Painless Authentication

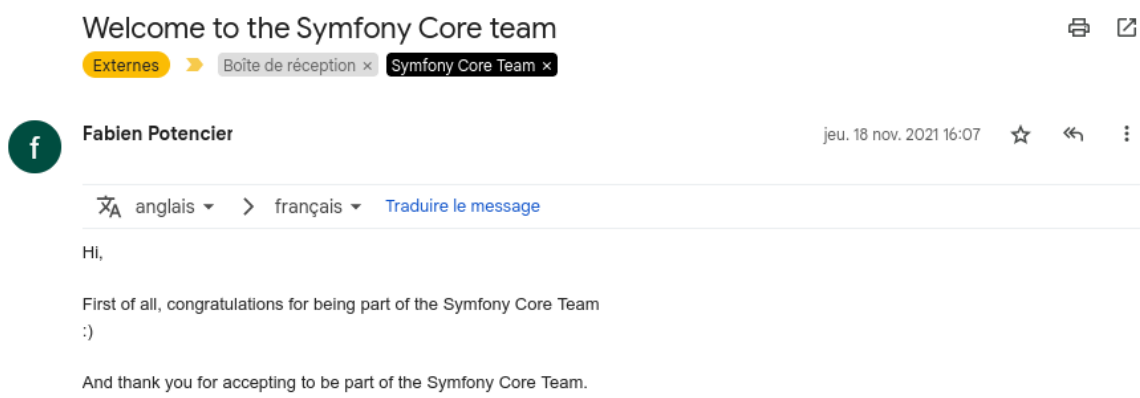
with Access Tokens

Photo by Kelly Sikkema on Unsplash

Mathieu Santostefano

- 👨‍💻 Web developer at <joliCode/>
- Symfony Core Team Member
- Twitter: @welcomattic
- GitHub: @welcomattic

🎉 Last year, on Nov. 18, I received an email from Fabien to join the Core Team



Welcome to the Symfony Core team

Externes Boîte de réception × Symfony Core Team ×

Fabien Potencier jeu. 18 nov. 2021 16:07

anglais › français Traduire le message

Hi,

First of all, congratulations for being part of the Symfony Core Team
:)

And thank you for accepting to be part of the Symfony Core Team.

🔐 Who has Access Token authentication in their apps?

🔒 Who is working with the new Symfony security?

🔒 Who is working with the new Symfony security?

Thank you for it, Wouter 🙏

The new Security System

- Removed everything but Guards
- Moved to an event-based system
- Passports and Badges
- Authenticator based

> Your job is to use or implement your own Authenticators

The new Security System

Event-based system

- You can interact on different levels
  - CheckPassportEvent
  - AuthenticationTokenCreatedEvent
  - AuthenticationSuccessEvent
  - LoginSuccessEvent
  - LoginFailureEvent
  - LogoutEvent
  - TokenDeauthenticatedEvent
  - SwitchUserEvent

The new Security System

Handle success or failure

- As before, you can still handle what happen in case of authentication sucess or failure

> 🤩 As many things since few years in Symfony, it improves the DX

What is an Access Token?

- `i-am-an-4cc3ss-t0k3n` could be an Access Token
- `mF_9.B5f-4.1JqM` could be an Access Token
- `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0I` could be an Access token

 What is an Access Token?

- `i-am-an-4cc3ss-t0k3n` could be an Access Token
- `mF_9.B5f-4.1JqM` could be an Access Token
- `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0I` could be an Access token

