Éviter les attaques par session en php

1. ## Objectifs

   - Mettre en place des mécanismes de sécurité pour éviter les attaques par session
   - Gérer des données utilisateur persistantes à travers les sessions

2. ## Présentation

   - Le détournement de session est une attaque par laquelle le pirate vole votre identifiant de session pour accéder au compte prévu. En utilisant cet identifiant de session, le pirate peut valider votre session en envoyant une requête au serveur, où un tableau $ _SESSION valide sa disponibilité sans rester à votre connaissance. Elle peut être effectuée via une attaque XSS ou en accédant aux données où les données de session sont stockées.

3. ## Mettre en place des mécanismes de sécurité pour éviter les attaques par session en php

   - Pour éviter les attaques par session en PHP, voici quelques mesures de sécurité que vous pouvez mettre en place :
     - ***1*-Utiliser des identifiants de session sécurisés**: Les identifiants de session générés par PHP peuvent être prévisibles et donc vulnérables à une attaque par devinette. Pour éviter cela, vous pouvez utiliser la fonction session_regenerate_id() pour générer des identifiants de session aléatoires et uniques.
     - **Exemple**
       - Voici un exemple de code en PHP qui utilise des identifiants de session sécurisés en combinant `session_regenerate_id()` et `random_bytes()`:

```
<?php// Démarrer la session
session_start();

// Régénérer l'identifiant de session
session_regenerate_id(true);

// Générer un identifiant de session aléatoire et unique
```

apcpedagogie.com

```
$random_id = bin2hex(random_bytes(32));

// Stocker l'identifiant de session dans la variable $_SESSION
$_SESSION['id'] = $random_id;
?>
```

- En utilisant cette approche, nous pouvons être sûr que chaque session aura un identifiant de session aléatoire et unique, ce qui rendra plus difficile pour les attaquants de deviner ou de voler des identifiants de session valides. De plus, en régénérant l'identifiant de session, nous réduisons le risque de fixation de session en empêchant les attaquants de forcer un identifiant de session valide sur l'utilisateur.

- ***2-*** Définir des limites de temps pour les sessions : En PHP, les sessions sont stockées sur le serveur pendant une période de temps définie dans la configuration. Il est important de limiter cette durée à un intervalle raisonnable pour éviter les attaques par session volée.
- ***3-*** Stocker les sessions dans une zone de mémoire protégée : Les sessions PHP peuvent être stockées sur le disque ou en mémoire. Il est recommandé de stocker les sessions en mémoire car cela les rend moins accessibles aux attaquants.
- ***4-*** Utiliser HTTPS pour les connexions sécurisées : Lorsque vous utilisez HTTPS pour vos connexions, les cookies et les sessions sont automatiquement chiffrés. Cela empêchera les attaquants de voler des sessions en interceptant le trafic réseau.
- ***5-*** Vérifier l'adresse IP de l'utilisateur : Il peut être utile de vérifier si l'adresse IP de l'utilisateur est la même que celle qui a été utilisée lors de la création de la session. Cela peut aider à empêcher les attaques par session volée.
- ***6-*** Vérifier le navigateur de l'utilisateur : Vous pouvez également vérifier le navigateur de l'utilisateur pour vous assurer que la session n'a pas été volée. Si le navigateur de l'utilisateur a été modifié, vous pouvez invalider la session.
- ***7-*** Stocker les informations de session sur le serveur : Les informations de session ne doivent pas être stockées sur le côté client (par exemple, dans des cookies) car cela peut les rendre vulnérables aux attaques XSS.

4. ## La fonction `session_regenerate_id()`

- La fonction `session_regenerate_id()` en PHP est utilisée pour renouveler l'identifiant de session d'un utilisateur. Voici quelques-uns des objectifs clés d'utilisation de cette fonction :
  - **Sécurité**: l'un des principaux objectifs de la fonction `session_regenerate_id()` est de renforcer la sécurité de la session. En renouvelant l'identifiant de session, cette

fonction peut aider à prévenir les attaques de session volées, où un attaquant tente d'utiliser l'identifiant de session d'un utilisateur légitime pour accéder à leur compte.

- **Protection des données** : en renouvelant l'identifiant de session, la fonction `session_regenerate_id()` peut également aider à protéger les données sensibles stockées dans la session. Si un attaquant parvient à accéder à une session volée, ils ne pourront pas utiliser l'identifiant de session pour accéder aux données sensibles, car il aura été remplacé.
- **Confidentialité**: la fonction `session_regenerate_id()` peut également aider à protéger la confidentialité des utilisateurs. En régénérant l'identifiant de session chaque fois qu'un utilisateur se connecte, il devient plus difficile pour un attaquant de suivre l'activité d'un utilisateur en ligne.

