Rapport d’audit de l’application « Shop »

Table des matières

[Introduction 2](#_Toc119153288)

[Authentification 3](#_Toc119153289)

[Description 3](#_Toc119153290)

[Éléments audités 3](#_Toc119153291)

[Https : 3](#_Toc119153292)

[Login : 4](#_Toc119153293)

[Hash avec sel : 4](#_Toc119153294)

[Restriction : 4](#_Toc119153295)

[Corrections à apporter 4](#_Toc119153296)

[Https : 4](#_Toc119153297)

[Login : 5](#_Toc119153298)

[Hash avec sel : 6](#_Toc119153299)

[Restriction : 7](#_Toc119153300)

[XSS/CSRF 7](#_Toc119153301)

[Description 7](#_Toc119153302)

[Éléments audités 7](#_Toc119153303)

[Protection des inputs et outputs 7](#_Toc119153304)

[Protection des formulaires 9](#_Toc119153305)

[Corrections à apporter 9](#_Toc119153306)

[Protection des inputs et outputs 9](#_Toc119153307)

[Protection des formulaires 11](#_Toc119153308)

[Injection SQL 12](#_Toc119153309)

[Description 12](#_Toc119153310)

[Éléments audités 12](#_Toc119153311)

[Rechercher un élément 12](#_Toc119153312)

[Sécurisation des variables 12](#_Toc119153313)

[Sécurisation URL 13](#_Toc119153314)

[Corrections à apporter 13](#_Toc119153315)

[Rechercher un élément 13](#_Toc119153316)

[Sécurisation des variables 14](#_Toc119153317)

[Sécurisation URL 15](#_Toc119153318)

[Protection des données 15](#_Toc119153319)

[Description 15](#_Toc119153320)

[Éléments audités 15](#_Toc119153321)

[Droit et protection des images 15](#_Toc119153322)

[Formulaire de contact et captcha 15](#_Toc119153323)

[Adresse mail 16](#_Toc119153324)

[Corrections à apporter 16](#_Toc119153325)

[Droit et protection des images 16](#_Toc119153326)

[Formulaire de contact et captcha 17](#_Toc119153327)

[Adresse mail 17](#_Toc119153328)

[Outils 17](#_Toc119153329)

[Vega 17](#_Toc119153330)

[High 18](#_Toc119153331)

[Medium 18](#_Toc119153332)

[Low 18](#_Toc119153333)

[DDOS 18](#_Toc119153334)

[DDOS 18](#_Toc119153335)

[Description 18](#_Toc119153336)

[Éléments audités 19](#_Toc119153337)

[Solution 19](#_Toc119153338)

[Spam 19](#_Toc119153339)

[Description 19](#_Toc119153340)

[Solution 19](#_Toc119153341)

# Introduction

Ce document regroupe les informations sur les menaces web examinées du site "Shop" en les décrivant et en mettant en exergue les éléments du code y faisant référence. Si besoin, les étapes nécessaires à la correction ont été décrites précisément.

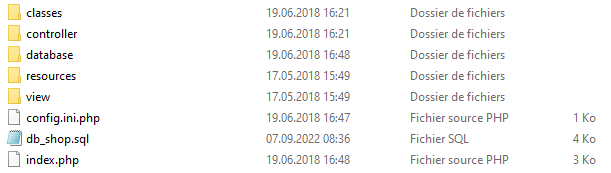
# Authentification

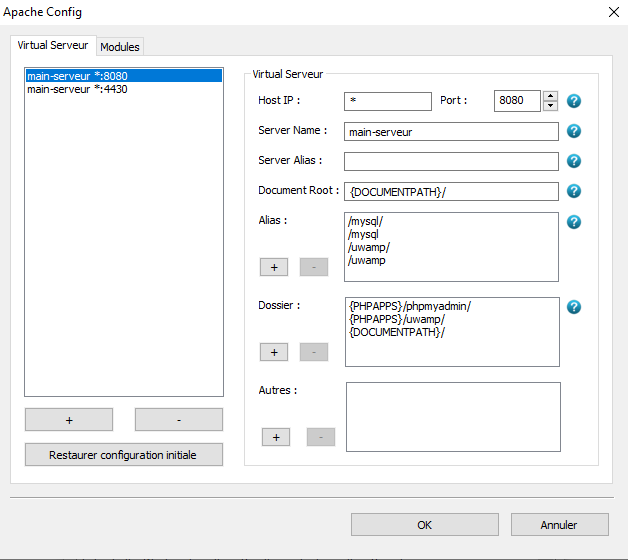
## Description

L’authentification est un processus permettant d’identifier une personne sur un réseau et protéger ses données personnelles. Il est donc important de sécuriser les données lors de la création d’un profil.