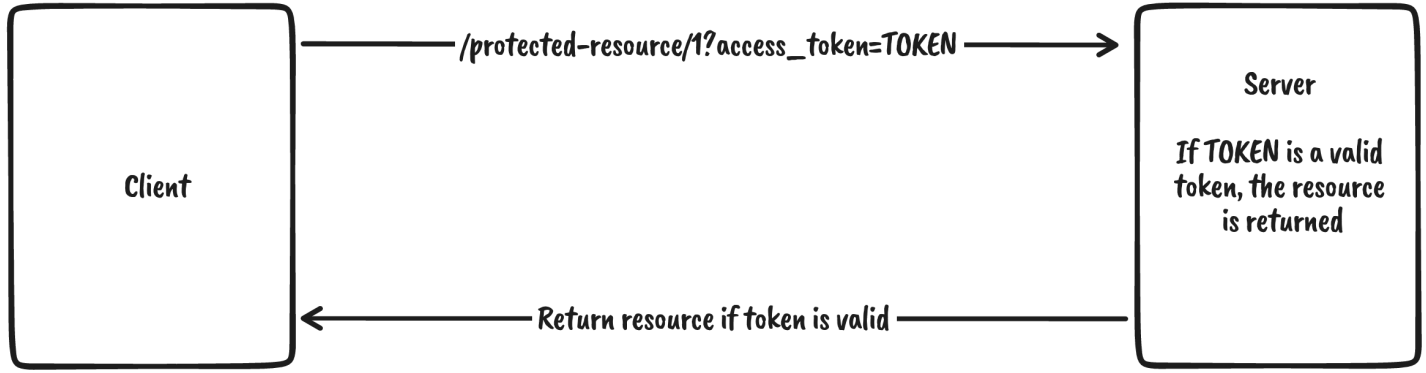
>  An Access Token is represented by a string.

Ok, but

- Must be sent in a HTTP request to fetch a protected resource
- The application expects to find a token, **verifies** it and decides to allow access or not

Ok, but

- Must be sent in a HTTP request to fetch a protected resource
- The application expects to find a token, **verifies** it and decides to allow access or not



Token issuer

Let’s assume our tokens comes from an external authorization server

✅ Verify the token?

The verification process is up to *you*

- Check if the string is present in a database
- Compute a hash and compare it to the expected one
- Decode the token (base64, ...) and make assertions on decoded values
- Call an external service to validate the token
- Verify the expiration date if needed
- Check if the token is revoked or not
- ...

Use cases

Authentication with Access Token is useful in some contexts, like

- Stateless user login
- API Gateway in front of private APIs
- Application that access to personal data provided by a third party
- Micro-services between them
- Client applications of a SaaS API
- ...

🤔 How to set up an Access Token auth with Symfony before 6.2?

🎬 The Project :
A private API for next Disney movies scenarios

- Form login authentication for users
- Users can create Scenarios
- Each users has an API key
- Users can request our private API with their API key
- We can revoke an API key at any time

With Symfony <= 6.1

- Create a custom Authenticator
  - Handle token extraction
  - Handle failure cases
  - Decode token if needed (JWT, SAML, ...)
  - Retrieve user identifier from the access token
  - Then load User object

> 👂 Have you heard about RFC6750?

With Symfony <= 6.1

- Handle token extraction
  - In request header?
    - Handle correctly the `Authorization` header with `Bearer` scheme
  - In query?
  - In request body?
- Fill up WWW-Authenticate response header in case of failure
- All requests must be done using HTTPS protocol to secure the token transport

With Symfony <= 6.1

```php
/** Simplified version */
private function extractToken(Request $req): ?string
{
    return match (true) {
        // Header
        $req->headers->has('Authorization') => str_replace('Bearer ', '', $req->headers->get('Authorization')),

        // Query string
        Request::METHOD_GET === $req->getMethod()
        && $req->query->has('access_token') => $req->query->get('access_token'),

        // Request body
        Request::METHOD_POST === $req->getMethod()
        && $req->request->has('access_token') => $req->request->get('access_token'),

        default => null,
    };
}
```

With Symfony <= 6.1

```php
public function authenticate(Request $request): Passport
{
    if (null === $apiKey = $this->extractToken($request)) {
        throw new AuthenticationException('No API Key provided');
    }

    /* Here's the token decoding if needed, then assertions on decoded values */

    return new SelfValidatingPassport(
        new UserBadge($apiKey, fn(string $apiKey) => $this->userRepository->findOneBy(['apiKey' => $apiKey]))
    );
}
```

