SQLI-101

Initiation à l'injection SQL (dans un cadre de CTF)
En 10 diapos!

SECUQAM 2022-06-23
Jean Privat

Structured Query Language

```
SELECT *
FROM users
WHERE name = 'john'
AND passwd = 'hunter2'; # J’❤ SQL
```

SQL est un langage, avec une syntaxe, une sémantique, des mots clés, des littéraux, des opérateurs, des expressions, des fonctions, des commentaires, etc.

Note: chaque saveur (et version!) de SQL a ses propres règles (sqlite, mysql, etc.)

SQL et (mauvaise) programmation

```php
<?php
# ...
$name = $_POST['name'];
$pass = $_POST['pass'];
$query = "SELECT * FROM users WHERE name = '" . $name .
         "' AND passwd = '" . $pass . "';";
$result = mysqli_query($db, $query);
# ...
?>
```

Si `$name` vaut `john` et `$pass` vaut `hunter2`, alors que vaut `$query`?

Exemple de base

Si `$name` vaut `john` et `$pass` vaut `hunter2`, alors `$query` vaut

```
SELECT * FROM users WHERE name = 'john' AND passwd = 'hunter2';
```

Que comprend sql?

```
SELECT * FROM users WHERE name = 'john' AND passwd = 'hunter2';
```

Ce qui retourne le tuple de John si et seulement si le mot de passe est le bon

Mais si `$name` vaut `admin';#` et `$pass` vaut `lol`, alors que vaut `$query`?

Exemple problématique

Si `$name` vaut `admin';#` et `$pass` vaut `lol`, alors `$query` vaut

```
SELECT * FROM users WHERE name = 'admin';#' AND passwd = 'lol';
```

Que comprend sql?

```
SELECT * FROM users WHERE name = 'admin';#' AND passwd = 'lol';
```

Et ça retourne le tuple de l'administrateur (sans connaitre son mot de passe)

Contre-mesures

Idéalement

Ne pas programmer comme des cochons

➜ Requêtes préparées (RTFM de vos langages et libs)

Autrement

Filtrage et nettoyage des données utilisateurs (parfois mal fait ou insuffisant)

➜ Par le programme (mais pourquoi se compliquer la vie?)

➜ Par un WAF (web application firewall) quand on ne fait pas confiance aux devs

Trois grandes familles de SQLI (traditionnelles)

Classique

L'attaquant voit **directement** le résultat de la requête
➜ Donc de sa requête injectée

En aveugle (blind)

L'attaquant **ne voit pas** le résultat
Mais récupère de l'information annexe **partielle** (ok ou erreur par exemple)
➜ Construire des expressions ± complexes pour extraire un bit d'information à la fois

Chronométrée (time-based)

L'attaquant **ne voit rien**, mais peut chronométrer le **temps de la réponse**
➜ Ajouter des `sleep` conditionnels (ou équivalents) à la requête

Impacts

Que cherche à gagner l'attaquant?

- Contournement d'authentification
  ➜ Et de validations analogues
- Exfiltration de données
  ➜ Même dans d'autres tables
- Modification de données
  ➜ Si la requête d'origine, la saveur et la configuration s'y prête
- Lectures et écritures sur disque, exécutions de commandes shell, etc.
  ➜ Il y a rarement les droits, mais vérifier quand même
- Autre
  ➜ Exemple: exploitation d'une vulnérabilité binaire (pwn) dans un plugin sqlite

SLQi et CTF (pourquoi c'est amusant)

Variation sur le code et les technos

Originalités sur la mauvaise construction de la requête
➜ Adapter les stratégies d'attaque

Code caché (totalement ou partiellement)

➜ Comprendre ce qui peut bien se passer (hypotétiser et expérimenter)

Variation sur les filtres

Le challenge peut filtrer bizarrement (simulation d'un WAF ou d'un filtrage maison)
➜ Chercher des stratégies alternatives

Etc. les gens ont de l'imagination

À l'attaque!

- Laboratoire SQLI de INF600C ➜ https://ctf.uqam.ca
- Ringzer0, track SQL Injection ➜ https://ringzer0ctf.com/

Trucs et astuces

RTFM: (doc officielle >> tutos superficiels et blogpost incorrects)

- https://dev.mysql.com/doc/refman/5.7/en/
- https://sqlite.org/docs.html

Une étape à la fois : injections simples pour confirmer des hypothèses
Testez chez vous: validez syntaxe et sémantique
ou chez quelqu'un d'autre https://extendsclass.com/mysql-online.html

Ressources complémentaires

- Sous-catégorie (et exemple) de OWASP A03:2021-Injection (3e du top 10) https://owasp.org/Top10/A03_2021-Injection/
- Page dédiée OWAPS https://owasp.org/www-community/attacks/SQL_Injection
- CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') https://cwe.mitre.org/data/definitions/89.html