AUTHENTIFICATION API REST (TOKEN + SSL)

Table des matières

ÉTAPE 1, BUT & PRÉREQUIS

BUT

Implémenter une connexion REST lors de l'accès à l'API afin de restreindre son accès qu'à certains rôles existants à l'aide du bundle LexikJWTAuthentification.

PRÉREQUIS

- Symfony 2.8 ou plus
- Librairie OpenSSL dans le cas où on garde l'encodeur par défaut

ÉTAPE 2, PROCÉDURE

Installer le bundle

Première étape pour installer le bundle nécessaire, rendez-vous au dossier racine de votre site en ligne de commande puis ajoutez le bundle au composer à l'aide de la commande :

```
composer require "lexik/jwt-authentication-bundle"
```

Une fois cela fait, il faut déclarer ce nouveau bundle dans le fichier "*app/AppKernel.php*" comme suit :

```
public function registerBundles()
{
    return array(
        // autres bundles…
        new Lexik\Bundle\JWTAuthenticationBundle\LexikJWTAuthenticationBundle(),
    );
}
```

Générer les clés SSH

L'étape suivante consiste à créer les clés SSH et les passphrases associées. Utilisez les trois commandes suivantes :

```
# Créer le dossier contenant les clés
# Pour les Symfony < v3.0, ajouter le paramètre -p
$ mkdir var/jwt

# Créer la première clé
$ openssl genrsa -out var/jwt/private.pem -aes256 4096

# Créer la deuxième clé
$ openssl rsa -pubout -in var/jwt/private.pem -out var/jwt/public.pem
```

Configurer le bundle

Commencez par aller dans le fichier « *app/config/config.yml* » et ajoutez-y les lignes suivantes :

```
# Créer le dossier contenant les clés
lexik_jwt_authentication:
    private_key_path: '%jwt_private_key_path%'
    public_key_path:  '%jwt_public_key_path%'
    pass_phrase:      '%jwt_key_pass_phrase%'
    token_ttl:        '%jwt_token_ttl%'

    token_extractors:
        authorization_header:
            enabled: true    # activer la vérification par header
            prefix:  token     # préfix du token
            name:    Token     # nom du header à envoyer
```

Rendez-vous ensuite dans le fichier « *app/config/parameters.yml.dist* » et ajoutez-y les lignes suivantes :

```
jwt_private_key_path: '%kernel.root_dir%/../var/jwt/private.pem' # Clé SSH privée
jwt_public_key_path: '%kernel.root_dir%/../var/jwt/public.pem'   # Clé SSH publique
jwt_key_pass_phrase: ''                                          # Passphrase
jwt_token_ttl: 3600                                              # Durée de vie
```

Il faut copier la même chose dans le fichier « app/config/parameters.yml », ce fichier est généré par le précèdent c'est pourquoi nous allons ici directement copier la même chose dedans pour nos tests.

Ensuite, rendez-vous dans le fichier « *app/config/security.yml* » et ajoutez-y :

```
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 } # À changer au besoin
```

Enfin, il ne reste plus qu'à ajouter une route dans le fichier « *app/config/routing.yml* » :

