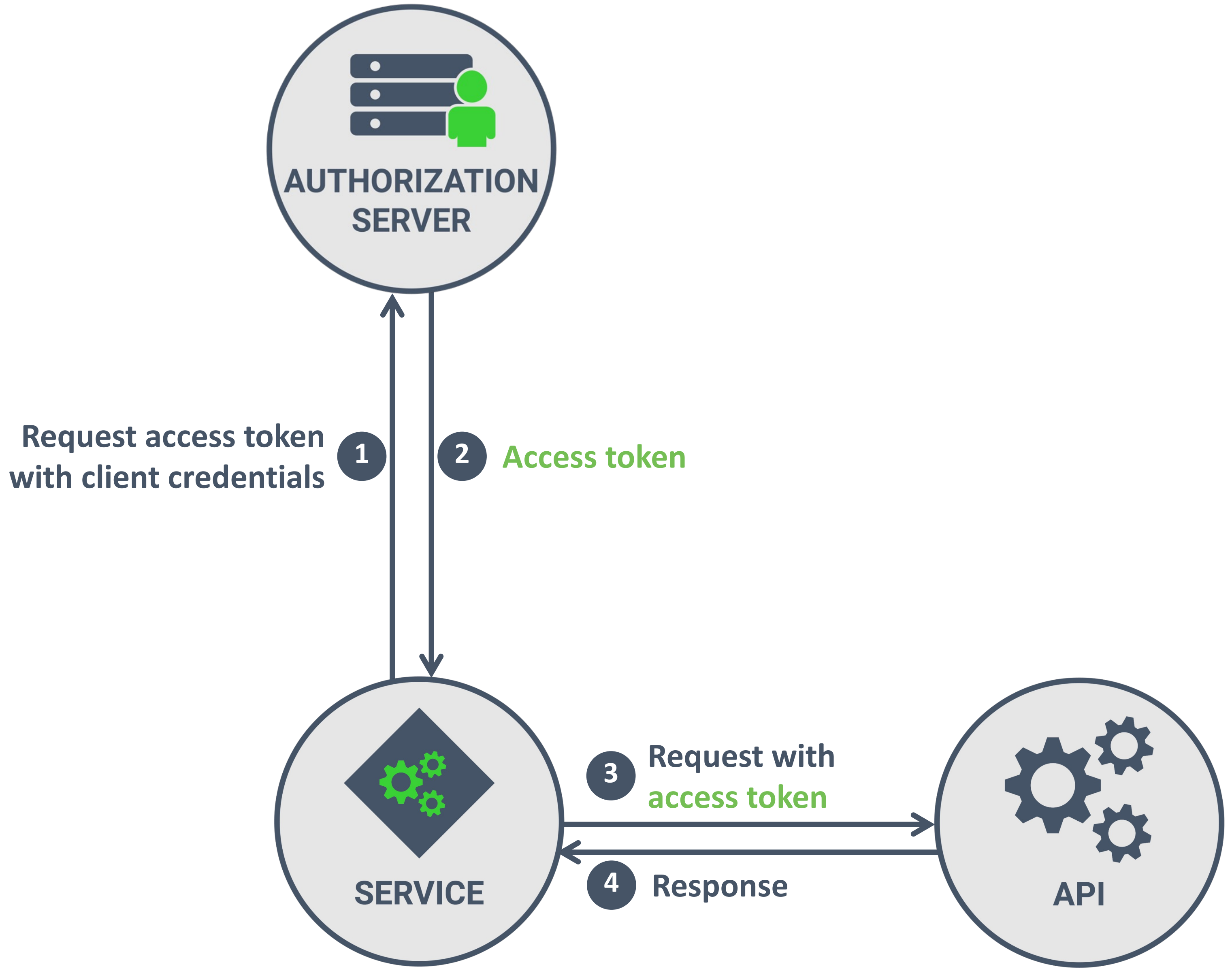
I help developers with security

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

https://pragmaticwebsecurity.com


CLIENT
BACKEND
CLIENT
SERVICE
Request with an access token
API

Scenarios that do not involve user-based access rely on the *Client Credentials* grant

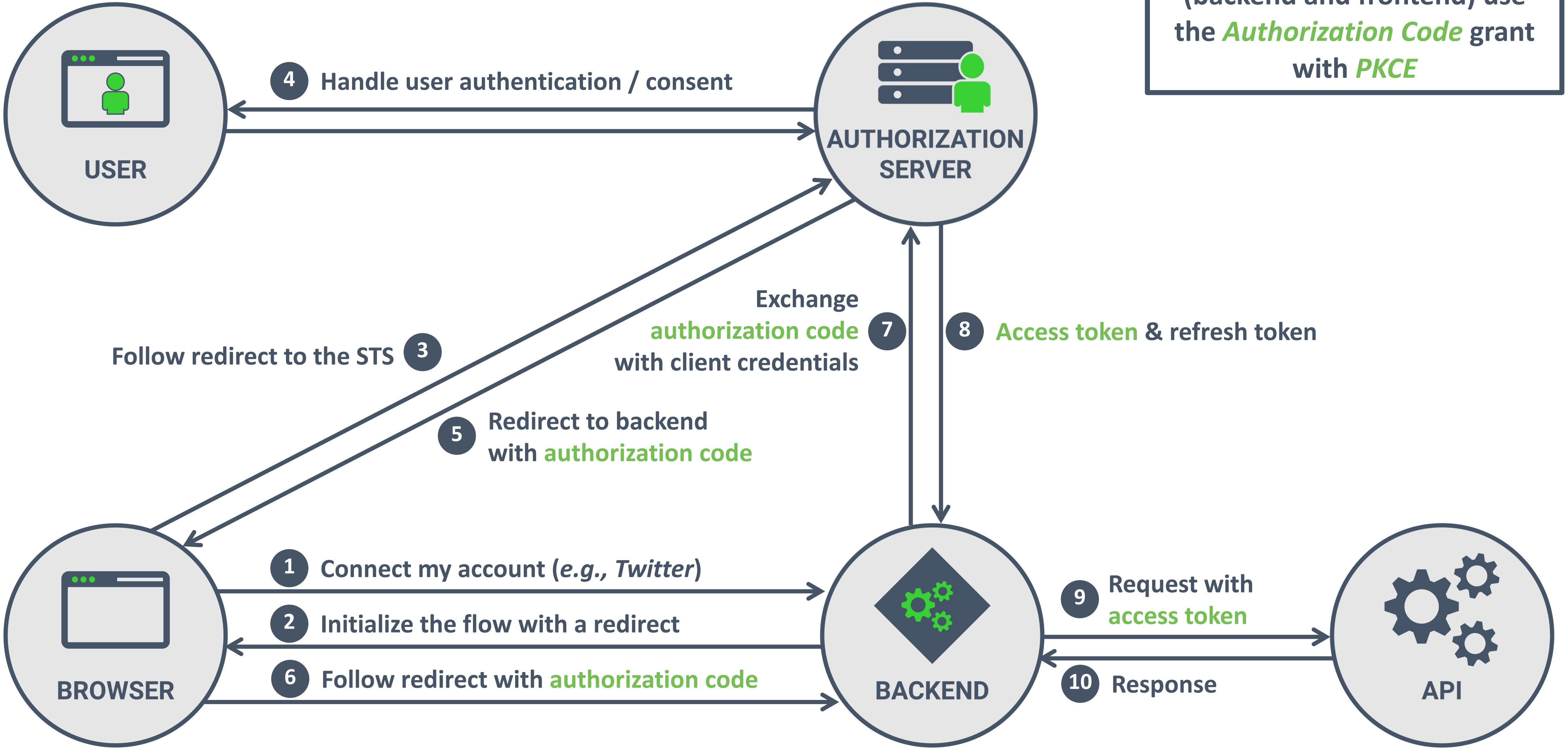

AUTHORIZATION SERVER
Request access token with client credentials
1
2
Access token
SERVICE
3
Request with access token
4
Response
API

The client credentials grant enables M2M access

The client credentials grant supports direct machine-to-machine access.

The grant relies on client credentials which have to be kept in a secure location (i.e., not on an untrusted device)

User-based access scenarios (backend and frontend) use the *Authorization Code* grant with *PKCE*

