Securizando APIs con JWT en Symfony

SPONSORS

CodelyTV

SensioLabs

espacio_RES
[universo de resiliencia creativa]

EUSA

INNN.

EMERGYA

¡Hola!

Soy Álex Martín

Hago APIs y otras mandangas en @innn

Puedes encontrarme en @alexdw

1

Alternativas para securizar APIs

El bueno El malo El feo

JWT
El bueno

Cookies
El malo

Oauth
El feo

1

Qué es JWT

JSON Web Token

- Formato de tokens para asegurar la comunicación entre dos partes
- Basado en estándar abierto (RFC-7519)
- Sencillo

El token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJz
dWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiYWRtaW4iOnRydWV9

El token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJz
dWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiYWRtaW4iOnRydWV9

El token

Header

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJz
dWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiYWRtaW4iOnRydWV9

El token

Header

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJz
dWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiYWRtaW4iOnRydWV9

Payload

El token

Header

Payload

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

```
{
    "sub": "1234567890",
    "name": "John Doe",
    "admin": true
}
```

El token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJz
dWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiYWRtaW4iOnRydWV9

El token **no está cifrado**, simplemente está codificado en **base 64 url**

El token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

Signature

El token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

Signature

El token **no está cifrado** pero sí **firmado**, por lo que nadie puede cambiar el contenido sin que nos enteremos

Partes del token

Header

```
{
    "alg": "HS256",  ——→ Algoritmo de hashing ( HMAC SHA256 RSA )
    "typ": "JWT"     ——→ Tipo de token
}
```

Partes del token

payload

```
{
    "sub": "1234567890",
    "name": "John Doe",
    "admin": true
}
```

Partes del token

payload

```
{
    "sub": "1234567890",
    "name": "John Doe",
    "admin": true
}
```

Existen 3 tipos de claims: *privados*, *públicos* , *registrados*

Tipos reservados:

- **jti** : Identificador del token
- **iss** : Emisor del token
- **aud** : Audiencia
- **sub** : Tema
- **iat** : Cuando se creó
- **exp** : Cuando expira
- **nbf** : No antes de

Partes del token

signature

HMACSHA256(
base64UrlEncode(*header*) + "." +
base64UrlEncode(*payload*),
secret)

La firma se obtiene a partir de la cabecera y el payload codificados tras aplicarle el algoritmo especificado en el header.

¡ JWT Exposed !

¡ JWT Exposed ! (en realidad no)

Usar ***RSA256*** o **superior**

y no

HS256 *(vulnerable a ataques por fuerza bruta)*

¡ JWT Exposed ! (en realidad no)

Usar ***RSA256*** o **superior**

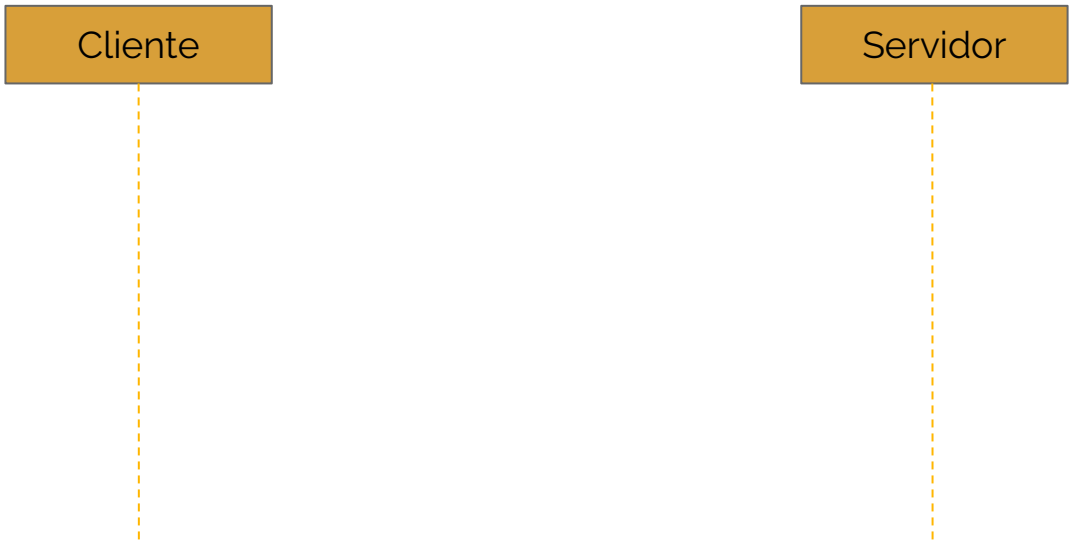
y no

HS256 *(vulnerable a ataques por fuerza bruta)*

Herramienta para un fin de semana divertido:

