Sécurité web - Formulaire

TD2 :
CREPIN CORENTIN
MAYEL MALEBE
KABEYA JUDE

1. Injection sql

```
sql New York'; DROP TABLE personne; --
```

"Pasted image 20230321225359.png" is not created yet. Click to create.

Résoudre ce problème

Pour se prémunir des injections SQL

- Utilisation des réquetes preparées
  Elles permettent de séparer les données des instructions SQL et de ne pas exécuter directement les données fournies par l'utilisateur

```php
$rs_req = $db->prepare($req);
```

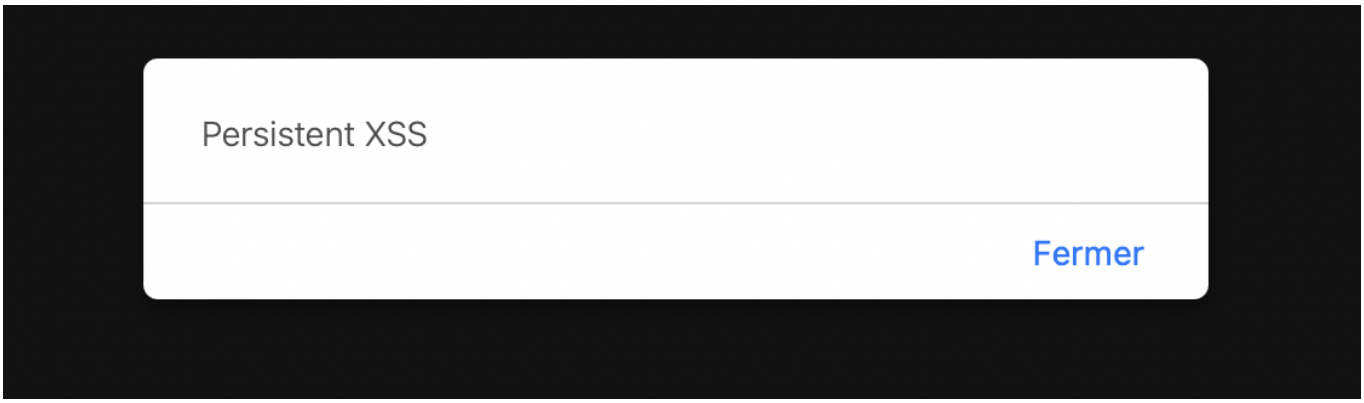
```php
$stmt->bindParam(':ville', $ville);
$stmt->bindParam(':numero', $numero, PDO::PARAM_INT);
```

Cette méthode permet d'associer des valeurs aux paramètres d'une requête SQL, qui peuvent être ensuite exécutés de manière sécurisée en évitant les injections SQL.

2. Démontrer une faille XSS

Le cross-site scripting (abrégé XSS) est un type de faille de sécurité des sites web permettant d'injecter du contenu dans une page, provoquant ainsi des actions sur les navigateurs web visitant la page.

Par exemple, un utilisateur décide d'insérer la ligne de code suivante dans un formulaire



Cette faille est totalement imprévisibles et dangereuses pour l'utilisateur, la personne malveillante peut récuperer des données de l'utilisateur, ou bien emmener l'utilisateur vers un site malveillant.

Résoudre ce problème

1. Le serveur effectue une vérification de la syntaxe, et initialise les ressources internes du serveur pour une utilisation ultérieure.
2. Utilisation de vérification via des méthodes :

- `htmlspecialchars()` : permet de convertir les caractères spéciaux en entités HTML. En échappant la valeur de `$_POST['ville']` avec `htmlspecialchars()`, cela permet de se prémunir contre les attaques de type XSS (Cross-Site Scripting), qui consistent à insérer du code malveillant dans des champs de formulaire ou des requêtes HTTP pour exécuter du code côté client.
- `trim()` : utilisée pour supprimer les espaces blancs en début et en fin de chaîne, afin d'éviter que des caractères non voulus ne soient inclus dans la valeur de la variable `$ville`.

