# RENDU – Sécurisation de l’application « Bibliothèque » (Symfony)

**Étudiant :** *[ton nom]*  
**Projet GitHub :** El-Profesor/EFREI-Municipal-Library  
**Date de rendu :** Mercredi 15 octobre 2025, 23:59

## ✅ Checklist de conformité (OWASP Top 10:2021)

* Validation des entrées (entités + contrôleurs + affichage erreurs Twig)
* Contrôle d’accès ROLE\_USER / ROLE\_ADMIN (redirection /login)
* Session « remember me » au plus 48h (172800 s)
* CSP appliquée globalement (default-src/script-src)
* Pages d’erreur personnalisées (401/403/404/405)
* Cookie dark/light mode (preuve via Storage)
* CSRF : token visible dans le DOM
* composer audit documenté (copie de la sortie)

**Note** : Les blocs ci‑dessous fournissent directement le code prêt à copier/coller.

## 1) Validation des entrées

### 1.1 Entité User (exemple d’annotations de validation)

// src/Entity/User.php  
use Symfony\Component\Validator\Constraints as Assert;  
  
class User  
{  
 /\*\*  
 \* @Assert\NotBlank  
 \* @Assert\Email(mode="html5")  
 \* @Assert\Length(max=180)  
 \*/  
 private ?string $email = null;  
  
 /\*\*  
 \* @Assert\NotBlank  
 \* @Assert\Length(min=8, max=255)  
 \*/  
 private ?string $password = null;  
  
 /\*\*  
 \* @Assert\NotBlank  
 \* @Assert\Length(min=2, max=100)  
 \*/  
 private ?string $username = null;  
}

### 1.2 Entité Book

// src/Entity/Book.php  
use Symfony\Component\Validator\Constraints as Assert;  
  
class Book  
{  
 /\*\* @Assert\NotBlank \*/  
 /\*\* @Assert\Length(min=2, max=255) \*/  
 private ?string $title = null;  
  
 /\*\* @Assert\NotBlank \*/  
 /\*\* @Assert\Length(min=2, max=255) \*/  
 private ?string $author = null;  
  
 /\*\* @Assert\Positive \*/  
 private ?int $year = null;  
}

### 1.3 Validation manuelle côté contrôleur

// src/Controller/BookController.php  
use Symfony\Component\Validator\Validator\ValidatorInterface;  
use Symfony\Component\HttpFoundation\JsonResponse;  
  
public function create(Request $request, ValidatorInterface $validator): Response  
{  
 $book = new Book();  
 // ... hydratation depuis le formulaire ...  
  
 $errors = $validator->validate($book);  
 if (count($errors) > 0) {  
 // Exemple d’API JSON ; pour un rendu Twig, passer $errors au template  
 return new JsonResponse(["errors" => (string) $errors], 400);  
 }  
  
 // persister puis redirect  
}

### 1.4 Affichage des erreurs Twig

{# templates/book/\_form.html.twig #}  
{{ form\_start(form) }}  
 {{ form\_errors(form) }}  
 {{ form\_row(form.title) }}  
 {{ form\_row(form.author) }}  
 {{ form\_row(form.year) }}  
 <button class="btn btn-primary">Enregistrer</button>  
{{ form\_end(form) }}

## 2) Contrôle d’accès

### 2.1 security.yaml

# config/packages/security.yaml  
security:  
 password\_hashers:  
 Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'  
  
 providers:  
 app\_user\_provider:  
 entity:  
 class: App\Entity\User  
 property: email  
  
 firewalls:  
 dev:  
 pattern: ^/(\_(profiler|wdt)|css|images|js)/  
 security: false  
 main:  
 lazy: true  
 provider: app\_user\_provider  
 form\_login:  
 login\_path: login  
 check\_path: login  
 logout:  
 path: app\_logout  
 remember\_me:  
 secret: '%kernel.secret%'  
 lifetime: 172800 # 48h  
 path: /  
  
 access\_control:  
 - { path: ^/admin, roles: ROLE\_ADMIN }  
 - { path: ^/book, roles: ROLE\_USER }  
 - { path: ^/login, roles: PUBLIC\_ACCESS }

**Vérification** : un utilisateur non connecté qui visite /book est redirigé vers /login.

## 3) Session limitée à 48h (Remember Me)

* **Déjà inclus** dans le bloc security.yaml ci‑dessus (lifetime: 172800).
* **Capture d’écran attendue** : cookies montrant REMEMBERME (ou cookie équivalent) avec expiration < **48h**.

## 4) Politique de Contenu (CSP)

### 4.1 Abord simple via un Event Subscriber global

// src/EventSubscriber/CspSubscriber.php  
namespace App\EventSubscriber;  
  
use Symfony\Component\EventDispatcher\EventSubscriberInterface;  
use Symfony\Component\HttpKernel\Event\ResponseEvent;  
use Symfony\Component\HttpKernel\KernelEvents;  
  
class CspSubscriber implements EventSubscriberInterface  
{  
 public static function getSubscribedEvents(): array  
 {  
 return [KernelEvents::RESPONSE => 'onResponse'];  
 }  
  
 public function onResponse(ResponseEvent $event): void  
 {  
 $response = $event->getResponse();  
 $policy = "default-src 'self'; script-src 'self' https://trusted.cdn.com";  
 $response->headers->set('Content-Security-Policy', $policy);  
 // Optionnel: rapport en mode Report-Only au début  
 // $response->headers->set('Content-Security-Policy-Report-Only', $policy);  
 }  
}

**Tests attendus** : dans DevTools → Network → onglet **Headers**, on voit l’en‑tête Content-Security-Policy avec les directives ci‑dessus.

