



Audit de sécurité d'une application web basée sur symfony (symfony-demo)

OUHMAN Abdessamad - ADARDOUR Youssef

*Pr. SADQI Yassine
Pr. HAZMAN Chaimae*

*LST Cybersécurité
2024/2025*

Contents

1 INTRODUCTION:	1
2 Présentation de l'Environnement:	1
2.1 Le framework symfony:	1
2.1.1 Qu'est ce qu'un framework ?	1
2.1.2 Pourquoi un framework ?	1
2.1.3 Pourquoi symfony ?	1
2.1.4 Modèle MVC (Model, View, controller):	2
2.1.5 MVC dans une application Web :	4
2.1.6 Création d'un projet symfony:	4
2.2 Symfony-demo: Application de démonstration Symfony	5
2.3 Docker	7
2.3.1 Qu'est ce que Docker ?	7
2.3.2 Architecture Docker:	8
2.3.3 Pourquoi intégrer Docker dans le projet ?	9
2.4 Préparation de l'environnement et intégration de docker:	9
2.5 Les outils utilisés pour l'audit:	13
2.5.1 OWASP ZAP :	13
3 Méthodologie de l'Audit de Sécurité:	14
3.1 Planification et cadrage	14
3.2 Analyse statique du code source	15
3.3 Tests d'intrusion manuels	15
3.4 Evaluation des vulnérabilités	15
3.5 Reporting et communication	15
3.6 Contrôle des correctifs et clôture	16
4 Analyse des Vulnérabilités:	16
4.1 Généralités:	16
4.1.1 Qu'est-ce que l'analyse des vulnérabilités ?	16
4.1.2 Qu'est-ce qu'une vulnérabilité de sécurité ?	16
4.1.3 Pourquoi l'analyse des vulnérabilités est importante ?	16
4.1.4 Types d'analyses des vulnérabilités:	17
4.1.5 Comment fonctionne l'analyse des vulnérabilités ?	17
4.2 Sécurité des dépendances:	18
4.2.1 Mettre à jour régulièrement Symfony et ses dépendances	18
4.2.2 Protéger les fichiers de configuration sensibles:	18
4.2.3 Activer l'authentification et la gestion des droits:	18
4.2.4 Protéger contre les attaques XSS et CSRF Attaques XSS (Cross-Site Scripting) et Attaques CSRF (Cross-Site Request Forgery)	18
4.2.5 Sécuriser les entêtes HTTP:	19
4.3 Sécurité des configurations:	19
4.4 Sécurité des contrôleurs et routes:	19
4.4.1 Le pare-feu : " Firewall "	19
4.4.2 Authentification des utilisateurs:	19
4.5 Sécurité des formulaires et des entrées utilisateur:	20
4.6 Sécurité API et communication avec d'autres services:	22
4.6.1 Qu'est-ce que la sécurité des API ?	22
4.6.2 Principaux types d'attaques d'API	23
4.6.3 Qu'est-ce qu'une API ?	23
4.6.4 Pourquoi la sécurité des API est-elle si importante ?	23
4.6.5 Risques liés à la sécurité des API	23
5 Test d'intrusion:	24
5.1 Analyse automatique:	24
5.2 Analyse manuel:	27

6 Recommandations et Corrections:	27
6.1 Signification des alertes générées générées:	27
6.2 Recommandations:	27
7 Conclusion:	27
8 Disclaimer :	28
9 Ressources:	29

1 INTRODUCTION:

Dans un contexte où les cyberattaques deviennent de plus en plus sophistiquées, il est essentiel pour les entreprises de s'assurer que leurs applications Web sont suffisamment sécurisées. Pour y parvenir, l'audit de sécurité des applications est un processus indispensable permettant de détecter les vulnérabilités et de renforcer la protection des données.

2 Présentation de l'Environnement:

2.1 Le framework symfony:



Selon le site officiel de symfony:

Symfony est un ensemble de composants PHP, un framework d'application Web, une philosophie et une communauté, tous fonctionnant ensemble en harmonie.

2.1.1 Qu'est ce qu'un framework ?

Un framework permet de s'assurer qu'une application est bien structurée, qu'elle est maintenable et qu'il est possible de facilement la mettre à jour. Il permet également de réutiliser des fonctionnalités déjà codées, en utilisant des modules. En résumé, il s'agit d'un ensemble de logiciels permettant d'améliorer l'expérience et la productivité des développeurs.

2.1.2 Pourquoi un framework ?

Un framework n'est pas absolument nécessaire : c'est « juste » l'un des outils disponibles pour aider à développer mieux et plus rapidement !

Mieux, car un framework offre la certitude de développer une application entièrement conforme aux règles métier, structurée et à la fois **maintenable** et **évolutive**.

Plus rapide, car il permet aux développeurs de gagner du temps en réutilisant des modules génériques afin de se concentrer sur d'autres domaines. Sans toutefois jamais être lié au framework lui-même.

2.1.3 Pourquoi symfony ?

- **Réputation:**

Rapidement adopté par les professionnels actifs dans ce domaine dès son lancement en 2005, Symfony est aujourd'hui un environnement stable, connu et reconnu à l'échelle internationale. Le nombre de ses références en témoigne, puisqu'elles ont considérablement augmenté depuis son lancement. Symfony, c'est aussi une communauté active ; développeurs, intégrateurs, utilisateurs et autres contributeurs participent à l'enrichissement permanent de cet outil.

- **Permanence:**

Symfony a été créé à l'origine par l'agence interactive **SensioLabs**. Conçu par des professionnels pour des professionnels, Symfony est avant tout un outil pragmatique, dont les fonctionnalités répondent à des besoins concrets.

La pérennité est également un élément lié au support à long terme. Le support professionnel de Symfony est assuré par SensioLabs, mais il existe également tout un écosystème qui s'est développé autour de Symfony depuis son lancement : la communauté (Slack, GitHub Discussions, etc.) et les nombreuses autres sociétés de services qui ont investi dans le framework.

Enfin, c'est également dans une optique de développement durable que Symfony est distribué sous licence Open Source MIT, qui n'impose pas de contraintes et permet le développement d'applications Open Source comme propriétaires.

- **Innovation:** Symfony est tout ce que l'on attend d'un framework : rapidité, flexibilité, composants réutilisables, etc. Ensuite, il y a la structure de ce qui a été développé et l'utilisation des meilleures pratiques. Pas mal !

Mais ce n'est pas tout ! SensioLabs ayant pris l'habitude de bousculer l'ordre établi et recherchant toujours l'excellence, Symfony (et toute sa communauté) a développé un sens de la curiosité qui va bien au-delà de PHP. Et nous n'hésitons pas à innover en allant chercher des idées ailleurs pour ensuite les adapter au monde de PHP, comme par exemple l'injection de dépendances venue du monde Java.

De plus, Symfony, cherchant à améliorer en permanence la productivité des développeurs, a imaginé la « barre d'outils de débogage Web », empruntée à d'autres frameworks, qu'ils soient PHP ou non.

- **Ressources:** L'utilisateur de Symfony a l'assurance de ne jamais « être seul avec son écran ». Qu'il s'agisse de support communautaire (listes de diffusion, IRC, etc.) ou de support d'entreprise (conseil, formation , etc.), il trouvera toujours les réponses à ses questions.

Partant du principe qu'« une ligne non documentée est une ligne qui n'existe pas », Il y a également de nombreux ouvrages dédiés à Symfony, qui aideront les développeurs tout au long du développement de leurs sites et applications.

- **Interopérabilité:** L'idée derrière Symfony : Ne vous enfermez pas dans Symfony ! Permettez-vous de construire des applications qui répondent précisément à vos besoins !

Symfony respecte les « standards de fait » existants de PHP : PHPUnit, conventions de nommage des classes, etc. De plus, Symfony permet également d'utiliser certaines parties de ses briques logicielles (injecteur de dépendances, gestion des traductions, gestion des formulaires, etc.) sans nécessairement utiliser le framework dans son intégralité.

D'ailleurs, Symfony est tellement interopérable qu'il utilise à la base des blocs de construction logiciels externes (ORM Doctrine, Swiftmailer, etc.) !

2.1.4 Modèle MVC (Model, View, controller):

Au cours des dernières années, les sites Web sont passés de simples pages HTML avec un peu de CSS à des applications incroyablement complexes sur lesquelles travaillent des milliers de développeurs en même temps. Pour travailler avec ces applications Web complexes, les développeurs utilisent différents modèles de conception pour concevoir leurs projets, afin de rendre le code moins complexe et plus facile à utiliser. Le plus populaire de ces modèles est MVC, également connu sous le nom de Model View Controller.

Le framework Modèle-Vue-Contrôleur (MVC) est un modèle d'architecture/conception qui sépare une application en trois composants logiques principaux : Modèle , Vue et Contrôleur . Chaque composant architectural est conçu pour gérer des aspects de développement spécifiques d'une application. Il isole la logique métier et la couche de présentation l'une de l'autre. Il était traditionnellement utilisé pour les interfaces utilisateur graphiques (GUI) de bureau . De nos jours, MVC est l'un des frameworks de développement Web standard les plus fréquemment utilisés pour créer des projets évolutifs et extensibles. Il est également utilisé pour la conception d'applications mobiles.

MVC a été créé par Trygve Reenskaug . L'objectif principal de ce modèle de conception était de résoudre le problème des utilisateurs contrôlant un ensemble de données volumineux et complexes en divisant une grande application en sections spécifiques ayant chacune leur propre objectif.

Le framework MVC comprend les 3 composants suivants :

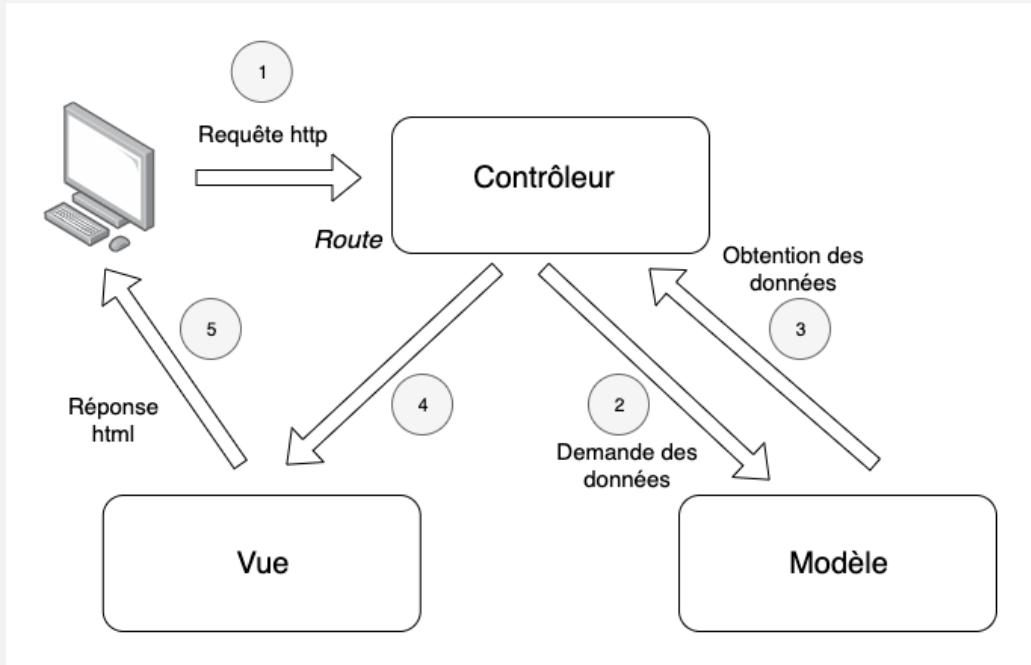


Figure 1: La logique MVC

- **Contrôleur :**

Le contrôleur est le composant qui permet l'interconnexion entre les vues et le modèle, agissant ainsi comme intermédiaire. Le contrôleur n'a pas à se soucier de la gestion de la logique des données, il indique simplement au modèle ce qu'il doit faire. Il traite toute la logique métier et les requêtes entrantes, manipule les données à l'aide du composant Modèle et interagit avec la vue pour restituer le résultat final.