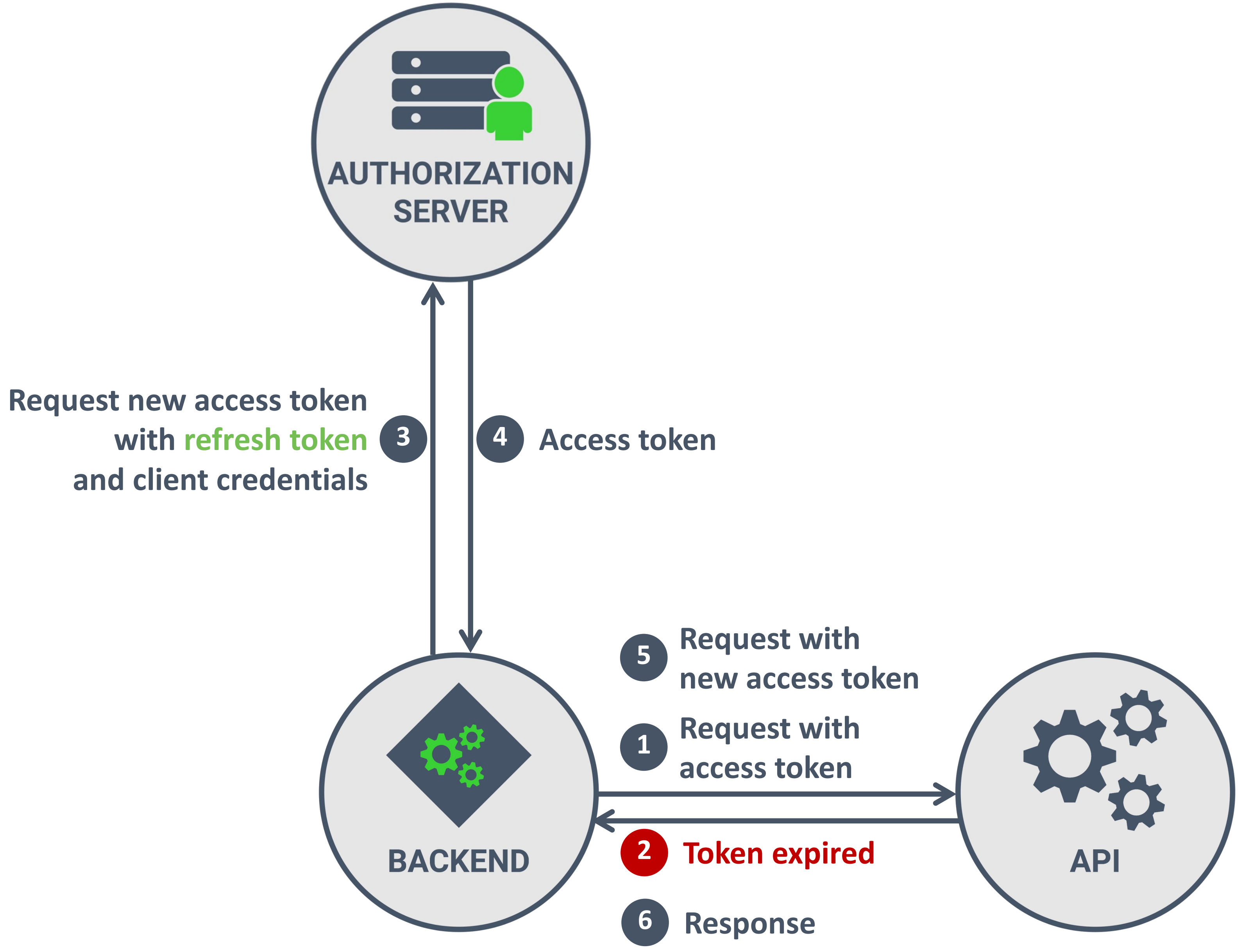


USER
4 Handle user authentication / consent
AUTHORIZATION SERVER
Follow redirect to the STS 3
Exchange authorization code with client credentials 7
8 Access token & refresh token
5 Redirect to backend with authorization code
BROWSER
1 Connect my account (*e.g., Twitter*)
2 Initialize the flow with a redirect
6 Follow redirect with authorization code
BACKEND
9 Request with access token
10 Response
API

@PhilippeDeRyck

THE AUTHORIZATION CODE GRANT ENABLES ACCESS ON BEHALF OF A USER

The authorization code grant with PKCE allows the user to delegate authority to an application to access APIs on their behalf


AUTHORIZATION SERVER
Request new access token with refresh token and client credentials
3
4
Access token
5
Request with new access token
1
Request with access token
BACKEND
2
Token expired
API
6
Response

USE SHORT-LIVED ACCESS TOKENS

OAuth 2.0 supports short-lived access tokens by also issuing a refresh token. Refresh tokens are under full control of the authorization server and can be easily revoked.

MISUSING TOKENS CAUSES AUTHORIZATION ISSUES

A common pitfall is to misuse tokens for a purpose they are not intended for. APIs only consume access tokens, not identity tokens or refresh tokens.

A reference access token

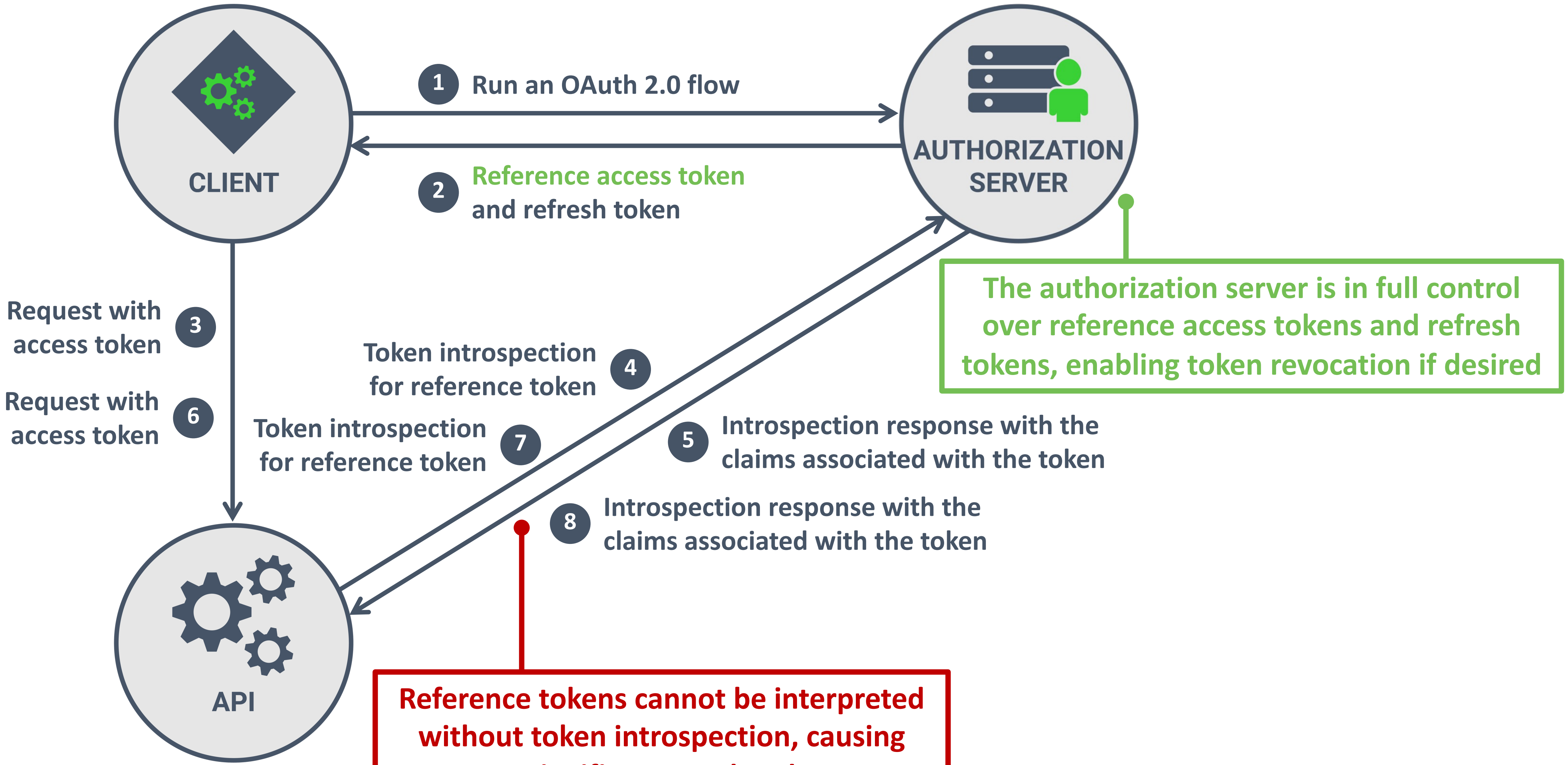
```
vSvhNDeQLqrzRbvA2eeYE2PthB1cBimS
```

A self-contained access token

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZC
I6Ik5UVkJPVFUzTXpCQk9FVXdOemhCUTBWR01rUTBR
VVU1UVRZeFFVVXlPVU5FUVVVeE5qRXlNdyJ9.eyJpc
3MiOiJodHRwczovL3N0cy5yZXN0b2dyYWRlLmNvbS8
iLCJzdWIiOiJhdXRoMHw1ZWI5MTZjMjU4YmRiNTBiZ
jIwMzY2YzYiLCJhdWQiOlsiaHR0cHM6Ly9hcGkucmV
zdG9ncmFkZS5jb20iLCJodHRwczovL3Jlc3RvZ3JhZ
GUuZXUuYXV0aDAuY29tL3VzZXJpbmZvIl0sImlhdCI
6MTU4OTc3NTA3MiwiZXhwIjoxNTg5ODYxNDcyLCJhe
nAiOiJPTEtObjM4OVNVSW11ZkV4Z1JHMVJpbExTZ2R
ZeHdFcCIsInNjb3BlIjoib3BlbmlkIHByb2ZpbGUgZ
W1haWwgb2ZmbGluZV9hY2Nlc3MifQ.XzJOXtTXOGOS
bCFvp4yZGJzh7XhMmOmI2XxtjWdlODz_siI-u8h11e
lcr8LwX6-hL20QOW0eStzBzmm1FM_tS7MxuKkYx8Ql
TWOURPembVKZOhNi8kN-1j0pyc0uzve7Jib5vcxmkP
wqpcVDFACgP85_0NYe4zXHKxCA5_8VOn05cRCDSkNM
TFzGJCT9ipCcNXaVGdksojYGqQzezjpzzzwrtPEkiy
FLFtDPZAl0MleF3oFAOCBK0UKuNjJ_cSBbUsaIwfvK
0WH47AwFrRn_TxL4S1P3j3b1GgBm8tAqXysY84VZu0
rSg3zrZj1PnoqPD4mbOXds20xafCr9wR4WTQ
```

A reference access token

