Mysql False注入及Mysql隐式转换

原创

baynk 于 2019-08-08 02:42:39 发布 469 收藏 2

分类专栏： # WEB安全 文章标签： CTF MySQL 隐式转换 False注入

本文链接：https://blog.csdn.net/u014029795/article/details/98803880

此文是由于一个CTF题而引发的学习和思考，以后坚持做CTF，还是能学不少东西的，有点后悔没早点开始了。。。

False注入

- 我也不知道为啥要叫这个名字，可能是因为利用系统错误来完成的注入，但是这个又和报错注入(error)不一样，所以才叫False注入吧。
- 来看这样一张普通的表，非常简单的一张表。

```
mysql> select * from test;
+------+--------+---------+
| id   | name   | flag    |
+------+--------+---------+
| 1    | baynk  | ayaya   |
| 2    | tutub  | ohoho   |
| 3    | tusiji | 1san579 |
+------+--------+---------+
3 rows in set (0.00 sec)
```

- 接着查询的时候指定一下name，但是指定的值为0。看如下表中的记录。

```
mysql> select * from test where name=0;
+------+--------+---------+
| id   | name   | flag    |
+------+--------+---------+
```

```
| 1    | baynk   | ayaya    |
| 2    | tutub   | ohoho    |
| 3    | tusiji  | 1san579  |
+------+---------+----------+
3 rows in set (0.00 sec)
```

- 是不是很神奇，没有name是0，但是就是这样还可以查询出表内的所有数据，这里涉及到了Mysql数据中的数据类型的隐式转换。接下来查看官方文档中的一段关于隐式转换的原文：

- The following rules describe how conversion occurs for comparison operations:

  If one or both arguments are NULL, the result of the comparison is NULL, except for the NULL-safe <=> equality comparison operator. For NULL <=> NULL, the result is true. No conversion is needed.

  If both arguments in a comparison operation are strings, they are compared as strings.

  If both arguments are integers, they are compared as integers.

  Hexadecimal values are treated as binary strings if not compared to a number.

  If one of the arguments is a TIMESTAMP or DATETIME column and the other argument is a constant, the constant is converted to a timestamp before the comparison is performed. This is done to be more ODBC-friendly. Note that this is not done for the arguments to IN()! To be safe, always use complete datetime, date, or time strings when doing comparisons. For example, to achieve best results when using BETWEEN with date or time values, use CAST() to explicitly convert the values to the desired data type.

  If one of the arguments is a decimal value, comparison depends on the other argument. The arguments are compared as decimal values if the other argument is a decimal or integer value, or as floating-point values if the other argument is a floating-point value.

  In all other cases, the arguments are compared as floating-point (real) numbers.

  为了大家看起来理解起来更加方便点，我贴段百度翻译过来，感觉这次机翻还是挺准确的。。。

  以下规则描述比较操作的转换方式：

  如果一个或两个参数都为空，则比较结果为空，但空安全<=>相等比较运算符除外。对于空值<=>空值，结果为真。不需要转换。

  如果比较操作中的两个参数都是字符串，则将它们作为字符串进行比较。

  如果两个参数都是整数，则将它们作为整数进行比较。

  如果不与数字进行比较，十六进制值将被视为二进制字符串。

  如果其中一个参数是timestamp或datetime列，而另一个参数是常量，则在执行比较之前，该常量将转换为timestamp。这样做是为了更好地支持ODBC。注意，这不是为in()中的参数所做的！为了安全起见，在进行比较时始终使用完整的日期时间、日期或时间字符串。例如，要在与日期或时间值之间使用时获得最佳结果，请使用cast()将值显式转换为所需的数据类型。

  如果其中一个参数是十进制值，则比较取决于另一个参数。如果另一个参数是十进制或整数值，则将这些参数作为十进制值进行比较；如果另一个参数是浮点值，则将这些参数作为浮点值进行比较。

  在所有其他情况下，参数都是作为浮点数（实数）进行比较的。

- 在看完了隐式转换的说明后，再来理解我们之前的语句`select * from test where name=0`。在我们表中name所有的数据都是字符串，现在要拿字符串和0(整数)来比较，所以要把字符串和0都转成浮点数，但是字符串转成浮点数会失败的，所以

字符串就会转成0，结果就变成了0=0，于是匹配成功，可以显示出整张表的数据。

- 然后我们接着在看这样的一张表记录。

```
mysql> select * from test where flag=0;
+------+-------+-------+
| id   | name  | flag  |
+------+-------+-------+
| 1    | baynk | ayaya |
| 2    | tutub | ohoho |
+------+-------+-------+
2 rows in set (0.00 sec)
```

