对ShirneCMS的一次审计思路

漏洞的审计

源头是在 `/src/extend/extcore/ImageCrop.php/crop` 这个方法里面发现有个 `getimagesize` 函数，这个函数是能够触发 `phar` 反序列化漏洞的，而这个 cms 是基于 thinkphp5.1 框架二次开发的，这个框架有个反序列化漏洞相信大家都很熟悉了，所以我们的目的就是能控制 `$imgData` 这个变量就行了

```
32          if (empty($img)){
33              exit();
34          }
35          if($imgWidth<1 && $imgHeight<1){
36              $imgWidth = config('upload.default_size');
37          }
38          if($imgQuality<1){
39              $imgQuality = config('upload.default_quality');
40          }
41          echo $img;
42          $imgData=$this->getImgData($img);
43
44          if($imgData!==false && !empty($imgData)) {
45              $imageinfo = getimagesize($imgData);
46              $image = imagecreatefromstring($imgData);
47
48              $photoWidth = $imageinfo[0];
49              $photoHeight = $imageinfo[1];
50
```

可以看到 `$imgData` 是由 `$this->getImgData($img);` 控制的，我们跟踪进去

```
private function getImgData($img){        if(strripos($img, 'http://')!==FALSE OR strrip
os($img,'https://') !==FALSE) { //站外图片            $data=file_get_contents($im
g);        }else{ //站内图片            $file=DOC_ROOT.'/'.$img;            if(is_file($fi
le)) {                $data = file_get_contents($file);            }else
{                return false;            }        }        return $data;    }
```

可以看到这里会限制只能由 `http://` 或者 `https://` 开头的参数才能获取站外的图片信息

再看看全局搜索 `crop` 这个方法看看哪里会调用他

```php
    public function cacheimage($img){
        Log::close();
        $paths=explode('.',$img);
        if(count($paths)==3) {
            preg_match_all('/(w|h|q|m)(\d+)(?:_|$)/', $paths[1], $matches);
            $args = [];
            foreach ($matches[1] as $idx=>$key){
                $args[$key]=$matches[2][$idx];
            }

            // echo $paths[0].'.'.$paths[2];
            $response = crop_image($paths[0].'.'.$paths[2], $args);
            if($response->getCode()==200) {
                file_put_contents(DOC_ROOT . '/' . $img, $response->getData());
            }
            $response->cacheControl('max-age=2592000');
            return $response;
        }else{
            return redirect(ltrim(config('upload.default_img'),'.'));
        }
    }
```

我们在 `src/application/task/controller/UtilController.php/cropimage` 发现有个 `crop_image` 函数，我们跟踪进去

```php
function crop_image($file, $options){    // echo $file;    $imageCrop=new \extcore\ImageCrop($file, $options);    return $imageCrop->crop();}
```

发现这里会调用到我们上面的 `crop` 函数

这里的 `$file` 参数也就是我们传给 `getImgData` 函数的 `$img` 变量，所以这里我们看看如何去控制他，可以看到 `crop_image` 方法里面有一个 `$paths=explode('.',$img);` ，就是会根据点去分隔我们的 `$img` 参数，然后又要 `count($paths)==3` ，我们可以回想到 `getImgData` 限制了 `http` 的开头，我们想要 `phar` 反序列化的话，必须是 `phar://` 的开头，那么我们直接在 `vps` 上放置我们的 `phar` 文件的路径不就可以了

但是这里有一个问题，我们正常输入一个 IP 地址的话肯定是不行的，因为他的 `count($paths)==3` ，所以我们可以使用十六进制绕过的方法，所以也就限制了这种方法只能在 `linux` 下面使用，这里顺便贴一下之前写的一个转进制的脚本

```php
<?php$ip = '127.0.0.1';$ip = explode('.',$ip);$r = ($ip[0] << 24) | ($ip[1] << 16) | ($ip[2] << 8) | $ip[3] ;if($r < 0) {    $r += 4294967296;}echo "十进制:";echo $r;echo "八进制:";echo decoct($r);echo "十六进制:";echo dechex($r);?>
```