```
vSvhNDeQLqrzRbvA2eeYE2PthB1cBimS
```



CLIENT
AUTHORIZATION SERVER
API
1 Run an OAuth 2.0 flow
2 Reference access token and refresh token
3 Request with access token
6 Request with access token
4 Token introspection for reference token
7 Token introspection for reference token
5 Introspection response with the claims associated with the token
8 Introspection response with the claims associated with the token

The authorization server is in full control over reference access tokens and refresh tokens, enabling token revocation if desired

Reference tokens cannot be interpreted without token introspection, causing significant overhead

@PhilippeDeRyck

A reference access token

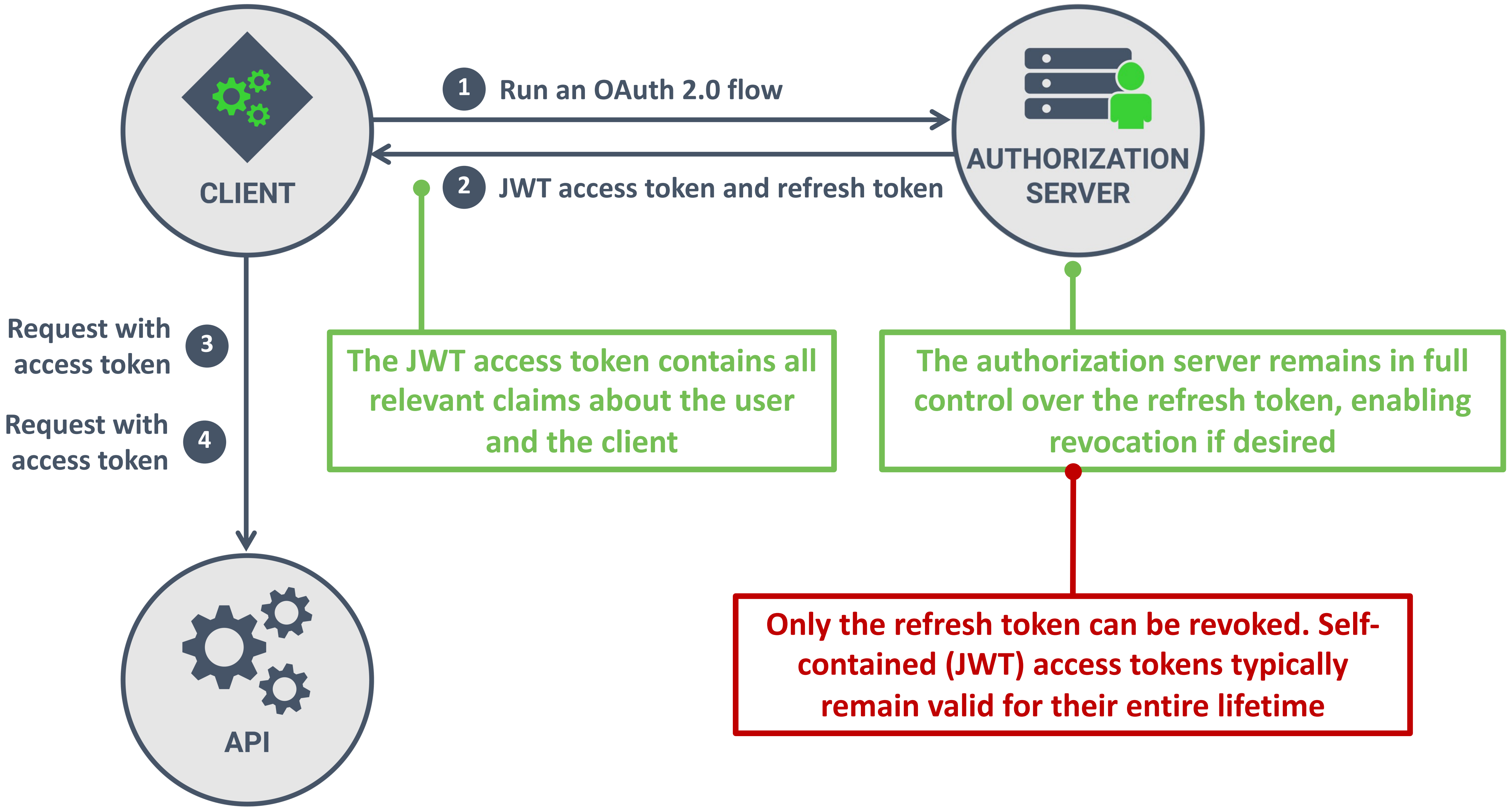
```
vSvhNDeQLqrzRbvA2eeYE2PthB1cBimS
```

A self-contained access token

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZC
I6Ik5UVkJPVFUzTXpCQk9FVXdOemhCUTBWR01rUTBR
VVU1UVRZeFFVVXlPVU5FUVVVeE5qRXlNdyJ9.eyJpc
3MiOiJodHRwczovL3N0cy5yZXN0b2dyYWRlLmNvbS8
iLCJzdWIiOiJhdXRoMHw1ZWI5MTZjMjU4YmRiNTBiZ
jIwMzY2YzYiLCJhdWQiOlsiaHR0cHM6Ly9hcGkucmV
zdG9ncmFkZS5jb20iLCJodHRwczovL3Jlc3RvZ3JhZ
GUuZXUuYXV0aDAuY29tL3VzZXJpbmZvIl0sImlhdCI
6MTU4OTc3NTA3MiwiZXhwIjoxNTg5ODYxNDcyLCJhe
nAiOiJPTEtObjM4OVNVSW11ZkV4Z1JHMVJpbExTZ2R
ZeHdFcCIsInNjb3BlIjoib3BlbmlkIHByb2ZpbGUgZ
W1haWwgb2ZmbGluZV9hY2Nlc3MifQ.XzJOXtTXOGOS
bCFvp4yZGJzh7XhMmOmI2XxtjWdlODz_siI-u8h11e
lcr8LwX6-hL20QOW0eStzBzmm1FM_tS7MxuKkYx8Ql
TWOURPembVKZOhNi8kN-1j0pyc0uzve7Jib5vcxmkP
wqpcVDFACgP85_0NYe4zXHKxCA5_8VOn05cRCDSkNM
TFzGJCT9ipCcNXaVGdksojYGqQzezjpzzzwrtPEkiy
FLFtDPZAl0MleF3oFAOCBK0UKuNjJ_cSBbUsaIwfvK
0WH47AwFrRn_TxL4S1P3j3b1GgBm8tAqXysY84VZu0
rSg3zrZj1PnoqPD4mbOXds20xafCr9wR4WTQ
```

A self-contained access token

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZC
I6Ik5UVkJPVFUzTXpCQk9FVXdOemhCUTBWR01rUTBR
VVU1UVRZeFFVVXlPVU5FUVVVeE5qRXlNdyJ9.eyJpc
3MiOiJodHRwczovL3N0cy5yZXN0b2dyYWRlLmNvbS8
iLCJzdWIiOiJhdXRoMHw1ZWI5MTZjMjU4YmRiNTBiZ
jIwMzY2YzYiLCJhdWQiOlsiaHR0cHM6Ly9hcGkucmV
zdG9ncmFkZS5jb20iLCJodHRwczovL3Jlc3RvZ3JhZ
GUuZXUuYXV0aDAuY29tL3VzZXJpbmZvIl0sImlhdCI
6MTU4OTc3NTA3MiwiZXhwIjoxNTg5ODYxNDcyLCJhe
nAiOiJPTEtObjM4OVNVSW11ZkV4Z1JHMVJpbExTZ2R
ZeHdFcCIsInNjb3BlIjoib3BlbmlkIHByb2ZpbGUgZ
W1haWwgb2ZmbGluZV9hY2Nlc3MifQ.XzJOXtTXOGOS
bCFvp4yZGJzh7XhMmOmI2XxtjWdlODz_siI-u8h11e
lcr8LwX6-hL20QOW0eStzBzmm1FM_tS7MxuKkYx8Ql
TWOURPembVKZOhNi8kN-1j0pyc0uzve7Jib5vcxmkP
wqpcVDFACgP85_0NYe4zXHKxCA5_8VOn05cRCDSkNM
TFzGJCT9ipCcNXaVGdksojYGqQzezjpzzzwrtPEkiy
FLFtDPZAl0MleF3oFAOCBK0UKuNjJ_cSBbUsaIwfvK
0WH47AwFrRn_TxL4S1P3j3b1GgBm8tAqXysY84VZu0
rSg3zrZj1PnoqPD4mbOXds20xafCr9wR4WTQ
```


CLIENT
1 Run an OAuth 2.0 flow
2 JWT access token and refresh token
AUTHORIZATION SERVER
3 Request with access token
4 Request with access token
API

The JWT access token contains all relevant claims about the user and the client

The authorization server remains in full control over the refresh token, enabling revocation if desired

Only the refresh token can be revoked. Self-contained (JWT) access tokens typically remain valid for their entire lifetime

@PhilippeDeRyck

Revocation vs performance

Token security is often a trade-off between performance and security. Short-lived self-contained access tokens typically offer a good balance

The claims obtained via token introspection

```
{
  "active": true,
  "iss": "https://sts.restograde.com",
  "aud": "https://api.restograde.com",
  "sub": "5eb916c258bdb50bf20366c6",
  "client_id": "0LKNn389SU…ilLSgdYxwEp",
  "scope": "reviews:read reviews:write"
}
```

The payload of a JWT-based access token

```
{
  "iss": "https://sts.restograde.com",
  "aud": "https://api.restograde.com",
  "sub": "5eb916c258bdb50bf20366c6",
  "exp": 1589861472,
  "azp": "0LKNn389SUufExgRG1RilLSgdYxwEp",
  "scope": "reviews:read reviews:write"
}
```

The claims obtained via token introspection

```
{
  "active": true,
  "iss": "https://sts.restograde.com",
  "aud": "https://api.restograde.com",
  "sub": "5eb916c258bdb50bf20366c6",
  "client_id": "0LKNn389SU…ilLSgdYxwEp",
  "scope": "reviews:read reviews:write"
}
```

Token introspection responses contain an *active* claim

The payload of a JWT-based access token

