30
Mars 2017

JSON Web Token Sécurisez vos APIs

Présentation par André Tapia

webnet
intégrateur de solutions digitales

SymfonyLive
PARIS 2017
30-31 MARS

SymfonyLive
PARIS 2017
30-31 MARS

ORDRE DU JOUR

1 /

Introduction

2 /

JWT: JSON Web Token

3 /

Intégration dans Symfony

SymfonyLive
PARIS 2017
30-31 MARS

QUI SUIS-JE ?

Qui suis-je ?

André Tapia

@dedeparisg

Architecte technique chez webnet
depuis 2011

+5 ans d'expérience sur une quinzaine de projets Symfony2 de tous types

SymfonyLive
PARIS 2017
30-31 MARS

1/3

/// Première partie

INTRODUCTION

Définition d'une API Web

Une API Web :

- expose de **l'information potentiellement critique**
- permet de manipuler cette information

Une API Web :

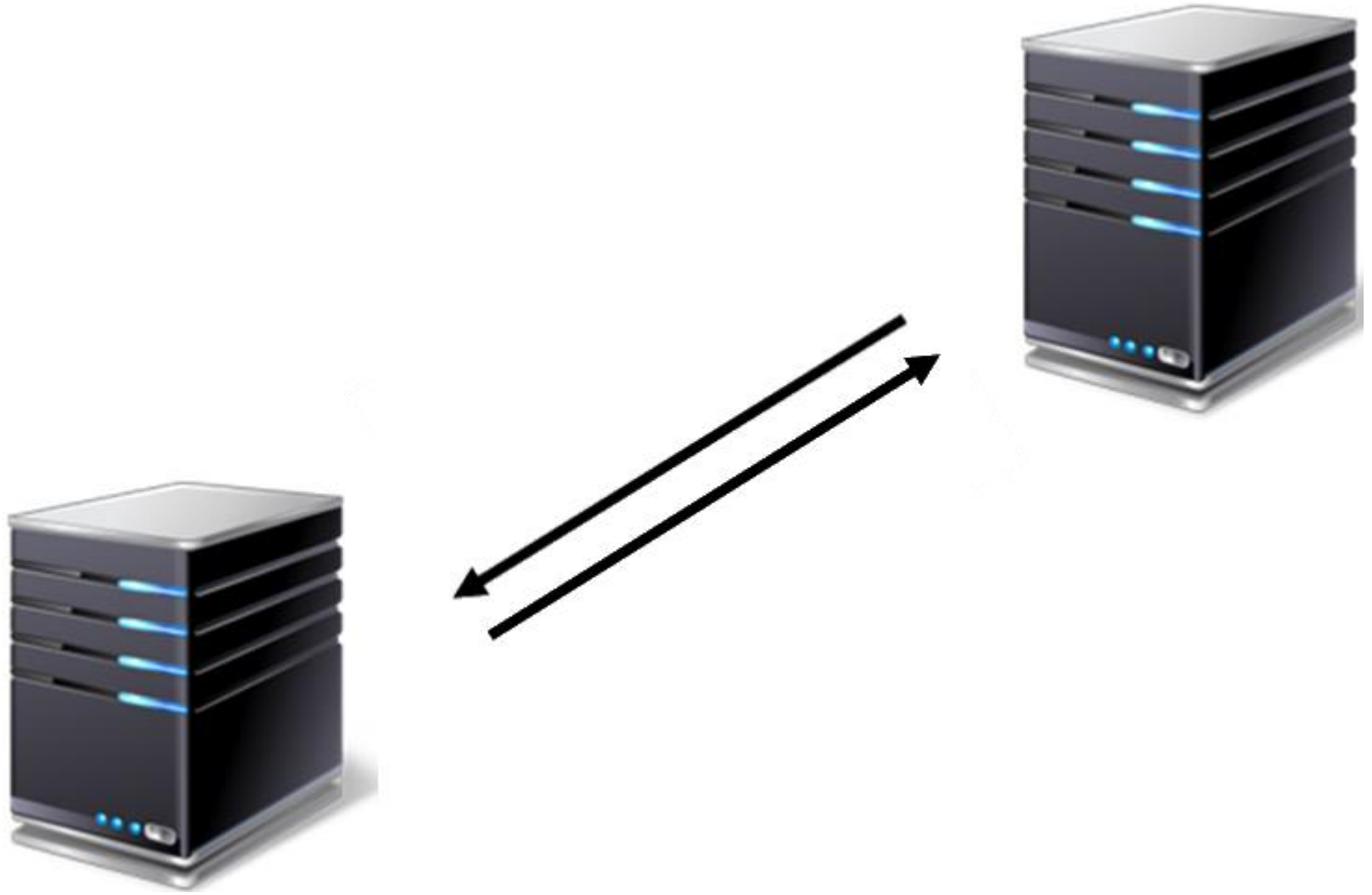
- expose de **l'information potentiellement critique**
- permet de manipuler cette information