En résumé, le code `$ville = htmlspecialchars(trim($_POST['ville']));` permet de récupérer de manière sécurisée la valeur de la variable `$_POST['ville']` en échappant les caractères spéciaux et en supprimant les espaces blancs.

```
$ville = htmlspecialchars(trim($_POST['ville']));
$numero = trim($_POST['numero']);
$stmt = $db->prepare("INSERT INTO personne (ville, numero) VALUES (:ville,
:numero)");
```

3. Créer un système d’authentification forte via OTP email.

3.1 Le formulaire

Après avoir construit le formulaire de connexion en prenant soin de me protéger contre les injections et les failles de sécurité.

```php
$username = htmlspecialchars(trim($_POST[USERNAME_FIELD]);

$password = htmlspecialchars(trim($_POST[PASSWORD_FIELD]));

$email = htmlspecialchars(trim($_POST[EMAIL_FIELD]));
```

Le formulaire de connexion envoie les informations saisies à la base de données, et pour assurer la confidentialité du mot de passe, celui-ci est crypté à l'aide de l'algorithme Blowfish, tel que présenté dans le cours.

```php
$encrypted_password = password_hash($password, PASSWORD_BCRYPT);
```

```
 username |                          password                          |        email        |  code
----------+------------------------------------------------------------+---------------------+--------
 judekby  | $2y$10$I/whxHWqYe2bwrJfHro2geXiQbcytaCTtO2MDcLlTryitA9vY8yEK | judek9392@gmail.com |
```

3.2 One time password(otp)

Pour envoyer un e-mail, j'utilise le protocole SMTP via Gmail, après avoir effectué quelques configurations sur mon compte Gmail.

Serveur de courrier sortant (SMTP)	smtp.gmail.com SSL requis : oui TLS requis : oui (si disponible) Authentification requise : oui Port pour TLS/STARTTLS : 587

Après avoir importé le package PhpMailer, je génère un mot de passe d'application spécifique pour PhpMailer.

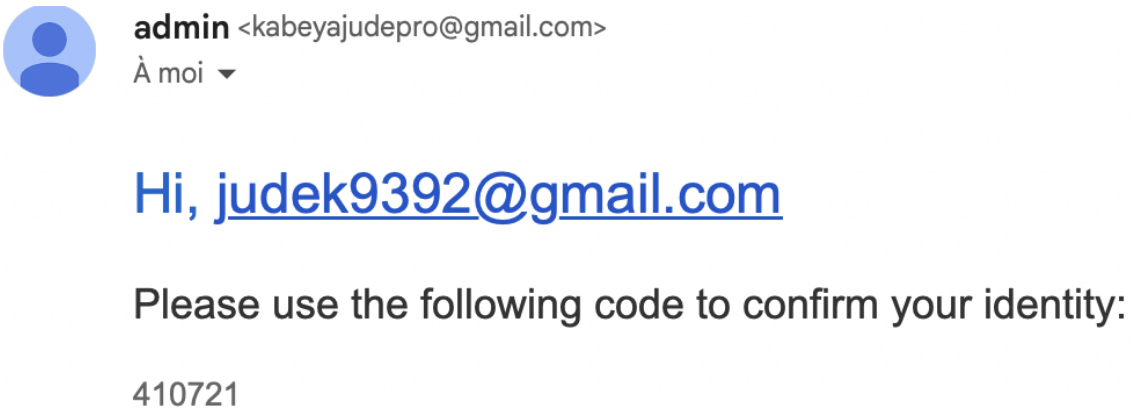
L'intérêt d'utiliser un mot de passe d'application Gmail est d'améliorer la sécurité du compte Gmail. En effet, la connexion à des applications tierces (comme des clients de messagerie ou des applications mobiles), peut présenter un risque de sécurité le compte Gmail, car vous partagez ainsi le mot de passe principale

```php
$mail->SMTPDebug = 0;
$mail->isSMTP();
$mail->Host = 'smtp.gmail.com';
$mail->SMTPAuth = true;
$mail->Username = 'kabeyajudepro@gmail.com';
```

```php
$mail->Password = 'qylsawntlhndycup';
$mail->SMTPSecure = 'ssl';
$mail->Port = 465;
$mail->setFrom('ne-pas-repondre@gmail.com', 'admin');
$mail->addAddress($email);
$mail->isHTML(true);
$mail->Subject = 'Verification code';
$mail->Body = '<html>
```