```
{
  "iss": "https://sts.restograde.com",
  "aud": "https://api.restograde.com",
  "sub": "5eb916c258bdb50bf20366c6",
  "exp": 1589861472,
  "azp": "0LKNn389SUufExgRG1RilLSgdYxwEp",
  "scope": "reviews:read reviews:write"
}
```

JWT tokens include an *expiration* timestamp (and optionally *issued at (iat)* / *not before (nbf)*)

The claims obtained via token introspection

```
{
  "active": true,
  "iss": "https://sts.restograde.com",
  "aud": "https://api.restograde.com",
  "sub": "5eb916c258bdb50bf20366c6",
  "client_id": "0LKNn389SU…ilLSgdYxwEp",
  "scope": "reviews:read reviews:write"
}
```

The payload of a JWT-based access token

```
{
  "iss": "https://sts.restograde.com",
  "aud": "https://api.restograde.com",
  "sub": "5eb916c258bdb50bf20366c6",
  "exp": 1589861472,
  "azp": "0LKNn389SUufExgRG1RilLSgdYxwEp",
  "scope": "reviews:read reviews:write"
}
```

The *iss* claim indicates which service issued the token.

The *aud* claim indicates which API is supposed to consume the token.

The claims obtained via token introspection

```
{
  "active": true,
  "iss": "https://sts.restograde.com",
  "aud": "https://api.restograde.com",
  "sub": "5eb916c258bdb50bf20366c6",
  "client_id": "OLKNn389SU…ilLSgdYxwEp",
  "scope": "reviews:read reviews:write"
}
```

The payload of a JWT-based access token

```
{
  "iss": "https://sts.restograde.com",
  "aud": "https://api.restograde.com",
  "sub": "5eb916c258bdb50bf20366c6",
  "exp": 1589861472,
  "azp": "OLKNn389SUufExgRG1RilLSgdYxwEp",
  "scope": "reviews:read reviews:write"
}
```

The client_id/*azp* claim indicates which client is authorized to use the token

Make sure the access token is acceptable

Access tokens have various claims that describe the token metadata (e.g., issuer, audience). Make sure these values make sense to your API.

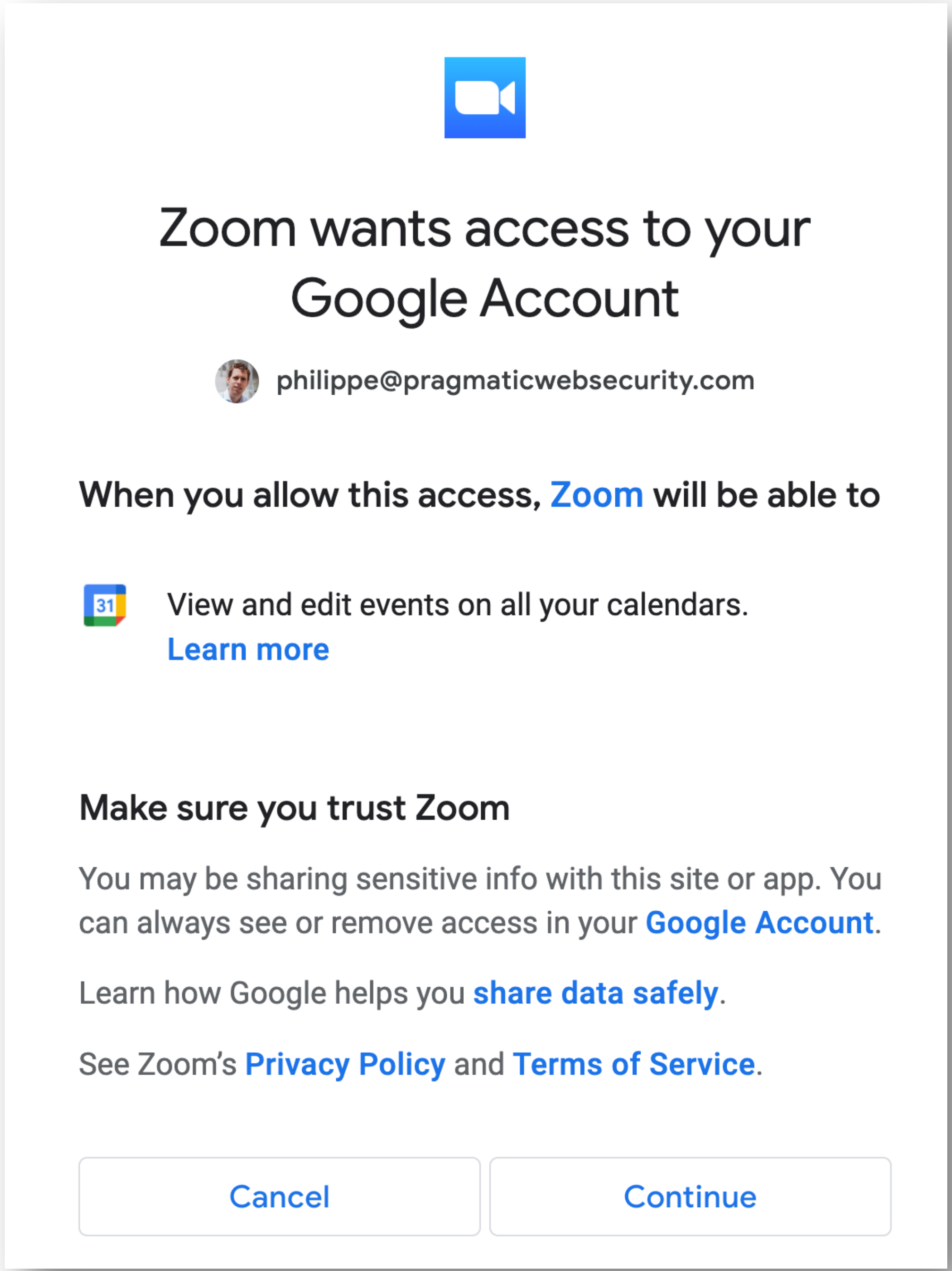
The claims obtained via token introspection

```
{
  "active": true,
  "iss": "https://sts.restograde.com",
  "aud": "https://api.restograde.com",
  "sub": "5eb916c258bdb50bf20366c6",
  "azp": "OLKNn389SUufExgRG1RilLSgdYxwEp",
  "scope": "reviews:read reviews:write"
}
```

The payload of a JWT-based access token