ℹ️ We consider the access token as user identifier. In case of JWT, we must decode it to get the user identifier

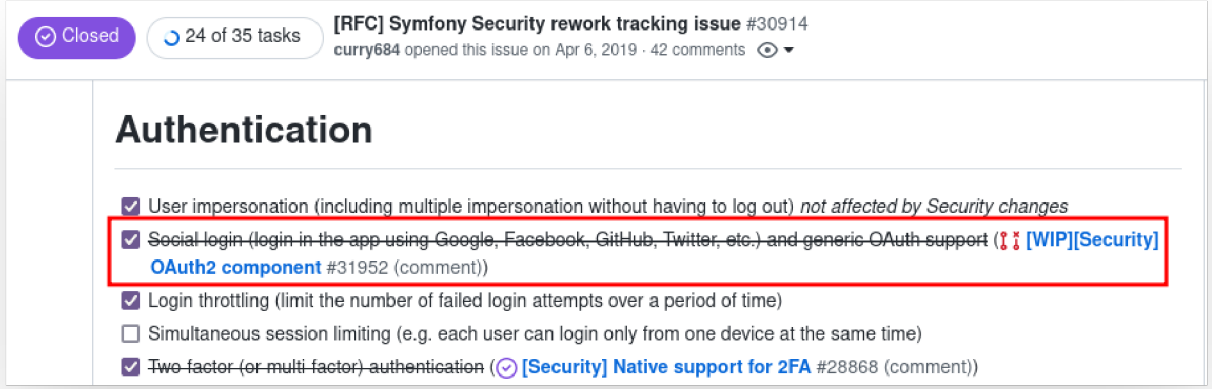
🥱 Boring code

- We have to repeat this code in all our applications, **boring**
- Our responsibility to respect RFC6750, **boring**
- No body likes boring code
- Poor **D**eveloper e**X**perience, Symfony tends to improve DX