## Éléments audités

Https :

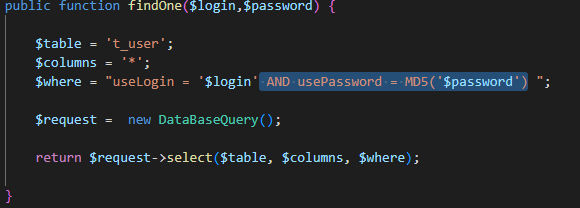
Le site est actuellement en http. Et donc, par conséquent, pas protégé. Si vous avez une ancienne version de Uwamp, il y de grandes chances que les ports d’Apache ne soient pas ceux de base.   




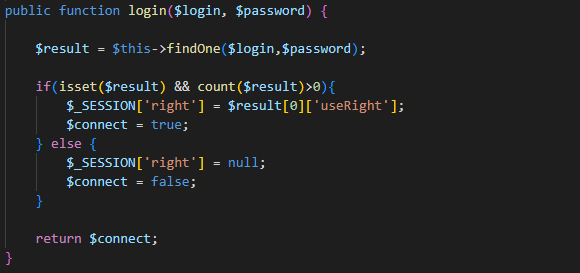
Login :

Les login Admin et Utilisateur n’ont pas un mot de passe sécurisé, il faudrait les changer.

Hash avec sel :

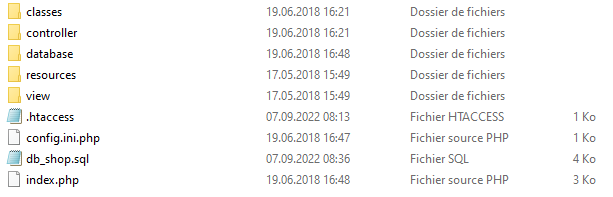
Les mots de passe qui sont stockés en base de données sont encryptés en MD5 et ne sont pas hashé avec un sel.  


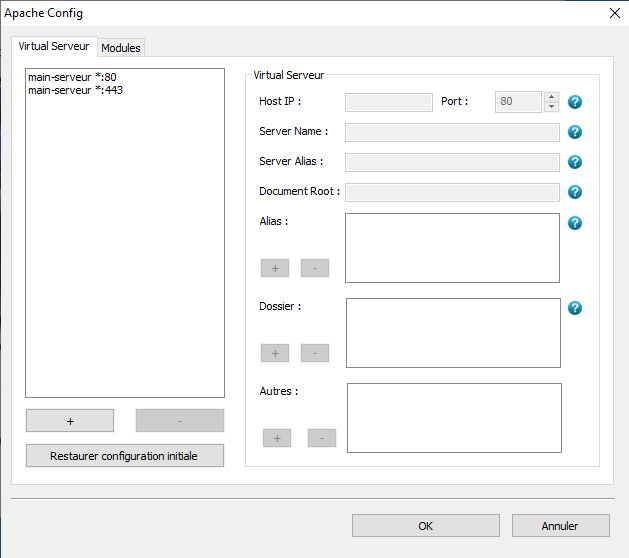
Restriction :

Il n’y a aucune restriction sécuritaire pour les essais de connexion 

## Corrections à apporter

### Https :

Pour rediriger une requête http nous devons créer un fichier .htaccess qui va permettre une redirection automatique en https. A la racine du site, il faut ajouter un fichier nommé « .htaccess ». Ce fichier contient les éléments suivants :   
Une image contenant texte

Description générée automatiquement  
Si les ports sont différents, il faut les modifier. 