```
{
  "iss": "https://sts.restograde.com",
  "aud": "https://api.restograde.com",
  "sub": "5eb916c258bdb50bf20366c6",
  "exp": 1589861472,
  "azp": "OLKNn389SUufExgRG1RilLSgdYxwEp",
  "scope": "reviews:read reviews:write"
}
```

The *scope* represents the authority that the user has delegated to the client



Zoom wants access to your Google Account

philippe@pragmaticwebsecurity.com

When you allow this access, Zoom will be able to

View and edit events on all your calendars.
Learn more

Make sure you trust Zoom

You may be sharing sensitive info with this site or app. You can always see or remove access in your Google Account.

Learn how Google helps you share data safely.

See Zoom's Privacy Policy and Terms of Service.

Gmail API, v1

Scopes	
https://mail.google.com/	Read, compose, send, and permanently delete all your email from Gmail
https://www.googleapis.com/auth/gmail.addons.current.action.compose	Manage drafts and send emails when you interact with the add-on
https://www.googleapis.com/auth/gmail.addons.current.message.action	View your email messages when you interact with the add-on
https://www.googleapis.com/auth/gmail.addons.current.message.metadata	View your email message metadata when the add-on is running
https://www.googleapis.com/auth/gmail.addons.current.message.readonly	View your email messages when the add-on is running
https://www.googleapis.com/auth/gmail.compose	Manage drafts and send emails
https://www.googleapis.com/auth/gmail.insert	Insert mail into your mailbox
https://www.googleapis.com/auth/gmail.labels	Manage mailbox labels
https://www.googleapis.com/auth/gmail.metadata	View your email message metadata such as labels and headers, but not the email body
https://www.googleapis.com/auth/gmail.modify	View and modify but not delete your email
https://www.googleapis.com/auth/gmail.readonly	View your email messages and settings
https://www.googleapis.com/auth/gmail.send	Send email on your behalf
https://www.googleapis.com/auth/gmail.settings.basic	Manage your basic mail settings
https://www.googleapis.com/auth/gmail.settings.sharing	Manage your sensitive mail settings, including who can manage your mail

Google Analytics API, v3

Scopes	
https://www.googleapis.com/auth/analytics	View and manage your Google Analytics data
https://www.googleapis.com/auth/analytics.edit	Edit Google Analytics management entities
https://www.googleapis.com/auth/analytics.manage.users	Manage Google Analytics Account users by email address
https://www.googleapis.com/auth/analytics.manage.users.readonly	View Google Analytics user permissions
https://www.googleapis.com/auth/analytics.provision	Create a new Google Analytics account along with its default property and view
https://www.googleapis.com/auth/analytics.readonly	View your Google Analytics data
https://www.googleapis.com/auth/analytics.user.deletion	Manage Google Analytics user deletion requests

Google Sheets API, v4

Scopes	
https://www.googleapis.com/auth/drive	See, edit, create, and delete all of your Google Drive files
https://www.googleapis.com/auth/drive.file	View and manage Google Drive files and folders that you have opened or created with this app
https://www.googleapis.com/auth/drive.readonly	See and download all your Google Drive files
https://www.googleapis.com/auth/spreadsheets	See, edit, create, and delete your spreadsheets in Google Drive
https://www.googleapis.com/auth/spreadsheets.readonly	View your Google Spreadsheets

Google Sign-In

Scopes	
profile	View your basic profile info
email	View your email address
openid	Authenticate using OpenID Connect

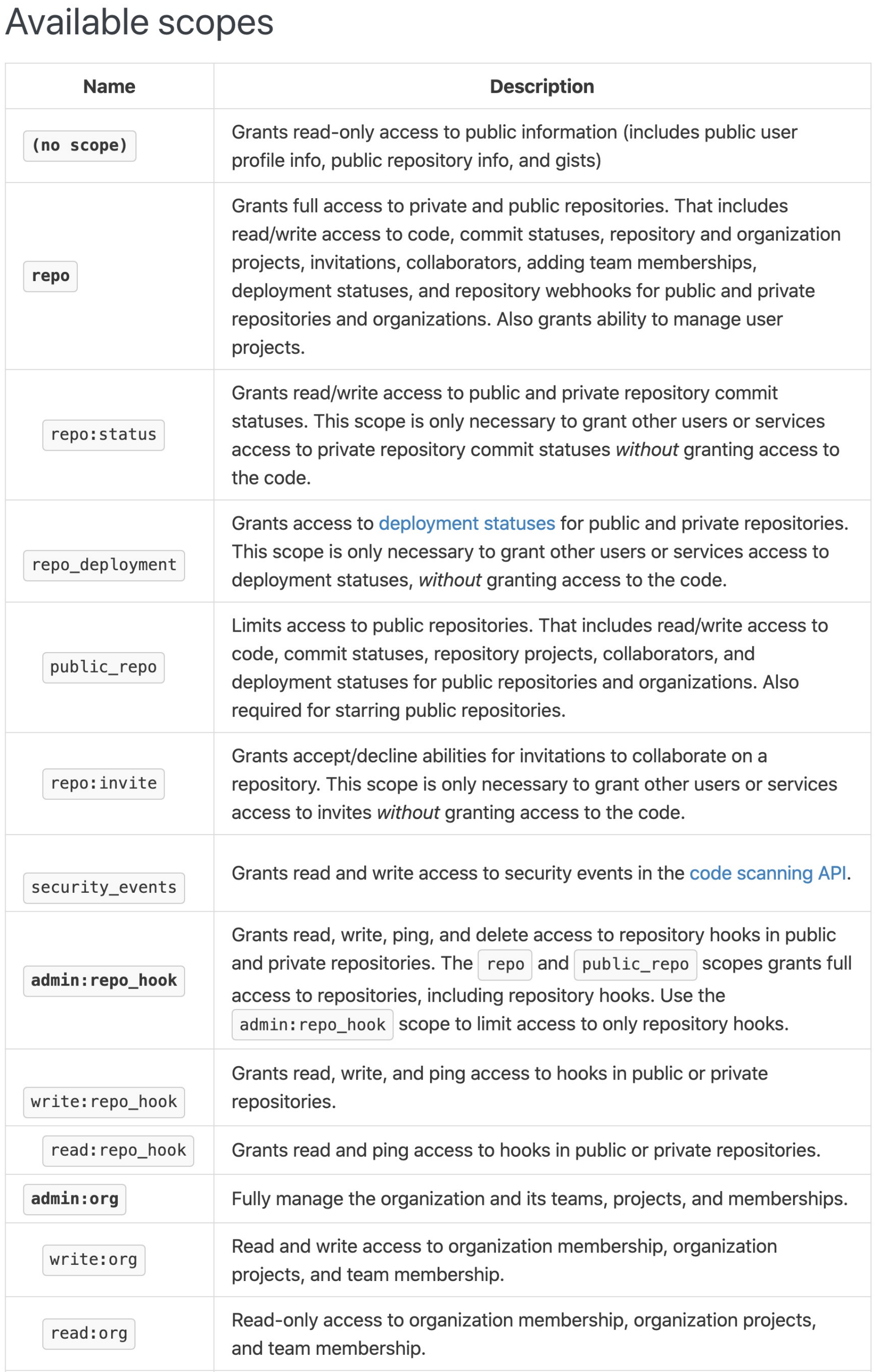
Google Site Verification API, v1

Scopes	
https://www.googleapis.com/auth/siteverification	Manage the list of sites and domains you control
https://www.googleapis.com/auth/siteverification.verify_only	Manage your new site verifications with Google

Google Slides API, v1

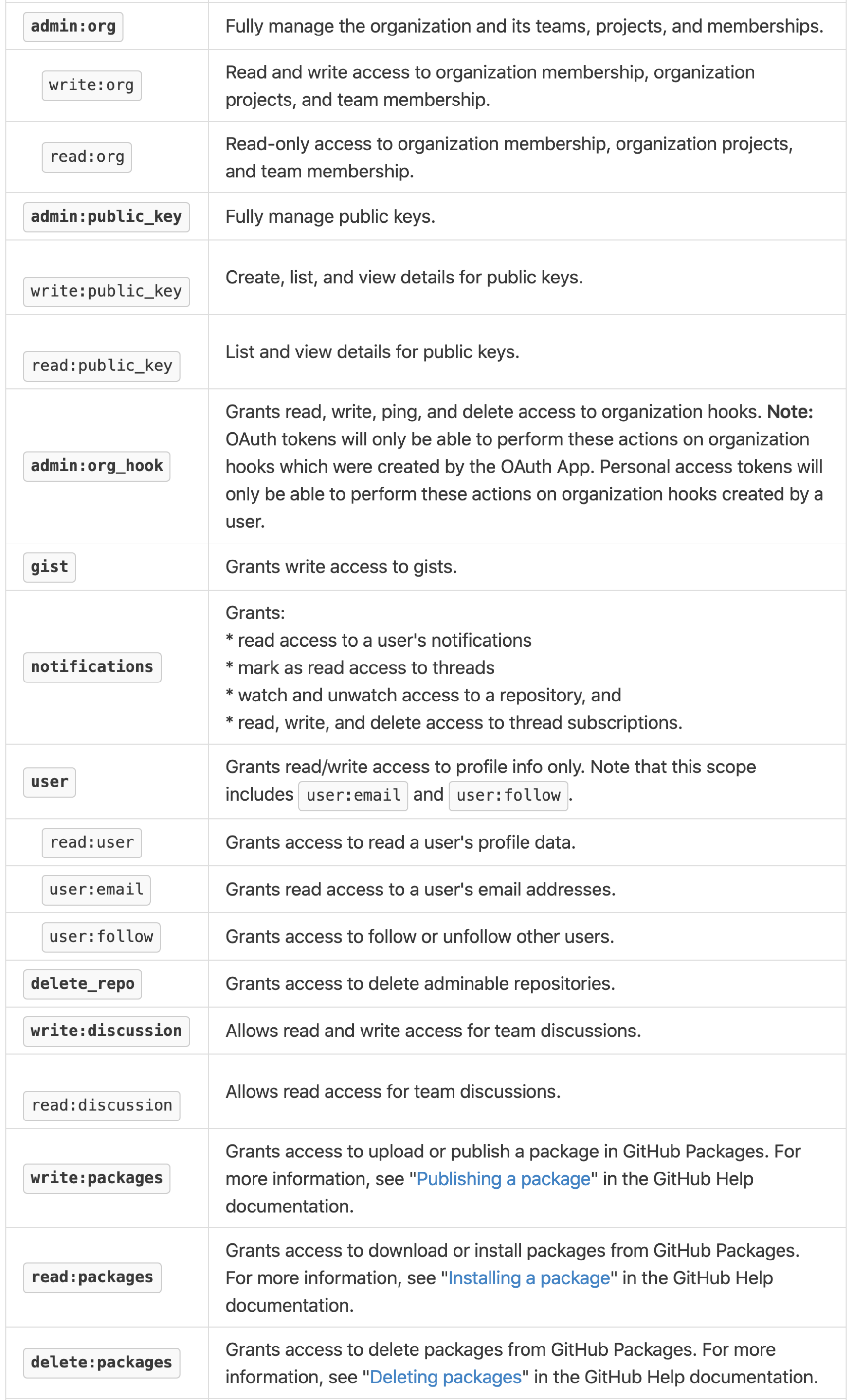
Scopes	
https://www.googleapis.com/auth/drive	See, edit, create, and delete all of your Google Drive files
https://www.googleapis.com/auth/drive.file	View and manage Google Drive files and folders that you have opened or created with this app
https://www.googleapis.com/auth/drive.readonly	See and download all your Google Drive files
https://www.googleapis.com/auth/presentations	View and manage your Google Slides presentations
https://www.googleapis.com/auth/presentations.readonly	View your Google Slides presentations
https://www.googleapis.com/auth/spreadsheets	See, edit, create, and delete your spreadsheets in Google Drive
https://www.googleapis.com/auth/spreadsheets.readonly	View your Google Spreadsheets

GitHub



Available scopes

Name	Description
`(no scope)`	Grants read-only access to public information (includes public user profile info, public repository info, and gists)
`repo`	Grants full access to private and public repositories. That includes read/write access to code, commit statuses, repository and organization projects, invitations, collaborators, adding team memberships, deployment statuses, and repository webhooks for public and private repositories and organizations. Also grants ability to manage user projects.
`repo:status`	Grants read/write access to public and private repository commit statuses. This scope is only necessary to grant other users or services access to private repository commit statuses *without* granting access to the code.
`repo_deployment`	Grants access to deployment statuses for public and private repositories. This scope is only necessary to grant other users or services access to deployment statuses, *without* granting access to the code.
`public_repo`	Limits access to public repositories. That includes read/write access to code, commit statuses, repository projects, collaborators, and deployment statuses for public repositories and organizations. Also required for starring public repositories.
`repo:invite`	Grants accept/decline abilities for invitations to collaborate on a repository. This scope is only necessary to grant other users or services access to invites *without* granting access to the code.
`security_events`	Grants read and write access to security events in the code scanning API.
`admin:repo_hook`	Grants read, write, ping, and delete access to repository hooks in public and private repositories. The `repo` and `public_repo` scopes grants full access to repositories, including repository hooks. Use the `admin:repo_hook` scope to limit access to only repository hooks.
`write:repo_hook`	Grants read, write, and ping access to hooks in public or private repositories.
`read:repo_hook`	Grants read and ping access to hooks in public or private repositories.
`admin:org`	Fully manage the organization and its teams, projects, and memberships.
`write:org`	Read and write access to organization membership, organization projects, and team membership.
`read:org`	Read-only access to organization membership, organization projects, and team membership.



`admin:org`	Fully manage the organization and its teams, projects, and memberships.
`write:org`	Read and write access to organization membership, organization projects, and team membership.
`read:org`	Read-only access to organization membership, organization projects, and team membership.
`admin:public_key`	Fully manage public keys.
`write:public_key`	Create, list, and view details for public keys.
`read:public_key`	List and view details for public keys.
`admin:org_hook`	Grants read, write, ping, and delete access to organization hooks. **Note:** OAuth tokens will only be able to perform these actions on organization hooks which were created by the OAuth App. Personal access tokens will only be able to perform these actions on organization hooks created by a user.
`gist`	Grants write access to gists.
`notifications`	Grants: * read access to a user's notifications * mark as read access to threads * watch and unwatch access to a repository, and * read, write, and delete access to thread subscriptions.
`user`	Grants read/write access to profile info only. Note that this scope includes `user:email` and `user:follow`.
`read:user`	Grants access to read a user's profile data.
`user:email`	Grants read access to a user's email addresses.
`user:follow`	Grants access to follow or unfollow other users.
`delete_repo`	Grants access to delete adminable repositories.
`write:discussion`	Allows read and write access for team discussions.
`read:discussion`	Allows read access for team discussions.
`write:packages`	Grants access to upload or publish a package in GitHub Packages. For more information, see "Publishing a package" in the GitHub Help documentation.
`read:packages`	Grants access to download or install packages from GitHub Packages. For more information, see "Installing a package" in the GitHub Help documentation.
`delete:packages`	Grants access to delete packages from GitHub Packages. For more information, see "Deleting packages" in the GitHub Help documentation.

@PhilippeDeRyck

Use scopes for function-level access control

Scopes define the authority to perform certain operations.

The API can rely on the presence of a certain scope in the access token claims for authorization purposes.

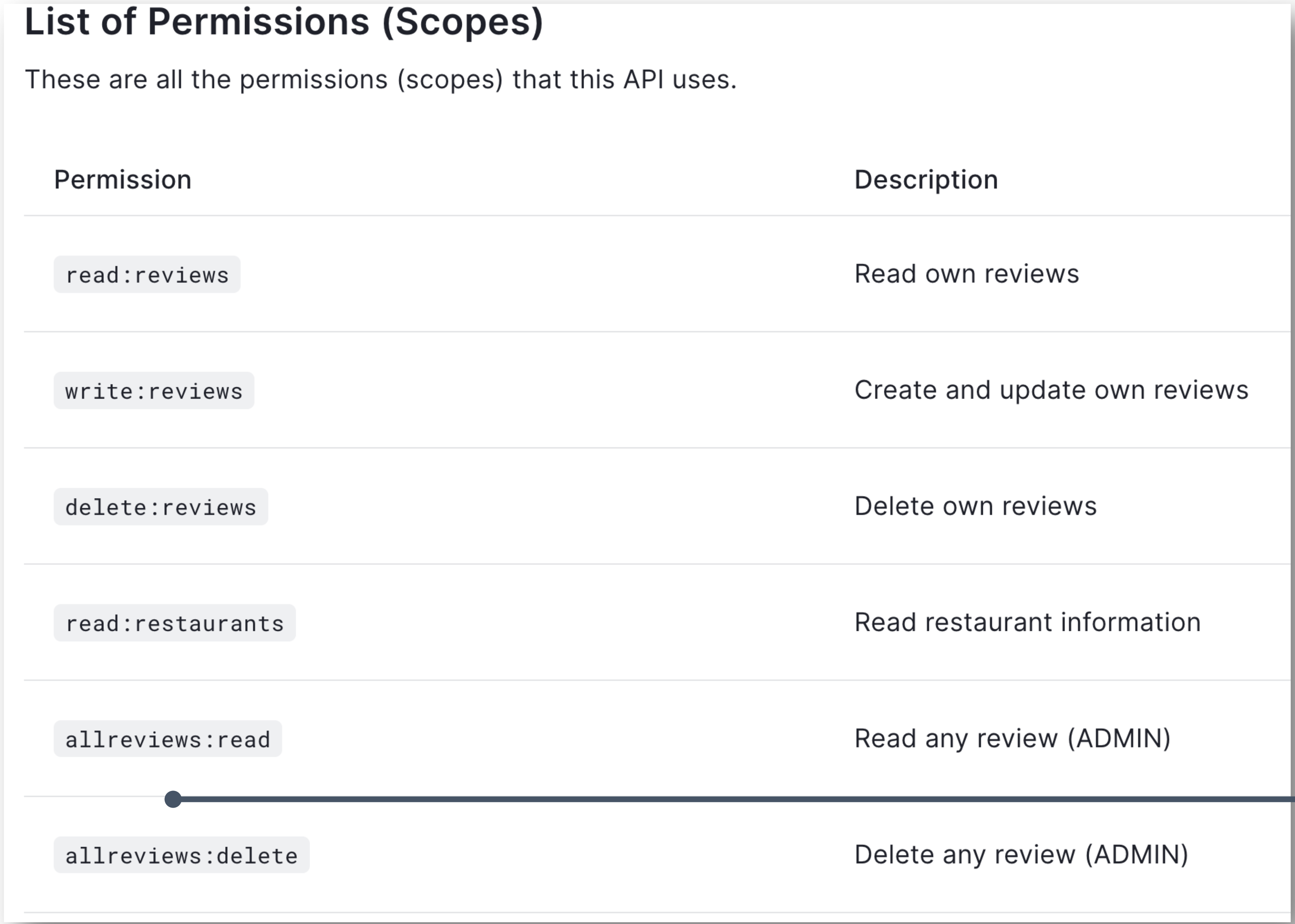
The use of custom permission claims