Responsabilités:

- Recevoir les entrées des utilisateurs et les interpréter.
- Mise à jour du modèle en fonction des actions de l'utilisateur.
- Sélection et affichage de la vue appropriée.

Exemple : dans une application de librairie, le contrôleur gérerait des actions telles que la recherche d'un livre, l'ajout d'un livre au panier ou le paiement.

- **Modèle :** Le composant Modèle correspond à toute la logique liée aux données avec laquelle l'utilisateur travaille. Il peut représenter soit les données transférées entre les composants Vue et Contrôleur, soit toute autre donnée liée à la logique métier. Il peut ajouter ou récupérer des données de la base de données. Il répond à la demande du contrôleur car ce dernier ne peut pas interagir avec la base de données par lui-même. Le modèle interagit avec la base de données et renvoie les données requises au contrôleur.

Responsabilités:

- Gestion des données : opérations CRUD (Créer, Lire, Mettre à jour, Supprimer).
- Application des règles commerciales.
- Notification de la vue et du contrôleur des changements d'état.

Exemple : dans une application de librairie, le modèle gérerait les données relatives aux livres, telles que le titre du livre, l'auteur, le prix et le niveau de stock.

- **Vue :** Le composant View est utilisé pour toute la logique d'interface utilisateur de l'application. Il génère une interface utilisateur pour l'utilisateur. Les vues sont créées par les données collectées par le composant modèle, mais ces données ne sont pas récupérées directement mais via le contrôleur. Il interagit uniquement avec le contrôleur.

Responsabilités:

- Restituer les données à l'utilisateur dans un format spécifique.
- Affichage des éléments de l'interface utilisateur.
- Mise à jour de l'affichage lorsque le modèle change.

Exemple : dans une application de librairie, la vue afficherait la liste des livres, les détails du livre et fournirait des champs de saisie pour rechercher ou filtrer des livres.

2.1.5 MVC dans une application Web :

- L'utilisateur envoie une requête HTTP, par exemple POST /inscription.
- **Le routeur** de l'application va appeler le bon contrôleur en fonction de la route (c'est-à-dire en fonction du verbe utilisé et de l'URI). Dans notre exemple, ce sera le contrôleur responsable de l'inscription d'un utilisateur.
- **Le contrôleur** est responsable de coordonner les actions à effectuer. Ici par exemple, il peut vérifier le contenu de la requête, puis demander au modèle d'inscrire l'utilisateur dans la base de données. Une fois que le modèle aura inscrit l'utilisateur, il pourra préparer la vue à retourner à l'utilisateur.
- **Le modèle** est responsable de la validation de la lecture et de l'enregistrement des données. Dans notre exemple, il va s'assurer que toutes les données pour créer un utilisateur ont bien été fournies. Dans l'affirmative il va créer l'utilisateur dans la base de données et le passer au contrôleur.
- **La vue** est responsable de l'interface graphique. Elle contient la logique pour l'affichage des données qui ont été récupérées dans par le modèle.

2.1.6 Création d'un projet symfony:

2.1.7.1 Exigences techniques :

- Installez PHP 8.2 ou supérieur et ces extensions PHP (qui sont installées et activées par défaut dans la plupart des installations PHP 8).
- Installez Composer , qui est utilisé pour installer les packages PHP.

2.1.7.2 Création d'applications Symfony

- Ouvrez votre terminal de console et exécutez l'une de ces commandes pour créer une nouvelle application Symfony .

```
# run this if you are building a traditional web application
$ composer create-project symfony/skeleton:"7.2.x" my_project_directory
$ cd my_project_directory
$ composer require webapp

# run this if you are building a microservice, console application or API
$ composer create-project symfony/skeleton:"7.2.x" my_project_directory
```

Figure 2: my_project_directory

2.1.7.3 Configuration d'un projet Symfony existant :

En plus de créer de nouveaux projets Symfony, nous travaillerons également sur des projets existants développés par d'autres. Pour cela, il suffit de récupérer le code du projet et d'installer les dépendances avec Composer. Si notre équipe utilise Git, nous pouvons configurer notre projet avec les commandes suivantes :

```

# clone the project to download its contents
$ cd projects/
$ git clone ...

# make Composer install the project's dependencies into vendor/
$ cd my-project/
$ composer install

```

Figure 3: Configuration d'un projet

2.1.7.4 Exécution d'applications Symfony :

En production, il est essentiel d'installer un serveur web comme Nginx ou Apache et de le configurer pour exécuter Symfony. Cette méthode peut également être utilisée si vous ne souhaitez pas utiliser le serveur web local de Symfony pour le développement. Toutefois, pour un environnement de développement local, la solution la plus pratique est d'utiliser le serveur web intégré au binaire Symfony. Ce serveur prend en charge HTTP/2, les requêtes simultanées, TLS/SSL et génère automatiquement des certificats de sécurité. Pour le démarrer, ouvrez un terminal, accédez au répertoire de votre projet et exécutez la commande suivante :

```

$ cd my-project/
$ symfony server:start

```

Figure 4: Exécution du serveur Symfony

2.1.7.5 Installation de packages :

Lors du développement d'applications Symfony, il est courant d'installer des packages, appelés bundles, qui apportent des fonctionnalités prêtées à l'emploi. Ces packages nécessitent généralement une configuration initiale, comme l'activation du bundle ou l'ajout de paramètres dans un fichier de configuration.

Pour automatiser cette tâche, Symfony inclut Symfony Flex, un outil qui facilite l'installation et la suppression de packages. Symfony Flex est un plugin Composer installé par défaut lors de la création d'une nouvelle application Symfony, et il permet d'automatiser les configurations nécessaires, simplifiant ainsi la gestion des dépendances dans les projets Symfony.

Si vous exécutez cette commande dans une application Symfony qui n'utilise pas Symfony Flex, une erreur Composer apparaîtra, indiquant que le nom du package logger n'est pas valide. En revanche, si Symfony Flex est installé, cette commande installera et activera automatiquement tous les packages nécessaires à l'utilisation du logger officiel de Symfony.

Cela est possible grâce aux "recettes", qui sont des ensembles d'instructions automatisées permettant d'installer et d'activer des packages dans une application Symfony. Symfony Flex gère ces recettes et enregistre leur état dans le fichier `symfony.lock`, qui doit être validé dans le dépôt de code pour assurer la cohérence de l'installation à travers les environnements.

2.2 Symfony-demo: Application de démonstration Symfony

L'application de démonstration Symfony est une application de référence créée pour montrer comment développer des applications en suivant les meilleures pratiques Symfony

Installation :

- **Option 1.** Téléchargez Symfony CLI et utilisez le symfonybinaire installé sur votre ordinateur pour exécuter cette commande :

```
PS C:\Users\youss\Desktop\audit_smf\test\my_project> symfony new --demo my_project
```

Figure 5: commande

- Option 2. Téléchargez Composer et utilisez le composerbinaire installé sur votre ordinateur pour exécuter ces commandes :

```
PS C:\Users\youss\Desktop\audit_smf\test\my_project> composer create-project symfony/symfony-demo my_project
```

Figure 6: commande 1

```
PS C:\Users\youss\Desktop\audit_smf\test\my_project> git clone https://github.com/symfony/demo.git my_project
```

Figure 7: commande 2

```
PS C:\Users\youss\Desktop\audit_smf\test\my_project> cd my_project/
```

Figure 8: commande 3

```
PS C:\Users\youss\Desktop\audit_smf\test\my_project> composer install
```

Figure 9: commande 4

Usage :

- **Option 1.** Téléchargez Symfony CLI et exécutez cette commande :

```
PS C:\Users\youss\Desktop\audit_smf\test\my_project> symfony serve
```

Figure 10: symfony serve

Accédez ensuite à l'application dans votre navigateur à l'URL indiquée (<https://localhost:8000> par défaut).

- **Option 2.** Utilisez un serveur Web comme Nginx ou Apache pour exécuter l'application

```
PS C:\Users\youss\Desktop\audit_smf\test\my_project> php -S localhost:8000 -t public/
```

Figure 11: localhost:8000

Le serveur Web PHP intégré utilisé par défaut le port 8000. Cependant, il est possible de spécifier un autre port, comme 8080, si nécessaire.

Interface Graphique :

Bienvenue sur l'application Symfony Demo

French ▾

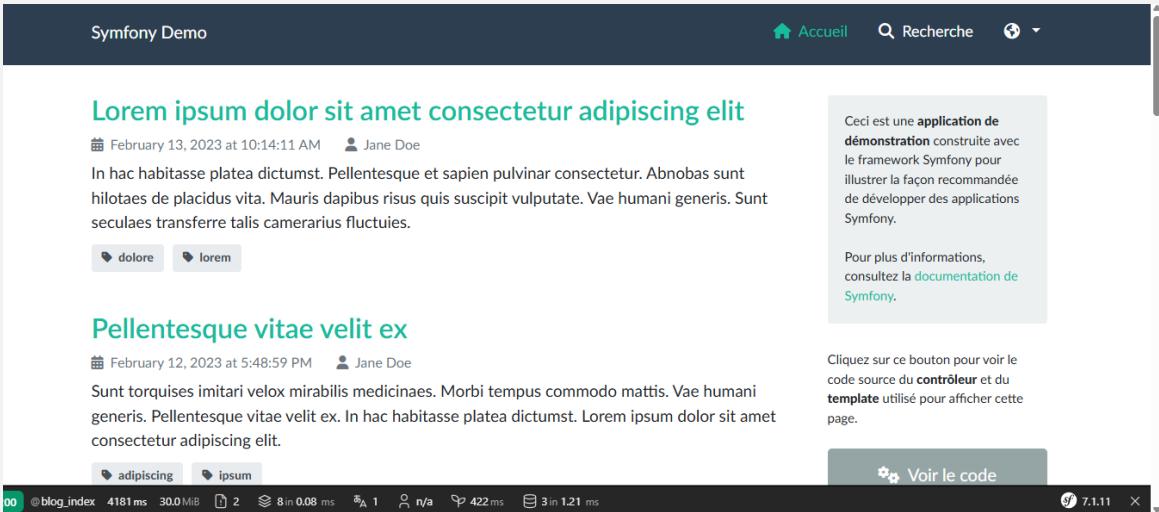
Parcourir la **section publique** de l'application de démonstration.

 Naviguer sur l'application

Parcourir l'**interface d'administration** de l'application de démonstration.

 Naviguer sur l'admin

Figure 12: Symfony Demo



The screenshot shows a web browser displaying the Symfony Demo application. The title bar says "Symfony Demo". The main content area shows two blog posts:

- Post 1:** "Lorem ipsum dolor sit amet consectetur adipiscing elit" (February 13, 2023 at 10:14:11 AM by Jane Doe). It contains placeholder text and categories "dolore" and "lorem".
- Post 2:** "Pellentesque vitae velit ex" (February 12, 2023 at 5:48:59 PM by Jane Doe). It also contains placeholder text and categories "adipiscing" and "ipsum".

To the right of the posts, there is a sidebar with the following text:

Ceci est une **application de démonstration** construite avec le framework Symfony pour illustrer la façon recommandée de développer des applications Symfony.

Pour plus d'informations, consultez la [documentation de Symfony](#).

At the bottom right of the sidebar is a button labeled "Voir le code". The footer of the browser window shows performance metrics: 4181 ms, 30.0 MB, 2 requests, 8 in 0.08 ms, 1 error, n/a, 422 ms, 3 in 1.21 ms, and sf 7.1.11.

