🔒 Sécurité des Applications Web

Bonnes Pratiques pour Développeurs Débutants et Intermédiaires

📋 Sommaire

1 📘 Introduction à la Sécurité Web

La **sécurité des applications web** est une **compétence essentielle** pour tout développeur. Une application mal sécurisée peut être victime de :

- Vol de données (identifiants, mots de passe, informations bancaires).
- Injection de code malveillant.
- Atteinte à la réputation de l'entreprise.
- Sanctions légales (RGPD).

> 💡 **Objectif :** Apprendre à **concevoir** et **développer** des applications **résistantes aux attaques**.

2 🧠 Principes de Base

Les principes fondamentaux de la sécurité sont :

- **Confidentialité** : Seules les personnes autorisées peuvent accéder aux données.
- **Intégrité** : Les données ne peuvent être modifiées par des personnes non autorisées.
- **Disponibilité** : Le service est accessible aux utilisateurs autorisés quand ils en ont besoin.

3 🚨 Vulnérabilités Courantes (OWASP Top 10)

L'OWASP (Open Web Application Security Project) publie une liste des 10 principales vulnérabilités :

Classement	Nom	Description
1	Injection	Données non fiables sont envoyées à un interpréteur (ex: **injection SQL**).
2	Failles d'authentification	Problèmes dans la gestion des sessions, des mots de passe.
3	Exposition de données sensibles	Données non chiffrées ou mal protégées.

Classement	Nom	Description
4	XML External Entities (XXE)	Attaques par entités XML externes.
5	Broken Access Control	Utilisateurs accèdent à des ressources non autorisées.
6	Security Misconfiguration	Paramètres par défaut ou erreurs de configuration.
7	Cross-Site Scripting (XSS)	Injection de scripts dans la page (côté client).
8	Insecure Deserialization	Problèmes lors de la conversion d'objets en chaînes et vice-versa.
9	Using Components with Known Vulnerabilities	Utilisation de bibliothèques ou frameworks obsolètes.
10	Insufficient Logging & Monitoring	Manque de journalisation pour détecter les attaques.

4 ✅ Bonnes Pratiques de Développement

- **Valider et filtrer** toutes les données entrantes (formulaire, URL, cookies).
- **Utiliser des requêtes préparées** (mysqli/PDO) pour éviter les injections SQL.
- **Échapper les données** sortantes (affichage dans HTML) avec **htmlspecialchars()**.
- **Utiliser des sessions sécurisées** (jeton CSRF, expiration).
- **Ne jamais stocker de mot de passe en clair** (utiliser **password_hash()** et **password_verify()**).
- **Ne jamais exposer de données sensibles** dans le code source ou les logs.

- **Mettre à jour régulièrement** les bibliothèques et frameworks utilisés.
- **Journaliser les actions importantes** (connexions, erreurs, modifications).

5 🐘 Sécurité en PHP (mysqli, PDO)

🔒 Injection SQL

✅ Solution : Requêtes préparées

```
// --- MAUVAIS ---
$sql = "SELECT * FROM users WHERE email = '".$_POST['email']."'"; //
$result = mysqli_query($conn, $sql);

// --- BON AVEC MYSQLI ---
$email = $_POST['email'];
$stmt = $conn->prepare("SELECT * FROM users WHERE email = ?"); // ? =
$stmt->bind_param("s", $email); // "s" = string
$stmt->execute();
$result = $stmt->get_result();

// --- BON AVEC PDO ---
$stmt = $pdo->prepare("SELECT * FROM users WHERE email = :email");
$stmt->bindParam(':email', $email);
$stmt->execute();
$result = $stmt->fetchAll();
```

XSS (Cross-Site Scripting)

✅ Solution : Échappement des sorties HTML

```
// --- MAUVAIS ---
echo "Bonjour " . $_POST['nom']; // ❌ Si l'utilisateur entre '" script

// --- BON ---
echo "Bonjour " . htmlspecialchars($_POST['nom']); // ✅ Échappe les
```

Authentification

```
 // --- Stocker le mot de passe ---
$motDePasse = $_POST['password'];
$motDePasseHashe = password_hash($motDePasse, PASSWORD_DEFAULT);

// --- Vérifier le mot de passe ---
$motDePasseSaisi = $_POST['password'];
if (password_verify($motDePasseSaisi, $motDePasseHashe)) {
    // Connexion réussie
}
```

6 Sécurité en JavaScript (DOM)

XSS via le DOM

> ✅ **Solution : Ne jamais injecter du HTML brut**

```
 // --- MAUVAIS ---
let userInput = 'alert("XSS")';
document.getElementById('output').innerHTML = userInput; // ❌ DANGERI

// --- BON ---
document.getElementById('output').textContent = userInput; // ✅ Affi
```

Gestion des Tokens (CSRF)

Utilisez des tokens CSRF pour valider les formulaires. C'est comme un **mot de passe unique** pour **chaque formulaire ou requête**.

```
------- STEPHANE  ----------
(Générer un UUID our rendre le formulaire unique + rendre ce UUID TOKE
function generateUUID() {
    return sprintf('%04x%04x-%04x-%04x-%04x-%04x%04x%04x',
        mt_rand(0, 0xffff), mt_rand(0, 0xffff),
```

```
        mt_rand(0, 0xffff),
        mt_rand(0, 0x0fff) | 0x4000,
        mt_rand(0, 0x3fff) | 0x8000,
        mt_rand(0, 0xffff), mt_rand(0, 0xffff), mt_rand(0, 0xffff)
    );
}

echo generateUUID(); // Ex: 550e8400-e29b-41d4-a716-446655440000


---------------------------
```

7 🧪 Exemples Concrets de Code Sécurisé

Exemple 1 : Formulaire de Contact Sécurisé (PHP + HTML)

Envoyer

```
prepare("INSERT INTO messages (nom, email, message) VALUES (?, ?, ?)")
        $stmt->execute([$nom, $email, $message]);
        echo "✅ Message envoyé !";
    } else {
        echo "❌ Email invalide.";
    }
}
?>
```

Exemple 2 : Gestion de Panier avec Validation (JavaScript)

```
function ajouterAuPanier(idProduit) {
    // Vérifier que l'ID est un nombre
    if (isNaN(idProduit)) {
        alert("ID de produit invalide.");
        return;
    }

    fetch('/api/panier.php', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ id: idProduit })
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            alert("Produit ajouté au panier.");
        } else {
            alert("Erreur : " + data.message);
        }
    });
}
```

8 📝 Exercices Pratiques

Exercice 1 : Formulaire d'inscription sécurisé

Créez un formulaire PHP/HTML avec champs nom, email, mot de passe. Validez les données, hachez le mot de passe et enregistrez-les dans une base via une requête préparée.

Exercice 2 : Empêcher les attaques XSS

Créez une page qui affiche une variable PHP dans une page HTML. Utilisez **htmlspecialchars()** pour empêcher les attaques XSS.

Exercice 3 : Validation de données côté client

Utilisez JavaScript pour valider un email et une quantité positive dans un formulaire avant envoi.

Exercice 4 : Gestion de session

Créez une page PHP qui vérifie si un utilisateur est connecté via une session. Redirigez-le vers la page de login s'il ne l'est pas.

Exercice 5 : Journalisation des erreurs

Créez un script PHP qui enregistre les erreurs dans un fichier **log.txt** avec l'heure et la nature de l'erreur.