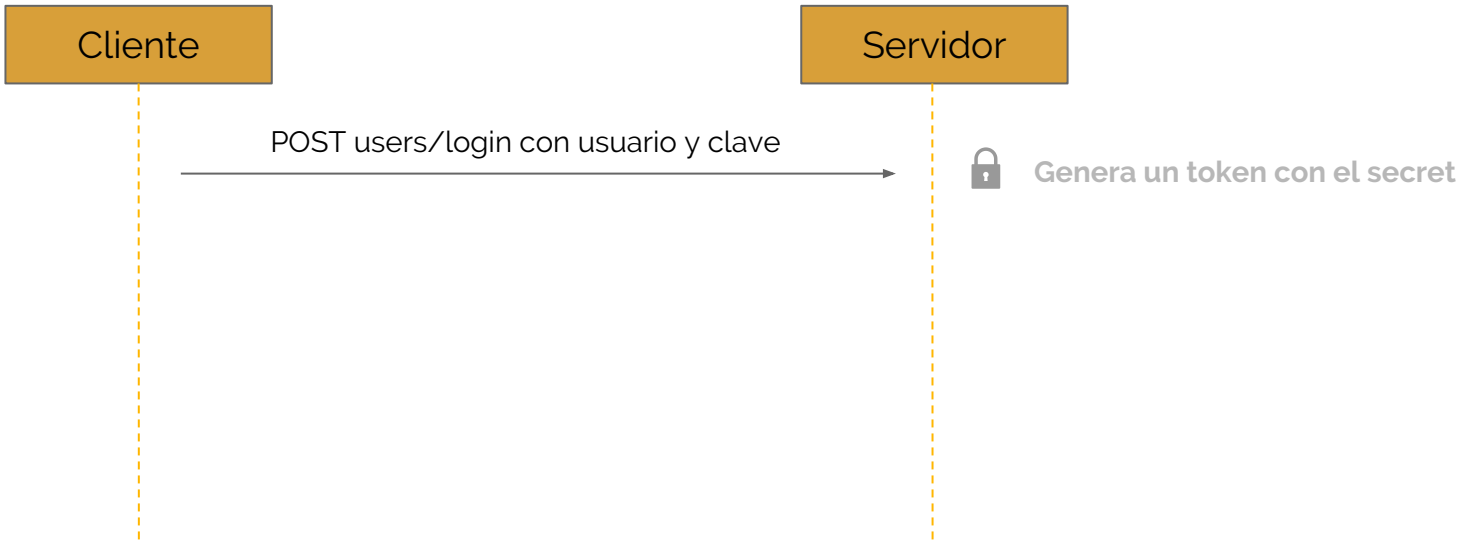
https://www.npmjs.com/package/jwt-cracker

¿Cómo funciona?

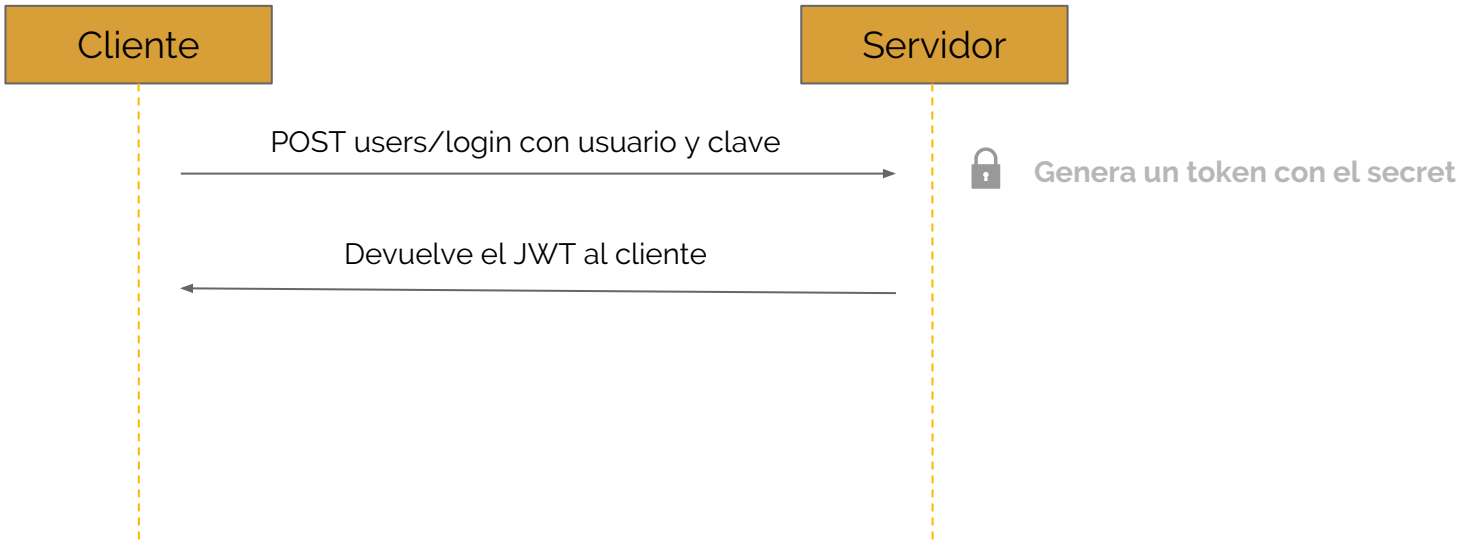


¿Cómo funciona?

