SIMPLON
.CO

Stratégie de Sécurisation de l'application pire2pire.com

Réalisé par :

- Yousra Chbib

Encadré par :

- Cyril MARCQ
- Hachemi
- Claire VAN WYNENDAELE

I. Introduction

introduction

II. Coté Client

1. Page de Connexion

1.1 Sécurisation du nom d'utilisateur

La sécurisation du champ « nom d'utilisateur » est une étape clé dans la protection de vos systèmes. Elle doit garantir que les données saisies sont conformes à un format attendu et ne peuvent pas être utilisées pour mener des attaques telles que l'injection de code ou l'énumération de comptes existants. Voici les axes essentiels à mettre en place :

1.1.1 Validation des entrées avec des expressions régulières (Regex):

Les expressions régulières sont des outils puissants permettant de définir des motifs précis pour contrôler la validité des données saisies par l'utilisateur. Elles servent à :

- **Définir un format précis** : Par exemple, vérifier qu'un nom d'utilisateur ne contient que des lettres, chiffres et certains caractères spéciaux autorisés.
- **Valider des formats spécifiques** :
  - Pour le adresse email : `^[a-zA-Z0-9._%+-]+@gmail\.com$`
  - Pour le numéro de téléphone : `^\+?[1-9]\d{1,14}$`

C*ela permet de s'assurer que l'email et/ou le numéro de téléphone suivent un schéma standard (bien que dans un cas spécifique, vous puissiez restreindre par exemple aux adresses Gmail).*

1.1.2 Filtrage et nettoyage des entrées (Sanitization) :

Même après validation, il est essentiel de nettoyer la saisie afin d'éviter les injections ou d'autres attaques malveillantes :

➢ **Suppression des espaces inutiles** : Pour enlever les espaces en début et fin de chaîne pour éviter des erreurs ou des incohérences dans le stockage.

➢ **Échappement ou refus des caractères dangereux :** I interdire ou échapper des caractères comme « `<, >, ;, --`», ou d'autres symboles susceptibles d'être exploités dans une injection **SQL** ou **XSS**.

1.1.3 Définition des Contraintes de longueur minimale et maximale :

La définition des contraintes de longueur vise à garantir un équilibre entre accessibilité et sécurité. Une exigence de longueur minimale de 6 caractères permet d'éviter les identifiants trop courts, souvent vulnérables aux attaques par dictionnaire. À l'inverse, fixer une longueur maximale,de 50 caractères, prévient les abus et réduit les risques liés aux dépassements de mémoire. Ces contraintes doivent être cohérentes avec la politique de sécurité globale de l'application et s'aligner avec les autres critères de validation et exigences métier.

1.2 La Sécurité des Mots de Passe

La gestion des mots de passe est un élément essentiel de la sécurité informatique. Cette section présente les mesures de sécurité pour assurer une protection efficace contre les attaques courantes, en mettant l'accent sur la longueur, la complexité et la gestion des mots de passe.

1.2.1 Longueur Minimale :

Un mot de passe sécurisé doit comporter au moins **12 caractères** pour résister aux attaques par **force brute**. Les mots de passe courts sont rapidement cassés, notamment par des attaques par dictionnaire et des méthodes de **calcul parallélisées**.

1.2.2 Complexité :

Pour renforcer la sécurité des comptes, il est essentiel que les mots de passe respectent des critères de complexité spécifiques. Lorsqu'un utilisateur tente de définir un mot de passe ne répondant pas à ces exigences, des messages d'avertissement clairs doivent être affichés pour guider l'utilisateur vers la création d'un mot de passe conforme.

Exemple :
L'utilisateur saisit un mot de passe composé d'un seul chiffre.
Messages d'avertissement affichés :

- **Longueur insuffisante**
- **Absence de majuscules**
- **Absence de minuscules**
- **Absence de caractères spéciaux**

Ces messages guident l'utilisateur pour qu'il crée un mot de passe conforme aux critères de sécurité recommandés, renforçant ainsi la protection de son compte.