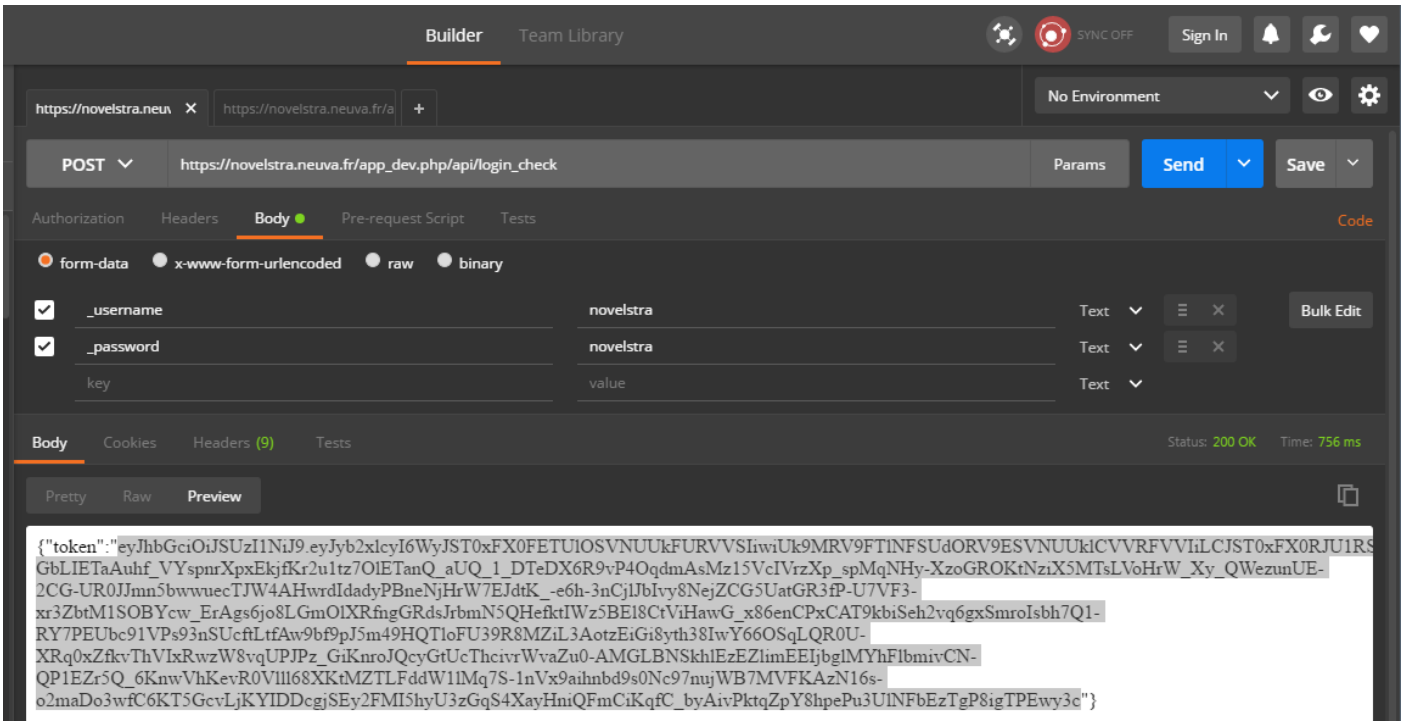
```
    api_login_check:
            path: /api/login_check
```

ÉTAPE 3, UTILISATION

Valider l'installation

Lorsque toutes les étapes précédentes ont été effectuées, vous pouvez tester la génération d'un token.

Pour cela, utilisez un logiciel tel que « *Postman* » permettent d'envoyer des requêtes de plusieurs types et accédez à l'adresse « */api/login_check* » (en POST) tout en renseignant dans le body votre identifiant et votre mot de passe comme ci-dessous :



Vous obtiendrez alors votre token. Il est assez long mais cela lui permet d'être sécurisé grâce à l'algorithme AES-256. Si vous souhaitez changer l'algorithme, une partie y est dédiée dans les étapes optionnelles de cette procédure.