Figure 13: section publique de l'application

2.3 Docker



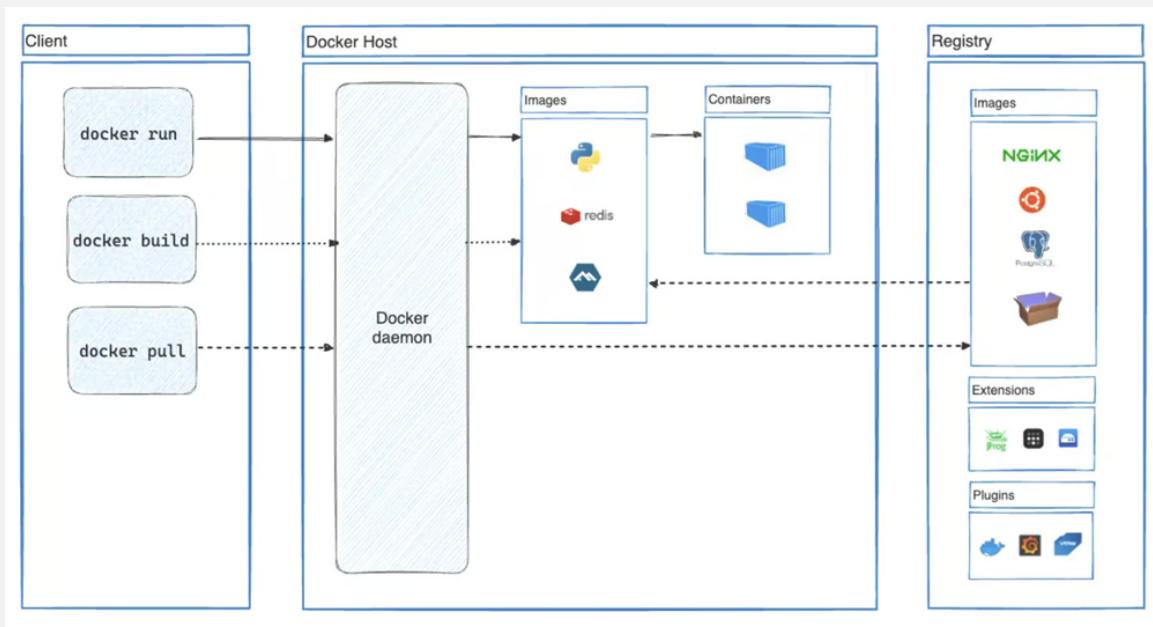
2.3.1 Qu'est ce que Docker ?

Docker est une plateforme ouverte pour le développement, la livraison et l'exécution d'applications. Docker vous permet de séparer vos applications de votre infrastructure afin de livrer vos logiciels rapidement. Avec Docker, vous gérez votre infrastructure de la même manière que vous gérez vos applications. En tirant parti des méthodologies Docker pour la livraison, les tests et le déploiement du code, vous pouvez réduire considérablement le délai entre l'écriture du code et son exécution en production.

- **Avantages de Docker :**
- **Cohérence** : Tous les développeurs travaillent avec la même configuration (versions de PHP, de MySQL, etc.).
- **Isolation** : Les conteneurs n'interagissent pas entre eux, limitant les risques de conflits.
- **Flexibilité** : Vous pouvez facilement mettre à jour ou remplacer des services (ex. changer de version de PHP) sans affecter le reste du système.

2.3.2 Architecture Docker:

Docker utilise une architecture client-serveur. Le client Docker communique avec le démon Docker, qui se charge de la création, de l'exécution et de la distribution de vos conteneurs Docker. Le client et le démon Docker peuvent s'exécuter sur le même système, ou vous pouvez connecter un client Docker à un démon Docker distant. Le client et le démon Docker communiquent via une API REST, des sockets UNIX ou une interface réseau. Docker Compose est un autre client Docker qui vous permet de travailler avec des applications composées d'un ensemble de conteneurs.



Le démon Docker:

Le démon Docker (dockerd) écoute les requêtes de l'API Docker et gère les objets Docker tels que les images, les conteneurs, les réseaux et les volumes. Un démon peut également communiquer avec d'autres démons pour gérer les services Docker.

Le client Docker:

Le client Docker (docker) est le principal moyen par lequel de nombreux utilisateurs interagissent avec Docker. Lorsque vous utilisez des commandes telles que docker run, le client les envoie à dockerd, qui les exécute. La dockercommande utilise l'API Docker. Le client Docker peut communiquer avec plusieurs démons.

Docker Desktop

Docker Desktop est une application facile à installer pour votre environnement Mac, Windows ou Linux. Elle vous permet de créer et de partager des applications conteneurisées et des microservices. Docker Desktop inclut le démon Docker (dockerd), le client Docker (docker), Docker Compose, Docker Content Trust, Kubernetes et Credential Helper.

Registres Docker

Un registre Docker stocke les images Docker. Docker Hub est un registre public accessible à tous, et Docker recherche les images sur Docker Hub par défaut. Vous pouvez même gérer votre propre registre privé.

Lorsque vous utilisez les commandes docker pullou docker run, Docker extrait les images requises de votre registre configuré. Lorsque vous utilisez la docker pushcommande, Docker envoie votre image vers votre registre configuré.

Objets Docker

Lorsque vous utilisez Docker, vous créez et utilisez des images, des conteneurs, des réseaux, des volumes, des plugins et d'autres objets. Cette section présente brièvement certains de ces objets.

Images

Une image est un modèle en lecture seule contenant des instructions pour créer un conteneur Docker. Souvent, une image est basée sur une autre image, avec quelques personnalisations supplémentaires. Par exemple, vous pouvez créer une image basée sur l'ubuntu image, mais qui installe le serveur web Apache et votre application, ainsi que les détails de configuration nécessaires à son exécution.

Vous pouvez créer vos propres images ou utiliser uniquement celles créées par d'autres et publiées dans un registre. Pour créer votre propre image, créez un Dockerfile avec une syntaxe simple définissant les étapes nécessaires à sa création et à son exécution. Chaque instruction d'un Dockerfile crée une couche dans l'image. Lorsque vous modifiez le Dockerfile et reconstruisez l'image, seules les couches modifiées sont reconstruites. C'est en partie ce qui rend les images si légères, compactes et rapides par rapport aux autres technologies de virtualisation.

Conteneurs

Un conteneur est une instance exécutable d'une image. Vous pouvez créer, démarrer, arrêter, déplacer ou supprimer un conteneur à l'aide de l'API Docker ou de la CLI. Vous pouvez connecter un conteneur à un ou plusieurs réseaux, lui attacher du stockage ou même créer une nouvelle image en fonction de son état actuel.

Par défaut, un conteneur est relativement bien isolé des autres conteneurs et de sa machine hôte. Vous pouvez contrôler le degré d'isolement du réseau, du stockage ou des autres sous-systèmes sous-jacents d'un conteneur par rapport aux autres conteneurs ou à la machine hôte.

Un conteneur est défini par son image ainsi que par les options de configuration que vous lui fournissez lors de sa création ou de son démarrage. Lorsqu'un conteneur est supprimé, toutes les modifications de son état qui ne sont pas enregistrées dans le stockage persistant disparaissent.

2.3.3 Pourquoi intégrer Docker dans le projet ?

L'utilisation de Docker présente plusieurs avantages sur divers aspects (sécurité, organisation, etc.), mais dans cette partie, nous allons mentionner uniquement les avantages qui ont motivé son intégration dans ce projet:

- **Sécurité:** Étant donné qu'il s'agit d'un projet d'audit de sécurité, dans lequel nous allons utiliser des outils de tests d'intrusion (pentest) pour attaquer et évaluer la sécurité de notre application web Symfony-demo, notre machine peut être susceptible d'être endommagée ou d'éprouver des problèmes en raison de l'utilisation de ces outils. Il est donc préférable d'isoler cet environnement de test d'intrusion dans un conteneur ou plutôt dans des conteneurs, afin de protéger notre machine et de ne pas compromettre sa sécurité.
- **Portabilité:** Docker permet de conteneuriser l'environnement de travail avec toutes les dépendances nécessaires. Cela signifie que l'image Docker que créée (avec l'application, les configurations, et les autres outils) peut être transférée facilement vers une autre machine.

Pas de dépendance au système hôte : Peu importe si on travaille sur un autre ordinateur avec un système d'exploitation différent (Peu importe l'infrastructure sous-jacente) Docker garantit que l'application et son environnement s'exécuteront de manière identique sur la nouvelle machine.

2.4 Préparation de l'environnement et intégration de docker:

L'environnement de travail de ce projet consiste à créer trois conteneurs, un pour l'application symfony-demo, un autre pour la base de données et le dernier conteneur pour l'outil de pentest (test d'intrusion) OWASP ZAP.

Après avoir cloné l'application Symfony-demo depuis GitHub, il y a plusieurs étapes à suivre afin de créer un environnement de travail basé sur docker.

D'abord, il faut installer les dépendances de notre projet symfony:

```
(abdes@hack:~/demo]
$ ls
assets composer.lock data sym migrations phpunit.xml.dist src tests
bin config importmap.php phpstan-baseline.neon public symfony.lock translations
composer.json CONTRIBUTING.md LICENSE phpstan.dist.neon README.md templates var translations
composer.json.dist Dockerfile migrations phpunit.xml.dist src tests vendor
[abdes@hack:~/demo]
$ composer install
Installing dependencies from lock file (including require-dev)
Verifying lock file contents can be installed on current platform.
Package operations: 143 installs, 0 updates, 0 removals
- Downloading staabm/side-effects-detector (1.0.5)
- Downloading sebastian/version (5.0.2)
- Downloading sebastian/type (5.1.0)
- Downloading sebastian/recursion-context (6.0.2)
- Downloading sebastian/object-reflector (4.0.1)
- Downloading sebastian/object-enumerator (6.0.1)
- Downloading sebastian/global-state (7.0.2)
```

Figure 14: installation des dépendances

Après l'installation de dépendances, et pour créer une image docker, il faut commencer par créer le fichier dockerfile dans le répertoire du projet à partir duquel l'image docker va être créée

```
GNU nano 8.3 Dockerfile
# Utiliser une image PHP avec Apache (version PHP 8.2)
FROM php:8.2-apache

# Installer les extensions PHP nécessaires pour Symfony
RUN docker-php-ext-install pdo pdo_mysql

# Installer des dépendances système supplémentaires
RUN apt-get update && apt-get install -y \
    libicu-dev \
    libzip-dev \
    libpng-dev \
    libjpeg62-turbo-dev \
    libfreetype6-dev \
    git \
    unzip \
    && docker-php-ext-configure gd --with-freetype --with-jpeg \
    && docker-php-ext-install gd

# Copier le fichier de configuration Apache (si nécessaire)
# COPY ./config/vhost.conf /etc/apache2/sites-available/000-default.conf
# Si vous avez un fichier de configuration Apache personnalisé, décommentez cette ligne.

# Définir le répertoire de travail
WORKDIR /var/www/symfony-demo

# Copier le projet Symfony dans le conteneur
```

Figure 15: La configuration du fichier dockerfile

Il est recommandé que chaque conteneur présente un seul service. pour cela, le projet symfony doit être hébergé dans un conteneur, et la base de données doit être hébergé dans un autre conteneur.