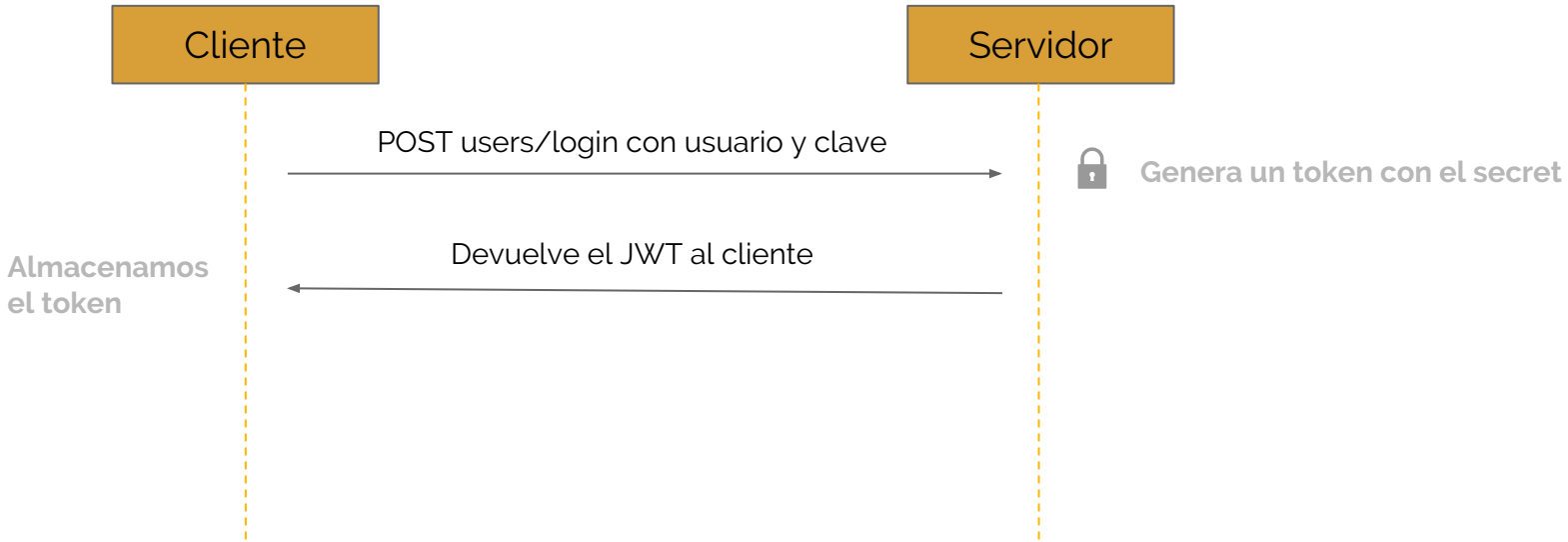
¿Cómo funciona?



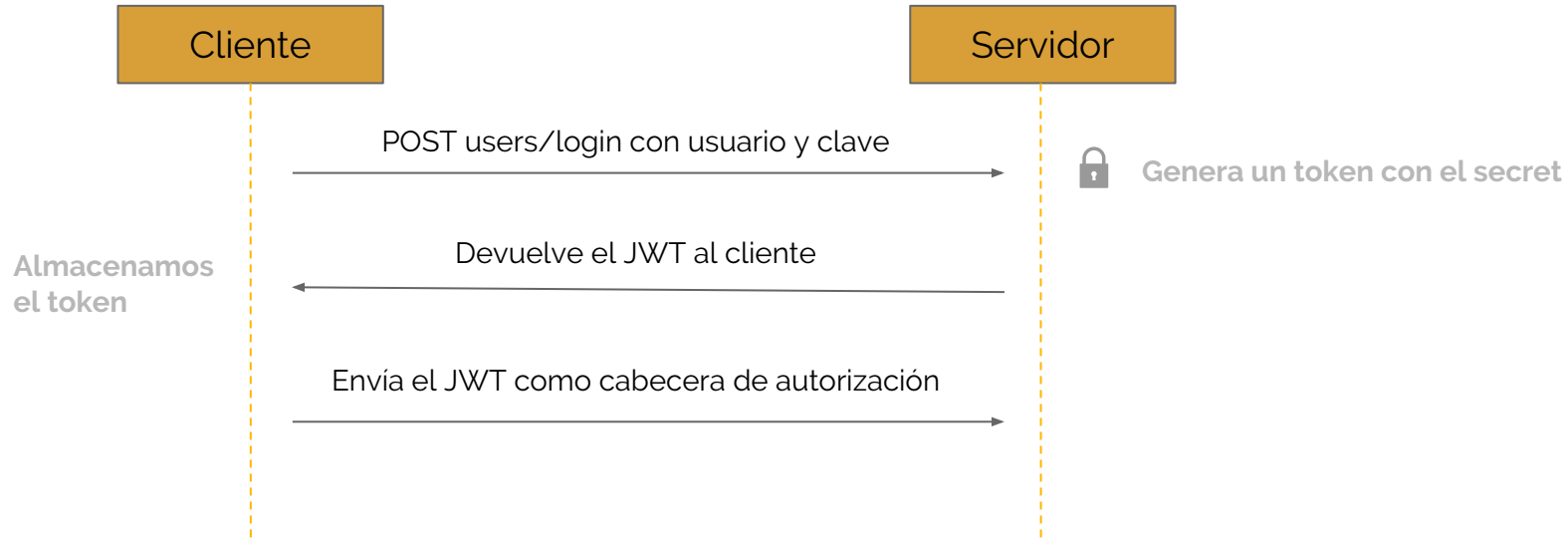
¿Cómo funciona?



