cours

1. Injection sql

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

> Pasted image 20230321225359.png

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

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

2. Démontrer une faille XSS

> Pasted image 20230320150514.png

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 : Cette fonction permet entre autres de convertir les balises < et > en < et >. Ainsi, si un utilisateur entre `script **<script>** dans la boîte de saisie, ceci sera converti en **<script>**. La tentative d'attaque deviendra tout à fait inoffensive.`

```
 $nom = htmlspecialchars($_POST['nom']);

$prenom = htmlspecialchars($_POST['prenom']);

$age = htmlspecialchars($_POST['age']);

$avis = htmlspecialchars($_POST['avis']); ```

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

##  3.1 Le formulaire
```

Après avoir construit mon 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]));
```

> Pasted image 20230325175201.png

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 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

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

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