Sécuriser un projet web – Rattrapage Picard

1. Introduction

Pour ce travail, j'ai choisi d'analyser le site picard.fr, qui correspond à l'entreprise Picard mentionnée dans l'énoncé. Ce choix me permet de cibler un exemple concret et réaliste de site web lié à la distribution alimentaire, afin d'identifier des attaques possibles et proposer des bonnes pratiques adaptées pour sécuriser ce type de plateforme.

2. Attaque n°1 : Injection SQL

L'injection SQL est une des failles les plus connues. En gros, elle consiste à insérer du code malicieux dans un formulaire (comme une connexion) pour forcer la base de données à faire des choses qu'elle ne devrait pas.

Exemple concret : un pirate entre `' OR 1=1 --` dans un champ de login. Résultat : la base pense qu'il est un utilisateur valide et le laisse passer.

Ce que ça peut provoquer :

- Connexion sans mot de passe
- Accès aux données utilisateurs
- Suppression ou modification de données sensibles

Comment s'en protéger :

- Utiliser des requêtes préparées (paramétrées)
- Ne jamais faire confiance aux données que l'utilisateur entre
- Utiliser un ORM qui gère ça pour nous (comme Sequelize, Prisma, etc.)

3. Attaque n°2 : Cross-Site Scripting (XSS)

Le XSS, c'est quand quelqu'un arrive à insérer du JavaScript malveillant dans une page web. En général, ça passe par des champs où on laisse l'utilisateur écrire du texte (comme des commentaires ou des pseudos).

Exemple : `<script>alert('Hacked')</script>` ajouté dans un formulaire non sécurisé. Tous ceux qui verront cette page verront le script s'exécuter.

Les dangers :

- Vol de cookies (donc usurpation de session)
- Redirection vers des sites frauduleux
- Injection de contenu indésirable

Pour éviter ça :

- Toujours “échapper” les caractères spéciaux avant d’afficher du contenu utilisateur
- Utiliser des frameworks modernes comme React, qui le font automatiquement
- Mettre en place une politique CSP (Content Security Policy)

4. Attaque n°3 : Cross-Site Request Forgery (CSRF)

Cette attaque consiste à piéger un utilisateur déjà connecté à un site pour qu’il exécute des actions sans le vouloir. Par exemple, changer son mot de passe, valider une commande, etc.

Comment ça marche :
Imaginons que tu sois connecté au site Picard, et que tu cliques sur un lien dans un mail ou un site qui déclenche une requête cachée vers ton compte. Si le site ne vérifie pas bien la provenance, l’action est validée.

Risques :

- Commandes envoyées à ton insu
- Changement d’email ou de mot de passe
- Réalisation d’actions sensibles sans le vouloir

Solutions :

- Mettre en place un token CSRF dans chaque formulaire
- Vérifier l’origine des requêtes (avec les headers Origin ou Referer)
- Bloquer les actions critiques via la méthode GET

5. Conclusion

Même si le site Picard peut sembler “simple”, il peut vite devenir une cible s’il n’est pas sécurisé dès le départ. Il ne faut pas attendre qu’un souci arrive pour se poser la question.

Les attaques comme l’injection SQL, le XSS ou le CSRF sont classiques mais très puissantes si on ne les anticipe pas. Heureusement, il existe des solutions simples à mettre en place, à condition d’avoir les bons réflexes dès la conception.

Sources :

Chaîne YouTube de Grafikart.fr mais surtout ses 3 vidéos suivantes :

- Sécuriser ses applications web : Les injections SQL
- Sécuriser ses applications web : Les failles XSS
- Sécuriser ses applications web : Attaques CSRF