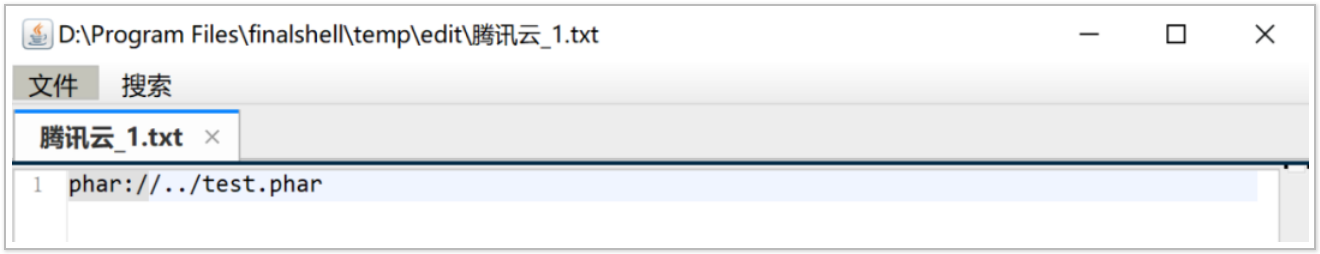
我们在 `$ip` 处贴上自己的 `vps` 的地址，这里要注意生成的十六进制前面要加上 `0x`

然后 `cacheimage` 函数的

```
$response = crop_image($paths[0].'.'.$paths[2], $args);
```

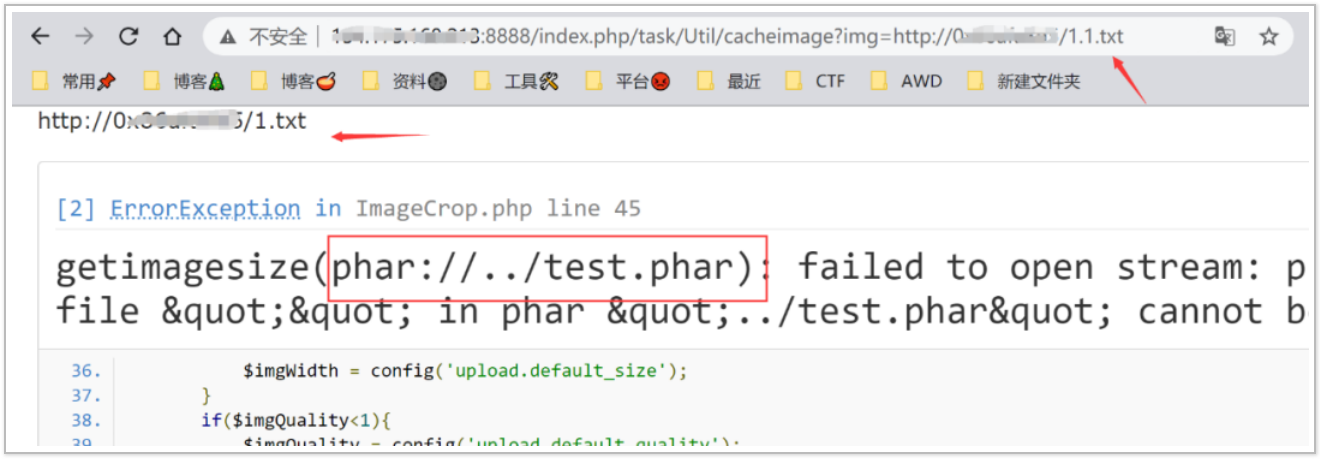
`$paths[0].'.'.$paths[2]` 就是我们想要控制的参数，因为前面 `explode` 把我们的 `url` 地址分成了 3 份，这里把第一份和第三份拼接了起来，于是我们可以构造类似于 `http://vps-ip/1.1.txt` 的形式，这里样我们的 `$paths[0].'.'.$paths[2]` 也就成为了 `1.txt` 也就是我们可控的东西了，同时这里也明白了为什么要将 `vps-ip` 转成 16 进制的原因了