当我使用flag=0时，居然只出现了两项记录，id=3的这条不见了，我们来看看第3条记录，flag是为1san579的，如果把此字符串转成浮点数，是可以转换成功的，结果为1，因为当字符串转浮点时如果是以数字开头就转到数字处再截断，于是结果就变成了1=0，所以第3条数据没了。

- 我接着又做了一次这样的测试，flag=1的情况，结果和我猜的是一样的，只显示第3条，不显示前2条。

```
mysql> select * from test where flag=1;
+------+--------+---------+
| id   | name   | flag    |
+------+--------+---------+
| 3    | tusiji | 1san579 |
+------+--------+---------+
1 row in set (0.00 sec)
```

- ~~所以之前那篇博文我到底写了啥垃圾玩意。。。马上去改了，消灭不良记录。。。~~
- 按照之前CTF题目来讲，应该是这样的一段代码，`select * from test where name='1'='0' and flag='1'='0'`，实际上是`select * from test where name=0 and flag=0`，也就是要把之前查询name=0和flag=0两张表相同的部分整理出来即可，所以应该是2条数据，因为flag=0的时候是没有第3条数据的，但是结果和我想的不一样。。。

```
mysql> select * from test where name='1'='0' and flag='1'='0';
+------+--------+---------+
| id   | name   | flag    |
+------+--------+---------+
| 1    | baynk  | ayaya   |
| 2    | tutub  | ohoho   |
| 3    | tusiji | 1san579 |
+------+--------+---------+
3 rows in set (0.00 sec)

mysql>
```

- 然后我又不死心去查询了这样一组数据，结果证明我刚刚的猜想是正确的，错的是我上段话中的“实际”。

```
mysql> select * from test where name=0 and flag=0;
```

```
+------+-------+-------+
| id   | name  | flag  |
+------+-------+-------+
| 1    | baynk | ayaya |
| 2    | tutub | ohoho |
+------+-------+-------+
2 rows in set (0.00 sec)
```

- 看来`select * from test where name='1'='0' and flag='1'='0'`实际上不是`select * from test where name=0 and flag=0`，啊啊啊！！！再接下来的测试中确定了是下面的问题。。。

```
mysql> select * from test where name='1'='0' and flag=0;
+------+-------+-------+
| id   | name  | flag  |
+------+-------+-------+
| 1    | baynk | ayaya |
| 2    | tutub | ohoho |
+------+-------+-------+
2 rows in set (0.00 sec)

mysql> select * from test where name='1'='0' and flag='1'='0';
+------+--------+---------+
| id   | name   | flag    |
+------+--------+---------+
| 1    | baynk  | ayaya   |
| 2    | tutub  | ohoho   |
| 3    | tusiji | 1san579 |
+------+--------+---------+
3 rows in set (0.00 sec)
```

- 我以为是and和=后优先顺序的问题，于是，我又错了。。。

```
mysql> select * from test where name='1'='0' and (flag='1'='0');
+------+--------+---------+
| id   | name   | flag    |
+------+--------+---------+
| 1    | baynk  | ayaya   |
| 2    | tutub  | ohoho   |
| 3    | tusiji | 1san579 |
+------+--------+---------+
3 rows in set (0.00 sec)
```

- 正准备求助的时候，我试了一下这条语句，结果我惊了。。。

```
mysql> select '1san579'='1'='0';
+----------------------+
| '1san579'='1'='0'    |
+----------------------+
|                    1 |
+----------------------+
```

```
1 row in set (0.00 sec)

mysql> select '1san579'=0;
+--------------+
| '1san579'=0 |
+--------------+
|            0 |
+--------------+
1 row in set, 1 warning (0.00 sec)

mysql>
```

原来如果有连续多个=号，是从左往后依次执行。。所以'1san579'='1'='0'这样的东西就变成了0='0'，结果为1，但是如果是从右往左的后就变成了'1san579'=0，结果为0。

- 所以 `select * from test where name='1'='0' and flag='1'='0'` 最后变成了 `select * from test where 1 and 1`，就是这样显示出全表的。

```
mysql> select * from test where 1 and 1;
+------+--------+---------+
| id   | name   | flag    |
+------+--------+---------+
| 1    | baynk  | ayaya   |
| 2    | tutub  | ohoho   |
| 3    | tusiji | 1san579 |
+------+--------+---------+
3 rows in set (0.00 sec)

mysql>
```

总结

- 我擦，之前看的Writeup都TM乱说的什么东西，我一直被误导，搞得我以为是从右往左比较。。。
- 哎，还是自己菜，不能怪其他人。。。！！！
- 此时心情真滴复杂。。。