Mais que permettre plusieurs conteneurs à se connectent, ainsi que pour pouvoir lancer le projet complet (application, base de données, dépendances, etc) en une seule commande sans avoir besoin à lancer chaque conteneur seul, il faut configurer le fichier docker-compose.yml.png dans notre répertoire.

```

GNU nano 8.3                               docker-compose.yml
Version: '3.8'

services:
  symfony:
    image: symfony-demo
    container_name: symfony-demo-container
    build:
      context: .
    ports:
      - "80:80"
    volumes:
      - ./var/www/symfony-demo
    depends_on:
      - mysql

  mysql:
    image: mysql:5.7
    container_name: symfony-mysql
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_DATABASE: symfony_demo
      MYSQL_USER: symfony_user
      MYSQL_PASSWORD: symfony_password
    ports:
      - "3306:3306"
    volumes:

```

Figure 16: La configuration du fichier compose.yml.png

La création de l'image docker pour l'application symfony-demo

```

$ sudo docker build -t symfony-demo .

[+] Building 28.5s (13/13) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 1.43kB
=> [internal] load metadata for docker.io/library/php:8.2-apache
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/8] FROM docker.io/library/php:8.2-apache@sha256:42de06c4cdff50b81b34112c700585ad1b194c961cb0e887a6906c7824223e2
=> [internal] load build context
=> => transferring context: 1.10MB
=> CACHED [2/8] RUN docker-php-ext-install pdo pdo_mysql
=> CACHED [3/8] RUN apt-get update && apt-get install -y libicu-dev libzip-dev libpng-dev libjpeg62-turbo
=> CACHED [4/8] WORKDIR /var/www/symfony-demo
=> [5/8] COPY ./var/www/symfony-demo
=> [6/8] RUN curl -S https://getcomposer.org/installer --dir=/usr/local/bin --filename=composer
=> [7/8] RUN composer install --no-dev --optimize-autoloader --no-scripts
=> [8/8] RUN chown -R www-data:www-data /var/www/symfony-demo
=> e: corting to image

```

Figure 17: Crédit de l'image docker pour le projet symfony-demo

à partir de l'image construite, on peut créer un conteneur pour notre projet, la commande "docker ps" est utilisée pour vérifier que le conteneur a été bien créé. en effet, elle permet d'afficher tous les conteneurs en cours d'exécution.

```

$ cd Bureau/symfony-demo
$ sudo docker run -d -p 80:80 --name symfony-demo-container symfony-demo
[sudo] Mot de passe de abdes:
7a4adfebf6fd250f770c12cd8fd9a8169d4cecbdf2dceec59bdb49bc7821e6c

$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS              NAMES
7a4adfebf6fd        symfony-demo        "docker-php-entrypoi..."   About a minute ago   Up About a minute   0.0.0.0:80->80/tcp, ::80->80/tcp   symfony-demo-container

$ 

```

Figure 18: Crédit de l'image docker à partir de l'image

Maintenant, on va installer une image docker d'owasp zap depuis son site officiel

```
(abdes㉿ hack)-[~/Bureau]
$ sudo docker pull ghcr.io/zaproxy/zaproxy:stable
stable: Pulling from zaproxy/zaproxy
7cf63256a31a: Downloading [=====] 11.47MB/28.22MB
4e78822cfa0e: Downloading [=>] 20.52MB/609.2MB
f6477a7a4bb4: Downloading [=====] 12.67MB/24.57MB
d02dab78c007: Waiting
```

Figure 19: L’installation de l’image docker d’owasp zap

Après la création des conteneurs et les configurer, on peut vérifier si l’environnement du travail a été bien construit.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
6c0d6f84cb50	ghcr.io/zaproxy/zaproxy:stable "bash"	/asset:17 seconds ago	Exited (0) 16 seconds ago		
ef1c1c7de953	symfony-demo	"docker-php-entrypoi..."	2 days ago	Exited (0) 25 minutes ago	80/tcp, 443/tcp, 127.0.0.1:5432/tcp, 127.0.0.1:51692/tcp
7ad40ccc1664	mysql:5.7	"docker-entrypoint.s..."	2 days ago	Exited (0) 25 minutes ago	3306/tcp
symfony-mysql					

Figure 20: Les conteneurs existants

La configuration n’est pas encore terminé, car il nous reste à configurer la base de données sur l’application symfony.

```
.env
# Format described at https://www.docker-project.org/projects/dockerfile-env/
# IMPORTANT: You MUST configure your server version, either here or in config/pac
#
#DATABASE_URL=sqlite:///%kernel.project_dir%/data/database.sqlite
DATABASE_URL="mysql://symfony_user:symfony_password@mysql:3306/symfony_demo"
# DATABASE_URL="mysql://app:!ChangeMe!@127.0.0.1:3306/app?serverVersion=8&charset=utf8mb4"
# DATABASE_URL="postresql://app:!ChangeMe!@127.0.0.1:5432/app?serverVersion=16500"
```

Figure 21: La configuration de la base de données sur l’application

Pour mettre à jour le schéma de la base de données, il faut d’abord générer des migrations et les appliquer sur cette base de données.

Pour la génération des migrations:

```
(abdes㉿ hack)-[~/demo]
$ php bin/console doctrine:migrations:diff
In SignatureHasher.php line 41:
  A non-empty secret is required.

doctrine:migrations:diff [-configuration CONFIGURATION] [-em EM] [-conn CONN] [-namespace NAMESPACE] [-filter-expression F
LITER-EXPRESSION] [-formatted] [-line-length LINE-LENGTH] [-check-database-platform [CHECK-DATABASE-PLATFORM]] [-allow-empty-
diff] [-from-empty-schema]
```

Figure 22: Erreur lors de la génération de la migration

Lors de la génération de la migration, nous avons rencontré l’erreur présentée dans la figure ci-dessus.

L’erreur est générée en raison de l’absence d’une clé secrète qui permet de garantir la sécurité, en particulier celle des sessions. En général, Symfony utilise une clé secrète pour signer les sessions et d’autres données sensibles afin de garantir leur intégrité et leur sécurité. Chaque session doit être signée avec cette clé. Même si un attaquant parvient à obtenir une session, il ne pourra pas la falsifier, car il lui faudra la clé secrète.

L’absence de cette clé secrète entraîne une erreur lors de la gestion des sessions, surtout si l’application essaie de créer ou de manipuler des sessions sans clé pour les signer. Et pour résoudre ce problème, nous devons créer une clé secret.

```
[abdes㉿ hack]:~$ php -r "echo bin2hex(random_bytes(16));" on in database "main" that could result in schema changes and data loss  
Do you want to continue? [yes/no] yes:  
82c71ad483d628c1f8e19250a43fea34
```



```
[abdes㉿ hack]:~$ | OR] The version "latest" couldn't be reached, there are no registered migrations.
```

Figure 23: Génération d'une clé secrète pour la signature

La commande présentant dans la figure permet de générer 16 octets (128 bits) aléatoires et de convertir ces octets en une chaîne hexadécimale.

Après avoir générée chaîne aléatoire, nous allons l'utilisé comme la clé secrète dans le fichier de configuration ".env".

```
① .env
② .env.dev
③ .env.test
④ .gitignore
⑤ .php_cs-fixer.dist.php
⑥ composer.json
⑦ composer.lock
⑧ CONTRIBUTING.md

13 #
14 # Run "composer dump-env prod" to compile .env files for production use (requires
15 # https://symfony.com/doc/current/best\_practices.html#use-environment-variables-for-production
16
17 ##### symfony/framework-bundle #####
18 APP_ENV=dev
19 APP_SECRET=82c71ad483d628c1f8e19250a43fea34
20 #####< symfony/framework-bundle #####
21
```

Figure 24: La configuration de la clé secrète

Maintenant nous pouvons surmonter l'erreur rencontré déjà puisque nous avons générée la clé secrète. nous allons essayer de régénérer une migration:

```
(abdes@hack:~/demo]$ php bin/console doctrine:migrations:diff
Generated new migration class to "/home/abdes/demo/migrations/Version20250322091922.php"
To run just this migration for testing purposes, you can use migrations:execute --up 'DoctrineMigrations\\Version20250322091922'
To revert the migration you can use migrations:execute --down 'DoctrineMigrations\\Version20250322091922'
```

Figure 25: Régénération de la migration

BINGO, le problème a été résolu.

Après la génération de la migration, nous allons l'appliquer à la base de données en utilisant la commande suivante:

```
(abdes@hack:~/demo]$ php bin/console doctrine:migrations:migrate
  [OK] Migrating up to DoctrineMigrations\Version20250322091922
  [notice] finished in 9.3ms, used 18M memory, 1 migrations executed, 13 sql queries
  [OK] Successfully migrated to version: DoctrineMigrations\Version20250322091922
```

Figure 26: Application de la migration à la base de données

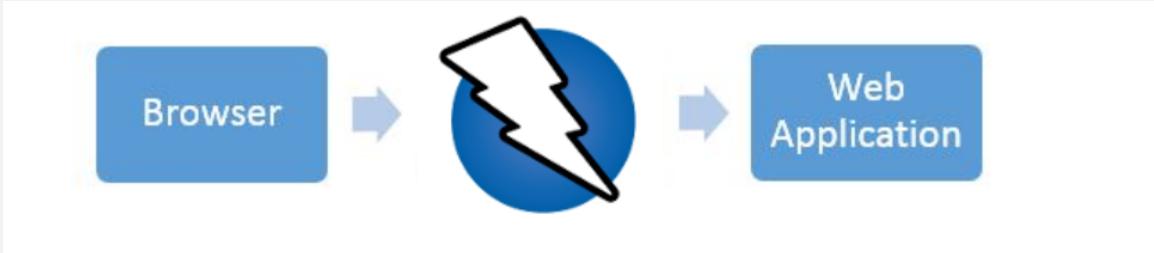
Maintenant notre environnement de travail a été bien construit et configuré.

2.5 Les outils utilisés pour l'audit:

2.5.1 OWASP ZAP :

Zed Attack Proxy (ZAP) de Checkmarx est un outil de test de pénétration gratuit et open source. ZAP est conçu spécifiquement pour tester les applications Web et est à la fois flexible et extensible.

À la base, ZAP est ce que l'on appelle un « proxy manipulateur intermédiaire ». Il se situe entre le navigateur du testeur et l'application Web afin de pouvoir intercepter et inspecter les messages envoyés entre le navigateur et l'application Web, modifier le contenu si nécessaire, puis transmettre ces paquets à la destination. Il peut être utilisé comme une application autonome et comme un processus démon.



S'il existe un autre proxy réseau déjà utilisé, comme dans de nombreux environnements d'entreprise, ZAP peut être configuré pour se connecter à ce proxy.



ZAP propose des fonctionnalités adaptées à un large éventail de niveaux de compétence : des développeurs aux testeurs novices en matière de tests de sécurité, en passant par les spécialistes des tests de sécurité. ZAP propose des versions pour chaque système d'exploitation majeur et Docker, vous n'êtes donc pas lié à un seul système d'exploitation. Des fonctionnalités supplémentaires sont disponibles gratuitement à partir de divers modules complémentaires sur la place de marché ZAP, accessibles depuis le client ZAP.

Comme ZAP est open source, le code source peut être examiné pour voir exactement comment la fonctionnalité est implémentée. N'importe qui peut se porter volontaire pour travailler sur ZAP, corriger des bugs, ajouter des fonctionnalités, créer des demandes d'extraction pour intégrer des correctifs au projet et créer des modules complémentaires pour prendre en charge des situations spécialisées.

[Cliquez ici pour plus de détails](#)

[Cliquez ici pour télécharger l'outil ZAP](#)

3 Méthodologie de l'Audit de Sécurité:

Un audit manuel de sécurité des applications web est un processus méthodique et approfondi visant à identifier, analyser et évaluer les vulnérabilités de sécurité dans une application web. Il s'agit d'un exercice crucial pour garantir la protection des applications web contre les attaques malveillantes et les intrusions de données.

Un audit de sécurité manuel peut se dérouler en 6 étapes clefs.

3.1 Planification et cadrage

Cette phase initiale définit les contours de l'audit et établit une communication claire entre les parties prenantes. Elle comprend :

- **Définition des objectifs et du périmètre :** Déterminer les objectifs spécifiques de l'audit et les éléments de l'application à analyser.
- **Identification des parties prenantes :** Identifier les individus ou les équipes responsables de l'application, de l'audit et de la prise de décision concernant les correctifs.
- **Établissement des canaux de communication :** Définir les moyens de communication entre les auditeurs et les parties prenantes pour assurer un échange fluide d'informations.
- **Compréhension des exigences spécifiques :** Acquérir une connaissance approfondie des exigences de sécurité et des contraintes réglementaires applicables à l'application.

- **Collecte d'informations** : Rassembler des informations détaillées sur l'architecture de l'application, les technologies employées, ses fonctionnalités et les flux de données.

3.2 Analyse statique du code source

L'examen minutieux du code source de l'application permet de détecter les failles de sécurité potentielles qui pourraient passer inaperçues lors de l'exécution. Cette étape implique :

- **Révision manuelle du code**: Effectuer une analyse ligne par ligne du code source pour identifier les erreurs de programmation, les configurations incorrectes et les pratiques de codage non sécurisées.
- **Utilisation d'outils d'analyse statique** : Compléter l'analyse manuelle avec des outils automatisés qui scannent le code pour des vulnérabilités connues et des failles de sécurité courantes.
- **Focalisation sur les composants critiques** : Prioriser l'analyse des composants sensibles de l'application qui gèrent les données confidentielles ou contrôlent les accès critiques.