> Pasted image 20230325181135.png

Génération d'un nombre aléatoire un nombre et celui-ci est envoyé à l'utilisateur pour une double authentification et une stocké en base pour la comparaison



4. Gestion d'un profil

4.1 Controle front des données

- Le Mail

```
<input type="text" class ="form-control" id="email" placeholder="Entrez votre email" name="email" pattern="[^@\s]+@[^@\s]+\.[^@\s]+" title="Invalid email address">
```

Afin de réaliser un controle au près de l'utilisateur, un controle au niveau de l'email à été réalisé avec le respect du formalisme d'un mail. Le pattern vérifie s' il le mail possède au moins un caractère(sauf un autre @ et un espace) avant le @, au moins deux caractères(sauf un autre @ ou un espace) après le @ et un point au millieu.

- *Le mot de passe* :

```
<input type="password" class="form-control" id="password" placeholder="Entrez votre mot de passe" name="password" minlength="8">
```

Le site accepte uniquement un mot de passe avec une longueur minimum de 8 caracteres

- Le nom d'utilisateur

```
required pattern="^[A-Za-z '-]+$" maxlength="15"
```

Le prénom est désormais obligatoire et ne doit comporter que des lettres + éventuellement des espaces, tirets ou apostrophes. Sa taille ne doit pas accéder 15 caractères ;

4.2 Controle back des données

```
$email = filter_var(htmlspecialchars($_POST['email'], FILTER_SANITIZE_EMAIL));
```

Gestion d'un profil utilisateur

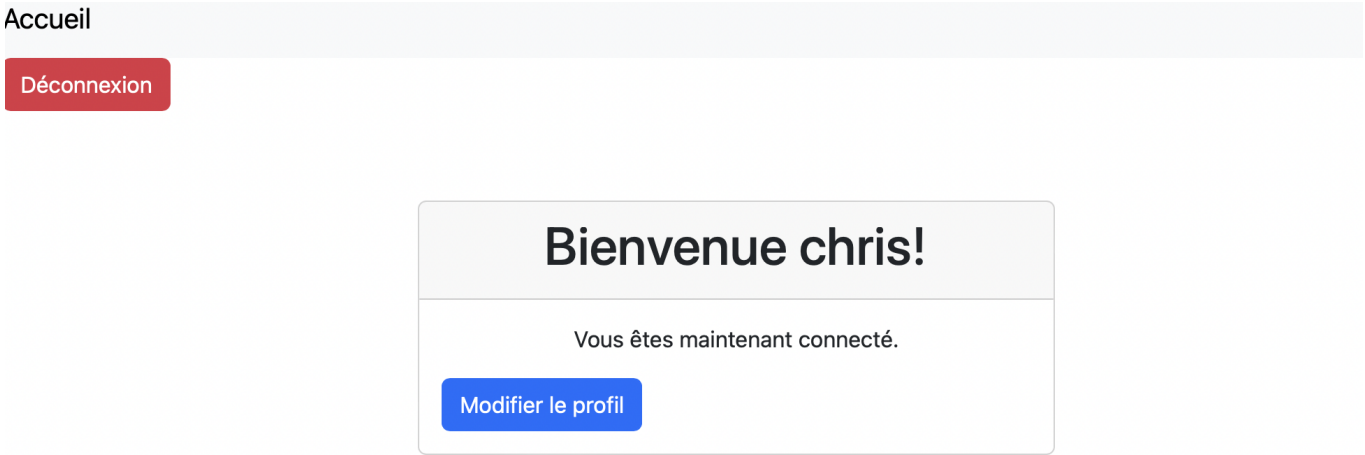
L'utilisateur peut dès à présent se connecter et se déconnecter tout en sécurité, nous avons utilisé les sessions en php en effet, Les sessions en PHP sont une fonctionnalité qui permet de gérer l'état d'une application web entre différentes pages ou requêtes du même utilisateur. Une session PHP est un moyen de stocker et de récupérer des données temporaires spécifiques à un utilisateur, qui sont accessibles sur différentes pages de l'application pendant la durée de sa session active.