```
{
  "iss": "https://sts.restograde.com",
  "aud": "https://api.restograde.com",
  "sub": "5eb916c258bdb50bf20366c6",
  "azp": "OLKNn389SUufExgRG1RilLSgdYxwEp",
  "permissions": ["reviews:fullaccess"]
}
```

Permissions are typically used in a first-party scenario, where the authorization server enforces a specific authorization policy

OAuth 2.0 supports the use of custom claims in access tokens. A *permissions* claim is quite common to include concrete user or client permissions in a token

@PhilippeDeRyck


List of Permissions (Scopes)

These are all the permissions (scopes) that this API uses.

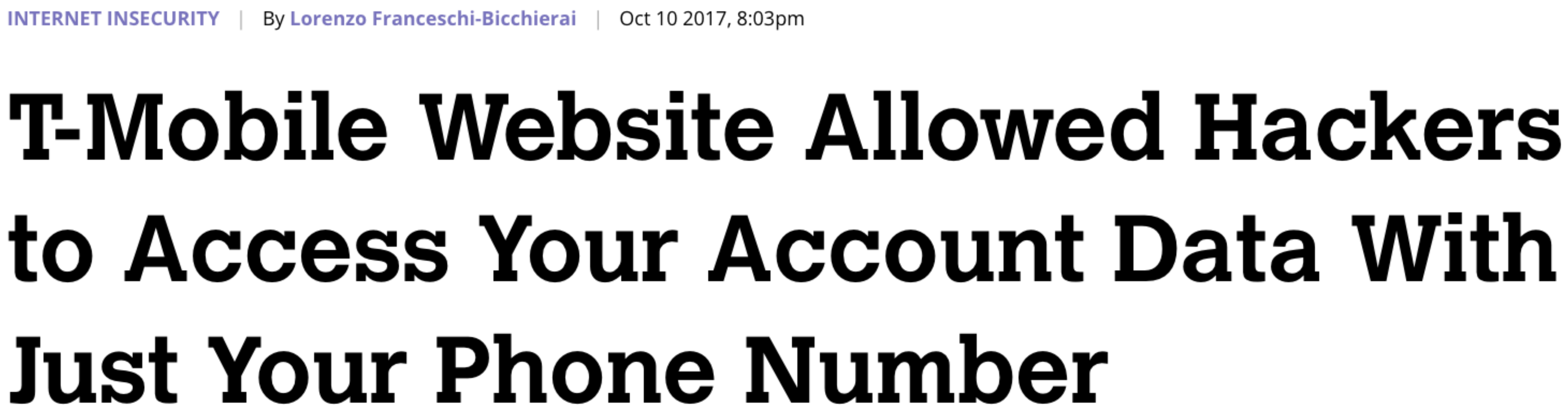
Permission	Description
`read:reviews`	Read own reviews
`write:reviews`	Create and update own reviews
`delete:reviews`	Delete own reviews
`read:restaurants`	Read restaurant information
`allreviews:read`	Read any review (ADMIN)
`allreviews:delete`	Delete any review (ADMIN)

The *allreviews:* permissions represent administrative access to the API

@PhilippeDeRyck

Custom permission claims offer more flexibility

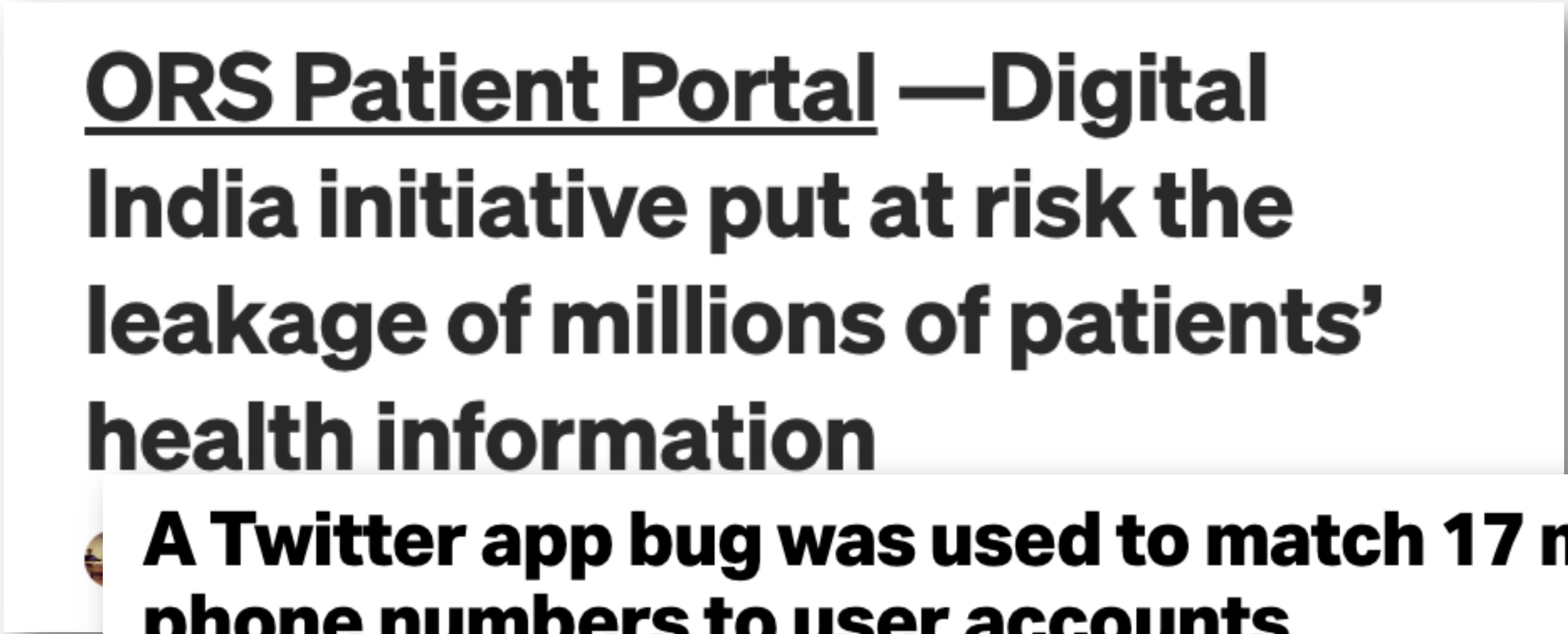
When the entire ecosystem is tightly controlled, the access token often includes client/user-specific permissions instead of coarse-grained scopes.



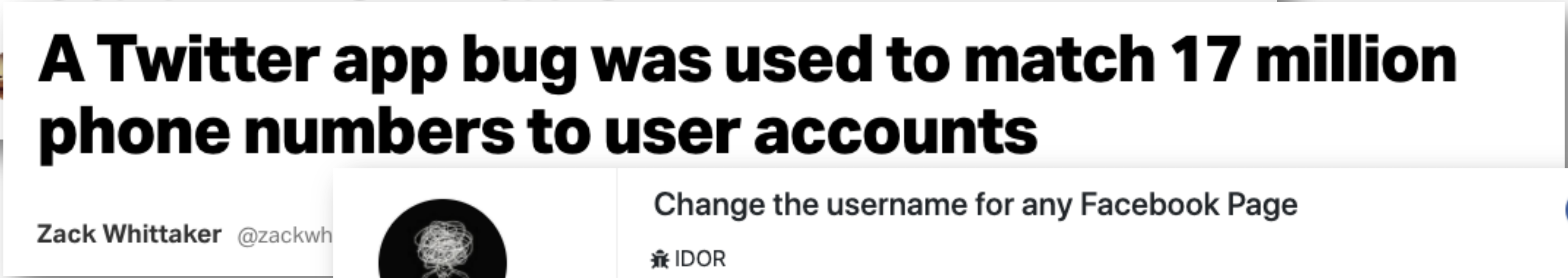
INTERNET INSECURITY | By Lorenzo Franceschi-Bicchierai | Oct 10 2017, 8:03pm

T-Mobile Website Allowed Hackers to Access Your Account Data With Just Your Phone Number

“he could query for someone else's phone number and the API would simply send back a response containing the other person's data.”

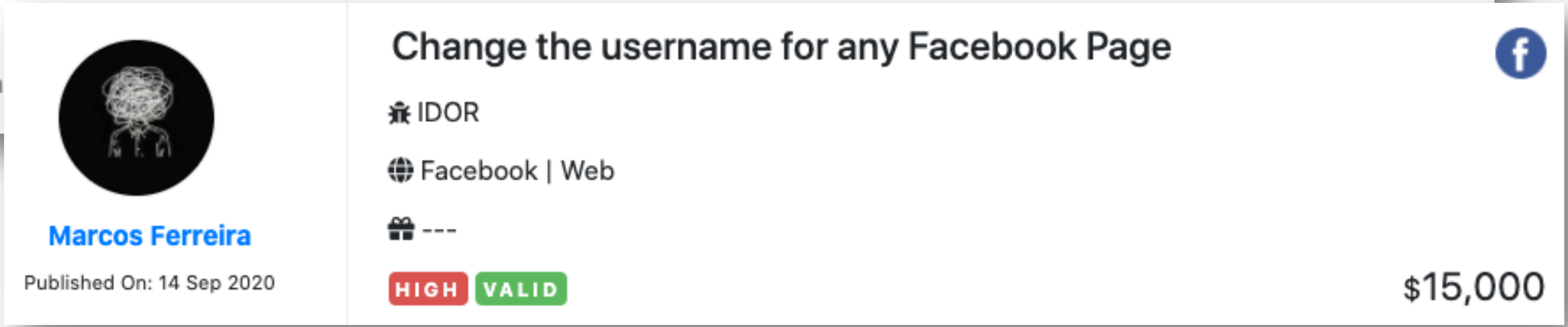


ORS Patient Portal —Digital India initiative put at risk the leakage of millions of patients' health information



A Twitter app bug was used to match 17 million phone numbers to user accounts

Zack Whittaker @zackwh



Marcos Ferreira

Published On: 14 Sep 2020

Change the username for any Facebook Page

IDOR

Facebook | Web

HIGH VALID

$15,000

https://techcrunch.com/2019/12/24/twitter-android-bug-phone-numbers/
https://bugreader.com/marcos@change-the-username-for-any-facebook-page-219
https://medium.com/@logicbomb_1/ors-patient-portal-digital-india-initiative-put-at-risk-the-leakage-of-millions-of-patients-7f093a1768e2

A user-specific access token