3.3 Tests d'intrusion manuels

Les experts simulent des attaques réelles contre l'application pour exploiter les vulnérabilités identifiées et mesurer leur impact potentiel. Leur processus :

- **Reconnaissance et cartographie** : Découvrir les points d'entrée potentiels de l'application, tels que les pages web, les API et les interfaces externes.
- **Pentest Black Box** : Agir comme un attaquant externe sans aucune connaissance interne de l'application pour tenter de la pénétrer.
- **Pentest Grey Box** : Disposer d'informations partielles sur l'application pour cibler des vulnérabilités spécifiques et des points d'accès.
- **Pentest White Box** : Avoir un accès complet au code source et à l'architecture de l'application pour effectuer des tests approfondis et exploiter des failles complexes.
- **Exploitation des vulnérabilités** : Tenter d'exploiter les vulnérabilités découvertes pour exécuter des actions malveillantes telles que le vol de données, la prise de contrôle de comptes ou la défiguration du site web.

3.4 Evaluation des vulnérabilités

Une fois les vulnérabilités identifiées, elles sont analysées en détail pour déterminer leur gravité et leur priorité de correction.

- **Analyse de l'impact** : Évaluer les conséquences potentielles de l'exploitation d'une vulnérabilité, telles que la perte de données, la compromission des systèmes ou les perturbations opérationnelles.
- **Évaluation de la facilité d'exploitation** : Déterminer le niveau de difficulté requis pour exploiter une vulnérabilité, en tenant compte des compétences et des ressources nécessaires à un attaquant.
- **Priorisation des vulnérabilités** : Classer les vulnérabilités en fonction de leur gravité et de leur priorité de correction, en se basant sur l'analyse de l'impact et de la facilité d'exploitation.

3.5 Reporting et communication

Les résultats de l'audit sont documentés de manière claire et concise pour informer les parties prenantes et faciliter la prise de décision concernant les correctifs. Le rapport d'audit inclut :

- **Résumé des conclusions** : Présenter un aperçu global des vulnérabilités identifiées, de leur gravité et de leur impact potentiel.
- **Détail des vulnérabilités** : Décrire chaque vulnérabilité de manière détaillée, en incluant des informations techniques, des preuves d'exploitation et des captures d'écran si nécessaire.
- **Recommandations pour la correction** : Fournir des instructions claires et précises pour corriger les vulnérabilités identifiées, en tenant compte des contraintes techniques et la criticité de l'application.
- **Évaluation du risque résiduel** : Évaluer le niveau de risque persistant après la correction des vulnérabilités identifiées.

3.6 Contrôle des correctifs et clôture

Après la mise en œuvre des actions correctives, il est important de vérifier leur efficacité et de s'assurer que les vulnérabilités ont bien été corrigées. Les pentesters réalisent :

- **Nouveaux tests d'intrusion ciblés** : Effectuer des tests supplémentaires pour confirmer que les vulnérabilités précédemment identifiées ont été corrigées et qu'aucune nouvelle faille n'a été introduite.
- **Revue du code corrigé** : Examiner les modifications apportées au code source pour s'assurer qu'elles corrigeant efficacement les vulnérabilités sans introduire de régressions.
- **Clôture de l'audit** : Documenter la finalisation de l'audit et consigner les actions entreprises pour corriger les vulnérabilités.

4 Analyse des Vulnérabilités:

4.1 Généralités:

4.1.1 Qu'est-ce que l'analyse des vulnérabilités ?

- L'analyse des vulnérabilités, également appelée « évaluation des vulnérabilités », est le processus d'évaluation des réseaux ou des actifs informatiques à la recherche de vulnérabilités de sécurité : des failles ou des faiblesses que les acteurs malveillants externes ou internes peuvent exploiter. L'analyse des vulnérabilités est la première étape du cycle de vie plus large de la gestion des vulnérabilités.

4.1.2 Qu'est-ce qu'une vulnérabilité de sécurité ?

- **Une vulnérabilité de sécurité**, c'est tout point faible dans la structure, le fonctionnement ou la mise en œuvre d'un actif informatique ou d'un réseau que des pirates ou autres acteurs malveillants peuvent exploiter pour obtenir un accès non autorisé aux systèmes et nuire au réseau, aux utilisateurs ou à l'entreprise. Voici quelques vulnérabilités courantes :
 - Les défauts de codage, comme dans les applications web sujettes au cross-site scripting, à l'injection de code SQL et à d'autres attaques par injection en raison de la façon dont elles gèrent les entrées utilisateur.
 - Ports ouverts non protégés sur les serveurs, ordinateurs portables et autres terminaux, que les pirates peuvent utiliser pour diffuser des logiciels malveillants.
 - Mauvaises configurations, par exemple un compartiment de stockage cloud qui expose des données sensibles sur le réseau Internet public à cause d'autorisations d'accès inappropriées.
 - Des correctifs manquants, des mots de passe faibles ou d'autres problèmes de cyber hygiène.

Des milliers de nouvelles vulnérabilités sont découvertes chaque mois. Aux États-Unis, deux agences gouvernementales tiennent des catalogues consultables des failles de sécurité connues : le National Institute of Standards and Technologies (NIST) et la Cybersecurity and Infrastructure Security Agency (CISA).

4.1.3 Pourquoi l'analyse des vulnérabilités est importante ?

Malheureusement, même si les vulnérabilités sont soigneusement documentées une fois découvertes, ce sont souvent les pirates informatiques et autres acteurs malveillants qui les trouvent en premier, leur permettant de prendre les organisations par surprise.

Pour adopter une posture de sécurité plus proactive face à ces cybermenaces, les équipes informatiques mettent en œuvre des programmes de gestion des vulnérabilités. Ces programmes s'appuient sur un processus continu qui permet d'identifier et de résoudre les risques de sécurité avant que les pirates ne puissent les exploiter. L'analyse des vulnérabilités constitue généralement la première étape du processus de gestion des vulnérabilités : son objectif étant de découvrir les failles de sécurité que les équipes chargées de l'informatique et de la sécurité devront résoudre.

Voici comment de nombreuses équipes de sécurité utilisent également l'analyse des vulnérabilités :

- **Validation des mesures et des contrôles de sécurité :** après avoir mis en place de nouveaux contrôles, les équipes effectuent souvent une autre analyse. Il s'agit de confirmer que les vulnérabilités identifiées ont été corrigées et que les efforts de résolution n'ont pas introduit de nouveau problème. Pour maintenir la conformité réglementaire : certaines réglementations exigent explicitement l'exécution d'une analyse des vulnérabilités. Par exemple, la norme PCI-DSS (Payment Card Industry Data Security Standard) exige que les organisations qui gèrent les données des titulaires de carte fassent des analyses trimestrielles .

4.1.4 Types d'analyses des vulnérabilités:

Les équipes de sécurité peuvent exécuter différents types d'analyses en fonction de leurs besoins. Les techniques utilisées pour l'analyse des vulnérabilités peuvent être actives ou passives :

- **Analyse active**, également appelée analyse non accréditée, consiste à envoyer des attaques, des requêtes ou des requêtes simulées à la cible pour identifier les vulnérabilités potentielles, telles que les débordements de mémoire tampon, les données non chiffrées et les processus d'authentification défaillants.
- **Analyse passif**, également appelée analyse des informations d'identification, consiste à analyser discrètement le trafic réseau pour détecter les vulnérabilités que les attaquants peuvent exploiter pour diffuser des logiciels malveillants ou voler/manipuler des données.

L'analyse des vulnérabilités peut également être classée en différents cas d'utilisation, tels que :

- **Analyse des vulnérabilités du réseau :** Analyse du réseau à la recherche de vulnérabilités, notamment les ports ouverts, les logiciels non corrigés et les protocoles réseau faibles.
- **Analyse des vulnérabilités des applications Web :** Recherche les failles de sécurité telles que l'injection SQL, le cross-site scripting (XSS) et d'autres vulnérabilités propres aux applications Web.
- **Analyse des vulnérabilités des bases de données :** Se concentre sur l'identification des vulnérabilités dans les bases de données, telles que les erreurs de configuration, l'authentification faible et les autorisations trop permissives.
- **Analyse des vulnérabilités de l'hôte :** Effectué sur des hôtes individuels (serveurs ou postes de travail) pour identifier les vulnérabilités au niveau du système d'exploitation ou des logiciels installés.
- **Analyse des conteneurs et des environnements virtualisés :** Conçu pour identifier les vulnérabilités dans les applications conteneurisées et les environnements virtuels. Cela inclut l'analyse des vulnérabilités dans les images de conteneur et la gestion des conteneurs et des machines virtuelles.

4.1.5 Comment fonctionne l'analyse des vulnérabilités ?

- **Étape 1 : Définition de la portée**

Avant l'analyse, vous devez déterminer les réseaux et les applications cibles, cartographier les points de terminaison et identifier les dépendances. La définition de la portée consiste également à déterminer si les périphériques internes, les systèmes externes ou une combinaison des deux doivent être analysés.

- **Étape 2 : Sélection de l'outil**

Vous devez choisir une solution, parmi le pool d'outils commerciaux et open source disponibles, qui correspond à votre organisation exigeante en matière de sécurité. La solution doit également disposer d'une console conviviale pour faciliter l'analyse des vulnérabilités et fonctionner de manière optimale sur les réseaux hybrides distribués afin de faciliter l'identification des risques dans tous vos environnements.

- **Étape 3 : Configuration**

L'outil de numérisation doit être configuré pour numériser en fonction des paramètres souhaités. Les détails de configuration peuvent inclure la spécification d'adresses IP ou de noms de domaine cibles, la définition de l'intensité ou de la vitesse d'analyse et la définition des techniques d'analyse.

- **Étape 4 : Lancement de l'analyse**

Lancez le processus à l'aide de commandes ou à l'aide des options fournies par l'outil de votre choix, tel qu'une interface graphique. Certaines ressources vous permettront de programmer vos analyses, ce qui rend cette étape automatique une fois que vous avez sélectionné vos préférences.

- **Étape 5 : Détection des vulnérabilités**

Les analyseurs recherchent les types de vulnérabilités courants ou comparent la surface d'attaque du système avec les paramètres enregistrés dans la base de données des vulnérabilités utilisée. Les vulnérabilités analysées correspondent généralement à la spécialité du scanner, qu'il s'agisse de bases de données, de réseaux, etc

- **Étape 6 : Correction et nouvelle analyse**

Sur la base des résultats de l'analyse, votre équipe de sécurité résoudra les vulnérabilités identifiées en déployant des correctifs de sécurité, en mettant à jour les versions logicielles ou en configurant les paramètres de sécurité, en fonction des recommandations du rapport de vulnérabilité.

Après la correction, une nouvelle analyse des systèmes cibles doit être effectuée pour vérifier que les vulnérabilités ont été résolues avec succès

- **Étape 7 : Surveillance continue**

De nouvelles vulnérabilités peuvent toujours faire surface. L'analyse des vulnérabilités doit être planifiée à intervalles réguliers afin d'identifier et de traiter rapidement les menaces émergentes.

4.2 Sécurité des dépendances:

4.2.1 Mettre à jour régulièrement Symfony et ses dépendances

- Les mises à jour contiennent souvent des correctifs de sécurité. Il est donc crucial de : Mettre à jour Symfony et ses bundles en exécutant la commande composer update.
- Surveiller les avis de sécurité publiés sur le site officiel de Symfony.
- Utiliser security-checker de Symfony pour détecter les vulnérabilités.

4.2.2 Protéger les fichiers de configuration sensibles:

Les fichiers de configuration, notamment .env, peuvent contenir des informations sensibles (clés API, identifiants de base de données, etc.). Voici quelques bonnes pratiques :

- Ne jamais commettre ces fichiers dans Git.
- Utiliser les variables d'environnement au lieu de stocker des données sensibles en dur.
- Configurer les permissions des fichiers pour limiter l'accès.