Edition

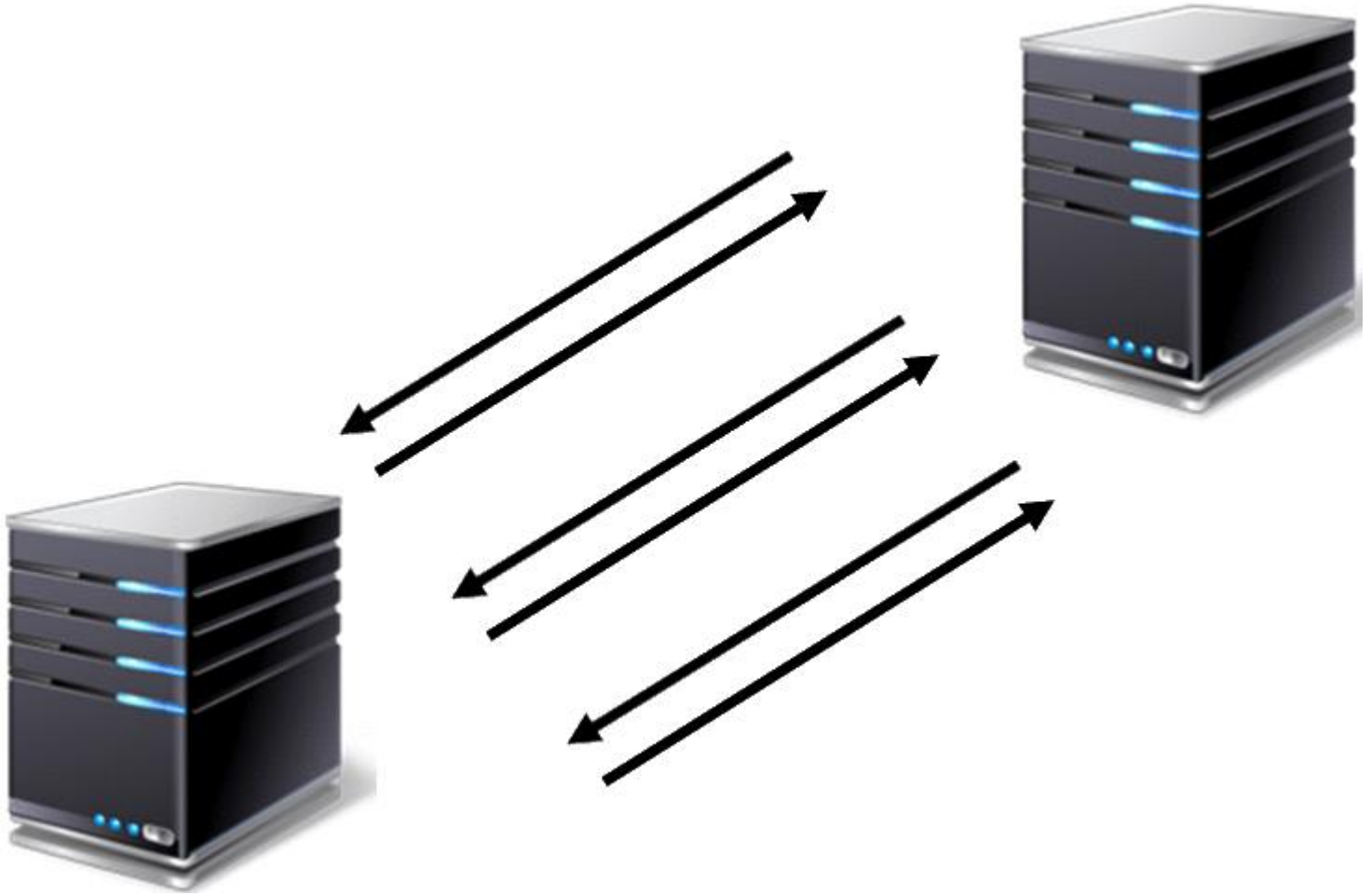

Ajout


Suppression

Sécurité d'une API Web



Sécurité d'une API Web



CRUCIAL de veiller à une **sécurité** accrue de chaque API

Une API Web :

- expose de **l'information potentiellement critique**
- permet de manipuler cette information
- est normalement **stateless**
  - ✓ Pas de session
  - ✓ Appel isolé

Une API Web :

- expose de **l'information potentiellement critique**
- permet de manipuler cette information
- est normalement **stateless**
  - ✓ Pas de session
  - ✓ Appel isolé

Une API Web :

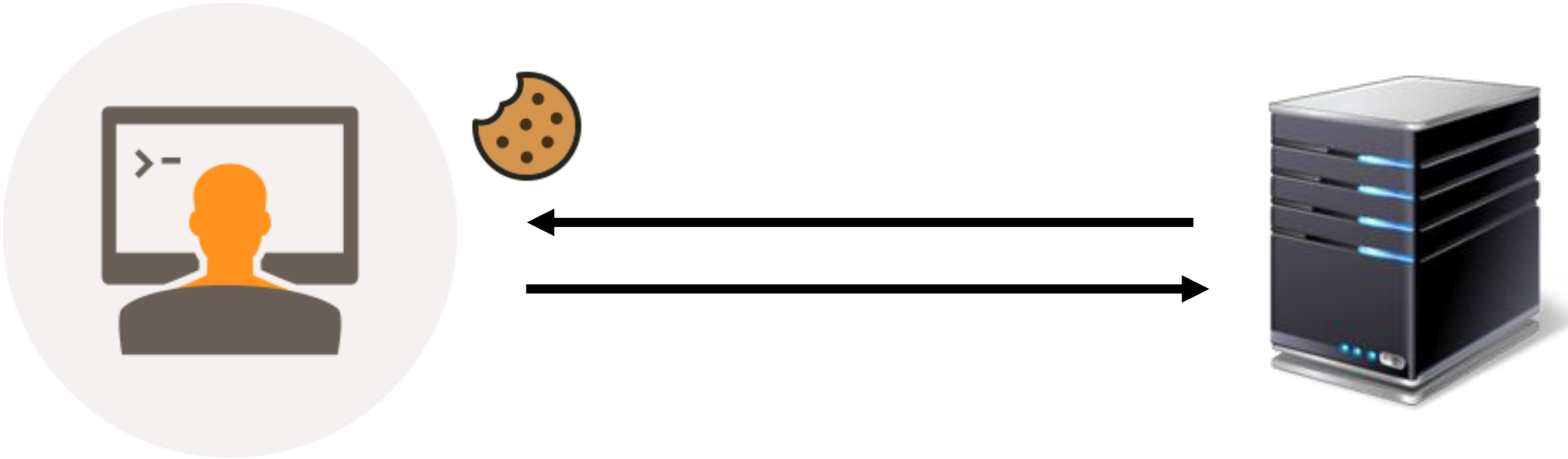
- expose de **l'information potentiellement critique**
- permet de manipuler cette information
- est normalement **stateless**
  - ✓ Pas de session
  - ✓ Appel isolé
  - ✓ Authentification à chaque appel

Une API Web :

- expose de **l'information potentiellement critique**
- permet de manipuler cette information
- est normalement **stateless**
- doit être utilisée en **HTTPS**

Quelles solutions pour sécuriser une API Web

- Authentification basée sur la session



Inconvénients

- CORS (Cross-origin resource sharing)
- Évolutivité

- Authentification basée sur les clefs d'API

⊕ Pas de session

- Authentification basée sur les clefs d'API

⊕ Pas de session

⊖ Gestion des clefs en bdd

id	username	…	api_key
1	andre	…	z654df84sSdDLfs3
2	amine	…	Ohg2v5x6df2fFspoa1fdffds8
3	antoine	…	khHp5se8w2xf1t9823tz3

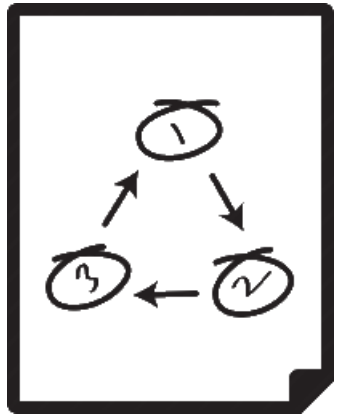
- Authentification basée sur les clefs d'API

- ⊕ Pas de session
- ⊖ Gestion des clefs en bdd
- ⊖ Pas de mécanisme d'expiration

