Liste des moyens mis en place pour assurer la sécurité de l'application

Faille XSS

Une faille XSS est la possibilité qu'un utilisateur malveillant puisse exécuter du code indésirable sur le navigateur d'un autre utilisateur.
→ Le moteur de template TWIG est utilisé sur cette application afin de s'en protéger.

Injection SQL

Une injection SQL consiste à introduire des données malsaines dans la base de données d'une application.
→ Validation des données reçues par les formulaire à l'aide du composant Validator
→ Utilisation de doctrine qui contre naturellement cette attaque
→ Utilisation de setParameter dans les requêtes DQL

HTTPS

Https est l'association de http et d'une méthode de chiffrement comme SSL par exemple. Il certifie également l'identité du site web à l'aide d'un certificat.
→ Utilisant un plan heroku gratuit, je ne profite pas de cette fonctionnalité. En revanche, pour une application réelle, https ne serait pas une option.

Debug mode

Le mode debug est très pratique en développement. En revanche, en production, il est nécessaire de le désactiver afin de ne pas afficher des données sensibles de l'application.
→ APP_DEBUG=0 dans le fichier .env

Téléchargement de fichiers

Avec un compte employé ou administrateur, il est possible d'ajouter une image à un livre. Bien que ce soit des comptes à hautes responsabilités, le risque qu'ils importent un script shell ou php n'est pas nul.
→ Mise en place, dans tous les formulaires concernant les livres, d'une obligation d'importer uniquement des fichiers de type jpeg ou png d'une taille maximale de 1024K.

Contraintes (Validator)

Ne jamais faire confiance aux valeurs des formulaires.
→ Le composant Validator permet de vérifier chaque donnée d'une entité afin qu'elle corresponde exactement au format attendu. Par exemple, tous les champs textes des entités User et Book ont une contrainte de taille de 255 caractères et ne peuvent être nuls.

Hashage des mots de passes

Un mot de passe stocké en clair dans une base de donnée est un cadeau pour un pirate informatique
→ Les mots de passe sont hachés avec le meilleur algorithme avant d'être stockés.

Attaque CSRF

Une attaque CSRF consiste à récupérer les données d'un utilisateur connecté à l'aide d'un site externe puis de se servir de celles-ci pour duper l'authentification de l'application.
→ Dans le formulaire de connexion, un token CSRF est placé dans un input caché. Grâce à cela, une requête http venant de l'extérieur ne fonctionnera pas.

Autorisation

Toutes les pages d'une application ne doivent pas être accessibles à tout le monde. Afin de protéger l'accès à des pages à plus haute responsabilité, il est nécessaire d'implémenter un système de rôle.
→ Chaque utilisateur qui s'inscrit obtient automatiquement le ROLE_REGISTRANT (inscrit)
→ Les autres rôles (EMPLOYEE & ADMIN) ne peuvent pas être donnés lors de la soumission du formulaire de création de compte.
→ Ajout d'un access_control dans le fichier security.yaml permettant de restreindre l'accès selon le rôle de l'utilisateur.