4.2.3 Activer l'authentification et la gestion des droits:

Symfony offre un puissant système d'authentification et d'autorisation grâce à Symfony Security. Voici quelques recommandations :

- Utiliser le composant security-bundle pour gérer l'accès aux routes.
- Stocker les mots de passe de manière sécurisée en utilisant bcrypt ou Argon2.
- Restreindre l'accès aux zones sensibles avec des rôles (ROLE_USER, ROLE_ADMIN).

4.2.4 Protéger contre les attaques XSS et CSRF Attaques XSS (Cross-Site Scripting) et Attaques CSRF (Cross-Site Request Forgery)

Symfony intègre Twig, qui échappe automatiquement les variables pour éviter l'injection malveillante de scripts.

- Toujours utiliser {{ variable }} pour afficher du contenu côté frontend.
- Si vous devez autoriser du HTML, utilisez explicitement { autoescape false }.
- Activer la protection CSRF pour tous les formulaires avec csrf_protection: true.
- Utiliser des jetons CSRF pour protéger les actions sensibles

4.2.5 Sécuriser les entêtes HTTP:

Les en-têtes HTTP permettent de limiter les attaques courantes :

- Utiliser nelmio/cors-bundle pour contrôler les requêtes CORS.
- Configurer Content Security Policy (CSP) pour empêcher l'exécution de scripts malveillants.
- Cacher les informations sur le serveur en désactivant X-Powered-By: PHP.

4.3 Sécurité des configurations:

Le composant de sécurité de Symfony est configuré via security.yaml, définissant les fournisseurs d'authentification, les pare-feu et les règles d'accès. Éléments clés :

- Les pare-feu isolent la logique de sécurité pour différentes sections d'application.
- Le contrôle d'accès applique des restrictions d'URL basées sur les rôles.
- Les hacheurs de mots de passe garantissent un stockage sécurisé des informations d'identification

4.4 Sécurité des contrôleurs et routes:

Afin d'assurer la sécurité des contrôleurs et des routes dans une application Symfony, plusieurs bonnes pratiques doivent être mises en œuvre :

4.4.1 Le pare-feu : " Firewall "

- La firewalls section Pare-feu config/packages/security.yaml est la section la plus importante. Un pare-feu est votre système d'authentification : il définit les parties de votre application qui sont sécurisées et comment vos utilisateurs pourront s'authentifier (par exemple, formulaire de connexion, jeton API, etc.). Un seul pare-feu est actif à chaque requête : Symfony utilise la patternclé pour trouver la première correspondance. Toutes les URL réelles sont gérées par le mainpare-feu. Un pare-feu peut proposer plusieurs modes d'authentification ; autrement dit, il permet de poser la question **Qui êtes-vous ?** de plusieurs manières.
- Consulter une URL sous un pare-feu ne nécessite pas nécessairement d'être authentifié (par exemple, le formulaire de connexion doit être accessible ou certaines parties de l'application doivent être publiques). En revanche, toutes les pages que vous souhaitez informer de la présence d'un utilisateur connecté doivent être sous le même pare-feu. Ainsi, si nous souhaitons afficher le message "**Vous êtes connecté en tant que...**" sur chaque page, elles doivent toutes être incluses dans le même pare-feu.

4.4.2 Authentification des utilisateurs:

Lors de l'authentification, le système tente de trouver un utilisateur correspondant au visiteur de la page web. Traditionnellement, cela se faisait via un formulaire de connexion ou une boîte de dialogue HTTP de base dans le navigateur. Cependant, le SecurityBundle inclut de nombreux autres authenticateur:

- **Formulaire de connexion** : La plupart des sites web disposent d'un formulaire de connexion permettant aux utilisateurs de s'authentifier à l'aide d'un identifiant (par exemple, une adresse e-mail ou un nom d'utilisateur) et d'un mot de passe. Cette fonctionnalité est fournie par le **FormLoginAuthenticator** intégré .
- **Connexion JSON** : Certaines applications proposent une **API sécurisée** par des jetons. Elles peuvent utiliser un point de terminaison qui fournit ces jetons en fonction d'un nom d'utilisateur (ou d'une adresse e-mail) et d'un mot de passe. L'authenticateur de connexion JSON
- **HTTP de base** : L'authentification HTTP Basic est un framework d'authentification HTTP standardisé. Elle demande des informations d'identification (nom d'utilisateur et mot de passe) via une boîte de dialogue dans le navigateur, et l'authenticateur HTTP Basic de Symfony les vérifie.
- **Lien de connexion** : Les liens de connexion constituent un mécanisme d'authentification sans mot de passe. L'utilisateur recevra un lien temporaire (par exemple par e-mail) qui l'authentifiera sur le site web.

- **Certificats clients X.509** : Lorsque nous utilisons des certificats clients, notre serveur web effectue lui-même l'authentification. **L'authentificateur X.509** fourni par Symfony extrait l'adresse e-mail du « nom distinctif » (DN) du certificat client. Il utilise ensuite cette adresse e-mail comme identifiant utilisateur dans le fournisseur d'utilisateurs.
- **Utilisateurs distants** : Outre l'authentification par certificat client, d'autres modules de serveur web pré-authentifient un utilisateur (par exemple, Kerberos). L'authentificateur d'utilisateur distant offre une intégration de base pour ces services. Ces modules exposent souvent l'utilisateur authentifié dans la REMOTE_USER variable d'environnement. L'authentificateur d'utilisateur distant utilise cette valeur comme identifiant pour charger l'utilisateur correspondant.
- **Limitation des tentatives de connexion** : Symfony offre une protection de base contre les attaques de connexion par force brute grâce au composant **Rate Limiter**. Symfony utilise le composant Rate Limiter qui, par défaut, utilise le cache de Symfony pour stocker les tentatives de connexion précédentes.

4.5 Sécurité des formulaires et des entrées utilisateur:

Les formulaires web sont devenus indispensables pour les entreprises. Ils servent d'outils vitaux pour collecter des informations, qu'il s'agisse de retours clients, de détails pour la génération de leads, ou de données financières sensibles. Cependant, la commodité de collecter des données en ligne vient avec le risque inhérent d'exposer ces données à des acteurs malveillants. L'accès non autorisé, les violations de données et les cyberattaques peuvent tous découler de vulnérabilités dans la sécurité des formulaires web.

Considérez, par exemple, l'Injection SQL, une attaque répandue où les hackers insèrent du code SQL malveillant à travers les entrées de formulaire pour exploiter votre base de données. Une autre menace courante est le cross-site scripting (XSS), où des scripts malveillants sont injectés dans des sites web de confiance pour voler des données ou détourner des sessions d'utilisateurs.

Le compromis des données de formulaire ne s'arrête pas là ; même des informations apparemment inoffensives comme les adresses email et les numéros de téléphone peuvent être utilisées pour des escroqueries de phishing et le vol d'identité. Dans un autre cas, l'exposition des données des employés peut conduire à de graves violations de la vie privée, causant une méfiance au sein de votre personnel.

Outre les répercussions financières et personnelles, les violations de données entraînent souvent des ramifications légales, y compris des amendes importantes et des pénalités réglementaires, surtout avec des lois strictes comme le **RGPD**.

Étant donné ces risques, l'emploi des meilleures fonctionnalités de sécurité pour les formulaires web est non négociable.

Comment vérifier si votre formulaire web est sécurisé :

1. Validation et assainissement des entrées :

La validation des entrées est l'une des mesures de sécurité fondamentales pour la protection des formulaires web. En validant les entrées des utilisateurs, vous vous assurez que seules les données correctement formatées entrent dans votre système. Cela est crucial pour se défendre contre des attaques telles que l'injection SQL, le cross-site scripting (XSS) et plus encore. La validation des entrées vérifie que les données répondent à certains critères avant de les traiter, réduisant ainsi le risque d'exécution de code malveillant sur votre serveur.

Des mécanismes de validation d'entrée robustes garantissent que chaque champ d'entrée est examiné pour son format, sa longueur et les caractères acceptables, ce qui peut atténuer le risque d'accès non autorisé et de manipulation des données.

En revanche, l'assainissement est le processus de nettoyage et de filtrage des entrées des utilisateurs pour éliminer tout code ou caractère potentiellement dangereux. Alors que la validation assure que les entrées répondent aux critères attendus, l'assainissement va plus loin en éliminant tout contenu indésirable. Cette approche double fait partie des meilleures pratiques pour la sécurité des formulaires web, réduisant considérablement les chances de réussite des attaques par injection.

Une assainissement efficace implique le codage des sorties, l'utilisation de fonctions intégrées pour nettoyer les données d'entrée et la mise en œuvre de techniques de liste blanche pour spécifier les valeurs autorisées. Mettre régulièrement à jour vos méthodes d'assainissement selon les dernières tendances de sécurité est vital pour maintenir une sécurité robuste des formulaires en ligne.

2. Systèmes CAPTCHA :

Les systèmes CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) sont un outil essentiel dans l'arsenal des outils de sécurité des formulaires en ligne. Ils aident à distinguer les utilisateurs humains des bots automatisés, prévenant ainsi le spam et les attaques automatisées sur vos formulaires web. L'implémentation de systèmes CAPTCHA réduit efficacement le risque d'attaques pilotées par des bots, protégeant votre site web des abus et des violations potentielles.

Plusieurs types de systèmes CAPTCHA sont disponibles, offrant chacun différents niveaux de sécurité et d'expérience utilisateur. Les types les plus courants incluent les CAPTCHAs basés sur le texte, les CAPTCHAs basés sur des images, et les CAPTCHAs audio. En exigeant des utilisateurs qu'ils complètent ces tests avant la soumission d'un formulaire, vous pouvez considérablement renforcer la sécurité de vos formulaires web.

Bien que les systèmes CAPTCHA soient des outils puissants, ils doivent être implantés de manière réfléchie pour équilibrer sécurité et expérience utilisateur. Des CAPTCHAs trop compliqués peuvent frustrer les utilisateurs légitimes, conduisant à une mauvaise expérience utilisateur et à des abandons potentiels. Il est donc crucial de choisir un CAPTCHA à la fois sécurisé et convivial.

Lors de la mise en place de CAPTCHA pour la sécurité des formulaires Web, envisagez d'utiliser des systèmes CAPTCHA modernes et adaptatifs qui ajustent la difficulté en fonction du comportement de l'utilisateur. Le reCAPTCHA de Google, par exemple, est largement utilisé car il offre une expérience fluide pour la plupart des utilisateurs tout en bloquant efficacement les activités suspectes. Assurez-vous toujours que vos solutions CAPTCHA sont à jour et configurées pour faire face aux menaces de sécurité actuelles.

3. Transmission de données sécurisée :

Chiffrer les données transmises entre l'utilisateur et le serveur garantit que les informations sensibles, telles que les identifiants de connexion et les détails personnels, sont protégées contre l'interception pendant le transit. Cette pratique joue un rôle significatif dans la protection du contenu sensible contre les attaques de l'homme du milieu et autres cybermenaces.

L'utilisation de protocoles comme HTTPS (HyperText Transfer Protocol Secure) est une meilleure pratique standard pour la sécurité des formulaires Web. HTTPS utilise le chiffrement SSL/TLS pour protéger les données, offrant une couche supplémentaire de sécurité. En garantissant que toutes les données transmises via vos formulaires Web sont chiffrées, vous pouvez empêcher les parties non autorisées d'accéder ou de manipuler ces données, préservant ainsi la confidentialité et l'intégrité des informations.

Transitionner votre site Web vers HTTPS est une étape cruciale pour améliorer la sécurité des formulaires Web. Pour mettre en œuvre HTTPS, vous devez obtenir un certificat SSL/TLS auprès d'une autorité de certification de confiance. Une fois installé sur votre serveur, ce certificat chiffre toutes les données échangées entre le navigateur de l'utilisateur et votre site Web, sécurisant ainsi les formulaires en ligne contre les menaces potentielles.

Mettre régulièrement à jour vos certificats SSL/TLS et utiliser les protocoles cryptographiques les plus récents sont des meilleures pratiques pour la sécurité des formulaires Web. De plus, configurer votre serveur pour utiliser des suites de chiffrement fortes et activer des fonctionnalités comme la Sécurité de Transport Strict HTTP (HSTS) peut renforcer davantage votre défense contre les menaces en ligne.