- Authentification basée sur les clefs d'API

- ⊕ Pas de session
- ⊖ Gestion des clefs en bdd
- ⊖ Pas de mécanisme d'expiration
- ⊖ Token non exploitable

Solution idéale :



- Stateless
- Gestion de l'expiration
- Auto-porteuse et sécurisée

SymfonyLive
PARIS 2017
30-31 MARS

2/ 3

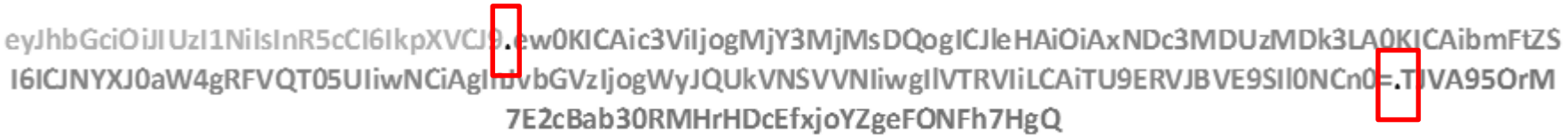
/// Seconde partie

JWT: JSON WEB TOKEN

Présentation

- Standard industriel qui repose sur une RFC (7519)
- Permet de fournir un mécanisme d'authentification fiable
- Repose sur un token qui va contenir les données
- Token sécurisé
  - JWS (RFC 7515)
  - JWE (RFC 7516)
- Fournit un système d'expiration

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ew0KICAic3ViIjogMjY3MjMsDQogICJleHAiOiAxNDc3MDUzMDk3LA0KICAibmFtZS
I6ICJNYXJ0aW4gRFVQT05UIiwNCiAgInJvbGVzIjogWyJQUkVNSVVNIiwgIlVTRVIiLCAiTU9ERVJBVE9SIl0NCn0=.TJVA95OrM
7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ


eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ew0KICAic3ViIjogMjY3MjMsDQogICJleHAiOiAxNDc3MDUzMDk3LA0KICAibmFtZS
I6ICJNYXJ0aW4gRFVQT05UIiwNCiAgIm bGVzIjogWyJQUkVNSVVNIiwgIlVTRVIiLCAiTU9ERVJBVE9SIl0NCn0=.TJVA95OrM
7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

Header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ew0KICAic3ViIjogMjY3MjMsDQogICJleHAiOiAxNDc3MDUzMDk3LA0KICAibmFtZSI6ICJNYXJ0aW4gRFVQT05UIiwNCiAgInJvbGVzIjogWyJQUkVNSVVNIiwgIlVTRVIiLCAiTU9ERVJBVE9SIl0NCn0=.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

Header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Payload

```
{
  "sub": 26723,
  "exp": 1477053097,
  "name": "Martin DUPONT",
  "roles": [
    "PREMIUM",
    "USER",
    "MODERATOR"
  ]
}
```

Liste des propriétés réservées :

Nom: **sub**
Description: Subject

Nom: **exp**
Description: Expiration Time

Nom: **nbf**
Description: Not Before

Nom: **iat**
Description: Issued At

Nom: **iss**
Description: Issuer

Nom: **aud**
Description: Audience

Nom: **jti**
Description: JWT ID

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ew0KICAic3ViIjogMjY3MjMsDQogICJleHAiOiAxNDc3MDUzMDk3LA0KICAibmFtZSI6ICJNYXJ0aW4gRFVQT05UIiwNCiAgInJvbGVzIjogWyJQUkVNSVVNIiwgIlVTRVIiLCAiTU9ERVJBVE9SIl0NCn0=.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

Header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Payload

```
{
  "sub": 26723,
  "exp": 1477053097,
  "name": "Martin DUPONT",
  "roles": [
    "PREMIUM",
    "USER",
    "MODERATOR"
  ]
}
```

Signature

```
HMACSHA256(
  base64UrlEncode(Header) +
  "." +
  base64UrlEncode(Payload),
  secret
)
```

JOSE : Javascript Object Signing and Encryption

HMAC + SHA	RSA + SHA	ECDSA + SHA
HS256	RS256	ES256
HS384	RS384	ES384
HS512	RS512	ES512

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ew0KICAic3ViIjogMjY3MjMsDQogICJleHAiOiAxNDc3MDUzMDk3LA0KICAibmFtZSI6ICJNYXJ0aW4gRFVQT05UIiwNCiAgInJvbGVzIjogWyJQUkVNSVVNIiwgIlVTRVIiLCAiTU9ERVJBVE9SIl0NCn0=.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

Header

```
{
  "alg": "none",
  "typ": "JWT"
}
```

Payload

```
{
  "sub": 26723,
  "exp": 1477053097,
  "name": "Martin DUPONT",
  "roles": [
    "PREMIUM",
    "USER",
    "MODERATOR"
  ]
}
```

- Implémentation disponible pour la grande majorité des langages de développement

PHP

Sign	HS256
Verify	HS384
`iss` check	HS512
`sub` check	RS256
`aud` check	RS384
`exp` check	RS512
`nbf` check	ES256
`iat` check	ES384
`jti` check	ES512

Spomky ☆145 View Repo

```
composer require spomky-labs/jose
```

PHP

MINIMUM VERSION 2.0.0

Sign	HS256
Verify	HS384
`iss` check	HS512
`sub` check	RS256
`aud` check	RS384
`exp` check	RS512
`nbf` check	ES256
`iat` check	ES384
`jti` check	ES512

Firebase ☆1735 View Repo

```
composer require firebase/php-jwt
```

```
composer require namshi/jose
```

```
composer require lcobucci/jwt
```