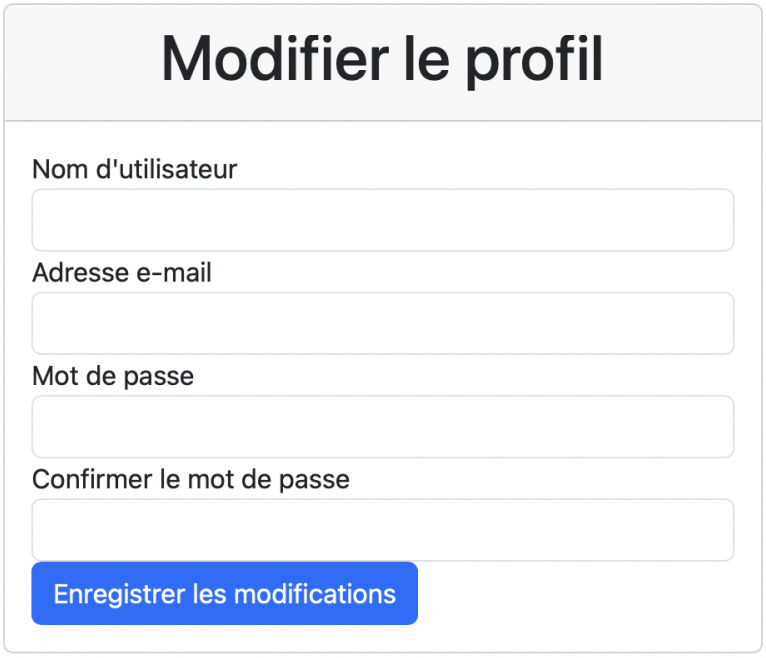
```
session_start
```

Quand un utilisateur se connecte, nous créons une session à partir de son nom d'utilisateur

```
$_SESSION['username'] = $_POST['username']
```

Les sessions sont tout de suite ecrasé quand l'utilisateur décide de se déconnecter.





La session d'un utilisateur est effacé avec la ligne de code suivante

```
session_destroy()
```

5. Mettre en place un système simple de traçabilité via des fichiers de logs

1. Vérifier la méthode de soumission des données :

La première étape consiste à vérifier que la méthode utilisée pour soumettre les données est POST. Cela signifie que les données ont été envoyées depuis un formulaire et qu'elles ne peuvent pas être accessibles directement via l'URL.

2. Récupérer les données soumises par le formulaire :

Le code récupère les données soumises par le formulaire en utilisant la méthode $_POST. Les données sont stockées dans des variables (nom, email, password) pour une utilisation ultérieure.

3. Vérifier que tous les champs sont remplis et que l'e-mail est valide :

Cette étape vérifie si les champs nom, email et password sont remplis en utilisant la fonction empty(). Si l'un de ces champs est vide, le script affiche un message d'erreur et s'arrête.

De plus, cette étape vérifie que l'e-mail est valide en utilisant la fonction filter_var() avec le paramètre FILTER_VALIDATE_EMAIL. Si l'e-mail n'est pas valide, le script affiche un message d'erreur et s'arrête.

4. Se connecter à la base de données MySQL :

Le code établit une connexion avec la base de données MySQL en utilisant la fonction mysqli_connect(). Les paramètres nécessaires pour se connecter à la base de données sont le nom d'hôte, le nom d'utilisateur, le mot de passe et le nom de la base de données.

