Rapport d’audit ICT-183



David Dieperink, Stefan Petrovic – CID3A

CFPV

Table des matières

[1 INtroduction 3](#_Toc116465061)

[2 authentification 3](#_Toc116465062)

[2.1 Description 3](#_Toc116465063)

[2.2 Eléments audités 3](#_Toc116465064)

[2.3 Corrections à apporter 4](#_Toc116465065)

[3 xSS 5](#_Toc116465066)

# INtroduction

Ce document regroupe les différentes failles de sécurités trouvées sur le site d’e-commerce. Il décrit les failles de sécurité, explique leur fonctionnement et décrit différentes corrections mises en place avec leurs avantage et désavantages.

# authentification

## Description

L’authentification permet à un utilisateur de prouver son identité au site. Cela lui permet d’avoir des droits et des accès à des données du site.

## Eléments audités

Voici les fonctions utilisées dans l’application pour l’authentification :

Figure 1



Figure 2

Une image contenant texte

Description générée automatiquement

1. ***Figure 1 :*** Un utilisateur est connecté à l’application quand il est trouvé directement dans la base de données en faisant une condition qui prend en compte le nom d’utilisateur et le mot de passe.

* Il n’y a pas de vérification du mot de passe avec un sel, cela crée des failles. Une Rainbow table peut être utilisé.
* Une vérification de la longueur du mot de passe n’existe pas, donc des **mots de passe par défaut** peuvent être utilisés (ex : admin, root).

1. ***Figure 2*** : La fonction utilisée pour effectuer la requête des informations de l’utilisateur dans la base de données fait un hachage avec le **MD5** mais ne prend en compte aucun sel.

## Corrections à apporter

* Une vérification plus complète peut être effectuée en vérifiant si l’utilisateur existe en faisant une condition avec son nom d’utilisateur dans la requête SQL (fonction **findOne**) puis en utilisant la fonction PHP **password\_verify** qui vérifie le hachage d’un mot de passe avec un sel. Cela permet de contourner la Rainbow table.

Une image contenant texte

Description générée automatiquement

* Une vérification sur la longueur du mot de passe doit également être implémentée pour éviter les mots de passe par défaut.

Une image contenant texte

Description générée automatiquement

* Pour contourner une attaque de **force brute**, la fonction PHP **sleep()** peut être utilisé pour mettre un délai lors de chaque tentative de connexion.



* Pour forcer HTTPS dans le serveur, un fichier **.htaccess** peut être ajouté dans la racine de l’application avec le contenu suivant :

Une image contenant texte

Description générée automatiquement

# xSS

## Description

XSS veut dire Cross-Site-Scripting. Une attaque XSS consiste à insérer un code malveillant dans un champ de formulaire par exemple. Si un code malveillant est inséré dans un champ de formulaire, le site va exécuter le code.

## Eléments audités

Le site possède plusieurs formulaires. Ces formulaires sont sensibles aux attaques XSS. Lorsque cette ligne (<script>alert(‘XSS’)</script>) est rentrée dans un champ de formulaire et que le formulaire est envoyé le script s’exécutera.

## Corrections à apporter

* Pour éviter que cela se produise, une des solutions est de filtrer les informations reçues lors de l’envoi du formulaire. Si les informations reçues ne sont pas conformes à nos attente, l’envoi du formulaire est refusé.

Une image contenant texte

Description générée automatiquement

# CSRF

## Description

## Eléments audités

## Corrections à apporter