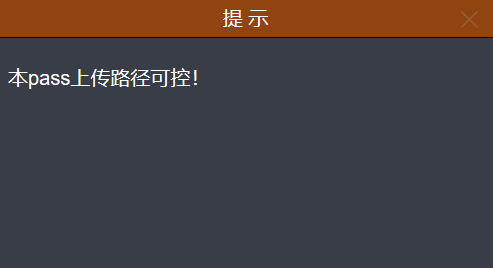
注意：在上一篇的最后有一个知识点没有提到的就是，关于00截断的利用条件。

1. 需满足 php 版本<5.3.4
2. php.ini中的magic\_quotes\_gpc是off状态的。(至于为什么下面会进行演示)

Pass12

打开第十二关老规矩先看提示



提示跟上一关一样是上传路径可控？？？怎么回事，难道两关考的考点是一样的吗？赶紧看看源码

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

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

$file\_ext = substr($\_FILES['upload\_file']['name'],strrpos($\_FILES['upload\_file']['name'],".")+1);

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

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

$img\_path = $\_POST['save\_path']."/".rand(10, 99).date("YmdHis").".".$file\_ext;

if(move\_uploaded\_file($temp\_file,$img\_path)){

$is\_upload = true;

} else {

$msg = "上传失败";

}

} else {

$msg = "只允许上传.jpg|.png|.gif类型文件！";