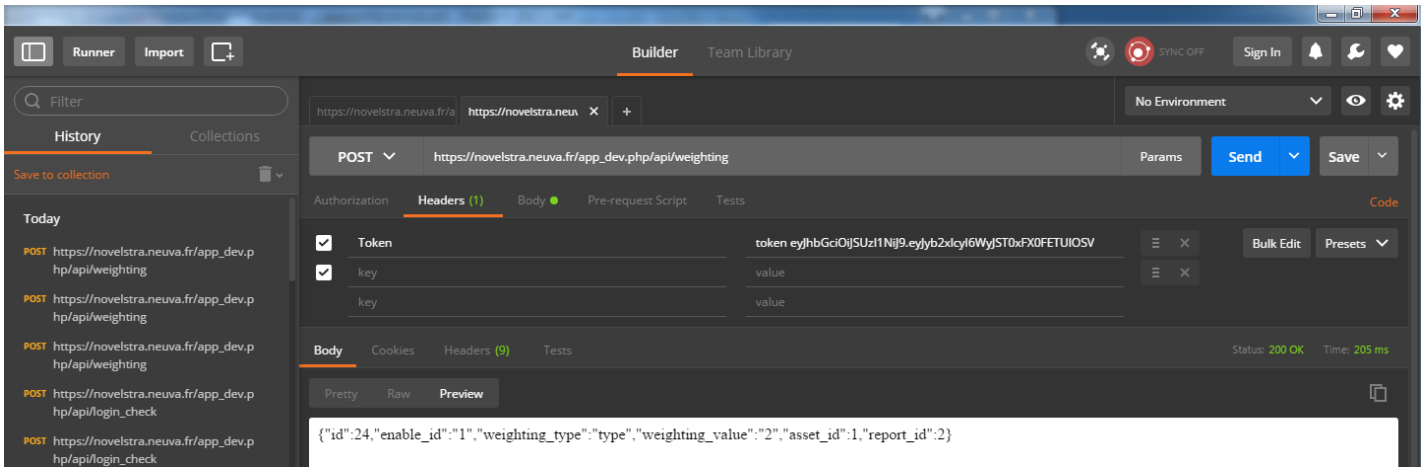
Vous pouvez aussi faire le même test à l'aide d'un terminal et de la commande suivante :

```
curl -X POST https://monsite.fr/app_dev/api/login_check -d _username=nomUtilisateur -d
_password=motDePasse
```

Une fois le token copié, ouvrez dans un nouvel onglet une adresse permettant l'utilisation de votre API, nous allons par exemple ici tenter d'ajouter un objet de type « *Weighting* » en *POST*.

Dans un premier temps, remplissez donc les champ requis à l'ajout (dans le cas où vous faites un ajout) dans le body puis, rendez-vous dans les headers et ajoutez-y la clé « Token » comme déclaré au début de cette procédure puis dans sa valeur écrivez en premier son préfixe, ici « token » suivi de la clé précédemment copiée.

Si tout s'est bien passé vous obtiendrez alors le message confirmant l'ajout :



Il faut savoir que si vous avez changé les rôles autorisés dans votre configuration, l'utilisateur pourra toujours se connecter à l'API mais il ne pourra pas ajouter de données, il recevra alors une erreur 403 « ACCESS DENIED ».

ÉTAPE 4, ÉTAPES OPTIONELLES

Changer l'algorithme de génération

Si vous souhaitez utiliser autre chose que l'algorithme par défaut qui est le AES-256, voici une liste de tous les algorithmes que peut utiliser le bundle :

Compatibles avec OpenSSL :

- RS256, RS384, RS512 (**RSA**)
- ES256, ES384, ES512 (**ECDSA**)
- HS256, HS384, HS512 (**HMAC**)

Compatibles avec phpseclib :

- RS256, RS384, RS512 (**RSA**)

Il faut alors changer la commande générant les clés SSH utilisé au début de cette procédure et ajouter les lignes suivantes dans le fichier « *app/config/config.yml* » :

```
lexik_jwt_authentication:
# ...
    encoder:
        service: lexik_jwt_authentication.encoder.default
        crypto_engine: openssl # Moteur de cryptage utilisé
        signature_algorithm: RS256 # Algorithme de cryptage utilisé
```

Vérifier la présence du token dans les cookies / query

Il faut pour cela ajouter le code suivant au fichier « *app/config/config.yml* » :

```
lexik_jwt_authentication:
# ...
    token_extractors:
        authorization_header:
            enabled: true
            prefix: token
            name: Token
        cookie:
            enabled: false # Mettre à true pour activer
            name: TokenCookie
        query
            enabled: false # Mettre à true pour activer
            name: TokenQuery
```