- La fonction `session_regenerate_id()` permet de régénérer l'identifiant de session actuel avec un nouveau identifiant de session aléatoire et unique. Voici un exemple d'utilisation :

Script PHP

```
<?php
session_start(); // Démarre la session

// Vérifie si l'identifiant de session existe
if (!isset($_SESSION['id'])) {
   // Génère un identifiant de session aléatoire
   $_SESSION['id'] = bin2hex(random_bytes(32));
}

// Régénère l'identifiant de session toutes les 30 minutes
if (isset($_SESSION['LAST_ACTIVITY']) && (time() - $_SESSION['LAST_ACTIVITY'] >
1800)) {
    session_regenerate_id(true); // Régénère l'identifiant de session
    $_SESSION['LAST_ACTIVITY'] = time(); // Met à jour le temps de la dernière
activité
}

// Met à jour le temps de la dernière activité
$_SESSION['LAST_ACTIVITY'] = time();
```

Description

- Dans cet exemple, nous commençons par démarrer la session avec la fonction session_start(). Ensuite, nous vérifions si l'identifiant de session existe en utilisant isset($_SESSION['id']). Si l'identifiant de session n'existe pas, nous générons un

identifiant de session aléatoire à l'aide de la fonction random_bytes() et l'enregistrons dans $_SESSION['id'].

- Ensuite, nous utilisons la variable $_SESSION['LAST_ACTIVITY'] pour stocker le temps de la dernière activité de l'utilisateur. Nous vérifions si la dernière activité a eu lieu il y a plus de 30 minutes (1800 secondes) en utilisant (time() - $_SESSION['LAST_ACTIVITY'] > 1800). Si c'est le cas, nous régénérons l'identifiant de session à l'aide de session_regenerate_id(true).
- Enfin, nous mettons à jour $_SESSION['LAST_ACTIVITY'] avec le temps actuel, ce qui signifie que la dernière activité est maintenant l'instant présent.
- En utilisant session_regenerate_id(), vous pouvez générer des identifiants de session aléatoires et uniques pour votre application PHP, ce qui renforce la sécurité de vos sessions.

- **Exercice d'application :**

- Dans cet exercice, vous devez créer une **page de connexion sécurisée** avec PHP qui utilise `session_regenerate_id()` pour générer des identifiants de session aléatoires et uniques. Lorsque l'utilisateur se connecte avec succès, vous devez stocker son nom d'utilisateur dans la session et afficher un message de bienvenue.
  Solution

- **Script php**

```
<?php
session_start(); // Démarre la session

// Vérifie si l'utilisateur est déjà connecté
if (isset($_SESSION['username'])) {
    header('Location: welcome.php'); // Redirige vers la page d'accueil
    exit;
}

// Vérifie si l'utilisateur a soumis le formulaire de connexion
if (isset($_POST['username']) && isset($_POST['password'])) {
    // Vérifie les informations d'identification de l'utilisateur
    if ($_POST['username'] === 'riadh' && $_POST['password'] === 'password123') {
        $_SESSION['username'] = $_POST['username']; // Stocke le nom d'utilisateur
dans la session
        session_regenerate_id(true); // Régénère l'identifiant de session avec un
nouvel identifiant aléatoire et unique
```

apcpedagogie.com

```
        header('Location: welcome.php'); // Redirige vers la page d'accueil
        exit;
    } else {
        $error = 'Mauvaises informations de connexion'; // Message d'erreur
    }
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Page de connexion</title>
</head>
<body>
    <h1>Page de connexion</h1>
    <?php if (isset($error)) : ?>
        <p><?php echo $error; ?></p>
    <?php endif; ?>
    <form method="post">
        <div>
            <label for="username">Nom d'utilisateur:</label>
            <input type="text" name="username" required>
        </div>
        <div>
            <label for="password">Mot de passe:</label>
            <input type="password" name="password" required>
        </div>
        <div>
            <button type="submit">Se connecter</button>
        </div>
    </form>
</body>
</html>
```

5. **Description**

   - Dans cet exemple, nous commençons par démarrer la session avec **session_start()**. Ensuite, nous vérifions si l'utilisateur est déjà connecté en vérifiant si **$_SESSION['username']** existe. Si l'utilisateur est déjà connecté, nous le redirigeons vers la page d'accueil avec header('Location: accueil.php').
   - Si l'utilisateur n'est pas déjà connecté, nous vérifions si le formulaire de connexion a été soumis en vérifiant si **$_POST['username']** et **$_POST['password']** existent. Si c'est le

apcpedagogie.com

cas, nous vérifions les informations d'identification de l'utilisateur en comparant les valeurs de **$_POST['username']** et **$_POST['password']** avec des valeurs en dur ('riadh' et 'password123').

- Si les informations d'identification de l'utilisateur sont correctes, nous stockons le nom d'utilisateur dans la session avec **$_SESSION['username'] = $_POST['username']** et régénérons l'identifiant de session avec **session_regenerate_id(true)**. Ensuite, nous redir

6. Applications

- **Application 01**

Solution

- **Script php**
- **Description**

Cours PHP

Source : https://apcpedagogie.com/eviter-les-attaques-par-session-en-php · téléchargé le 07/11/2026 · © APC Pédagogie · apcpedagogie.com

apcpedagogie.com