4. Jetons Anti-CSRF :

La falsification de requête inter-sites (CSRF) est une exploitation malveillante où des commandes non autorisées sont transmises à partir d'un utilisateur que l'application Web fait confiance. Ces attaques peuvent compromettre la sécurité des formulaires en ligne en trompant les utilisateurs pour qu'ils effectuent des actions indésirables, telles que changer les détails du compte ou effectuer des transactions sans leur connaissance. Prévenir les attaques CSRF est vital pour protéger les formulaires Web et les informations personnelles identifiables et les informations médicales protégées qu'ils collectent.

Ces jetons fonctionnent en générant des valeurs uniques et imprévisibles pour chaque session ou soumission de formulaire, garantissant que toute tentative de forger une requête échouera. Cette couche supplémentaire de sécurité aide à vérifier l'authenticité des actions des utilisateurs, empêchant ainsi les attaques CSRF de réussir.

Pour mettre en œuvre efficacement les jetons anti-CSRF, incluez un champ de saisie caché dans vos formulaires Web qui contient la valeur unique du jeton. Ce jeton doit être généré pour chaque session ou instance de formulaire et doit être validé par le serveur lors de la soumission du formulaire. Si le jeton est manquant ou invalide, le serveur doit rejeter la demande, déjouant ainsi les attaques CSRF potentielles.

Utiliser des cadres et des bibliothèques qui fournissent un support intégré pour les jetons anti-CSRF peut simplifier la mise en œuvre. Par exemple, de nombreux cadres de développement Web modernes offrent des intergiciels ou des fonctions d'aide pour générer et valider automatiquement les jetons CSRF. Mettre régulièrement à jour ces cadres et bibliothèques, ainsi que réaliser des revues de sécurité, sont des meilleures pratiques pour la sécurité des formulaires Web, garantissant que vos défenses restent robustes face aux menaces évolutives.

5. Contrôles d'accès :

Les contrôles d'accès sont cruciaux pour définir qui peut accéder et modifier vos formulaires Web. Le contrôle d'accès basé sur les rôles (RBAC) est une stratégie puissante qui restreint l'accès en fonction des rôles des utilisateurs au sein de l'organisation. En attribuant des permissions spécifiques à différents rôles, vous pouvez garantir que seules les personnes autorisées ont l'accès nécessaire pour modifier les configurations de formulaire ou voir des données sensibles.

Le RBAC améliore la sécurité des formulaires Web en limitant la surface d'attaque et en minimisant le risque de menaces internes. Il garantit que les utilisateurs n'ont accès qu'aux fonctionnalités requises pour leur rôle, réduisant le potentiel d'accès non autorisé et de violations de données. Réviser et mettre à jour régulièrement les rôles et les permissions des utilisateurs est essentiel pour maintenir des formulaires en ligne sécurisés.

En plus du RBAC, la mise en œuvre de contrôles d'accès granulaires permet une gestion plus précise des permissions des utilisateurs. Cette approche implique de définir des politiques d'accès à un niveau granulaire, spécifiant qui peut voir, éditer ou supprimer des champs de formulaire individuels ou des soumissions. Les contrôles d'accès granulaires offrent une plus grande flexibilité et sécurité, garantissant que les informations sensibles ne sont accessibles qu'à ceux ayant le dégagement approprié.

L'adoption d'une combinaison de contrôles d'accès basés sur les rôles (RBAC) et de contrôles d'accès granulaires est recommandée pour une sécurité complète des formulaires Web. L'utilisation de cadres de contrôle d'accès et l'audit régulier des journaux d'accès garantissent que les politiques d'accès sont appliquées efficacement et aident à identifier toute tentative d'accès non autorisé, permettant une réponse rapide aux incidents de sécurité potentiels.

4.6 Sécurité API et communication avec d'autres services:

4.6.1 Qu'est-ce que la sécurité des API ?

La sécurité des API protège les interfaces de programmation applicative (API) contre toute utilisation malveillante ou tentative d'exploitation visant à perturber leur fonctionnement ou à dérober des données sensibles. Il s'agit d'un ensemble de stratégies, de techniques et de solutions permettant de s'assurer que seuls les utilisateurs autorisés peuvent accéder à l'API et que les données transmises via cette interface sont sécurisées de façon à empêcher toute manipulation ou tout accès prohibés.

Les API permettent aux systèmes et aux services d’interagir avec le back-end. Il est donc essentiel d’assurer leur sécurité afin de protéger les données sensibles qui y transitent, telles que les informations d'accès (données d'authentification, d'autorisation, de validation, de chiffrement, etc.). La sécurité des API englobe donc les méthodes et les outils conçus pour défendre ce framework back-end et neutraliser les menaces liées aux violations d'accès, aux attaques de bots ou aux tentatives de détournement.

4.6.2 Principaux types d'attaques d'API

1. DoS (déni de service)
2. DDoS (déni de service distribué)
3. MITM (Man-in-the-Middle)
4. Violation du contrôle d'accès
5. Injection

Une attaque d'API peut entraîner des pertes massives de données, des vols d'informations privées ou personnelles ainsi que des interruptions de service.

4.6.3 Qu'est-ce qu'une API ?

Une API, ou interface de programmation applicative (Application Programming Interface en anglais), est un ensemble de définitions et de protocoles facilitant la communication des composants logiciels. En tant qu'intermédiaire entre les différents systèmes, l'API permet aux applications ou aux services de partager des données et des fonctionnalités. Mais l'API ne se contente pas de fournir une infrastructure de connectivité. Elle régit également la manière dont les applications sont autorisées à communiquer et à interagir entre elles, en définissant notamment le type de requêtes qui peuvent être échangées, la manière dont ces requêtes doivent être effectuées et les formats de données autorisés.

Les API permettent ainsi aux entreprises de partager des données avec leurs clients ou d'autres utilisateurs externes. Parmi les applications les plus courantes, on peut par exemple citer le traitement des paiements, les visioconférences, les réseaux sociaux, les services de messagerie, la domotique et le suivi de santé ou des performances physiques. Ouvertes ou fermées, publiques ou privées, la plupart des architectures d'API suivent des standards tels que REST, SOAP ou GraphQL.

En leur donnant accès aux fonctionnalités d'autres applications, les API profitent aussi aux développeurs. Elles leur évitent en effet d'avoir à continuellement créer de nouvelles infrastructures de connectivité ou à maîtriser toutes les subtilités de l'architecture et du code sous-jacent.

4.6.4 Pourquoi la sécurité des API est-elle si importante ?

Les applications sont omniprésentes. Elles font désormais partie intégrante de la vie des entreprises et de la société dans son ensemble. Or, derrière presque toutes les applications se trouvent des API. Il est donc essentiel de les protéger ces interfaces efficacement.

Les API offrent un framework back-end à la plupart des applications cloud-native, dont les applications mobiles, web et SaaS, mais aussi aux applications internes ou destinées aux partenaires et aux clients. Leur utilisation est de ce fait extrêmement répandue. Pour donner une idée de l'ampleur du phénomène, la plateforme de gestion d'API Postman a, par exemple, enregistré 1,13 milliard d'appels d'API en 2022. Mais cet essor éveille aussi l'intérêt des acteurs malveillants.

Les API exposent en effet la logique des applications, les ressources et les données sensibles (y compris les données à caractère personnel). Elles sont donc devenues la cible privilégiée des attaquants, qui peuvent désormais profiter de cette porte d'entrée pour perturber le fonctionnement de l'entreprise et accéder à des informations sensibles dans le but de les exfiltrer ou de les détruire.

4.6.5 Risques liés à la sécurité des API

Les erreurs de configuration, les failles logiques et les vulnérabilités des API mettent les applications et les données en danger. Dans ce contexte, certaines entreprises pensent s'offrir une sécurité complète en se dotant d'une passerelle API, qui se borne pourtant à surveiller le trafic acheminé via la passerelle et n'offre aucune visibilité sur le trafic interne.

Pourtant, les équipes de sécurité ont besoin d'une visibilité totale sur leur surface API, car on ne peut protéger que ce que l'on voit. Or, un acteur malveillant peut se cacher derrière chaque angle mort et chaque API fantôme.

Bien que les passerelles surveillent efficacement les API et leur utilisation, elles ne sont pas en mesure de détecter et de bloquer les attaques. La sécurité des API doit donc conjuguer visibilité, gestion des risques et protection en temps réel contre les acteurs malveillants.

5 Test d'intrusion:

L'analyse de vulnérabilité est un processus utilisé pour identifier et évaluer les faiblesses dans un système, un réseau ou une application afin de prévenir les potentielles attaques. Il existe deux principaux types d'analyse de vulnérabilité : l'analyse automatique et l'analyse manuelle. OWASP ZAP est un outil très puissant qui permet d'effectuer tout les deux :

5.1 Analyse automatique:

Cet analyse s'effectuer en utilisant des outils dédiés afin de détecter des faiblesses et des vulnérabilités dans un système d'information même si on a pas des connaissances sur les vulnérabilités détectées.

Pour effectuer ce teste automatique sur notre application, on nous allons procéder comme suit:

- Lancer l'outil de test d'intrusion (pentest) OWASP ZAP, une fenêtre pour choisir si on veut enregistrer la session ou non:

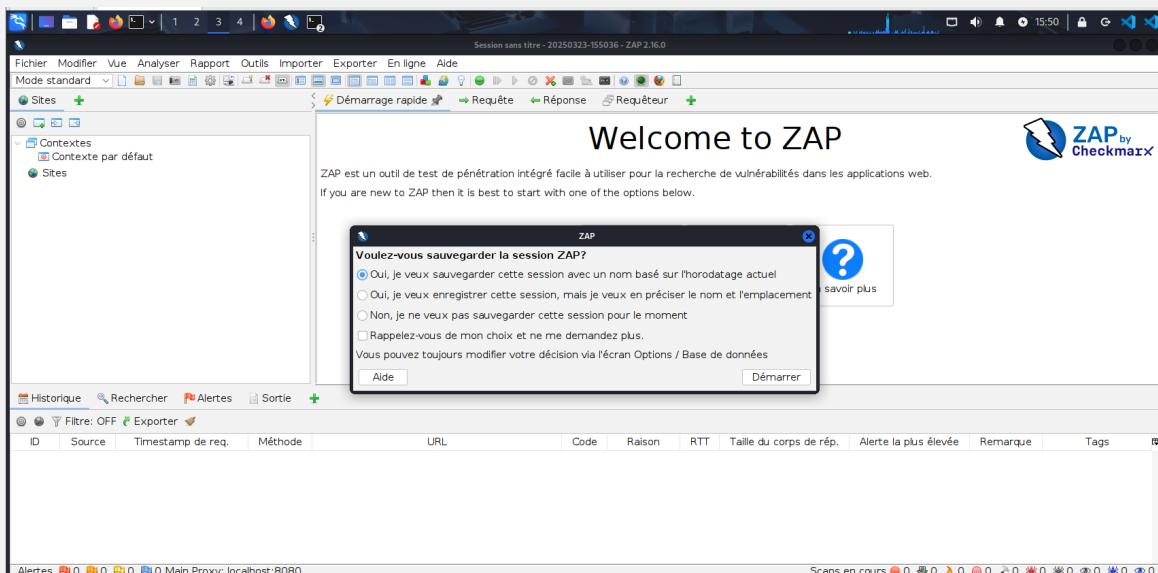


Figure 27: L'interface graphique de OWASP ZAP

- Puis on peut choisir entre les deux types de tests, soit le test manuel ou le test automatique;