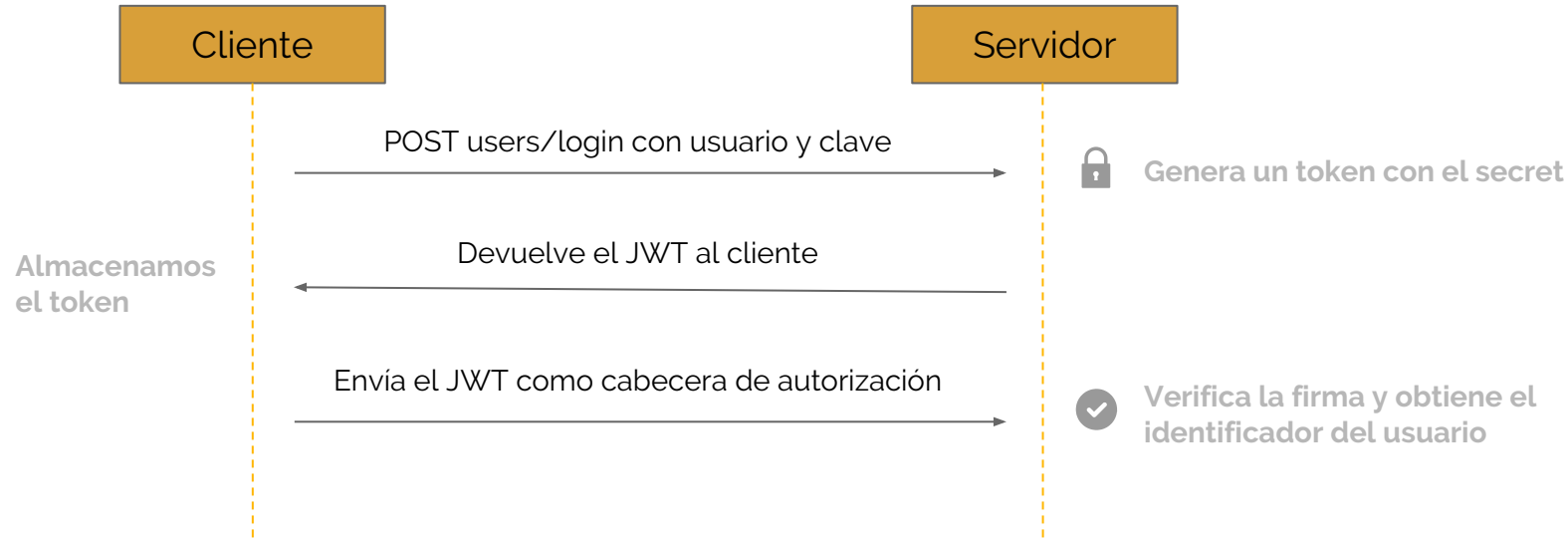
¿Cómo funciona?



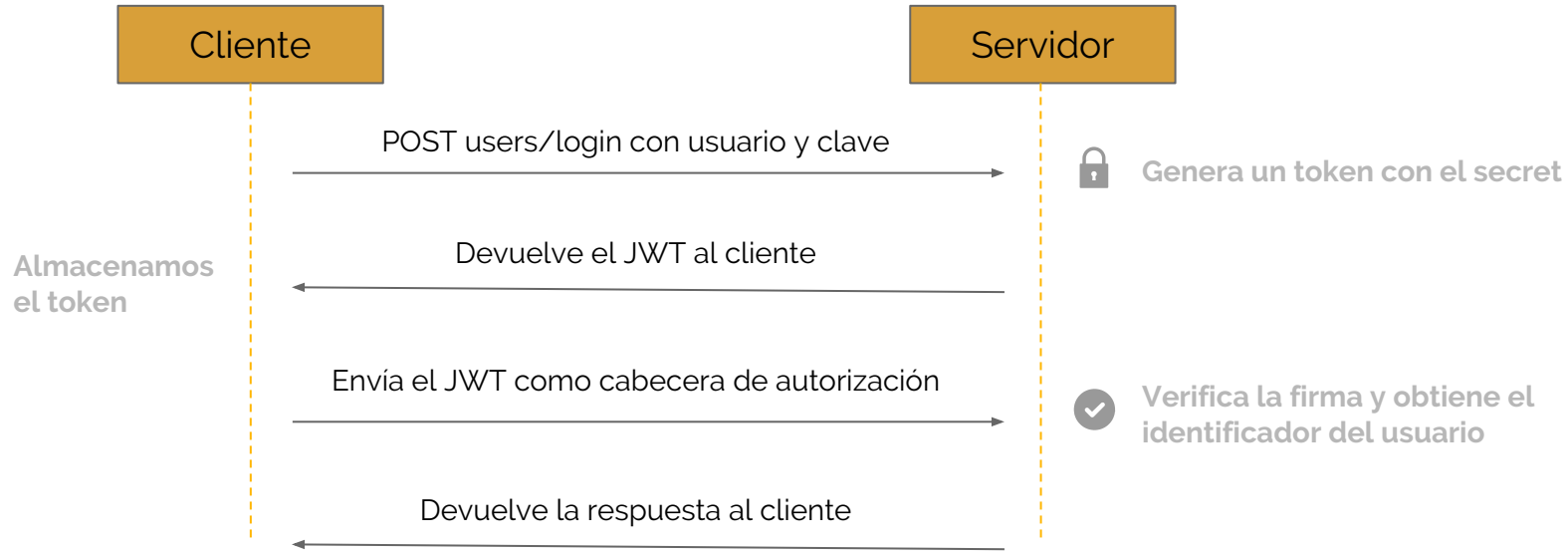
¿Cómo funciona?



¿Cómo funciona?



¿Cómo funciona?