Login :

Les mots de passe ne sont actuellement pas sécurisés, il faudrait les changer par un mot de passe plus long et plus sécurisé. Pour avoir un mot de passe sûr et sécurisé, nous vous conseillons la norme suivante :

* Longueur du mot de passe minimum : 16 caractères
* Utiliser des majuscules et minuscules
* Utiliser les chiffres
* Utiliser des caractères spéciaux tel que : \*&@#

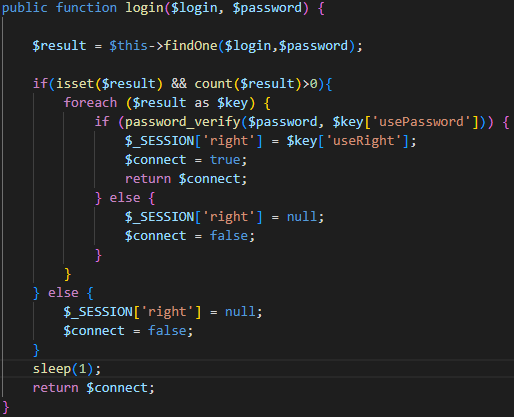
Admin : .Admin- -> j@1m3LeSC@Rr07tesV3rtEs  
Utilisateur : .User- -> M0nCh@73stM0R7¢e50iR

Hash avec sel :

Pour stocker un mot de passe sécurisé en base de données, on peut utiliser la fonction PHP de base, qui hash et ajoute du sel automatiquement :  
   
lors de la connexion de l’utilisateur sur son compte, il va falloir comparer le hash obtenu par l’entrée utilisateur et le hash stocké en base de données. On change le code pour qu’à chaque connexion, il vérifie le mot de passe :  
Une image contenant texte

Description générée automatiquement

### Restriction :

Pour la restriction, nous avons ajouté la fonction « sleep() » qui met en pause la lecture du code pendant x secondes. Cette restriction permet de ralentir considérablement les attaques par brute-force.   
  
On peut aussi mettre un compteur dans la variable $\_SESSION et après tant de tentatives de connexion, on empêche l’utilisateur de se connecter pendant un temps imparti et on envoie un mail au possesseur du compte pour lui en informer.

# XSS/CSRF

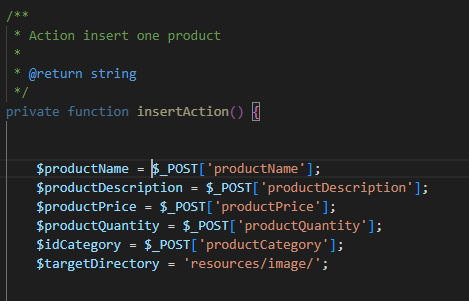
## Description

Le Cross Site Scripting ou XSS représente les attaques par Scripting qui permet de créer un lien entre le site de l’hacker et votre site. Le Cross Site Request Forgery ou CSRF est un type d’attaque qui vise les formulaires non sécurisés. Pour s’en protéger il existe des CSRF token qui vérifie si le formulaire à bien été envoyé par l’utilisateur et non quelqu’un d’autre.

## Éléments audités

### Protection des inputs et outputs

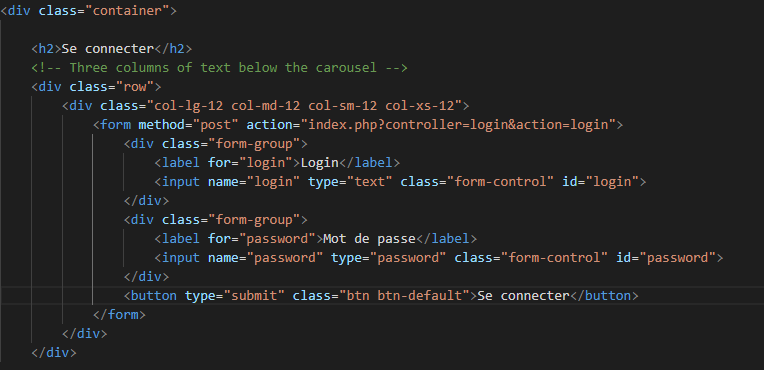
Les inputs et les outputs ne sont pas protéger contre les attaques XSS :

Input :  


Output :  