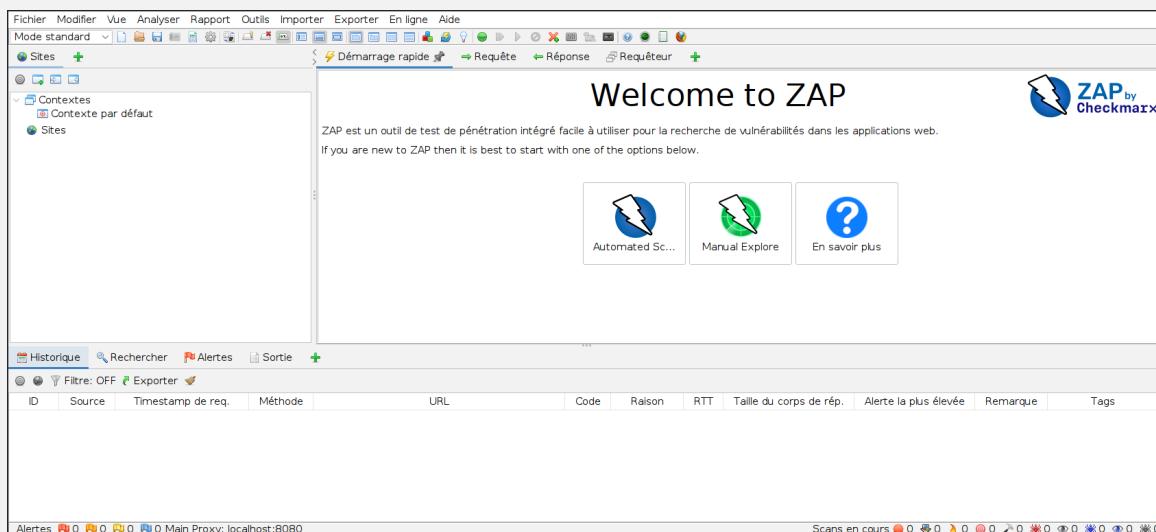


Figure 28: L'interface graphique OWASP ZAP

- Après avoir choisi l'analyse automatique, on doit saisir le URL de la cible à auditer, puis cliquer sur le bouton attaquer:

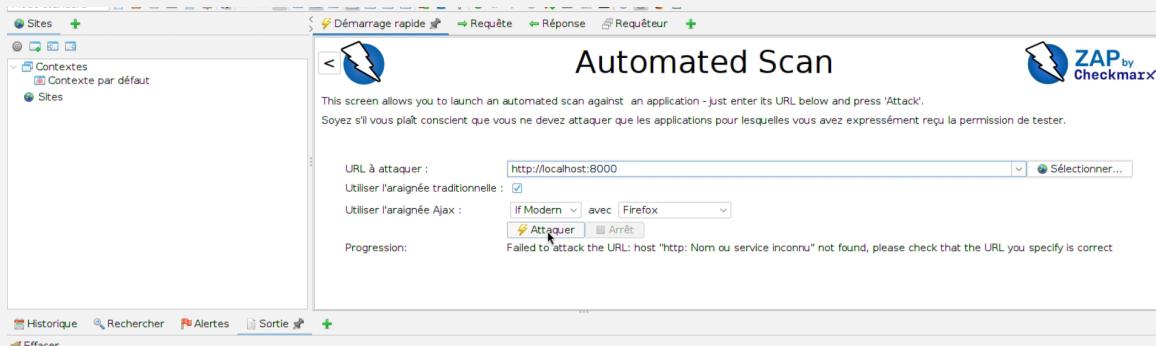


Figure 29: Analyse automatique

- On attend que le test se termine:

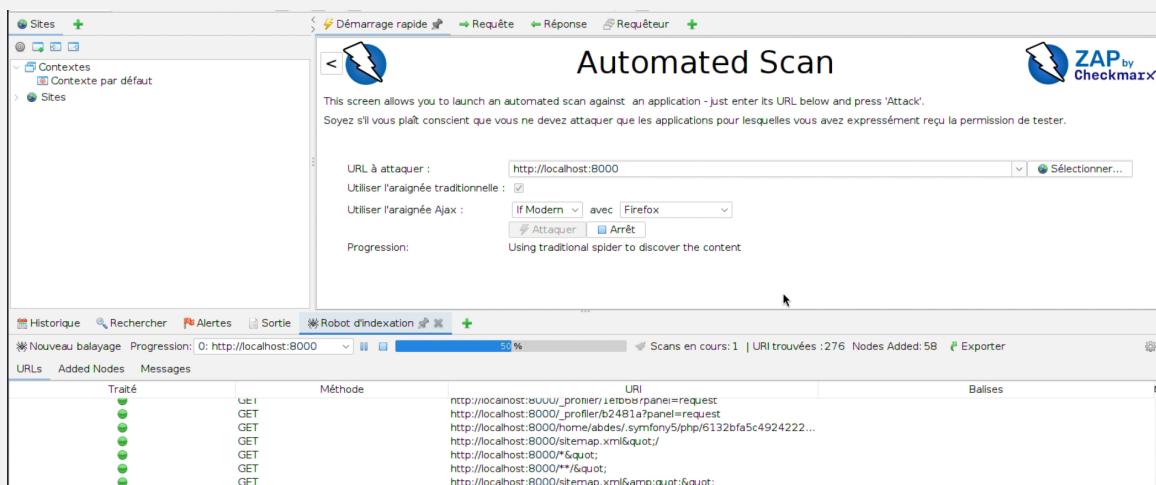


Figure 30: Test en cours

- Après on doit examiner les alertes générées afin d'évaluer la sécurité de notre application:

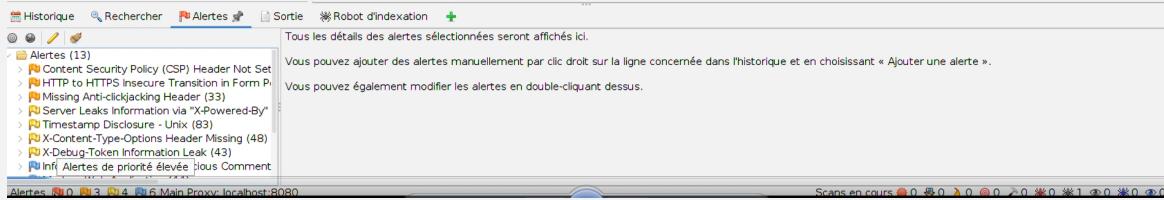


Figure 31: Alertes de sécurité

La figure ci-dessus montre qu'on a trouvé 13 alertes; 3 d'une gravité ou d'un risque moyen, 2 d'un risque faible et 6 alertes d'informations.

- OWASP ZAP affiche le niveau de gravité, une petite description et certaines informations pour chaque alerte:



Figure 32: La première alerte

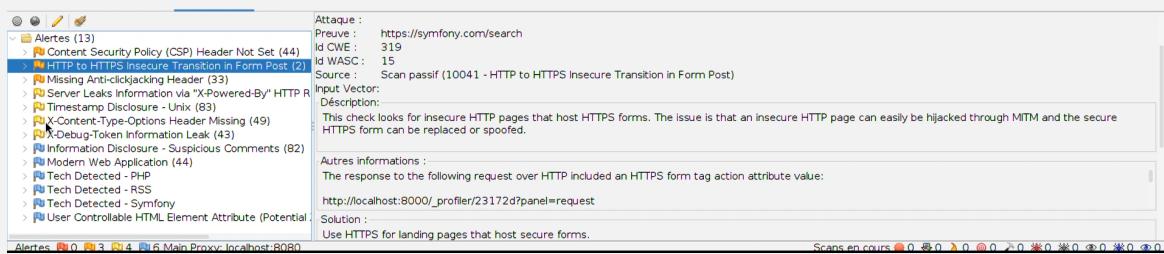


Figure 33: Deuxième alerte

- Après avoir effectué le test, on peut générer un rapport pour cet analyse



Figure 34: générer le rapport

- Le rapport a été généré avec succès:



Figure 35: Rapport de l'analyse

5.2 Analyse manuel:

Cet analyse est plus sofistiqué, il implique qu'un expert en sécurité avec des connaissances solides effectue des testes et des évaluation en profondeur du système, soit en s'appuyant sur certains outil pour simplifier la tâche ou en le faisant entièrement manuellement, une petite vidéo qui montre cette audit est intégrer dans la présentation.

6 Recommandations et Corrections:

6.1 Signification des alertes générées:

- Content Security Policy (CSP) Header Not Set

Signification : La Content Security Policy (CSP) est une couche de sécurité supplémentaire qui aide à détecter et atténuer certains types d'attaques, y compris les attaques de type Cross Site Scripting (XSS) et les attaques d'injection de données. L'absence de l'en-tête CSP signifie que le navigateur n'a aucune instruction sur les sources de contenu autorisées à être chargées par la page web. Cela rend l'application vulnérable aux attaques XSS, où un attaquant peut injecter du code malveillant qui sera exécuté par le navigateur de l'utilisateur.

- Missing Anti-clickjacking Header (2)

Signification : Le "clickjacking" est une technique où un attaquant trompe un utilisateur en lui faisant cliquer sur quelque chose de différent de ce qu'il perçoit visuellement. L'en-tête "X-Frame-Options" ou "Content-Security-Policy frame-ancestors" est utilisé pour empêcher l'inclusion de la page web dans un cadre (frame) ou un iframe, ce qui est une technique couramment utilisée dans les attaques de clickjacking. L'absence de cet en-tête rend l'application vulnérable aux attaques de clickjacking.

6.2 Recommandations:

Pour se remédier à «Content Security Policy (CSP) Header Not Set»: Implémenter une politique CSP stricte en définissant une politique CSP qui autorise uniquement les sources de contenu nécessaires à l'application. Par exemple, autoriser uniquement les scripts provenant de son propre domaine et bloquer tous les scripts provenant de domaines externes.

Utiliser l'en-tête "Content-Security-Policy" : Ajouter cet en-tête à toutes les réponses HTTP de l'application.

7 Conclusion:

Dans ce rapport, nous avons abordé les vulnérabilités courantes identifiées lors d'un audit de sécurité. Symfony offre, par défaut, une protection efficace contre de nombreuses vulnérabilités. Lorsque la protection n'est pas intégrée, le framework met à disposition des outils pour l'implémenter.

En termes de sécurité en PHP, nous recommandons fortement l'utilisation de frameworks reconnus comme Symfony. L'utilisation de tels frameworks améliore significativement la sécurité par rapport aux applications développées sans framework, grâce aux mécanismes intégrés et aux bonnes pratiques qu'ils imposent.

8 Disclaimer :

Ce rapport a bénéficié de l'utilisation de l'intelligence artificielle pour la correction, la reformulation et l'amélioration de la clarté du texte. L'IA a été utilisée dans le but d'optimiser la présentation et la rédaction, tout en préservant l'intégrité du contenu initial.

9 Ressources:

1. Ce rapport a été réalisé en utilisant *LATEX*, <https://fr.overleaf.com/>
2. L'une des ressources principales de ce projet est le site officiel de *Symfony*, <https://symfony.com/>
3. La console.dev comme ressource pour la partie sur l'architecture d'un projet *Symfony*, <https://laconsole.dev/formations/symfony/architecture-projet#:~:text=Symfony%20est%20un%20framework%20bas%C3%A9,fichiers%20de%20code%20s'ex%C3%A9cutent.>
4. Le site officiel de ZIWIT; une entreprise spécialisée dans la sécurité informatique, offrant des services de conseil et des solutions dans le domaine de la cybersécurité. Elle est particulièrement connue pour ses outils d'audit de sécurité et ses services de pentesting. <https://www.ziwit.com/fr/audit-applications-entreprise>
5. Pour le modèle *MVC*, consultez le site suivant : <https://www.geeksforgeeks.org/mvc-framework-introduction/>
6. Le site officiel d'*OWASP ZAP* : <https://www.zaproxy.org/docs/>
7. Le site officiel de *Docker* : <https://docs.docker.com/get-started/docker-overview/>
8. IBM - Vulnerability Scanning
9. Wiz - Vulnerability Scanning
10. Drosalys - Sécuriser une application *Symfony*
11. StudyRaid - Security Component Configuration
12. Dyma - *Symfony*: Le Framework PHP
13. *Symfony* - Demo Repository
14. Top 5 des fonctionnalités de sécurité pour les formulaires web
15. Sécurisation de l'authentification, des sessions et du contrôle d'accès
16. Qu'est-ce que la sécurité des API ?
17. Dynamic Application Security Testing (DAST)
18. [securite-des-applications-web](#)