This is definitely improvable

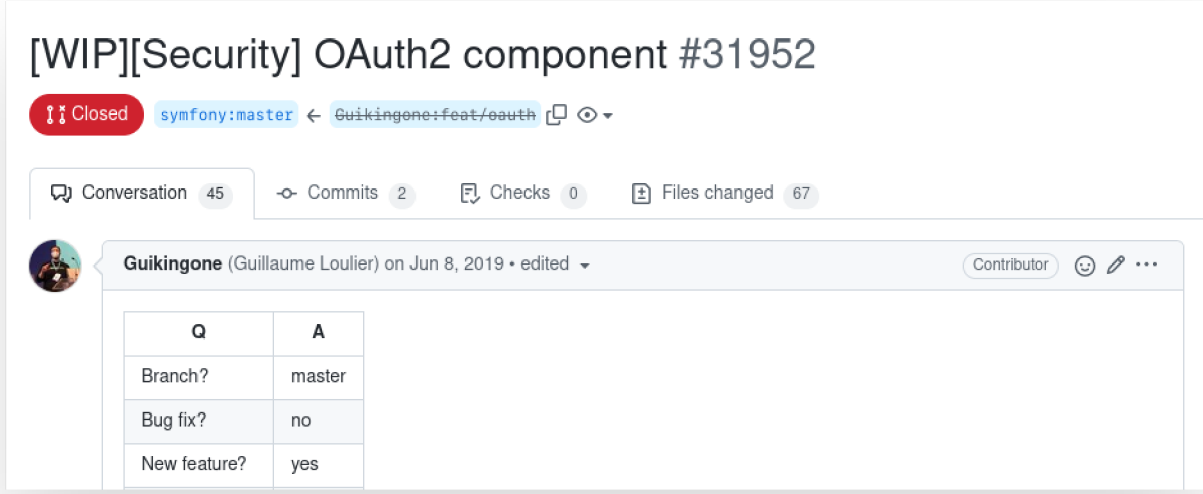
Discussions starts long time ago

- **April 2019**, at EU FOSSA Hackathon



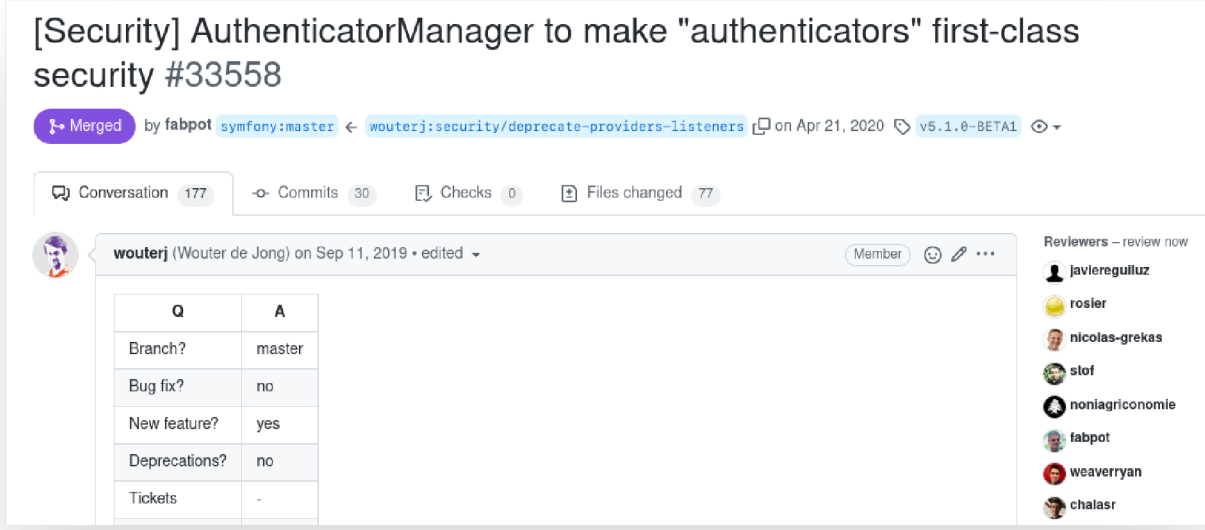
 Discussions starts long time ago

- **June 2019**, first PR about OAuth2 Component. ❌ Aborted



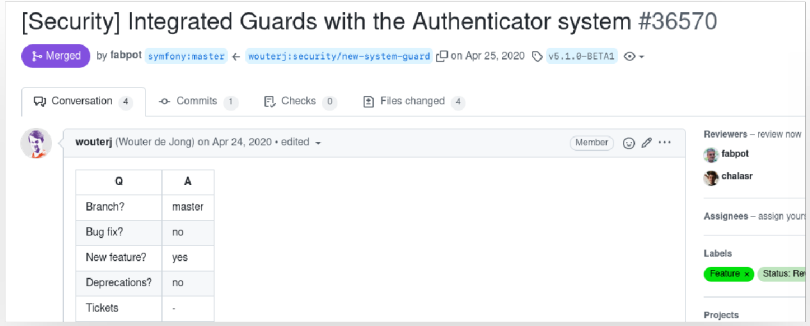
Discussions starts long time ago

- **September 2019**, Wouter's 1st PR about redesign of Security



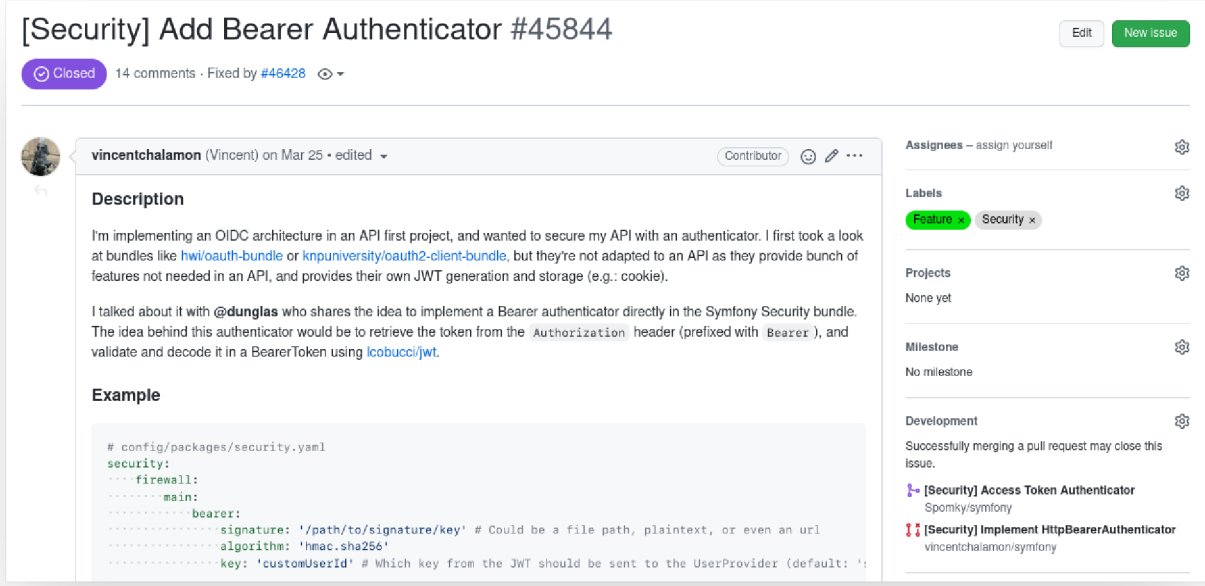
Discussions starts long time ago

- **April 2020**, continuation of Wouter's work on new Security system



Discussions starts long time ago

- **March 2022**, Vincent Chalamon opens an issue about "Bearer Authenticator"



[Security] Add Bearer Authenticator #45844

Edit | New issue

Closed · 14 comments · Fixed by #46428

vincentchalamon (Vincent) on Mar 25 • edited — Contributor

Description

I'm implementing an OIDC architecture in an API first project, and wanted to secure my API with an authenticator. I first took a look at bundles like hwi/oauth-bundle or knpuniversity/oauth2-client-bundle, but