### Protection des formulaires

Les formulaires ne sont pas protégés des attaques CSRF :



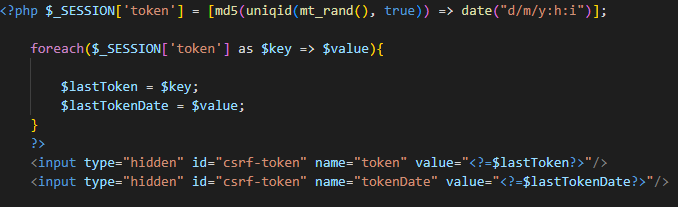
## Corrections à apporter

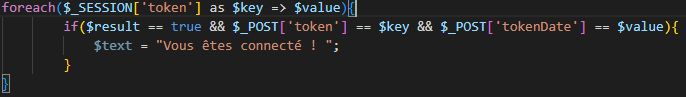
### Protection des inputs et outputs

Input :  
Lorsqu’on récupère les données que nous envoies l’utilisateur, il faut les analyser avec la méthode « htmlspecialchars ». Cette méthode va remplacer les « < », les « > » et les « & » respectivement par des « &lt », des « &gt » et des « &amp » soit leur correspondant en entités HTML. Cela empêche l’insertion de scripte.  


Output :  
Même chose lorsqu’on récupère des données de la base de données. Il faut aussi mettre la méthode « htmlspecialchars ». Cela empêche l’exécution de scripte lors de l’affichage de la donnée.  


### Protection des formulaires

Pour protéger les formulaires des attaques CSRF, le meilleur moyen est d’utiliser le CSRF token. Il permet de vérifier que la personne qui envoie le formulaire, lorsqu’on le récupéré, soit bel et bien l’utilisateur et non une tierce personne.  
Il faut ajouter dans le formulaire :

Et dans le code qui vérifie les inputes :

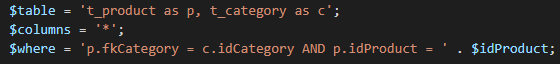
# Injection SQL

## Description

Les injection SQL consiste à mettre des commandes SQL dans les inputs des formulaires pour modifier, accéder, supprimer à une table de la DB. Pour ce faire, les hackers mettent souvent dans les inputs le code suivant : ' OR 1==1 --

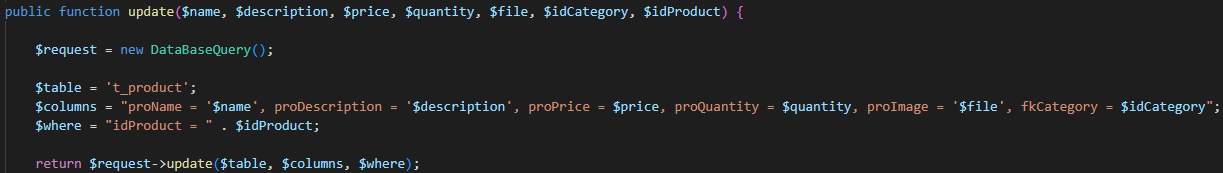
## Éléments audités

### Rechercher un élément

Lorsqu’on recherche un élément précis, il est possible de récupérer plus que 1 élément.  


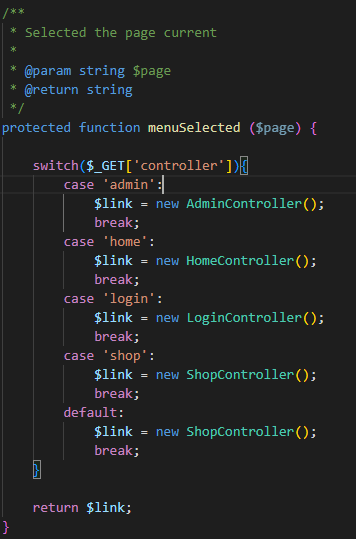
### Sécurisation des variables

Les variables sont présente directement dans la requête SQL.