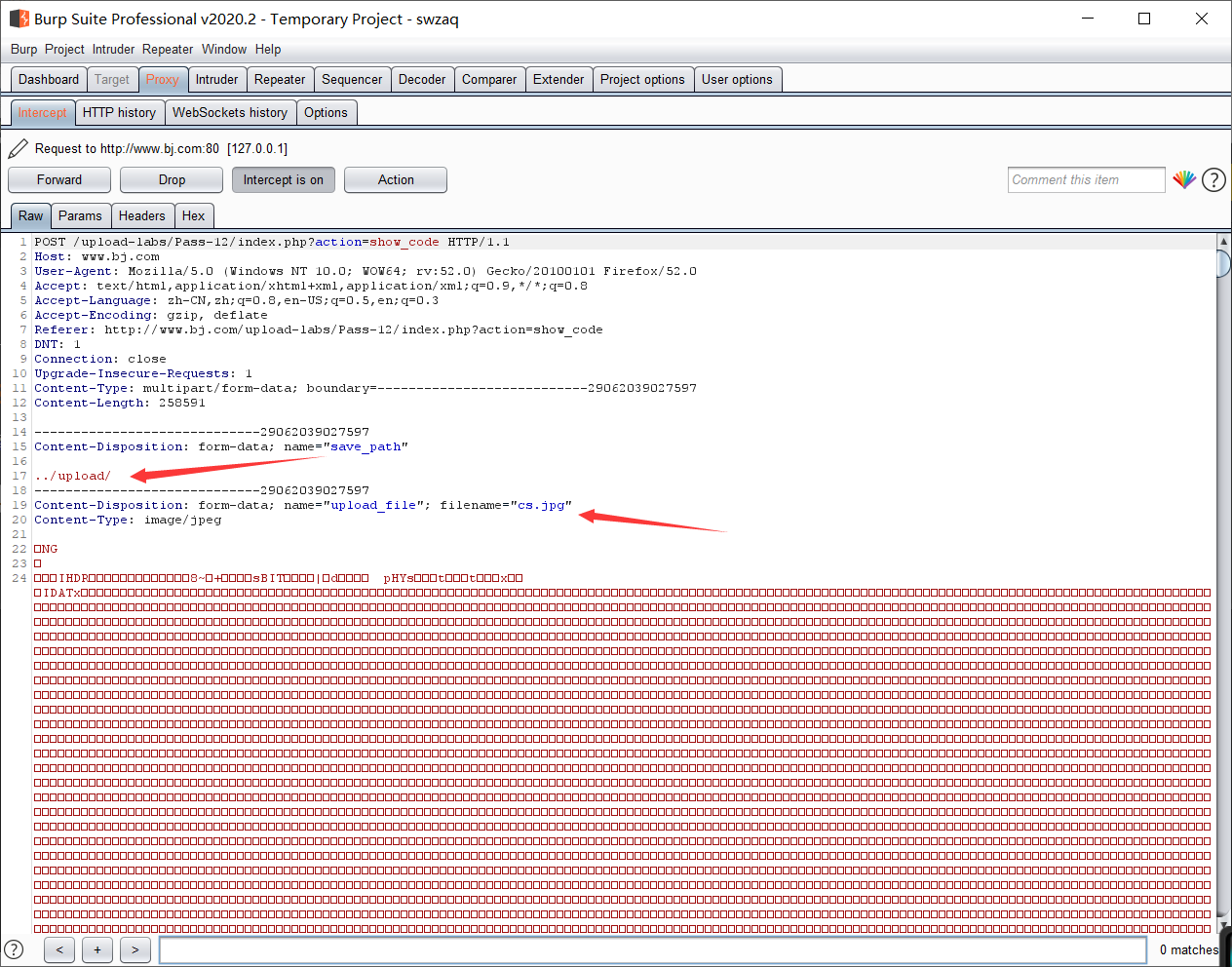
}}

通过源码对比发现，虽然都是上传路径可控，但是前面是用get方式获取路径参数的，而这里则是通过post方式来获取的。

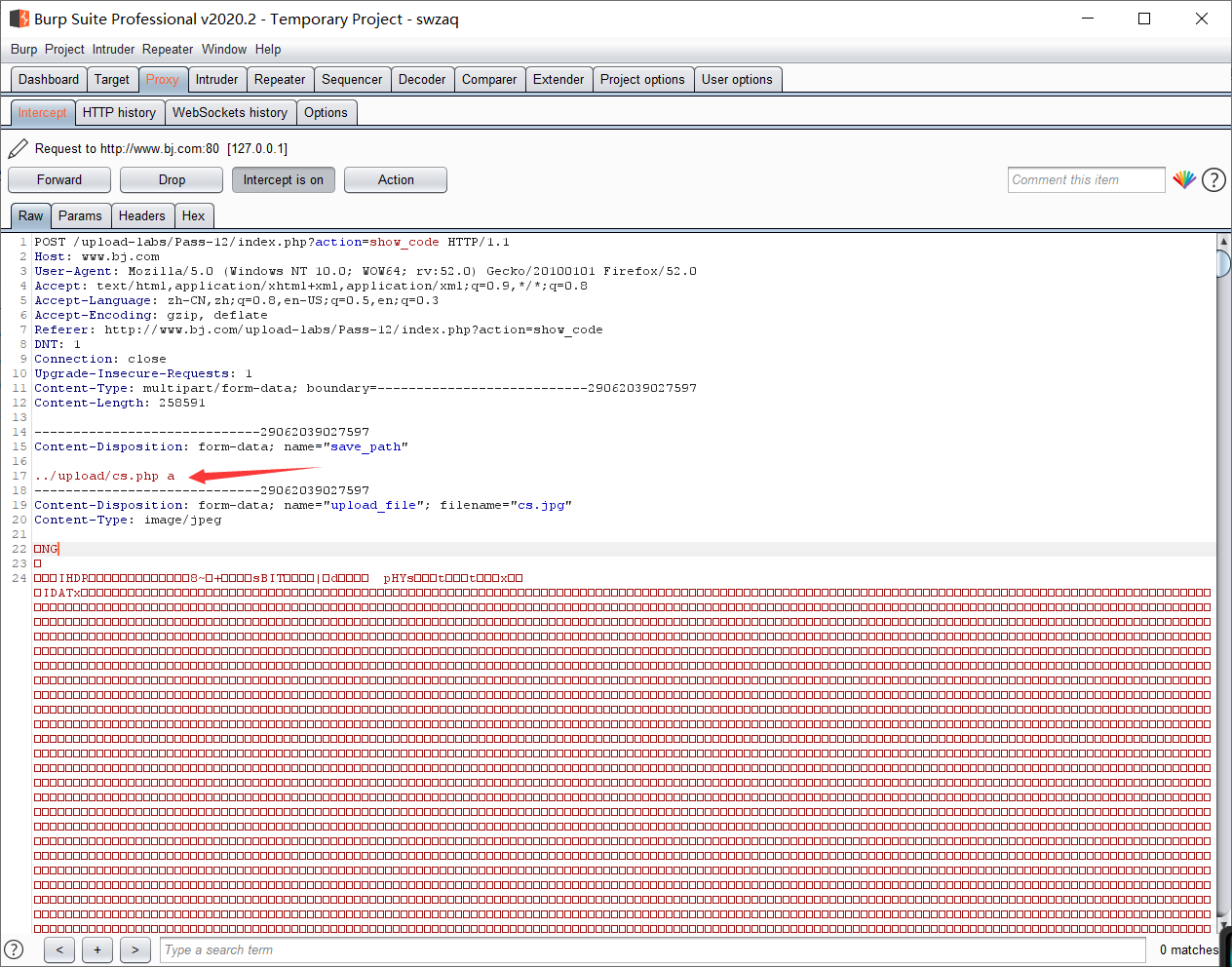
这里还是利用截断的方法去绕过过滤。不过需要注意的是两个截断有点区别，通过get方式是在url的参数中添加%00。这是因为%00通过get方式传递到服务器会被自动解码，所以就变成了ascii码中数值0对应的那个字符（null），这才能达到截断的目的。

