Exercice 1:

1. Expliquer en quelques lignes le cross-site scripting ?

2. Expliquer la fixation de session en une ligne.

3. Expliquer les différentes étapes exécutées par un pirate pour voler les cookies d'une victime.

4. Expliquez la méthode que pourra utiliser un pirate pour voler l'Id d'une session d'une victime.

5. A quoi sert la fonction htmlspecialchars ?

6. Expliquez en une ligne le risque d'attaque via un file upload ?

7. Quel est le rôle de HTMLPurifier ?

8. Si un site est vulnérable aux attaques XSS, donner deux actions que peut faire un pirate sur ce site.

9. Quel est le risque si un fichier de connexion à la base de données est nommé « bdd.txt » ?

Exercice 2:

1. Un site stocke et affiche du contenu qui provient d'une source extérieure. Le site doit intégrer un contenu HTML riche. Quelle est la fonction qu'il faut utiliser pour sécuriser ce site ?
    a. remove_tags()
    b. strip_tags()
    c. htmlspecialchars()
    d. purify()
    e. Aucune bonne réponse

2. Il est déconseillé :
    a. de nommer un fichier contenant du code php « config.bdd.php »
    b. d'utiliser le même utilisateur base de données pour deux applications
    c. d'utiliser l'utilisateur base de données MySQL root dans le code php
    d. de cacher les messages d'erreur en production
    e. Aucune bonne réponse

3. Une application Web a deux pages. La première « form.php » contient un formulaire de vote pour un candidat. La deuxième « vote.php » permet d'insérer dans la base de données les informations du candidat choisis par l'utilisateur. Pour protéger cette application des attaques CSRF, il faut :

a. sur form.php : générer un token unique, écraser la valeur de l'ID de session avec la valeur de ce token. Sur la page vote.php : vérifier si l'ID de session est bien renseigné.
b. sur form.php : générer un token unique, écraser la valeur de l'ID de session avec la valeur de ce token.
c. sur form.php : générer un token unique, ajouter un champ invisible au formulaire qui a comme valeur ce token et stocker ce token dans la session. Sur vote.php : vérifier si le token reçu via le formulaire est le même que celui stocké dans la session.
d. sur form.php : générer un token unique, ajouter un champ invisible au formulaire qui a comme valeur ce token et stocker ce token dans la session.
e. Aucune bonne réponse

4. Parmi les bonnes pratiques de sécurité Web :
   a. S'assurer que les données reçus sont au format attendu
   b. Ne jamais utiliser directement les données client
   c. Eviter d'utiliser les sessions
   d. Faire la validation de données seulement coté client
   e. Aucune bonne réponse

5. Un pirate a la possibilité de connaitre ou de deviner une partie du code source d'une application php si l'application web
   a. est open source
   b. utilise error_reporting(E_ALL)
   c. utilise HTMLPurifier
   d. n'utilise pas le PHP Objet
   e. Aucune bonne réponse

6. Le code php ci-dessous

```
…
$req = $bdd->prepare(' UPDATE comments SET text=? WHERE id=?');
$req->execute(array($_GET['content'], $_GET['id']));
…
```

   a. Peut être piraté si id = "1 OR 1=1"
   b. Peut être piraté si id = "1 OR '1'='1'"
   c. Peut être piraté si id = "1 AND 1=1"
   d. Peut être piraté si content = "X', approved='1"
   e. Aucune bonne réponse

7. Une attaque de type CSRF sert à
   a. Injecter du code sql dans un formulaire
   b. Voler l'identifiant de session
   c. Voler des cookies stockés coté client
   d. Voler des cookies stockés coté serveur
   e. Aucune bonne réponse

Exercice 3 :

Soit le code php de vérification d'une connexion à un site :

```
…
<input type="text" name="login">
<input type="password" name="password">
…
$login = $_POST['login'];
$password = $_POST['password'];
$req = $bdd->query("SELECT count(*) FROM User WHERE
login='".$login."' AND pswd='".$password. "'");
…
```