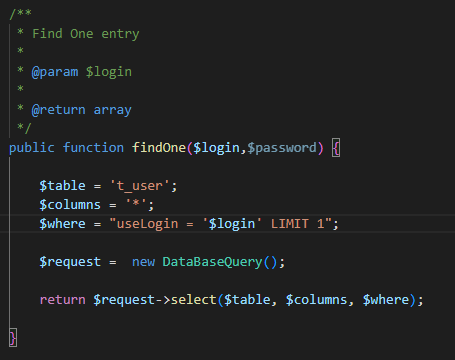
Il est donc possible d’accéder à la DB depuis les formulaires avec ‘OR 1==1 --.  


### Sécurisation URL

Il est possible d’accéder à des pages normalement réserver pour les admins en modifiant le lien URL  
  


## Corrections à apporter

### Rechercher un élément

Quand on cherche un élément précis, c’est une bonne pratique de préciser qu’il faut limiter le retour à 1 seule élément dans la requête. On peut aussi vérifier que je reçois bien un seul enregistrement.  
  


### Sécurisation des variables

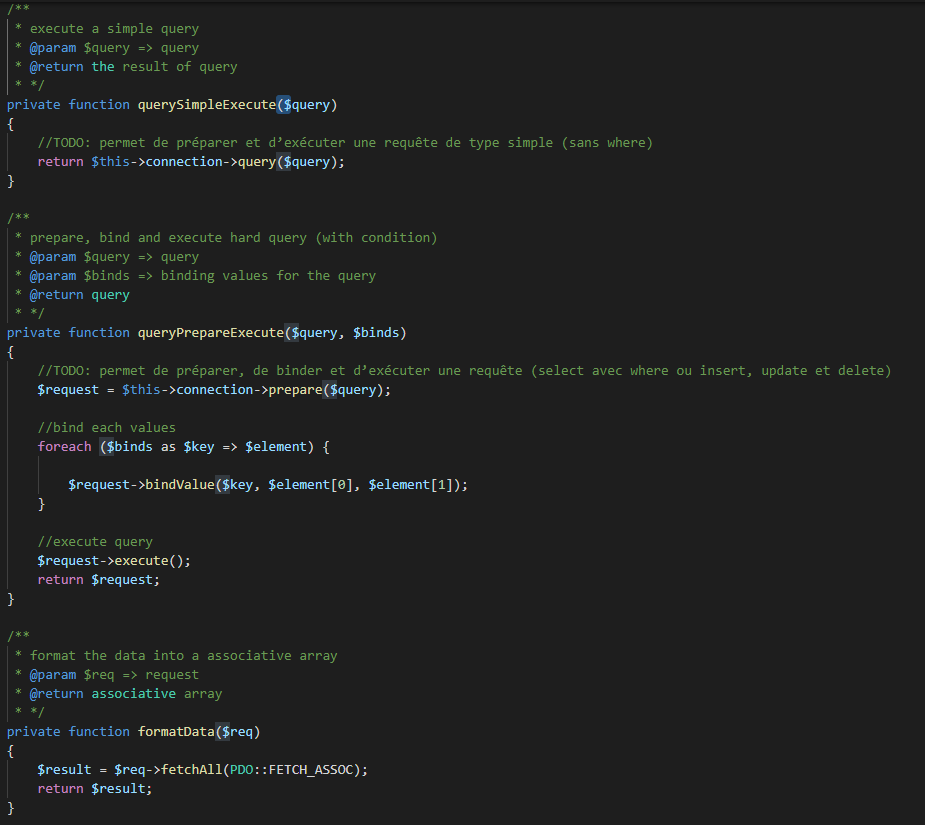
Il y a 2 manières dans sécuriser les variables qui sont introduite dans les requêtes SQL.

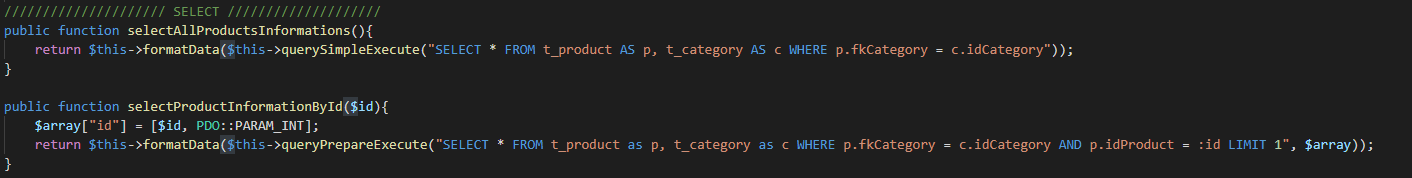
1. Htmlspecialchars

Si on utilise la méthode htmlspecialchars. On peut rajouter le paramètre « ENT\_QUOTES » qui permet de traiter le ‘, ce qui aura pour effet d’empêcher l’utilisation d’injection SQL de type :  
 ‘OR 1==1 --  