Cuidado al guardar el Token

- Estratégia local Storage:

Cuidado al guardar el Token

- Estratégia local Storage:

  Atención a los ataques XSS

Cuidado al guardar el Token

- Estratégia local Storage:

  Atención a los ataques XSS
- Estratégia cookies:

Cuidado al guardar el Token

- Estratégia local Storage:

  Atención a los ataques XSS
- Estratégia cookies:

  Atención con los ataques CSRF

Cuidado al guardar el Token

- Estratégia local Storage:

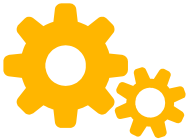
  Atención a los ataques XSS
- Estratégia cookies:

  Atención con los ataques CSRF

MÁS INFO:

https://stormpath.com/blog/where-to-store-your-jwts-cookies-vs-html5-web-storage

https://auth0.com/blog/ten-things-you-should-know-about-tokens-and-cookies/#token-storage



En Javascript...

```
fetch(apiURL,{
     method: “POST”,
     headers: {
          “Accept”: “application/json”,
          “Content-Type”: “application/json”
          “Authorization”: “Bearer ”+ token
     },
     body: JSON.stringify( data )
})
```

2

Lexik
JWTAuthenticationBundle

Instalación con Symfony Flex

Requisitos

OpenSSL

HTTPS recomendado

```
$ composer create-project symfony/skeleton phpsevilla_jwt
$ cd phpsevilla_jwt
$ composer require lexik/jwt-authentication-bundle
```

Instalación con Symfony Flex

Requisitos

OpenSSL

HTTPS recomendado

```
$ composer create-project symfony/skeleton phpsevilla_jwt
$ cd phpsevilla_jwt
$ composer require jwt-auth
```

Configuración

Generamos el par de claves

```
$ mkdir -p config/jwt
$ openssl genrsa -out config/jwt/private.pem -aes256 4096
$ openssl rsa -pubout -in config/jwt/private.pem -out config/jwt/public.pem
```

Configuración

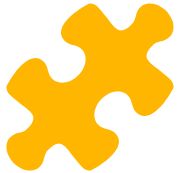
Generamos el par de claves

```
$ mkdir -p config/jwt
$ openssl genrsa -out config/jwt/private.pem -aes256 4096
$ openssl rsa -pubout -in config/jwt/private.pem -out config/jwt/public.pem
```

Definimos la clave en nuestro fichero de .env

```
###> lexik/jwt-authentication-bundle ###
# Key paths should be relative to the project directory
JWT_PRIVATE_KEY_PATH=config/jwt/private.pem
JWT_PUBLIC_KEY_PATH=config/jwt/public.pem
JWT_PASSPHRASE=89b1d0b5dc1168a0eb816feb13e67068
###< lexik/jwt-authentication-bundle ###
```

Configuración



Configuración firewall

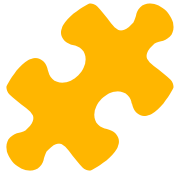
```
security:
  # ...
  firewalls:
    login:
      pattern:  ^/api/login
      stateless: true
      anonymous: true
      form_login:
        check_path: /api/login_check
        success_handler: lexik_jwt_authentication.handler.authentication_success
        failure_handler: lexik_jwt_authentication.handler.authentication_failure
        require_previous_session: false
    api:
     pattern:   ^/api
     stateless: true
     guard:
       authenticators:
         - lexik_jwt_authentication.jwt_token_authenticator
access_control:
- { path: ^/api/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
- { path: ^/api,       roles: IS_AUTHENTICATED_FULLY }
```

Configuración

Configuración rutas

```
// config/routes.yaml

api_login_check:
  path:  /api/login_check
```

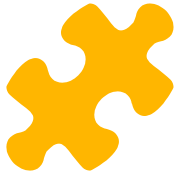
Obtener el token



Configuración proveedor

```yaml
security:
  # ...
  providers:
    in_memory:
      memory:
        users:
          alex:
            password: alex
            roles: 'ROLE_USER'
  encoders:
    Symfony\Component\Security\Core\User\User: plaintext
```

Obtener el token



Configuración proveedor

```
security:
  # ...
  providers:
    in_memory:
      memory:
        users:
          alex:
            password: alex
            roles: 'ROLE_USER'
  encoders:
    Symfony\Component\Security\Core\User\User: plaintext
```

```
$ curl -X POST http://localhost:8000/api/login_check -d _username=alex -d _password=alex
```

Obtener el token

Configuración proveedor

```
security:
  # ...
  providers:
    in_memory:
      memory:
        users:
          alex:
            password: alex
            roles: 'ROLE_USER'
  encoders:
    Symfony\Component\Security\Core\User\User: plaintext
```

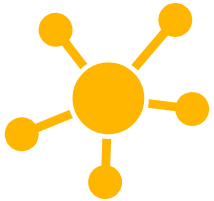
```
$ curl -X POST http://localhost:8000/api/login_check -d _username=alex -d _password=alex
```

```
{
"token":
"eyJhbGciOiJSUzI1NiJ9.eyJyb2xlcyI6WyJST0xFX1VTRVIiXSwidXNlcm5hbWUiOiJhbGV4IiwiaWF0IjoxNTE3MTM
5OTIxLCJleHAiOjE1MTcxNDM1MjF9. ... "
}
```

3

Extendiendo JWTAuthenticationBundle

Extendiendo JWTAuthenticationBundle



Clase Authentication

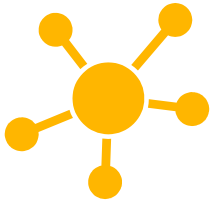
```php
// src/Security/Guard/JWTTokenAuthenticator.php

<?php

namespace AppBundle\Security\Guard;

use Lexik\Bundle\JWTAuthenticationBundle\Security\Guard\JWTTokenAuthenticator
as BaseAuthenticator;

class JWTTokenAuthenticator extends BaseAuthenticator
{
// ...
}
```