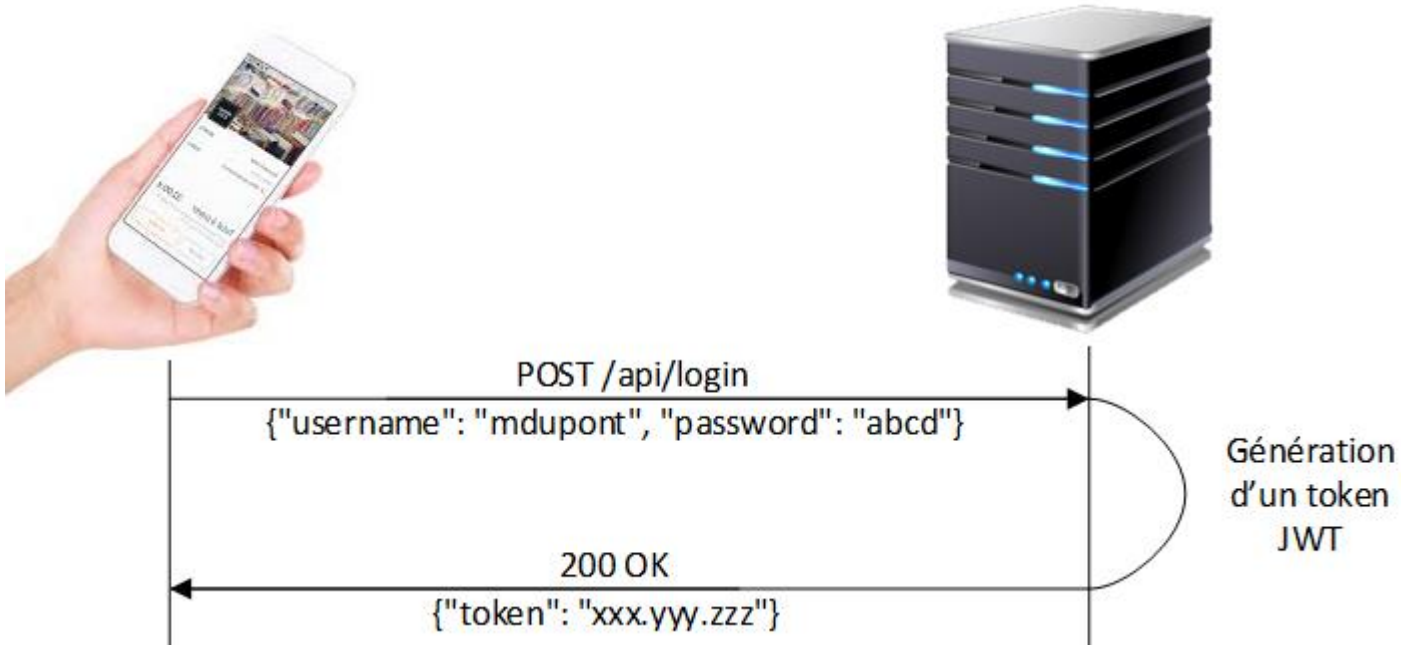
```
composer require emarref/jwt
```

```
composer require gree/jose
```

Exemple d'une application mobile utilisant une API

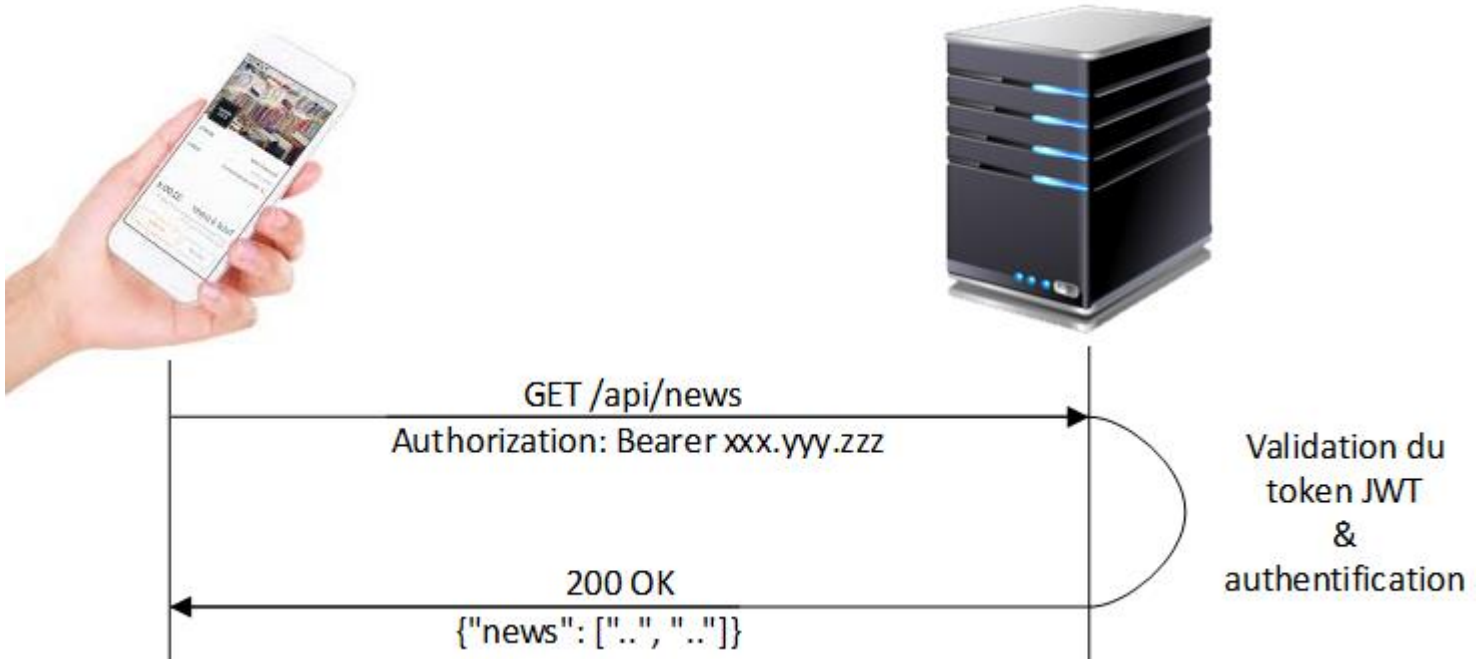
Etape 1 :

- L'utilisateur va s'authentifier sur l'API
- En cas d'authentification réussie, le serveur génère et renvoie un token JWT à l'application



Etape 2 à N :

- L'application transmet le token JWT pour chaque transaction suivante en header des requêtes