they're not adapted to an API as they provide bunch of features not needed in an API, and provides their own JWT generation and storage (e.g.: cookie).

I talked about it with **@dunglas** who shares the idea to implement a Bearer authenticator directly in the Symfony Security bundle. The idea behind this authenticator would be to retrieve the token from the `Authorization` header (prefixed with `Bearer`), and validate and decode it in a BearerToken using lcobucci/jwt.

Example

```
# config/packages/security.yaml
security:
    firewall:
        main:
            bearer:
                signature: '/path/to/signature/key' # Could be a file path, plaintext, or even an url
                algorithm: 'hmac.sha256'
                key: 'customUserId' # Which key from the JWT should be sent to the UserProvider (default: '
```

Assignees – assign yourself

Labels: Feature × Security ×

Projects: None yet

Milestone: No milestone

Development

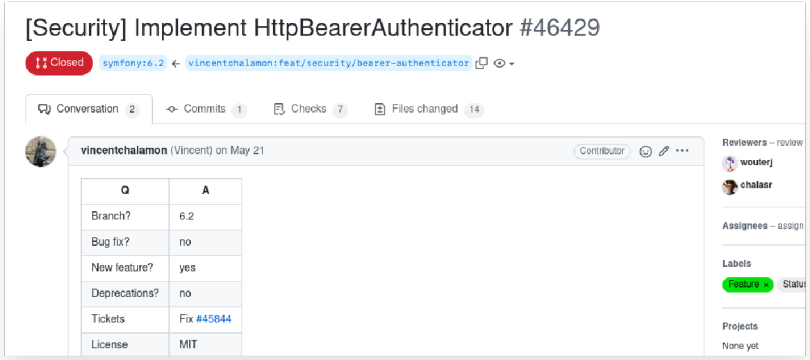
Successfully merging a pull request may close this issue.