```
{
  "iss": "https://sts.restograde.com",
  "aud": "https://api.restograde.com",
  "sub": "5eb916c258bdb50bf20366c6",
  "azp": "0LKNn389SUufExgRG1RilLSgdYxwEp",
  "permissions": ["reviews:read"]
}
```

The sub claim contains the unique identifier of the user associated with the access token

Use the user's identifier to make access control decisions (e.g., author of the review) or to collect additional information for more advanced authorization decisions

The user has permission to read reviews, but the permission does not specify *which* reviews

Perform object-level authorization checks

Broken Object-Level Authorization is the #1 API security failure. Use the sub *claim to establish the user's identity and make sure the user is allowed to perform the requested operation on the specified object.*

An access token with custom user-specific claims

```
{
  "iss": "https://sts.restograde.com",
  "aud": "https://api.restograde.com",
  "sub": "5eb916c258bdb50bf20366c6",
  "azp": "0LKNn389SUufExgRG1RilLSgdYxwEp",
  "permissions": "restaurants:manage",
  "tenant_id": "5eb916c258bdb50bf20366c6",
  "tenant_name": "International Foodie Group"
}
```

Access token claims should be related to the user and should not API-specific authorization details

Access tokens can contain additional metadata about the user to enable authorization

Avoid overloading the access token with details

A common pitfall is to cram the access token full of API-specific details, causing you to replicate API-specific authorization policies.

Limit the information in the access token to user-specific claims and enforce authorization at the API.

An access token containing a tenant_id

```
{
  …
  "sub": "5eb916c258bdb50bf20366c6",
  "azp": "OLKNn389SUufExgRG1RilLSgdYxwEp",
  "permissions": "restaurants:manage",
  "tenant_id": "5eb916c258bdb50bf20366c6",
  "tenant_name": "International Foodie Group"
}
```

Failing to verify the validity of request parameters against a trusted source (e.g., access token claims) can result in authorization bypasses

A Java Spring endpoint returning a list of restaurants for a tenant