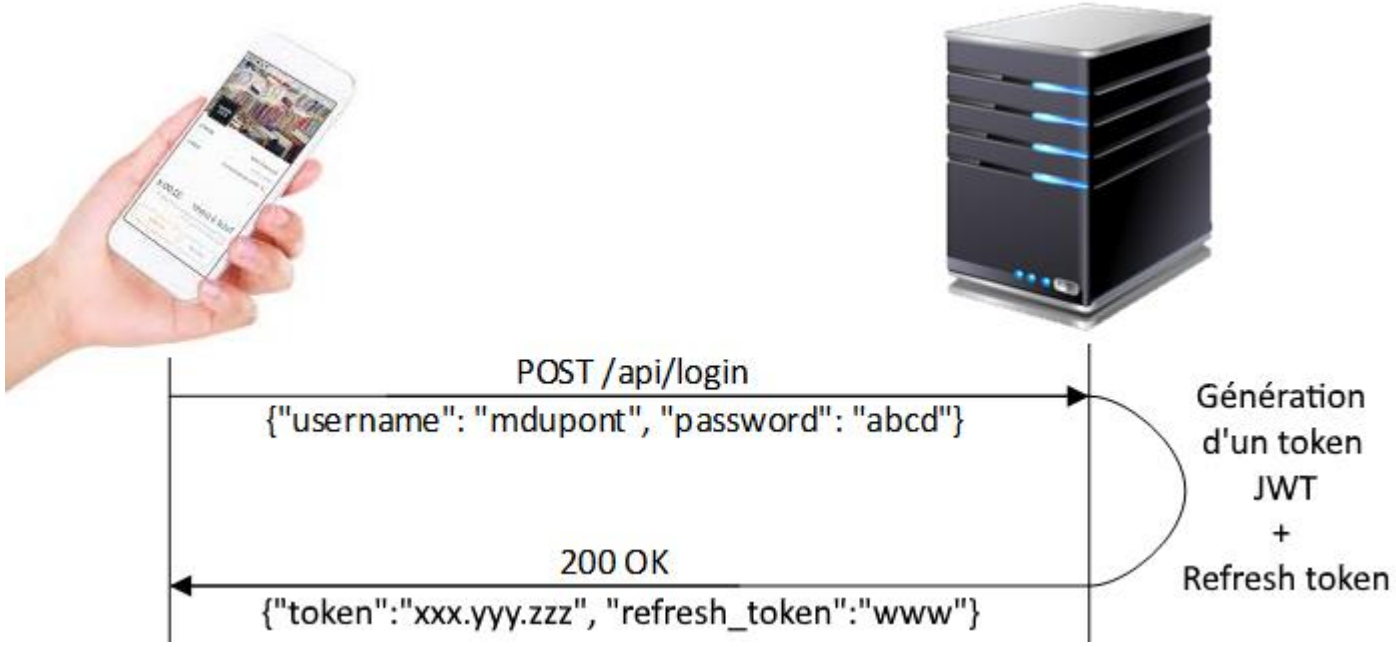


Gestion de l'expiration

Quelle durée choisir ?

- Pas de durée type
- En moyenne : entre 5 min et 1 heure
- Délai expiré :

Utilisation de Refresh token



Gestion de l'expiration

Gestion de l'expiration

SymfonyLive
PARIS 2017
30-31 MARS

3/ 3

/// Troisième partie

INTÉGRATION DANS UN PROJET SYMFONY


SymfonyLive
PARIS 2017
30-31 MARS
AbstractGuardAuthenticator

AbstractGuardAuthenticator


```php
namespace Symfony\Component\Security\Guard;

abstract class AbstractGuardAuthenticator
{
    public function createAuthenticatedToken(UserInterface $user, $providerKey);
}
```

AbstractGuardAuthenticator


```php
namespace Symfony\Component\Security\Guard;


abstract class AbstractGuardAuthenticator implements GuardAuthenticatorInterface
{
    public function createAuthenticatedToken(UserInterface $user, $providerKey);
}
```

AbstractGuardAuthenticator


```php
namespace Symfony\Component\Security\Guard;

interface GuardAuthenticatorInterface
{
    public function getCredentials(Request $request);

    public function getUser($credentials, UserProviderInterface $userProvider);

    public function checkCredentials($credentials, UserInterface $user);

    public function createAuthenticatedToken(UserInterface $user, $providerKey);

    public function onAuthenticationSuccess(Request $request, TokenInterface $token, $providerKey);

    public function onAuthenticationFailure(Request $request, AuthenticationException $exception);

    public function supportsRememberMe();
}
```

AbstractGuardAuthenticator


```php
namespace Symfony\Component\Security\Guard;

interface GuardAuthenticatorInterface extends AuthenticationEntryPointInterface
{
    public function getCredentials(Request $request);

    public function getUser($credentials, UserProviderInterface $userProvider);

    public function checkCredentials($credentials, UserInterface $user);

    public function createAuthenticatedToken(UserInterface $user, $providerKey);

    public function onAuthenticationSuccess(Request $request, TokenInterface $token, $providerKey);

    public function onAuthenticationFailure(Request $request, AuthenticationException $exception);

    public function supportsRememberMe();
}
```

AbstractGuardAuthenticator

SymfonyLive
PARIS 2017
30-31 MARS

```php
namespace Symfony\Component\Security\Http\EntryPoint;


interface AuthenticationEntryPointInterface
{
    public function start(Request $request, AuthenticationException $authException = null);
}
```

Exemple d'utilisation

AbstractGuardAuthenticator


```yaml
#app/config/security.yml
security:
    encoders:
        Symfony\Component\Security\Core\User\UserInterface: plaintext

    providers:
        in_memory:
            memory:
                users:
                    andre:
                        password: I_<3_Webnet
                        roles:    ROLE_ADMIN
```

AbstractGuardAuthenticator

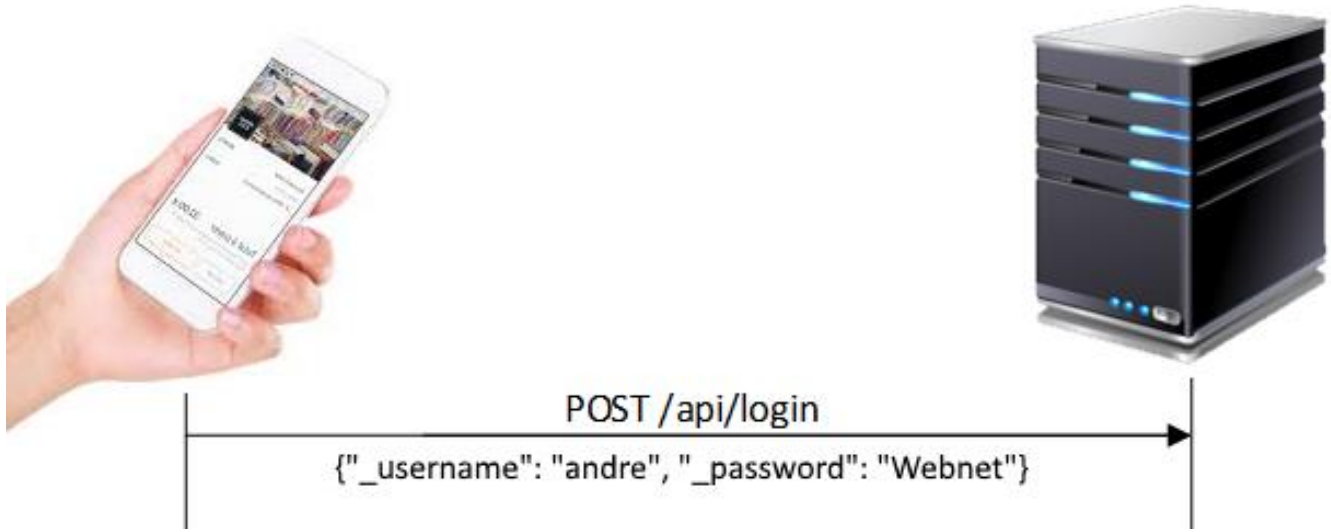

```yaml
#app/config/security.yml
security:
    firewalls:
        login:
            pattern:  ^/api/login
            stateless: true
            anonymous: true
            provider: in_memory
            form_login:
                check_path:      /api/login_check
                success_handler: webnet_authentication.handler.authentication_success
                failure_handler: webnet_authentication.handler.authentication_failure
                require_previous_session: false
                use_referer: true

    access_control:
        - { path: ^/api/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
```