Dans notre code fournit. Il s’agit de la manière la plus simple, car il aurait fallu modifier tout le code des Models.

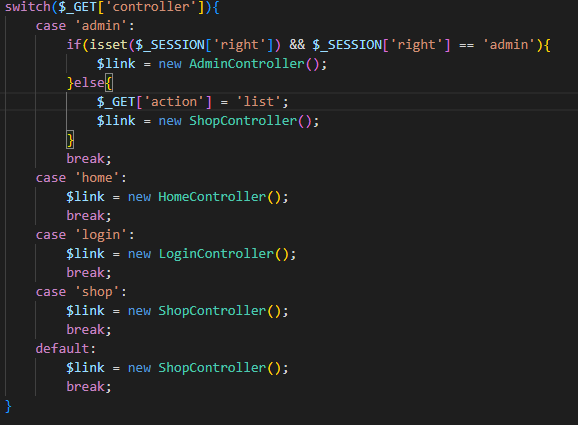
1. Bind value

Le bind permet de sécuriser le contenu des variables et il s’agit de la plus sécurisée des 2 manières.  




### Sécurisation URL

Pour empêcher les utilisateurs d’aller dans des endroits qu’ils n’ont normalement pas accès, il suffit de vérifier s’ils ont bien les droits avant de les laisser y accéder.



# Protection des données

## Description

La protection des données est un élément important pour un site web. En effet, plusieurs éléments devront être protégé tel que les images, les vidéos. Il est aussi important de protéger les adresse mail et les formulaires de contactes contre les spam automatiques.

## Éléments audités

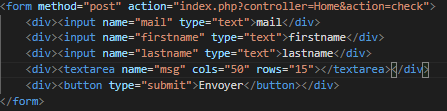
### Droit et protection des images

Les images ne sont pas protégées contre les vols d’images

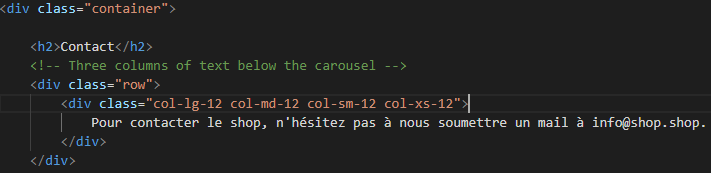


### Formulaire de contact et captcha

Le formulaire n’est pas sécurisé et peut être une entrée pour les attaques de type spams.



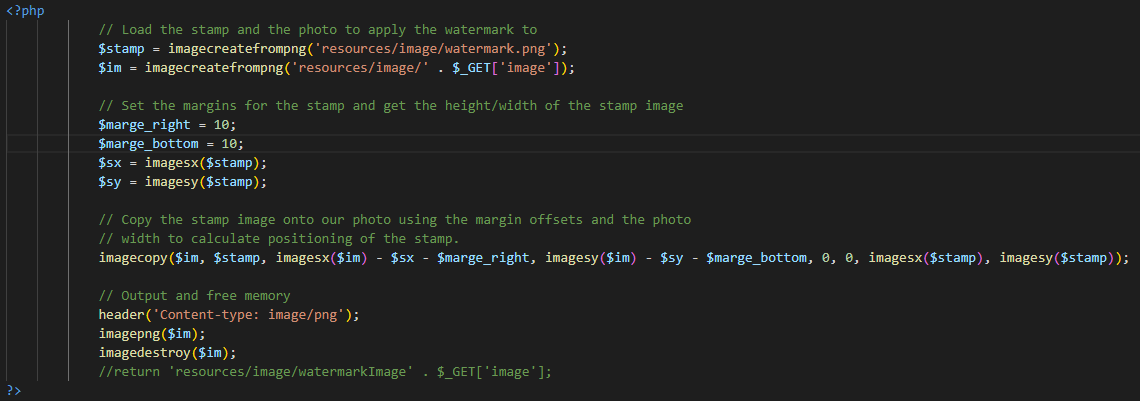
### Adresse mail

L’email du propriétaire du site est en dur en HTML ce qui permet à des robots de la récupérer facilement pour les personnes qui revende ces types d’information à des entreprises pour qu’ils envoient de la pub.  