```
@PreAuthorize("hasPermission('restaurants:manage')")
@RequestMapping(path = "/tenants/{tenantId}/restaurants", methoc
public void getTenantRestaurants(String tenantId) {
  RestaurantService.listRestaurantsForTenant (tenantId);
}
```

Requests often contain path variables or query parameters that identify a user/tenant/customer/…

Using untrusted values for authorization

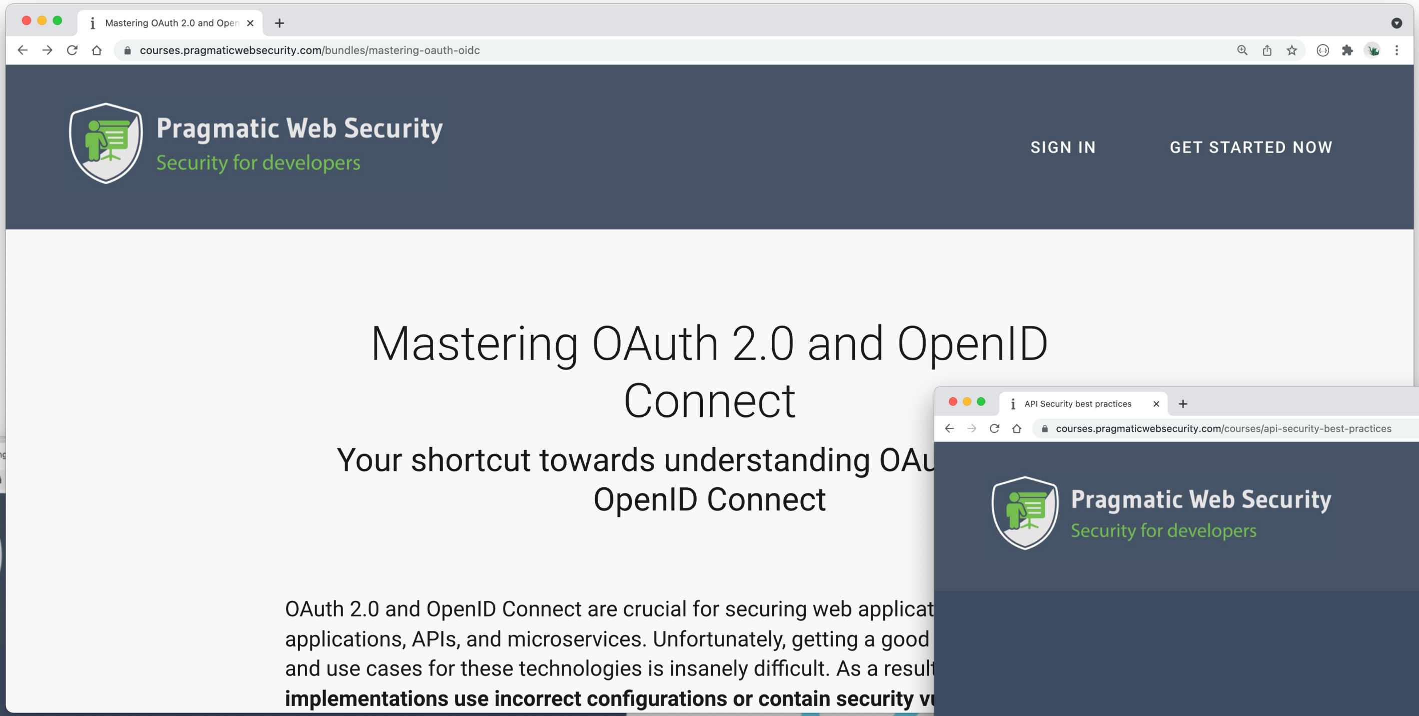
A common pitfall is to use an untrusted or unverified value for authorization purposes.

Ensure that every value from the request is properly checked against a trusted source.

KEY TAKEAWAYS

1. Use OAuth 2.0 to enable authorization in a complex architecture
2. Learn and respect the purpose of OAuth 2.0 flows and tokens
3. Enforce API authorization using information from the access token

Keep learning with these in-depth security courses!


Mastering OAuth 2.0 and Open

courses.pragmaticwebsecurity.com/bundles/mastering-oauth-oidc

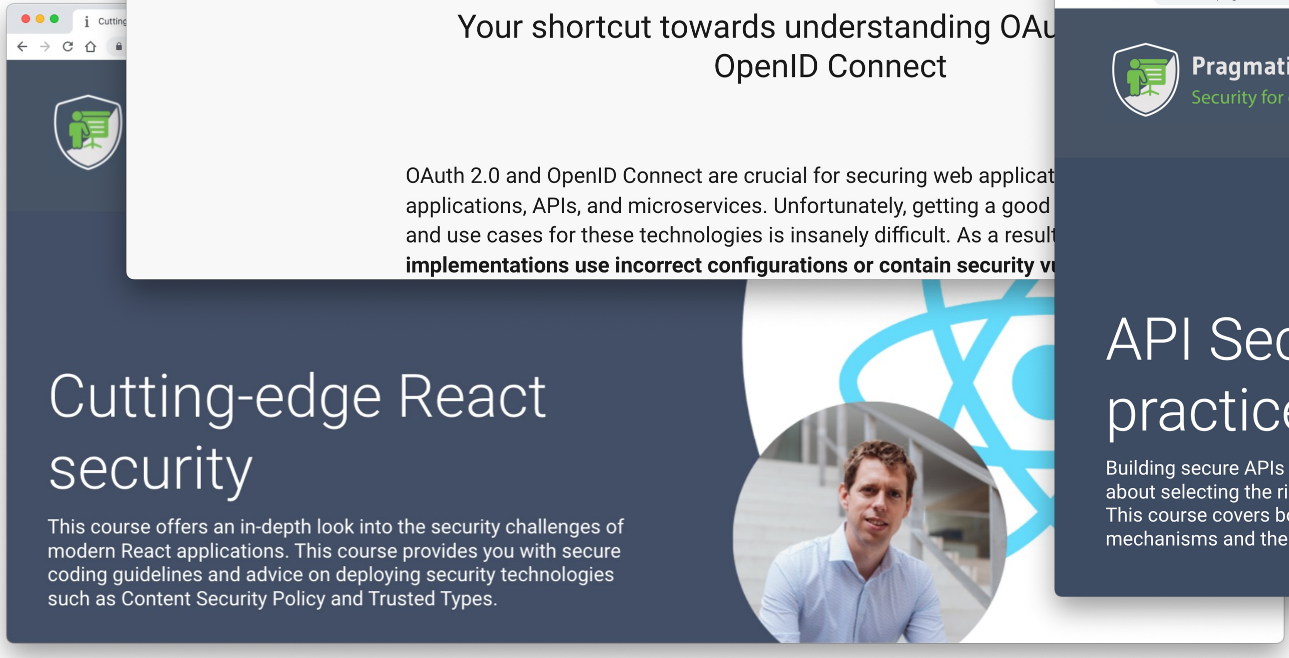
Pragmatic Web Security
Security for developers

SIGN IN GET STARTED NOW

Mastering OAuth 2.0 and OpenID Connect

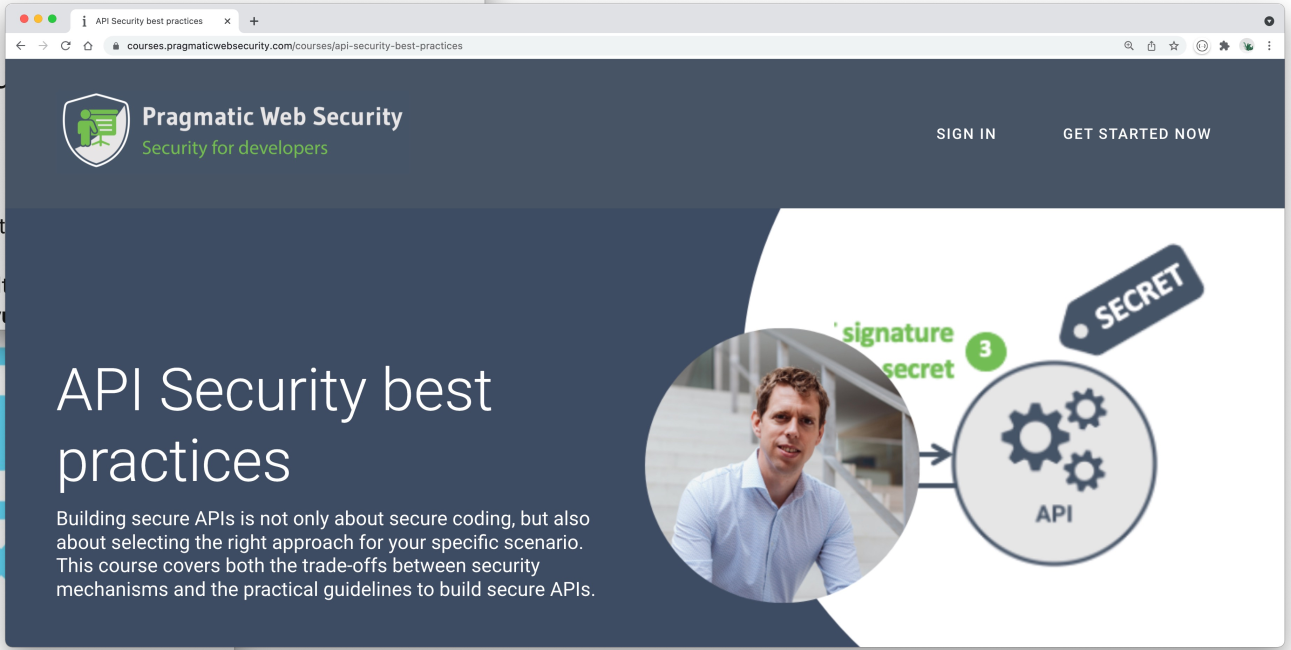
Your shortcut towards understanding OAu
OpenID Connect

OAuth 2.0 and OpenID Connect are crucial for securing web applicat
applications, APIs, and microservices. Unfortunately, getting a good
and use cases for these technologies is insanely difficult. As a result
implementations use incorrect configurations or contain security vu


Cutting

Cutting-edge React security

This course offers an in-depth look into the security challenges of modern React applications. This course provides you with secure coding guidelines and advice on deploying security technologies such as Content Security Policy and Trusted Types.


API Security best practices

courses.pragmaticwebsecurity.com/courses/api-security-best-practices

Pragmatic Web Security
Security for developers

SIGN IN GET STARTED NOW

API Security best practices

Building secure APIs is not only about secure coding, but also about selecting the right approach for your specific scenario. This course covers both the trade-offs between security mechanisms and the practical guidelines to build secure APIs.

HTTPS://COURSES.PRAGMATICWEBSECURITY.COM

Thank you!

Connect on social media
to stay in touch

@PhilippeDeRyck

/in/PhilippeDeRyck