AbstractGuardAuthenticator

SymfonyLive
PARIS 2017
30-31 MARS

```yaml
# app/config/service.yml
services:
    webnet_authentication.handler.authentication_success:
        class: AppBundle\Security\AuthenticationSuccessHandler
        arguments: []

    webnet_authentication.handler.authentication_failure:
        class: AppBundle\Security\AuthenticationFailureHandler
        arguments: []
```

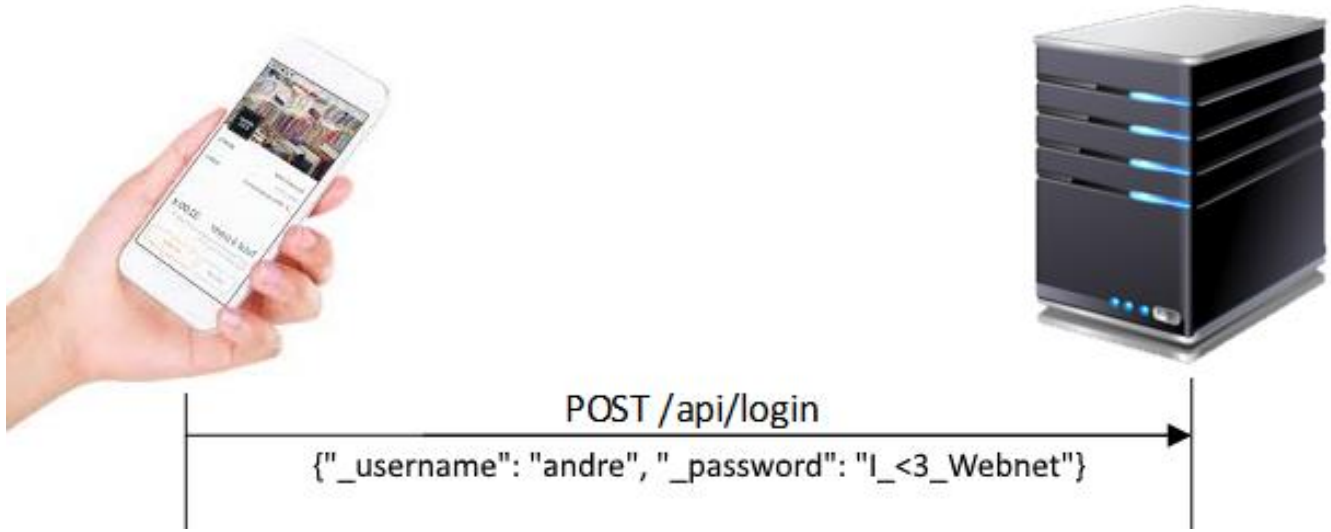
AbstractGuardAuthenticator



AbstractGuardAuthenticator


```php
/**
 * Class AuthenticationFailureHandler
 *
 * @package AppBundle\Security
 */
class AuthenticationFailureHandler implements AuthenticationFailureHandlerInterface
{
    /**
     * {@inheritdoc}
     */
    public function onAuthenticationFailure(Request $request, AuthenticationException $exception)
    {
        $data = array(
            'message' => strtr($exception->getMessageKey(), $exception->getMessageData())
        );

        return new JsonResponse($data, Response::HTTP_FORBIDDEN);
    }
}
```