我们同时在 `vps` 上放置 `test.phar` 的路径，这个 `cms` 后台是可以上传 `jpg` 文件的，当然 `phar` 反序列化的话即使是 `jpg` 后缀的文件也是能够成功反序列的，这里我为了方便直接放在根目录下

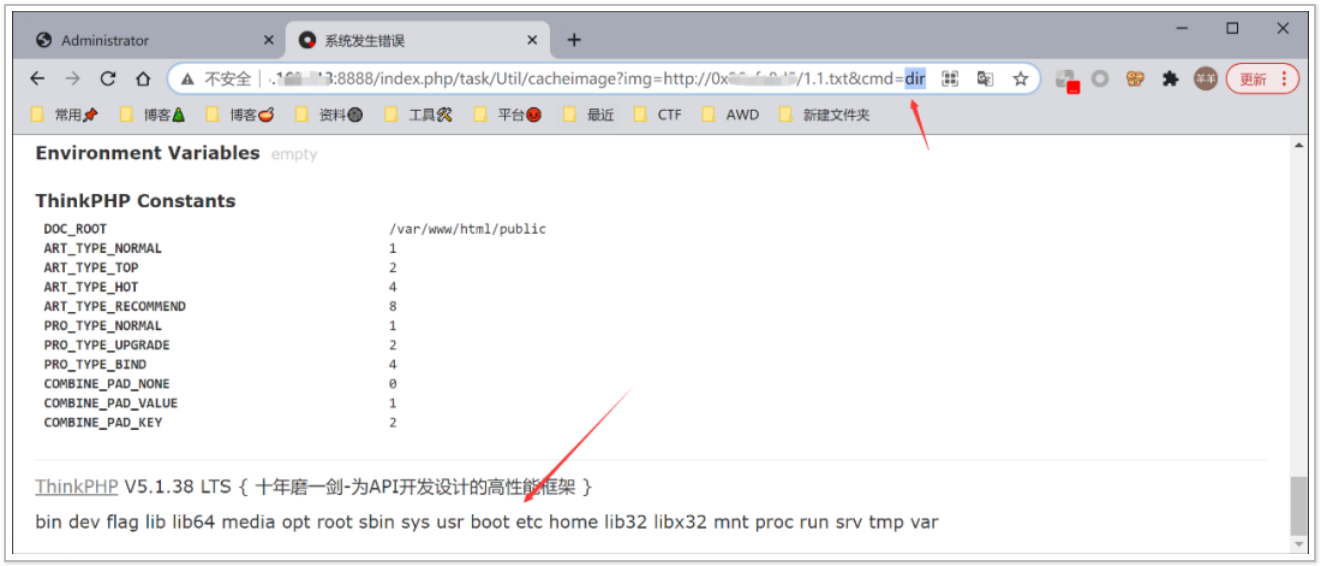


到了这一步我们的思路基本就清晰了，我们测试一下 `$img` 是否能够正确的打印出来，可以手动添加一个 `echo $img;`

我们访问一下 `cacheimage` 的路由



可以看到我们的 `$img` 变成了 `1.txt` ， `getimagesize` 函数里面也成功接收到我们放在 `1.txt` 里面的内容



我们再 `cmd` 传参我们的命令即可看到漏洞已经成功利用

总结

漏洞已经上交于 cnvd 平台，然后这个漏洞由于十六进制绕过的问题，只能在 `linux` 下才可以成功实现，所以可以把 `cms` 放在 `docker` 里面进行测试，然后在一些小的 cms 里面关于 `phar` 反序列化漏洞还是比较好找的，因为一般来说后台都是能够上传 `jpg` 格式的文件，能够触发 `phar` 的函数也蛮多的。