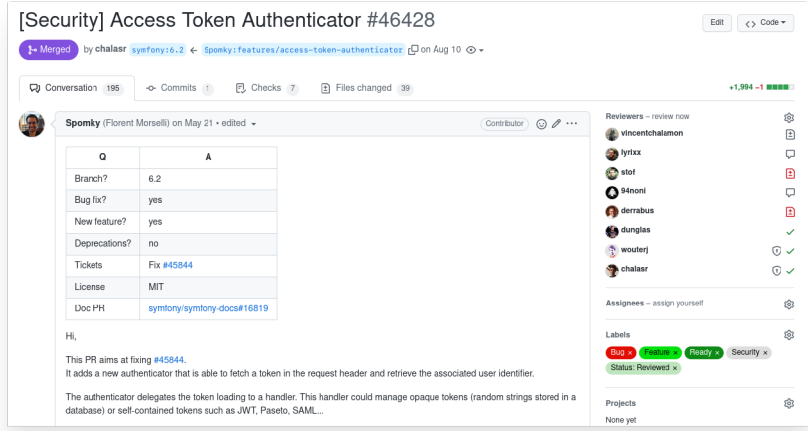
[Security] Access Token Authenticator
Spomky/symfony

[Security] Implement HttpBearerAuthenticator
vincentchalamon/symfony

Discussions starts long time ago

- Finally, on **21 May 2022**, Vincent Chalamon & Florent Morselli each open a PR to implement the Authenticator





🤩 Thanks to Florent Morselli  @Spomky

In Symfony 6.2 we won't have to worry about this code anymore!

[Security] Access Token Authenticator #46428

Edit | Code

Merged by chalasr `symfony:6.2` ← `Spomky:features/access-token-authenticator` on Aug 10

Conversation 195 | Commits 1 | Checks 7 | Files changed 39 | +1,994 −1

Spomky (Florent Morselli) on May 21 • edited — Contributor

Q	A
Branch?	6.2
Bug fix?	yes
New feature?	yes
Deprecations?	no
Tickets	Fix #45844
License	MIT
Doc PR	symfony/symfony-docs#16819

Hi,

This PR aims at fixing #45844.
It adds a new authenticator that is able to fetch a token in the request header and retrieve the associated user identifier.

The authenticator delegates the token loading to a handler. This handler could manage opaque tokens (random strings stored in a database) or self-contained tokens such as JWT, Paseto, SAML...

Reviewers – review now
vincentchalamon
lyrixx
stof
94noni
derrabus
dunglas
wouterj
chalasr

Assignees – assign yourself

Labels
Bug × | Feature × | Ready × | Security × | Status: Reviewed ×

Projects
None yet

🚀 Let's meet AccessTokenAuthenticator in Symfony 6.2

AccessTokenAuthenticator

- Takes care of token extraction
  - Header
  - Query string
  - Request body
  - Or Custom extractors
- Calls **your** Token Handler to check the token (revocation, expiration, signature, ...)
- Custom success / failure handlers if needed

> ✨ All this via configuration!

What kind of tokens are supported?

- JWT
- SAML2
- Biscuit
- Macaroons
- Homemade tokens
- ...
- Any kind of token, as it's up to you to handle them

How much easier is it than in 6.1?

