Sécurité Informatique

1. Principes de Sécurité

1.1 Injection SQL

Définition

L'injection SQL est une vulnérabilité qui permet à un attaquant d'exécuter des requêtes SQL malveillantes sur une base de données. Cela peut entraîner l'exposition, la modification ou la suppression de données.

Exemples

```
-- Requête SQL vulnérable
SELECT * FROM utilisateurs WHERE nom = '$nom'; -- Si $nom contient "'); DROP TABLE
utilisateurs; --", cela peut détruire la table.

-- Prévention
SELECT * FROM utilisateurs WHERE nom = ?; -- Utiliser des requêtes préparées.
```

1.2 XSS (Cross-Site Scripting)

Définition

Le XSS est une vulnérabilité qui permet à un attaquant d'injecter des scripts malveillants dans des pages web, souvent pour voler des cookies ou des informations sensibles.

Exemples

1.3 CSRF (Cross-Site Request Forgery)

Définition

Le CSRF est une attaque qui incite un utilisateur authentifié à exécuter une action non désirée sur une application web.

Exemples

Meilleures Pratiques de Sécurité

- **Validation des Entrées** : Toujours valider et échapper les entrées utilisateur.
- **Utilisation de HTTPS** : Chiffrer les communications avec SSL/TLS.
- **Gestion des Sessions** : Établir des sessions sécurisées avec des identifiants de session aléatoires.

---

2. Authentification et Autorisation

2.1 Authentification

Définition

L'authentification est le processus de vérification de l'identité d'un utilisateur. Cela peut être fait par des mots de passe, des biométries, ou des tokens.

2.2 Autorisation

Définition

L'autorisation détermine quels ressources et actions un utilisateur authentifié peut accéder ou exécuter.

2.3 OAuth

Définition

OAuth est un protocole d'autorisation qui permet d'accorder à des applications tierces l'accès aux informations d'un utilisateur sans partager son mot de passe.

2.4 JWT (JSON Web Token)

Définition

JWT est un standard ouvert qui permet d'échanger des informations de manière sécurisée entre deux parties. Un JWT est un token compact et auto-contenu, utilisé pour l'authentification.

Exemple de Création d'un JWT

```
const jwt = require('jsonwebtoken');

// Générer un token
const token = jwt.sign({ userId: 123 }, 'secret_key', { expiresIn: '1h' });

// Vérifier un token
jwt.verify(token, 'secret_key', (err, decoded) => {
  if (err) {
    console.log('Token invalide');
  } else {
    console.log('Utilisateur ID:', decoded.userId);
  }
});
```

---

3. Chiffrement et Hachage

3.1 SSL/TLS

Définition

SSL (Secure Sockets Layer) et TLS (Transport Layer Security) sont des protocoles de sécurité qui chiffrent les communications entre les serveurs et les clients, protégeant ainsi les données sensibles.

3.2 Chiffrement

Définition

Le chiffrement transforme des données lisibles en un format illisible. Il existe deux types principaux :

- **Chiffrement symétrique** : La même clé est utilisée pour chiffrer et déchiffrer les données (ex. : AES).
- **Chiffrement asymétrique** : Utilise une paire de clés (publique et privée) pour chiffrer et déchiffrer les données (ex. : RSA).

3.3 Hachage

Définition

Le hachage est une méthode pour transformer des données en une chaîne fixe de caractères (empreinte) et est souvent utilisé pour stocker les mots de passe.

Exemple de Hachage avec bcrypt

```
const bcrypt = require('bcrypt');

async function hashPassword(password) {
  const saltRounds = 10;
  const hash = await bcrypt.hash(password, saltRounds);
  return hash;
}

async function verifyPassword(password, hash) {
  const match = await bcrypt.compare(password, hash);
  return match; // true ou false
}
```

Ces concepts de sécurité sont essentiels pour protéger les applications web contre les attaques et garantir la sécurité des données des utilisateurs.