Servicio

```yaml
// config/service.yaml

app.jwt_token_authenticator:
    class:  AppBundle\Security\Guard\JWTTokenAuthenticator
    Parent: lexik_jwt_authentication.security.guard.jwt_token_authenticator
```

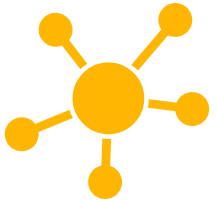
Extendiendo JWTAuthenticationBundle

Extendiendo JWTAuthenticationBundle



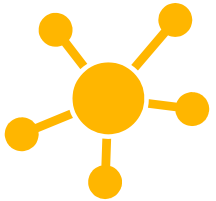
GUARD		JWTAuthenticator
getCredentials()	→	Validamos el token
getUser()	→	Obtenemos el usuario

Extendiendo JWTAuthenticationBundle



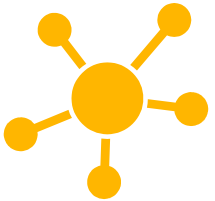
GUARD		JWTAuthenticator
getCredentials()	→	Validamos el token
getUser()	→	Obtenemos el usuario
getCredentials()	→	true

Extendiendo JWTAuthenticationBundle



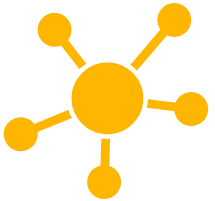
GUARD		JWTAuthenticator
getCredentials()	→	Validamos el token
getUser()	→	Obtenemos el usuario
getCredentials()	→	true
createAuthenticatedToken()	→	Creamos el token de Symfony

Extendiendo JWTAuthenticationBundle



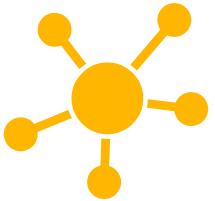
GUARD		JWTAuthenticator
getCredentials()	→	Validamos el token
getUser()	→	Obtenemos el usuario
getCredentials()	→	true
createAuthenticatedToken()	→	Creamos el token de Symfony
onAuthenticationFailure()	→	Devolvemos el código de respuesta correcto (401)

Extendiendo JWTAuthenticationBundle



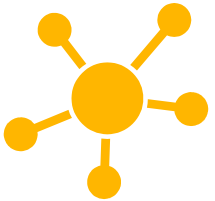
GUARD		JWTAuthenticator
getCredentials()	→	Validamos el token
getUser()	→	Obtenemos el usuario
getCredentials()	→	true
createAuthenticatedToken()	→	Creamos el token de Symfony
onAuthenticationFailure()	→	Devolvemos el código de respuesta correcto (401)
onAuthenticationSuccess()	→	No hacemos nada

Extendiendo JWTAuthenticationBundle



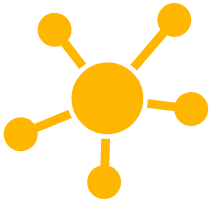
GUARD		JWTAuthenticator
getCredentials()	→	Validamos el token
getUser()	→	Obtenemos el usuario
getCredentials()	→	true
createAuthenticatedToken()	→	Creamos el token de Symfony
onAuthenticationFailure()	→	Devolvemos el código de respuesta correcto (401)
onAuthenticationSuccess()	→	No hacemos nada
supportsRememberMe()	→	false

Extendiendo JWTAuthenticationBundle



GUARD		JWTAuthenticator
getCredentials()	→	Validamos el token
getUser()	→	Obtenemos el usuario
getCredentials()	→	true
createAuthenticatedToken()	→	Creamos el token de Symfony
onAuthenticationFailure()	→	Devolvemos el código de respuesta correcto (401)
onAuthenticationSuccess()	→	No hacemos nada
supportsRememberMe()	→	false

Extendiendo JWTAuthenticationBundle

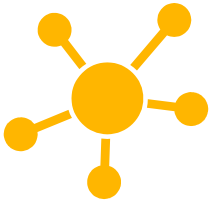


TIP

Crear un comando para generar tokens

```
//src/Command/GenerateTokenCommand.php

...
public function __construct(JWTTokenManagerInterface $jwtManager){
   $this->jwtManager = $jwtManager;
   parent::__construct();
}

protected function execute(InputInterface $input, OutputInterface $output){
  ...
  $encoder = $this->getContainer()->get('security.password_encoder');
  $user = $em->getRepository("AppBundle:User")->findOneByUsername($username)
  $isValid =  $encoder->isPasswordValid($user,$password);
    if ($isValid){
      $token = $this->jwtManager>create($user);
      $output->writeln('<success>Usuario correcto</success>');
      $output->writeln("Token: {$token}");
    }else{
      $output->writeln('<error>Usuario incorrecto</error>');
    }
}
```

Extendiendo JWTAuthenticationBundle

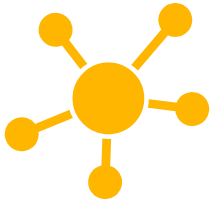


TIP

Crear un comando para generar tokens

```
//src/Command/GenerateTokenCommand.php

...
public function __construct(JWTTokenManagerInterface $jwtManager){
   $this->jwtManager = $jwtManager;
   parent::__construct();
}

protected function execute(InputInterface $input, OutputInterface $output){
  ...
  $encoder = $this->getContainer()->get('security.password_encoder');
  $user = $em->getRepository("AppBundle:User")->findOneByUsername($username)
  $isValid =  $encoder->isPasswordValid($user,$password);
    if ($isValid){
      $token = $this->jwtManager>create($user);
      $output->writeln('<success>Usuario correcto</success>');
      $output->writeln("Token: {$token}");
    }else{
      $output->writeln('<error>Usuario incorrecto</error>');
    }
}
```