6.1 security.yaml

```yaml
security:
  providers:
    client_app_provider:
      entity:
        class: App\Entity\App
        property: apiKey
  firewalls:
    api:
      pattern: ^/api
      lazy: true
      provider: client_app_provider
      custom_authenticator: App\Security\ApiKeyAuther
```

6.2 security.yaml

```yaml
security:
  providers:
    client_app_provider:
      entity:
        class: App\Entity\App
        property: apiKey
  firewalls:
    api:
      pattern: ^/api
      lazy: true
      provider: client_app_provider
      access_token:
        token_extractors:
          - 'security.access_token_extractor.query_st
          - 'security.access_token_extractor.request_
          - 'security.access_token_extractor.header'
        token_handler: App\Security\AccessTokenHandle
```

```php
/** 6.1 **/
class ApiKeyAuthenticator extends AbstractAuthenticator
{
    public function __construct(
        private readonly AppRepository $repository,
    ) {}

    public function supports(Request $request): ?bool
    {
        return $request->headers->has('Authorization')
            || $request->query->has('access_token')
            || $request->request->has('access_token');
    }

    public function authenticate(Request $request): Passport
    {
        $apiKey = $this->extractToken($request);

        if (null === $apiKey) {
            throw new AuthenticationException('No API Key provided');
        }

        // In case of JWT, decode it here

        return new SelfValidatingPassport(
            new UserBadge(
                $apiKey,
                function (string $apiKey) {
                    return $this->repository->findOneBy(['apiKey' => $apiKey, 'revoked' => false]);
                }
```

```php
/** 6.1 **/
class ApiKeyAuthenticator extends AbstractAuthenticator
{
    public function __construct(
        private readonly AppRepository $repository,
    ) {}

    public function supports(Request $request): ?bool
    {
        return $request->headers->has('Authorization')
            || $request->query->has('access_token')
            || $request->request->has('access_token');
    }

    public function authenticate(Request $request): Passport
    {
        $apiKey = $this->extractToken($request);

        if (null === $apiKey) {
            throw new AuthenticationException('No API Key provided');
        }

        // In case of JWT, decode it here

        return new SelfValidatingPassport(
            new UserBadge(
                $apiKey,
                function (string $apiKey) {
                    return $this->repository->findOneBy(['apiKey' => $apiKey, 'revoked' => false]);
                }
```

```php
/** 6.1 **/
class ApiKeyAuthenticator extends AbstractAuthenticator
{
    public function __construct(
        private readonly AppRepository $repository,
    ) {}

    public function supports(Request $request): ?bool
    {
        return $request->headers->has('Authorization')
            || $request->query->has('access_token')
            || $request->request->has('access_token');
    }

    public function authenticate(Request $request): Passport
    {
        $apiKey = $this->extractToken($request);

        if (null === $apiKey) {
            throw new AuthenticationException('No API Key provided');
        }

        // In case of JWT, decode it here

        return new SelfValidatingPassport(
            new UserBadge(
                $apiKey,
                function (string $apiKey) {
                    return $this->repository->findOneBy(['apiKey' => $apiKey, 'revoked' => false]);
                }
```

```php
/** 6.1 **/
class ApiKeyAuthenticator extends AbstractAuthenticator
{
    public function __construct(
        private readonly AppRepository $repository,
    ) {}

    public function supports(Request $request): ?bool
    {
        return $request->headers->has('Authorization')
            || $request->query->has('access_token')
            || $request->request->has('access_token');
    }

    public function authenticate(Request $request): Passport
    {
        $apiKey = $this->extractToken($request);

        if (null === $apiKey) {
            throw new AuthenticationException('No API Key provided');
        }

        // In case of JWT, decode it here

        return new SelfValidatingPassport(
            new UserBadge(
                $apiKey,
                function (string $apiKey) {
                    return $this->repository->findOneBy(['apiKey' => $apiKey, 'revoked' => false]);
                }
```

AccessTokenHandler 6.2

```php
class AccessTokenHandler implements AccessTokenHandlerInterface
{
    public function __construct(
        private readonly AppRepository $repository,
    ) {}

    public function getUserIdentifierFrom(string $token): string
    {
        // In case of JWT, decode it here

        $app = $this->repository->findOneBy(['apiKey' => $token]);

        if ($app === null || $app->isRevoked()) {
            throw new BadCredentialsException('Invalid credentials.');
        }

        return $token;
    }
}
```

TODO

Thank you ☕ 🥐

Any questions?

qrcode, slide link etc