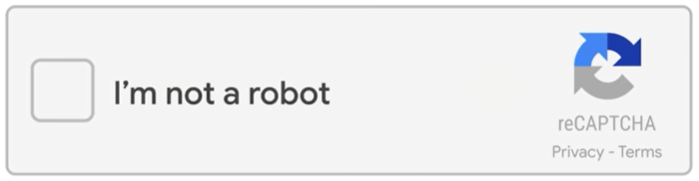
## Corrections à apporter

### Droit et protection des images

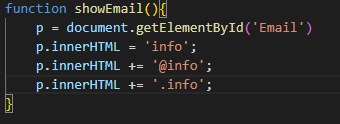
Pour protéger les images d’un site web, il est possible d’ajouter un “Watermark” qui va s’ajouter par-dessus l’images afin de la protéger et éviter le vol d’images. Il est conseillé d’ajouter le Watermark sur l’image avant son affichage ou lors de son insertion en base de données. Pour la seconde option, l’image doit être ensuite enregistrée en local avec le Watermark.

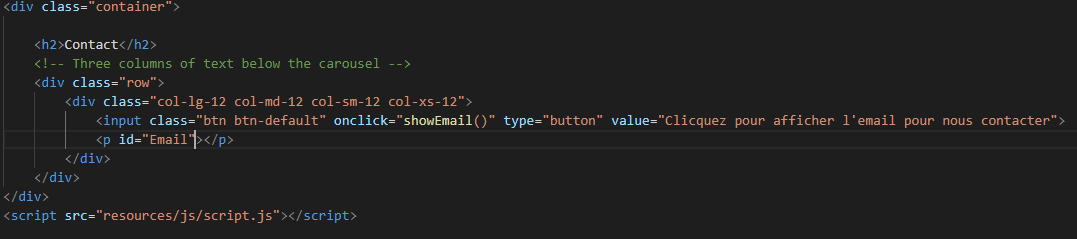


### Formulaire de contact et captcha

Pour éviter que la page contacte ou la page de login soient victime de spam d’un robot, il faut ajouter un captcha. Il existe plusieurs types de captcha mais le plus utilisé est le captcha de google  


### Adresse mail

Pour éviter que les robots ne détectent les emails, il suffit de faire appel au JS en mettant des bouts de l’email ensemble afin de l’afficher.   
  
Mais il faut d’abord que l’utilisateur appuie sur un bouton (oblige une action de l’utilisateur => limite le spam).



# Outils

## Vega

Vega est un scanner web qui permet de vérifier les entrées web (formulaires etc..) et les sorties détecter des failles ou des potentiel danger sur le site. Cela permet de gagner du temps et c’est un peu plus fiable que de tout regarder et tester soit même.

Voici le rapport automatique de Vega de notre site.



### High

Les 2 plus haut risque sont des faux positifs. Ils nous disent juste qu’il faut faire attention à la configuration du serveur Apache.

### Medium

Les trois premières alertes sont des faux positifs par rapport au certificat https qui est fourni par Apache mais ce n’est pas un vrai.

Le dernier attire juste l’attention sur le fait qu’on pourrait accéder aux fichiers personnels de l’ordinateur avec le Controlleur qui appelle d’autre fichier. Or le Controlleur n’appelle que les Models et les Views donc il s’agit d’un faux positif.

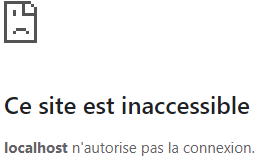
### Low

Les 9 alertes nous disent que l’on peut avoir accès à un index des dossiers. Mais c’est parce que la configuration du serveur Apach est en mode debug. S’il était en mode production, alors on n’aurait pas accès à cet index. Il s’agit donc d’un faux positif.