Extendiendo JWTAuthenticationBundle



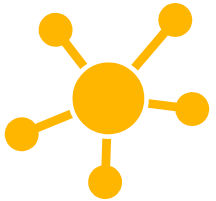
EVENTOS

Events::JWT_CREATED() → Cuando se crea un token

Ejemplo: Personalizar claims

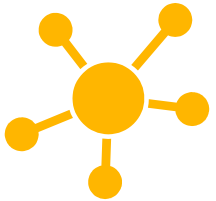
```
//src/EventListener/JWCCreatedListener.php

...
public function onJWTCreated(JWTCreatedEvent $event){
   $payload = $event->getData();
   $payload['custom'] = 'My custom claim';
   $event->setData($payload)
}
```

Extendiendo JWTAuthenticationBundle



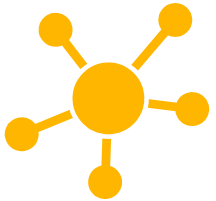
Events::JWT_DECODED() → Al decodificar el token (Ej. Añadir una validación extra)

Extendiendo JWTAuthenticationBundle



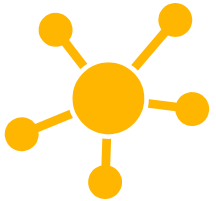
Events::JWT_DECODED()	→	Al decodificar el token (Ej. Añadir una validación extra)
Events::JWT_AUTHENTICATED()	→	Al autenticar (Ej. Añadir información extra al token de symfony)

Extendiendo JWTAuthenticationBundle



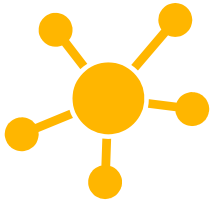
Events::JWT_DECODED()	→	Al decodificar el token (Ej. Añadir una validación extra)
Events::JWT_AUTHENTICATED()	→	Al autenticar (Ej. Añadir información extra al token de symfony)
Events::JWT_AUTHENTICATION_SUCCESS()	→	Al autenticar correctamente (Ej. Añadir datos al JWT Token)

Extendiendo JWTAuthenticationBundle



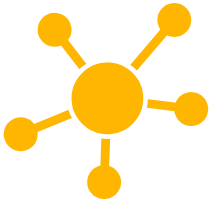
Events::JWT_DECODED()	→	Al decodificar el token (Ej. Añadir una validación extra)
Events::JWT_AUTHENTICATED()	→	Al autenticar (Ej. Añadir información extra al token de symfony)
Events::JWT_AUTHENTICATION_SUCCESS()	→	Al autenticar correctamente (Ej. Añadir datos al JWT Token)
Events::JWT_ENCODED()	→	Al crear el token (Ej. Crear un log con los tokens creados)

Extendiendo JWTAuthenticationBundle



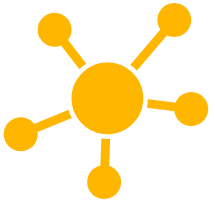
Events::JWT_DECODED()	→	Al decodificar el token (Ej. Añadir una validación extra)
Events::JWT_AUTHENTICATED()	→	Al autenticar (Ej. Añadir información extra al token de symfony)
Events::JWT_AUTHENTICATION_SUCCESS()	→	Al autenticar correctamente (Ej. Añadir datos al JWT Token)
Events::JWT_ENCODED()	→	Al crear el token (Ej. Crear un log con los tokens creados)
Events::JWT_AUTHENTICATION_FAILURE()	→	Error al autenticar (Ej. Crear un log con accesos fallidos)

Extendiendo JWTAuthenticationBundle



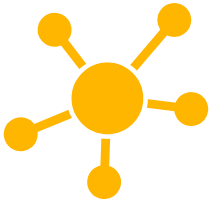
Events::JWT_DECODED()	→	Al decodificar el token (Ej. Añadir una validación extra)
Events::JWT_AUTHENTICATED()	→	Al autenticar (Ej. Añadir información extra al token de symfony)
Events::JWT_AUTHENTICATION_SUCCESS()	→	Al autenticar correctamente (Ej. Añadir datos al JWT Token)
Events::JWT_ENCODED()	→	Al crear el token (Ej. Crear un log con los tokens creados)
Events::JWT_AUTHENTICATION_FAILURE()	→	Error al autenticar (Ej. Crear un log con accesos fallidos)
Events::JWT_INVALID()	→	Token invalido (Ej. Crear un log con accesos inválidos)

Extendiendo JWTAuthenticationBundle



Events::JWT_DECODED()	→	Al decodificar el token (Ej. Añadir una validación extra)
Events::JWT_AUTHENTICATED()	→	Al autenticar (Ej. Añadir información extra al token de symfony)
Events::JWT_AUTHENTICATION_SUCCESS()	→	Al autenticar correctamente (Ej. Añadir datos al JWT Token)
Events::JWT_ENCODED()	→	Al crear el token (Ej. Crear un log con los tokens creados)
Events::JWT_AUTHENTICATION_FAILURE()	→	Error al autenticar (Ej. Crear un log con accesos fallidos)
Events::JWT_INVALID()	→	Token invalido (Ej. Crear un log con accesos inválidos)
Events::JWT_NOT_FOUND()	→	Token no encontrado (Ej. Personalizar respuesta)