5. Insertion des données dans la base de données :

Le code crée une requête SQL pour insérer les données du formulaire dans la table users de la base de données.

La requête est exécutée en utilisant la fonction mysqli_query(). Si la requête réussit, le code affiche un message de confirmation.

6. Récupération de l'identifiant de l'utilisateur :

Le code récupère l'identifiant de l'utilisateur en effectuant une requête SELECT sur la table users en utilisant l'adresse e-mail soumise dans le formulaire. Il stocke l'identifiant dans une variable $id_utilisateur.

7. Enregistrement de la trace de modification dans un fichier log :

Le code enregistre une trace de modification dans un fichier de logs en utilisant la fonction error_log(). Le nom du fichier de logs est généré à partir de la date courante (au format 'YYYY-MM-DD') et stocké dans une variable $log_file. Le message de log est stocké dans une variable $log_message et contient la date et l'heure courantes, l'identifiant de l'utilisateur et le message 'modification du profil'.

8. Fermeture de la connexion à la base de données :

Le code ferme la connexion à la base de données en utilisant la fonction mysqli_close().

6 - Mettre en place un chiffrement applicatif (Les données stockée doivent être chiffrées de manière réversible

1. Vérifier la méthode de soumission des données :

La première étape consiste à vérifier que la méthode utilisée pour soumettre les données est POST. Cela signifie que les données ont été envoyées depuis un formulaire et qu'elles ne peuvent pas être accessibles directement via l'URL.

2. Récupérer les données soumises par le formulaire :

Le code récupère les données soumises par le formulaire en utilisant la méthode $_POST. Les données sont stockées dans des variables (nom, email, password) pour une utilisation ultérieure.

3. Vérifier que tous les champs sont remplis et que l'e-mail est valide :

Cette étape vérifie si les champs nom, email et password sont remplis en utilisant la fonction empty(). Si l'un de ces champs est vide, le script affiche un message d'erreur et s'arrête.

De plus, cette étape vérifie que l'e-mail est valide en utilisant la fonction filter_var() avec le paramètre FILTER_VALIDATE_EMAIL. Si l'e-mail n'est pas valide, le script affiche un message d'erreur et s'arrête.

4. Se connecter à la base de données MySQL :

Le code établit une connexion avec la base de données MySQL en utilisant la fonction mysqli_connect(). Les paramètres nécessaires pour se connecter à la base de données sont le nom d'hôte, le nom d'utilisateur, le mot de passe et le nom de la base de données.

5. Générer une clé de chiffrement aléatoire :

Le code utilise la fonction openssl_random_pseudo_bytes() pour générer une clé de chiffrement aléatoire de 32 octets.

6. Chiffrer les données :

Le code utilise la fonction openssl_encrypt() pour chiffrer les données (nom, email, password) en utilisant l'algorithme AES-256-CBC et la clé de chiffrement aléatoire générée à l'étape précédente.

7. Stocker les données chiffrées dans la base de données :

Le code utilise la méthode préparée mysqli_prepare() pour préparer la requête SQL d'insertion dans la table 'users'. Les données chiffrées sont liées aux paramètres de la requête avec la méthode bind_param() et la requête est exécutée avec la méthode execute(). Si l'exécution est réussie, un message est affiché pour indiquer que l'enregistrement a été créé avec succès, sinon un message d'erreur est affiché.

8. Déchiffrer les données pour les afficher à l'utilisateur :

Le code utilise la fonction openssl_decrypt() pour déchiffrer les données (nom, email, password) chiffrées précédemment en utilisant la même clé de chiffrement aléatoire. Les données déchiffrées sont stockées dans des variables distinctes (decrypted_password, decrypted_email, decrypted_nom). Ces données sont ensuite affichées à l'utilisateur à l'aide de la fonction echo().

9. Fermer la requête et la connexion à la base de données :

Enfin, le code ferme la requête en utilisant la méthode close() et la connexion à la base de données en utilisant la fonction mysqli_close().