1. Donnez le code qu'un pirate pourra injecter pour pouvoir accéder au site sans avoir des login et mot de passe valides
2. Donnez le code qu'un pirate pourra injecter pour pouvoir mettre à vide tous les mots de passes, sachant que le pirate a pu savoir que le nom de la table est « User » et que le champ qui stocke le mot de passe est « pswd »
3. Donnez le code qu'un pirate pourra injecter pour pouvoir supprimer la base de données, sachant que le pirate a pu savoir que le nom de cette base est « geststock »

Pour chaque question il faut spécifiez le champ qu'il faut utiliser.

Exercice 4 :

Le code article.php ci-dessous représente une vulnérabilité.

```
...<form method="post" action = "article.php">
<input type="text" name="commentaire"><br/>
<input type="submit" > </form>...
<?php
if (isset($_POST['commentaire'])){
//mettre à jour la table des commentaires dans la base de données.
Insérer dans la table commentaire avec un prepare et un execute
...
}
//Affichage de la liste des commentaires déjà envoyés à partir de
base de données. Select de la table commentaire avec un prepare et un
execute
...
?>
```

1. Quel est le nom de l'attaque lié à ce problème ?
2. Expliquez comment un pirate pourra exploiter cette vulnérabilité.
3. Expliquez en une ou deux lignes la solution à faire pour corriger cette vulnérabilité.

Exercice 5:

Soit le code source suivant :

```
$content = $_POST['content'];
$id = $_POST['id'];
$mysqli->query("UPDATE    comments "."SET    text='$content'    WHERE
id=$id");
```

1. Ce code est vulnérable à quel type d'attaque ?

2. Comment on pourra sécuriser ce code ?

Solutions

Exercice 1:

1. C'est une attaque Web qui exploite une vulnérabilité sur un site qui affiche du contenu qui provient d'une source extérieure sans le filtrer de manière adéquate. Un pirate envoie une entrée utilisateur contenant un code JavaScript qui est stocké ensuite interprété par le navigateur de l'utilisateur.
   Ce type d'attaques peut modifier le contenu d'une page web, permettre le vol de cookie, …

2. L'attaquant impose à la victime de se connecter sur un site avec un session ID fixé.

3.
   - Via une attaque XSS, le pirate modifie le contenu d'une page en injectant un code JavaScript.
   - Ce code contient une redirection (un lien) vers une url d'un script écrit par le pirate.
   - Le cookie de la victime est lu via JavaScript et envoyé via cette url.

4. Attaquer en XSS et voler le cookie contenant l'id de la session via un JavaScript.

5.
   - Se protéger des attaques XSS
   - Filtrer le HTML de quelques caractères sensibles

6. Le fichier envoyé par l'utilisateur risque d'être un script malveillant

7. Filtrer le code HTML de quelques balises et attributs pour éviter les attaques XSS.

8.
   - Afficher une fenêtre d'alerte
   - Modifier le style ou le contenu d'une page web

9. Le code source PHP peut être exposé et lu par des internautes via leurs navigateurs. Ce code source pourra fournir les informations de connexion à la base de données à des utilisateurs mal intentionnés.

Exercice 2:

1. d
2. b, c
3. c
4. a, b
5. a, b
6. e
7. e

Exercice 3 :

Sur le deuxième champ

1. 1' OR '1'='1
2. 1'; UPDATE User SET password='' WHERE '1'='1
3. 1'; drop database geststock ;

Exercice 4 :

1. XSS.
2. Mettre un contenue JavaScript dans le champ commentaire qui pourra modifier le contenue de la page web, faire une redirection, voler des cookies, …
3. Filtrer les données, utiliser HTML purifier, faire de l'output escaping, …

Exercice 5:

1. Injections SQL

2. A l'aide de l'échappement spécifique à SQL, à laide de «mysql_real_escape_string », à l'aide des requêtes préparées, …