## 5) Pages d’erreurs personnalisées

Créez ces templates :

/templates/bundles/TwigBundle/Exception/  
 ├─ error401.html.twig  
 ├─ error403.html.twig  
 ├─ error404.html.twig  
 └─ error405.html.twig

Exemple minimal :

{# templates/bundles/TwigBundle/Exception/error403.html.twig #}  
{% extends 'base.html.twig' %}  
{% block title %}Accès refusé (403){% endblock %}  
{% block body %}  
 <h1>🚫 Accès refusé</h1>  
 <p>Vous n’avez pas les droits pour accéder à cette ressource.</p>  
 <a href="{{ path('app\_home') }}" class="btn btn-secondary">Retour</a>  
{% endblock %}

**Astuce** : tester en **env=prod** (APP\_ENV=prod APP\_DEBUG=0) pour forcer les vraies pages d’erreur.

## 6) Cookie mode sombre / clair

### 6.1 Contrôleur

// src/Controller/ThemeController.php  
namespace App\Controller;  
  
use Symfony\Component\HttpFoundation\Response;  
use Symfony\Component\HttpFoundation\Cookie;  
use Symfony\Component\Routing\Annotation\Route;  
  
class ThemeController  
{  
 #[Route('/theme/{mode}', name: 'app\_theme', requirements: ['mode' => 'dark|light'])]  
 public function set(string $mode): Response  
 {  
 $response = new Response('OK');  
 $value = $mode === 'dark' ? 'true' : 'false';  
 $cookie = Cookie::create('myapp\_dark\_mode', $value, strtotime('+1 year'), '/', null, false, true, false, Cookie::SAMESITE\_LAX);  
 $response->headers->setCookie($cookie);  
 return $response;  
 }  
}

### 6.2 Utilisation côté Twig/CSS

{# base.html.twig #}  
{% set dark = app.request.cookies.get('myapp\_dark\_mode') == 'true' %}  
<html class="{{ dark ? 'theme-dark' : 'theme-light' }}">

**Capture d’écran attendue** : onglet **Application/Storage → Cookies** montrant myapp\_dark\_mode.

## 7) Protection CSRF

* Assurez‑vous que framework.yaml contient csrf\_protection: true (valeur par défaut en Flex).
* Pour les formulaires générés par Symfony Form, le token est géré automatiquement.
* Pour un formulaire « manuel », ajoutez :

<form method="post" action="{{ path('book\_delete', {id: book.id}) }}" onsubmit="return confirm('Confirmer ?');">  
 <input type="hidden" name="\_token" value="{{ csrf\_token('delete' ~ book.id) }}">  
 <button class="btn btn-danger">Supprimer</button>  
</form>

**Capture d’écran attendue** : inspecteur DOM montrant l’input \_token.

## 8) Audit des dépendances

### 8.1 Commande

composer audit

* **But** : interroger la base de vulnérabilités de friends-of-php/security-advisories et signaler les paquets affectés (CVE, sévérité, version corrigée).
* **À fournir** : copier/coller **l’intégralité** de la sortie dans la section ci‑dessous.

**Sortie de composer audit :**

[Collez ici la sortie complète]

**Astuce** : lancer aussi composer outdated --direct pour voir les MAJ directes disponibles.

## 9) Captures et justification

* **Session < 48h** : capture du cookie de remember‑me avec date d’expiration ≤ maintenant+48h.
* **Cookie dark/light** : capture de myapp\_dark\_mode.
* **CSRF** : capture du champ \_token dans le DOM.
* **CSP** : capture de l’en‑tête Content-Security-Policy dans Network → Headers.

Collez les images ci‑dessous (une par sous‑section) :

![Session 48h](./screens/session-48h.png)  
![Cookie thème](./screens/cookie-theme.png)  
![CSRF token](./screens/csrf-token.png)  
![CSP header](./screens/csp-header.png)

**Checklist de prise de vue** 1. Ouvre DevTools → Application → Cookies : capture REMEMBERME (expiration < 48h). 2. Va sur /theme/dark puis /theme/light : capture myapp\_dark\_mode. 3. Ouvre DevTools → Network → une requête HTML : capture l’en‑tête Content-Security-Policy. 4. Sur la page Liste/Show avec bouton Supprimer : Inspect → capture l’<input name="\_token">.

## 10) Bonus (optionnel) – Renforcement mot de passe

### 10.1 Règles serveur (validator custom)

// src/Validator/StrongPassword.php (Constraint + Validator à créer)  
// Exigences : longueur ≥ 12, ≥1 maj, ≥1 min, ≥1 chiffre, ≥1 spécial

### 10.2 Feedback front simple

<input type="password" id="pwd" oninput="checkPwd(this.value)">  
<small id="pwd-hint"></small>  
<script>  
function checkPwd(v){  
 const ok = v.length>=12 && /[A-Z]/.test(v) && /[a-z]/.test(v) && /\d/.test(v) && /[^\w]/.test(v);  
 document.getElementById('pwd-hint').textContent = ok ? '✅ Mot de passe fort' : '❗ Trop faible';  
}  
</script>

## 11) Difficultés rencontrées

* *Décrivez brièvement les blocages et leur résolution.*

## 12) Bilan des acquis

* *Ce que vous avez appris / consolidé.*

## 13) Commandes utiles (mémo)

php bin/console cache:clear --env=prod  
php bin/console debug:router  
php bin/console security:hash-password 'MonSuperMot2Passe!'  
composer audit  
composer outdated --direct

## 14) Annexes

* Versions : php -v, symfony -v, composer -V
* Extraits de logs pertinents : var/log/\*.log