1.2.3 Interdiction des Informations Personnelles :

Pour renforcer la sécurité des mots de passe, il est essentiel d'éviter l'utilisation d'éléments facilement devinables tels que les noms, prénoms, dates de naissance, noms d'utilisateur ou des mots courants comme "password123". Dans ce cadre, des **vérifications** seront mises en place côté serveur pour comparer les mots de passe saisis avec une liste de mots interdits. Cette approche garantit que les utilisateurs ne peuvent pas définir de mots de passe contenant des informations personnelles ou des termes trop simples, renforçant ainsi la robustesse globale des mots de passe utilisés.

1.2.4 Vérification des Fuites de Données :

Pour renforcer la sécurité des mots de passe, il est essentiel de vérifier si ceux-ci ont été compromis lors de fuites de données précédentes. On a mis en place des vérifications côté serveur qui comparent les mots de passe saisis avec des bases de données de mots compromis, telles que **Have I Been Pwned**.

Exemple d'intégration en Python :

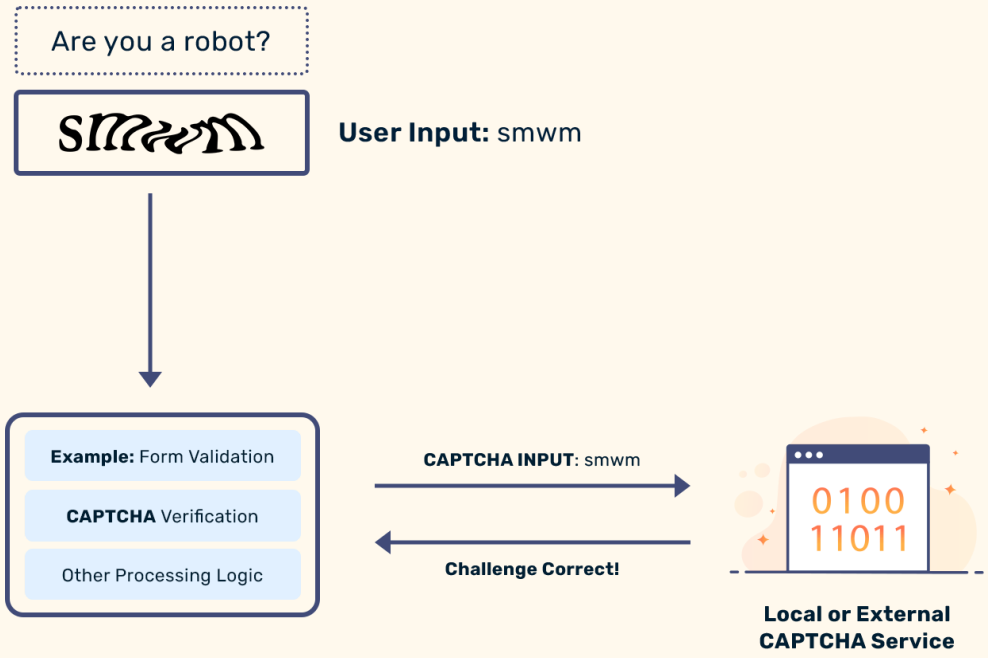
```python
import requests
import hashlib

def check_password(password):
    sha1_hash = hashlib.sha1(password.encode()).hexdigest().upper()
    prefix, suffix = sha1_hash[:5], sha1_hash[5:]
    response = requests.get(f"https://api.pwnedpasswords.com/range/{prefix}")

    return suffix in response.text

password = "monMotDePasseSuperSecret"
if check_password(password):
    print("Ce mot de passe a déjà été compromis !")
else:
    print("Ce mot de passe est sûr.")
```

1.3 Sécurisation du Formulaire d'Authentification

Sécuriser le formulaire d'authentification avant qu'il n'atteigne le serveur est essentiel pour protéger les informations sensibles des utilisateurs. Cette approche vise à prévenir diverses menaces, notamment les attaques par injection et la falsification de requêtes intersites (**CSRF**), qui ciblent souvent les formulaires comme points d'entrée vulnérables.

1.3.1 Implémentation de Systèmes CAPTCHA :

Pour protéger le formulaires contre les soumissions automatisées par des bots malveillants, nous intégrons des **systèmes CAPTCHA**. Ces tests permettent de différencier les utilisateurs humains des programmes automatisés, réduisant ainsi les risques d'abus et d'attaques par force brute.

