# **Troisième Jalon - Presque fin de projet**

Projet *Let’s chat* accessible sur mon compte github sur le lien suivant :

<https://github.com/touggani/cybersecurite_projet>

## **Objectif du jalon**

* Vous avez fait un projet qui contient des vulnérabilités :
  + Pour chaque vulnérabilité, vous la décrivez :
    - Vous décrivez en quoi votre code est vulnérable ;
    - Vous définissez le(s) mécanisme(s) ET le(s) principe(s) de sécurité que vous auriez dû appliquer pour éviter cette vulnérabilité
    - Vous indiquez la bibliothèque et sa ou ses fonctions permettant d'appliquer le mécanisme de sécurité en question
    - Décrivez les tests que vous feriez pour vérifier que cet endpoint (cette page web) n'est pas vulnérable (Les tests vont vous permettre de renforcer l'implémentation des principes de dev sécurisés)
    - ***(En bonus)*** Quel impact aurait eu cette vulnérabilité sur votre métier (Injection SQL => Fuite de données => Incident diplomatique => vous fermez votre business)

## **Les vulnérabilités**

### Configuration des routes

### L’une des failles qui nous saute aux yeux lorsque nous nous baladons souvent et attentivement sur l’application est la non-réécriture des url. Cette erreur amenée à une faille bien plus importante, l’accès à tous les fichiers par tous les utilisateurs. En effet, un utilisateur qui a identifié les fichiers utilisés sur le serveur, peut y accéder directement à travers la barre de recherche et aucune restriction ne lui sera faite.

### Pour cela nous aurions pu créer un router. En effet, le router en php est possible et permet de réécrire les url mais aussi de seulement donner accès aux pages aux utilisateurs autorisés.

### Fonctionnement de cette méthode :

### Pour régler cette faille il suffit de configurer un fichier php où nous allons y configurer notre classe route. Cette classe route contiendra plusieurs fonctions permettant d’accepter ou non un utilisateur à une page. Voici un exemple complet et bien expliqué sur la notion de route en php : <https://grafikart.fr/tutoriels/router-628>

### Test à réaliser :

### Pour tester cette vulnérabilité, il nous suffit d’essayer d’accéder aux autres pas de l’application à travers la barre de recherche. Dans notre application, nous pouvons essayer d’accéder au chat sans s’être identifié au préalable.

### Impacte de la faille :

### L’utilisateur peut accéder à des fichiers confidentiels, aux comptes d’autres utilisateurs et bien plus.

### Filtrage des entrées utilisateurs

### L’application comporte beaucoup d’entrée utilisateurs, des champs pour la connexion, pour l’inscription ou encore pour l’envoi de messages. Certains d’entre eux ne sont pas vérifier et peuvent comporter du code ou autre amenant à une injection sql par exemple.

### Pour cela nous aurions pu créer des fichiers annexes permettant à la vérification et la validité de ces entrées utilisateurs.

### Fonctionnement de cette méthode :

### La vérification a été faite sur quelque champs du site. En effet, nous avons des fichiers annexes permettant seulement à vérifier la validité des données. Ces fichiers se trouve dans le répertoire privé du projet (enregistrement\_utilisateur.php et connexion\_utilisateur.php). Pour la vérification et selon les champs il est possible d’utiliser différent moyen :

* Vérification des adresses mail : filter\_var($var, FILTER\_VALIDATE\_EMAIL)
* Vérification d’un integer : filter\_var('0755', FILTER\_VALIDATE\_INT)
* Utilisation du regex : preg\_match('/^[A-Za-z][A-Za-z0-9]{4,31}$/' , $var)

En utilisant cela, nous pouvons accepter ou non où bien formater les données.

### Test à réaliser :

Si les données sont bien travaillées, nous pourrons accéder à aucune des données de base de données ou lancer une commande JavaScript. Pour tester la faille, nous pouvons, dans les champs textes, lancer des commandes JavaScript où bien des commandes sql. Par la suite identifier si cela a un impact sur l’application.

### Impacte de la faille :

Cette faille peut être dangereuse. En effet, elle peut mener à une faille sql. Cela amener à être une menace sur les données utilisateurs et les base de données. L’attaquant peut réaliser n’importe quelle manipulation tels que l’ajout, la suppression ou la modification des tables/données. Cette faille peut être très dangereuse lorsque les données sont sensible tels que les informations bancaires et de sécurité sociale par exemple. Selon chaque type de données les procédures et les règles sont différentes.