AbstractGuardAuthenticator

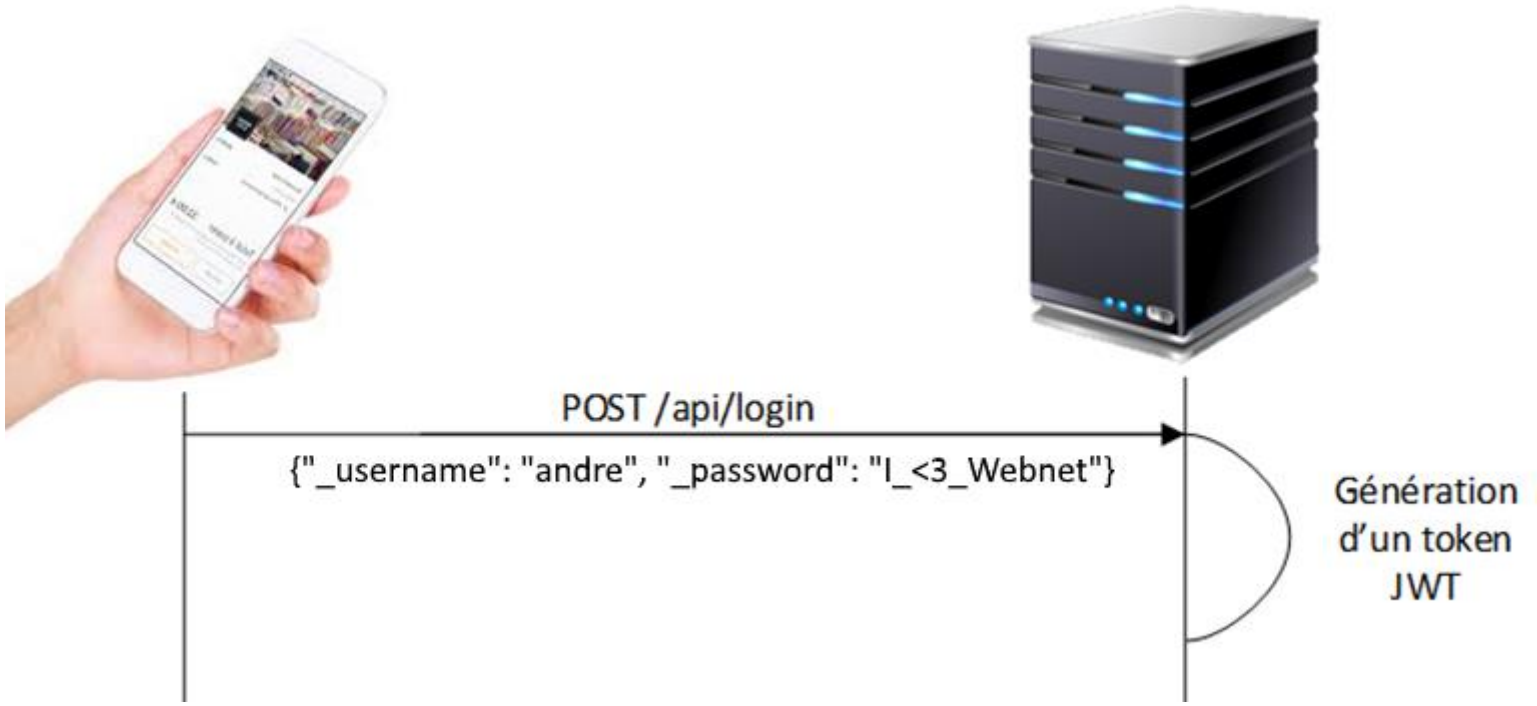


AbstractGuardAuthenticator

SymfonyLive
PARIS 2017
30-31 MARS

```php
/**
 * Class AuthenticationSuccessHandler
 *
 * @package AppBundle\Security
 */
class AuthenticationSuccessHandler implements AuthenticationSuccessHandlerInterface
{
    /**
     * @inheritdoc
     */
    public function onAuthenticationSuccess(Request $request, TokenInterface $token)
    {
        return $this->handleAuthenticationSuccess($token->getUser());
    }
}
```

AbstractGuardAuthenticator



namshi/jose

composer require namshi/jose

JSON Object Signing and Encryption library for PHP.

github.com/namshi/jose
Source
Issues

Installs:	2 084 981
Dependents:	36
Suggesters:	0
Stars:	392
Watchers:	32
Forks:	63
Open Issues:	8

dev-master
7.2.3
7.2.2

7.2.3 — 2016-12-05 07:27 UTC

requires

- ext-date: *
- ext-hash: *
- ext-json: *
- ext-pcre: *
- ext-spl: *
- php: >=5.5
- symfony/polyfill-php56: ^1.0

requires (dev)

- phpunit/phpunit: ^4.5|^5.0
- satooshi/php-coveralls: ^1.0
- phpseclib/phpseclib: ^2.0

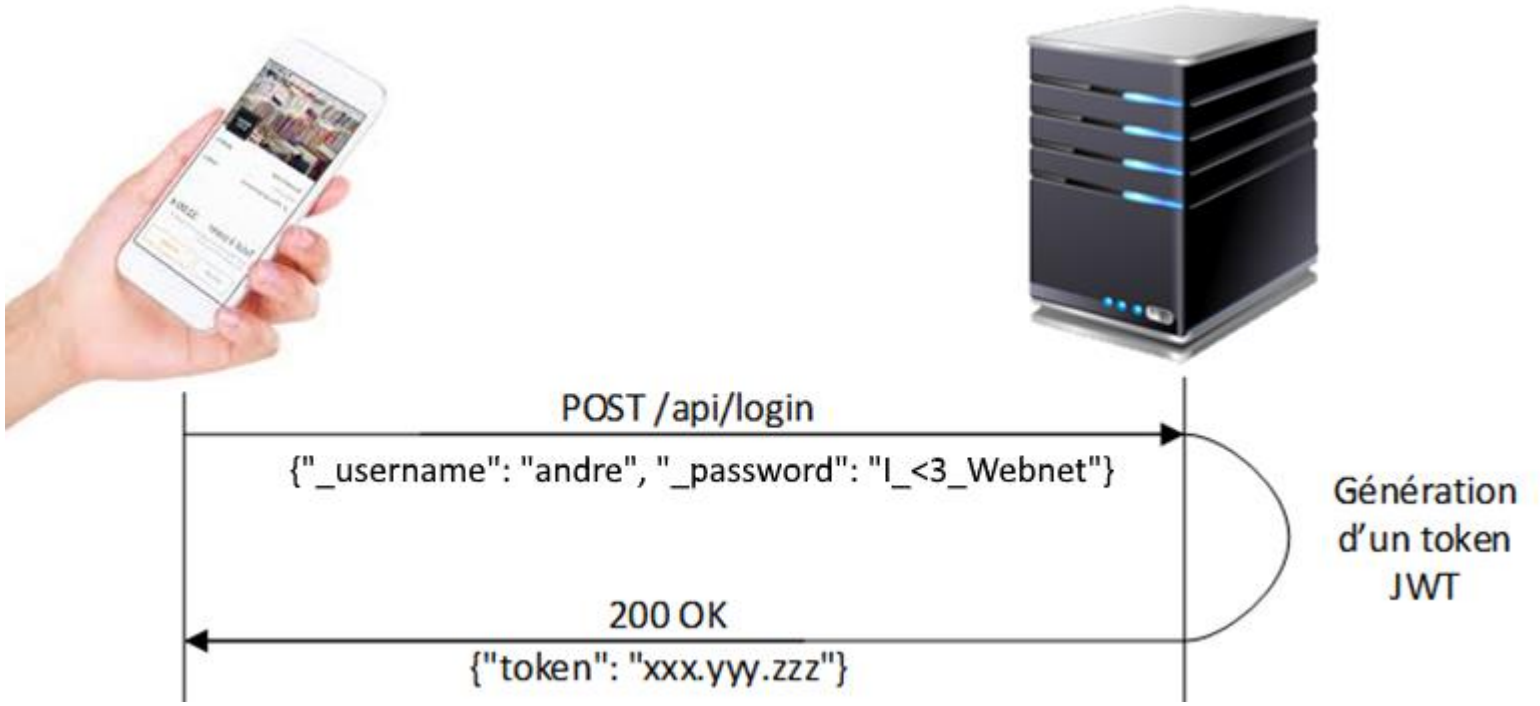
suggests

- ext-openssl: Allows to use OpenSSL as crypto engine.
- phpseclib/phpseclib: Allows to use Phpseclib as crypto engine, use version ^2.0.