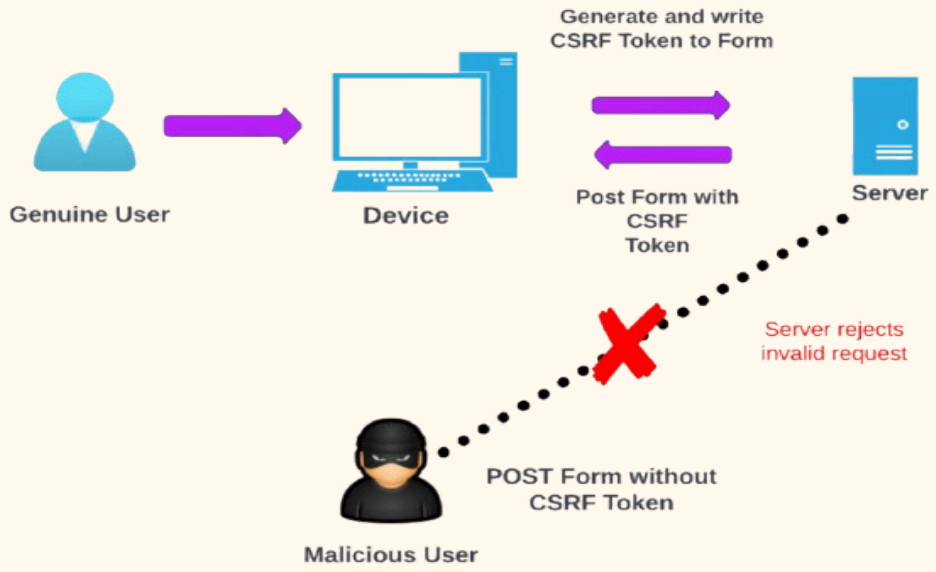


Fonctionnement d'un Test CAPTCHA pour la Protection des Formulaires

1.3.2 Utilisation de Jetons Anti-CSRF :

Une attaque CSRF est une menace sérieuse, qui se produit lorsqu'un utilisateur authentifié d'un site web est amené, à son insu, à exécuter une action non désirée sur ce site. Par exemple, un pirate pourrait envoyer une requête à votre insu, utilisant vos informations d'authentification pour exécuter une action que vous n'avez pas autorisée, comme modifier votre mot de passe ou effectuer un transfert bancaire.

C’est pour cela qu’on va utiliser la méthode de **tokens CSRF**. Un token CSRF est un jeton unique et aléatoire qui est généré par le serveur et inclus dans les formulaires HTML. Lors de la soumission du formulaire, ce token est vérifié par le serveur pour s'assurer qu'il correspond à celui stocké dans la session de l'utilisateur. Si le token ne correspond pas, la requête est rejetée.



1.4 Sécurisation d'Authentification

L’authentification est le processus qui suit la soumission du formulaire de connexion. C’est là où le serveur valide l’identité de l’utilisateur et lui accorde (ou non) l’accès à

l’application. A ce stade, plusieurs menaces peuvent compromettre la sécurité de la plateforme, notamment les attaques par **force brute**, le vol de sessions ou la divulgation d’informations sensibles.

1.4.1 Protection contre les attaques par force brute :

Les attaques par force brute consistent à tester une multitude de combinaisons de mots de passe jusqu'à en trouver un valide. Pour limiter ces tentatives :

- ➔ **Limiter le nombre d'essais** : Ajouter un délai entre les tentatives (ou bloquer un compte après un certain nombre d’échecs)
- ➔ **CAPTCHA après plusieurs échecs :** Empêche les bots d’essayer des milliers de combinaisons automatiquement**.**

1.4.2 Gestion des jetons d’authentification :

Les jetons d’authentification permettent aux utilisateurs de s’authentifier sans stocker leur session côté serveur.

- ➔ **Choix du type de jeton :**

  WT (JSON Web Token) est un jeton d’authentification compact et sécurisé. Les informations nécessaires à l'authentification de l'utilisateur sont contenues directement dans le token lui-même. Ainsi, contrairement aux sessions classiques qui requièrent un stockage côté serveur pour maintenir l'état de l'utilisateur, le JWT dispense en grande partie de ce mécanisme de stockage.