# DDOS

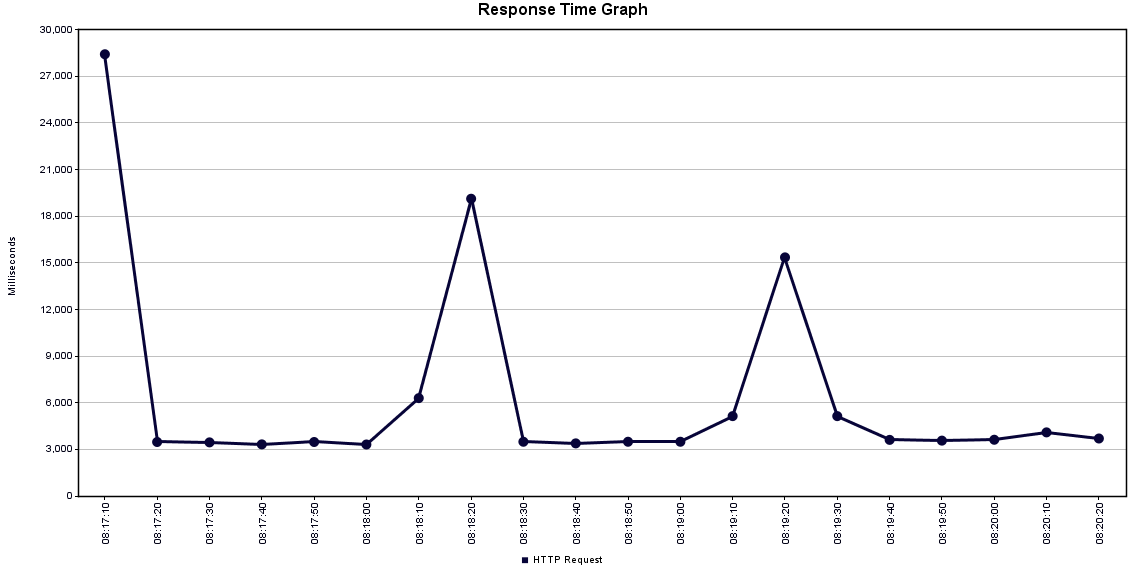
## DDOS

### Description

Les attaques DDOS sont très utilisés dans le domaine du web, que ce soit pour de vrai ou alors pour se préparer à de potentielles futures attaques. La plupart des attaques se finisse de cette manière :  


### Éléments audités

Ce graphique montre le temps en millisecondes que met le serveur pour répondre à un requêtes http. En général, il met environ 3 secondes pour répondre ce qui est énorme et ça avec juste 1000 threads, c’est très peu comparé à de vrai DDOS.



### Solution

Dans ces cas-là, il vaut mieux se protéger en utilisant un service de sécurité internet comme Cloudflare.  
Cette entreprise propose une infrastructure déjà pré-faite ainsi que plusieurs services pour nous protéger et cela à des prix différents. Étant donné que nous voulons juste nous protéger des attaques DDOS et que nous avons un site personnel, la version gratuite est amplement suffisante pour nous satisfaire.

## Spam

### Description

Le spam est un type de DDOS qui est spécifique à l’envoie de mail via un nom de domaine « détourné ». Dans la plupart des messageries, il y a une section spam où sont redirigé tous ces mails non désirables. Le problème est que certains mail (important ou non) sont parfois envoie dans cette section.

### Solution

Il faut mettre en place un SPF. SPF est l’abréviation de « Sender Policy Framework ». Cette méthode permet aux serveurs de messagerie de vérifier si un Email reçu provient réellement du serveur hôte déclaré. Vérification SPF : Le nom de domaine doit être indiqué comme l’adresse « enveloppe de ». Ensuite, le serveur entrant vérifie si l’adresse IP à partir de laquelle l’e-mail est envoyé est autorisée dans l’enregistrement SPF. Le courrier échoue à l’authentification SPF si l’une des vérifications échoue. Pour que la vérification soit correcte, il faudra ajouter des enregistrements SPF afin de créer une sorte de White-liste de nom de domaine autorisés.