Extendiendo JWTAuthenticationBundle



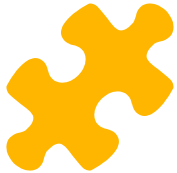
Events::JWT_DECODED()	→	Al decodificar el token (Ej. Añadir una validación extra)
Events::JWT_AUTHENTICATED()	→	Al autenticar (Ej. Añadir información extra al token de symfony)
Events::JWT_AUTHENTICATION_SUCCESS()	→	Al autenticar correctamente (Ej. Añadir datos al JWT Token)
Events::JWT_ENCODED()	→	Al crear el token (Ej. Crear un log con los tokens creados)
Events::JWT_AUTHENTICATION_FAILURE()	→	Error al autenticar (Ej. Crear un log con accesos fallidos)
Events::JWT_INVALID()	→	Token invalido (Ej. Crear un log con accesos inválidos)
Events::JWT_NOT_FOUND()	→	Token no encontrado (Ej. Personalizar respuesta)
Events::JWT_EXPIRED()	→	Token caducado

4

gesdinet

JWTRefreshTokenBundle

Instalación



Requisitos

Symfony 3.3 o Symfony 4.0

HTTPS recomendado

```
$ composer require gesdinet/jwt-refresh-token-bundle
```

Configuración

Configuración rutas

```
// config/routes.yaml

gesdinet_jwt_refresh_token:
    path:     /api/token/refresh
    defaults: { _controller: gesdinet.jwtrefreshtoken:refresh }
```

Configuración

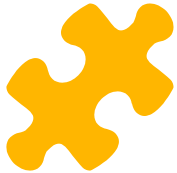
Configuración rutas

```
// config/routes.yaml

gesdinet_jwt_refresh_token:
    path:     /api/token/refresh
    defaults: { _controller: gesdinet.jwtrefreshtoken:refresh }
```

Configuración firewall

```
# app/config/security.yml

    firewalls:
        refresh:
            pattern:  ^/api/token/refresh
            stateless: true
            anonymous: true
    access_control:
        - { path: ^/api/token/refresh, roles: IS_AUTHENTICATED_ANONYMOUSLY }
```

Configuración



Configuración rutas

```
// config/routes.yaml

gesdinet_jwt_refresh_token:
    path:     /api/token/refresh
    defaults: { _controller: gesdinet.jwtrefreshtoken:refresh }
```

Configuración firewall

```
# app/config/security.yml

    firewalls:
        refresh:
            pattern:  ^/api/token/refresh
            stateless: true
            anonymous: true
    access_control:
        - { path: ^/api/token/refresh, roles: IS_AUTHENTICATED_ANONYMOUSLY }
```

Actualizar esquema

```
php bin/console doctrine:schema:update --force
```

Configuración

TTL refresh

```
// config/packages/jwt_refresh.yaml

gesdinet_jwt_refresh_token:
    ttl: 2592000
```

Configuración

TTL refresh

```
// config/packages/jwt_refresh.yaml

gesdinet_jwt_refresh_token:
    ttl: 2592000
```

TTL Update

```
# app/packages/jwt_refresh.yaml

gesdinet_jwt_refresh_token:
    ttl_update: true
```

Uso

Generar token

```
$ curl -X POST http://localhost:8000/api/login_check -d _username=alex -d _password=alex
```


```
{
"token":
"eyJhbGciOiJSUzI1NiJ9.eyJyb2xlcyI6WyJST0xFX1VTRVIiXSwidXNlcm5hbWUiOiJhbGV4IiwiaWF0IjoxNTE3MTM
5OTIxLCJleHAiOjE1MTcxNDM1MjF9. ... "
"refresh_token":"697567984c4a46443f1fa6ba6e0b6cfe80fdbb50b83dae821089630262cde467526d18121faf
0ae5b8dccc81ad30b0106643c6e22bcbdfe4d7a2926e7960bef"
}
```

Uso

Generar token

```
$ curl -X POST http://localhost:8000/api/login_check -d _username=alex -d _password=alex
```

```
{
"token":
"eyJhbGciOiJSUzI1NiJ9.eyJyb2xlcyI6WyJST0xFX1VTRVIiXSwidXNlcm5hbWUiOiJhbGV4IiwiaWF0IjoxNTE3MTM
5OTIxLCJleHAiOjE1MTcxNDM1MjF9. ... "
"refresh_token":"697567984c4a46443f1fa6ba6e0b6cfe80fdbb50b83dae821089630262cde467526d18121faf
0ae5b8dccc81ad30b0106643c6e22bcbdfe4d7a2926e7960bef"
}
```

Revocar tokens expirados

```
$ php bin/console gesdinet:jwt:clear
```

Uso

Generar token

```
$ curl -X POST http://localhost:8000/api/login_check -d _username=alex -d _password=alex
```

```
{
"token":
"eyJhbGciOiJSUzI1NiJ9.eyJyb2xlcyI6WyJST0xFX1VTRVIiXSwidXNlcm5hbWUiOiJhbGV4IiwiaWF0IjoxNTE3MTM
5OTIxLCJleHAiOjE1MTcxNDM1MjF9. ... "
"refresh_token":"697567984c4a46443f1fa6ba6e0b6cfe80fdbb50b83dae821089630262cde467526d18121faf
0ae5b8dccc81ad30b0106643c6e22bcbdfe4d7a2926e7960bef"
}
```

Revocar tokens expirados

```
$ php bin/console gesdinet:jwt:clear
```

Revocar un token

```
$ php bin/console gesdinet:jwt:revoke TOKEN
```

¡Gracias!

¿Preguntas?

Puedes encontrame en @alexdw