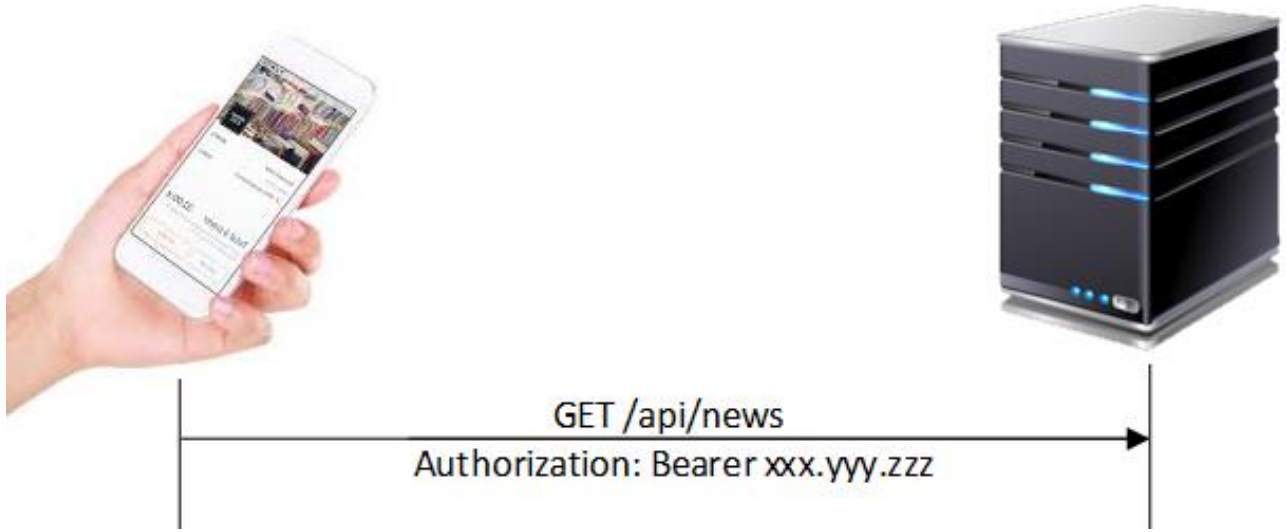
AbstractGuardAuthenticator


```php
/**
 * Class AuthenticationSuccessHandler
 * @package AppBundle\Security
 */
class AuthenticationSuccessHandler implements AuthenticationSuccessHandlerInterface
{
    const SSL_KEY_PASSPHRASE = 'tests';

    public function onAuthenticationSuccess(Request $request, TokenInterface $token)
    {
        return $this->handleAuthenticationSuccess($token->getUser());
    }

    public function handleAuthenticationSuccess(UserInterface $user)
    {
        $jws = new SimpleJWS(array('alg' => 'RS256'));
        $jws->setPayload(array('sub' => $user->getUsername(), 'exp' => time() + 3600));

        $privateKey = openssl_pkey_get_private("file://path_to_private.key", self::SSL_KEY_PASSPHRASE);
        $jws->sign($privateKey);

        return new JsonResponse(array('token' => $jws->getTokenString()));
    }
}
```

AbstractGuardAuthenticator



AbstractGuardAuthenticator



AbstractGuardAuthenticator


```yaml
# app/config/services.yml
services:
    app.token_authenticator:
        class: AppBundle\Security\WebnetTokenAuthenticator
```

AbstractGuardAuthenticator


```php
/**
 * Class WebnetAuthenticator
 *
 * @package AppBundle\Security
 */
class WebnetAuthenticator extends AbstractGuardAuthenticator
{

}
```

AbstractGuardAuthenticator


```php
/**
 * Class WebnetAuthenticator
 *
 * @package AppBundle\Security
 */
class WebnetAuthenticator extends AbstractGuardAuthenticator
{
    /**
     * @inheritdoc
     */
    public function getCredentials(Request $request)
    {
        if (!$tokenValue = $request->headers->get('Authorization')) {
            // no token? Return null and no other methods will be called
            return;
        }

        $token = explode(' ', $tokenValue);

        try {
            return ['token' => SimpleJWS::load($token[1])];
        } catch (\Exception $e) {
            return;
        }
    }
}
```

AbstractGuardAuthenticator


```php
/**
 * Class WebnetAuthenticator
 *
 * @package AppBundle\Security
 */
class WebnetAuthenticator extends AbstractGuardAuthenticator
{
    /**
     * @inheritdoc
     */
    public function start(Request $request, AuthenticationException $authException = null)
    {
        $data = array('message' => 'Authentication Required');

        return new JsonResponse($data, Response::HTTP_UNAUTHORIZED);
    }
}
```

AbstractGuardAuthenticator


```php
/**
 * Class WebnetAuthenticator
 *
 * @package AppBundle\Security
 */
class WebnetAuthenticator extends AbstractGuardAuthenticator
{
    /**
     * @inheritdoc
     */
    public function getCredentials(Request $request)
    {
        if (!$tokenValue = $request->headers->get('Authorization')) {
            // no token? Return null and no other methods will be called
            return;
        }

        $token = explode(' ', $tokenValue);

        try {
            return ['token' => SimpleJWS::load($token[1])];
        } catch (\Exception $e) {
            return;
        }
    }
}
```

AbstractGuardAuthenticator


```php
/**
 * Class WebnetAuthenticator
 *
 * @package AppBundle\Security
 */
class WebnetAuthenticator extends AbstractGuardAuthenticator
{
    /**
     * @inheritdoc
     */
    public function getUser($credentials, UserProviderInterface $userProvider)
    {
        $payload = $credentials['token']->getPayload();

        if (!isset($payload['sub']) || !$payload['sub']) {
            return;
        }

        return $userProvider->loadUserByUsername($payload['sub']);
    }
}
```

AbstractGuardAuthenticator


```php
/**
 * Class WebnetAuthenticator
 *
 * @package AppBundle\Security
 */
class WebnetAuthenticator extends AbstractGuardAuthenticator
{
    /**
     * @inheritdoc
     */
    public function checkCredentials($credentials, UserInterface $user)
    {
        $publicKey = openssl_pkey_get_public("file://path_to_public.key");

        // verify that the token is valid (exp) and had the same values
        return $credentials['token']->isValid($publicKey, 'RS256');
    }
}
```

AbstractGuardAuthenticator


```php
/**
 * Class WebnetAuthenticator
 *
 * @package AppBundle\Security
 */
class WebnetAuthenticator extends AbstractGuardAuthenticator
{
    /**
     * @inheritdoc
     */
    public function onAuthenticationSuccess(Request $request, TokenInterface $token, $providerKey)
    {
        // on success, let the request continue
        return null;
    }

    /**
     * @inheritdoc
     */
    public function onAuthenticationFailure(Request $request, AuthenticationException $exception)
    {
        $data = array(
            'message' => strtr($exception->getMessageKey(), $exception->getMessageData())
        );

        return new JsonResponse($data, Response::HTTP_FORBIDDEN);
    }
}
```

AbstractGuardAuthenticator

SymfonyLive
PARIS 2017
30-31 MARS

```php
/**
 * Class WebnetAuthenticator
 *
 * @package AppBundle\Security
 */
class WebnetAuthenticator extends AbstractGuardAuthenticator
{
    public function supportsRememberMe()
    {
        return false;
    }
}
```

Autre solution ?

« **There's a bundle for that !** »

- lexik/LexikJWTAuthenticationBundle
- gesdinet/JWTRefreshTokenBundle (refresh token)

Conclusion

SymfonyLive
PARIS 2017
30-31 MARS

Merci
pour votre attention,
et bon appétit ☺

RETROUVEZ WEBNET