但是如果是通过post的话，加入%00不会别服务器解码，只会原样被参数变量接收。所以这里要通过修改post数据的十六进制来达到截断的目的。

第一步：上传shell，抓包改数据

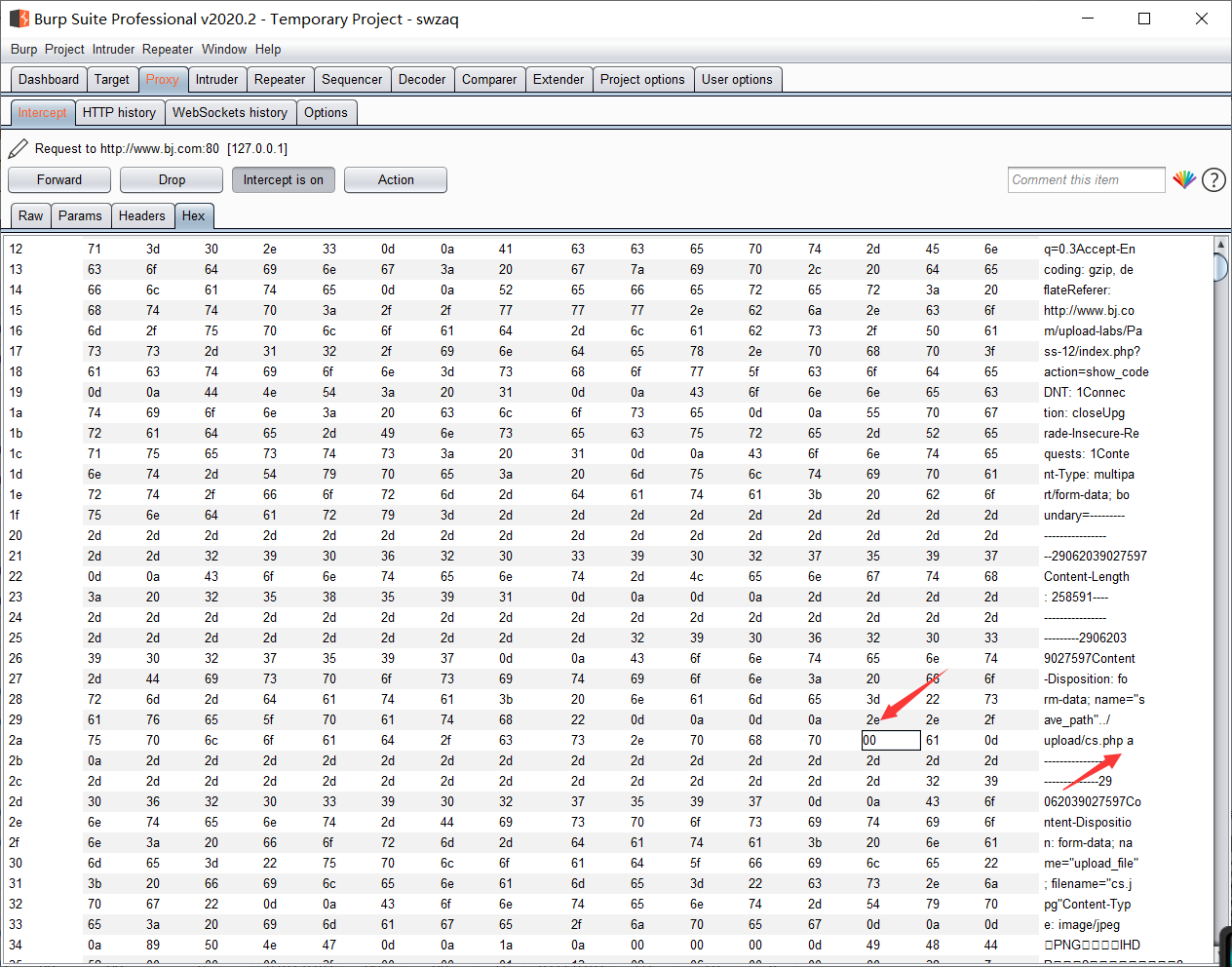


在路径参数后添加cs.php a

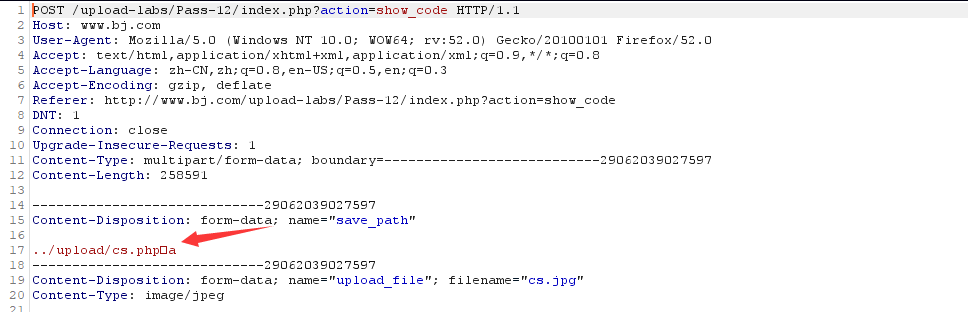


空格之后添加a的原因是这样在十六进制中比较好定位，a的十六进制是61



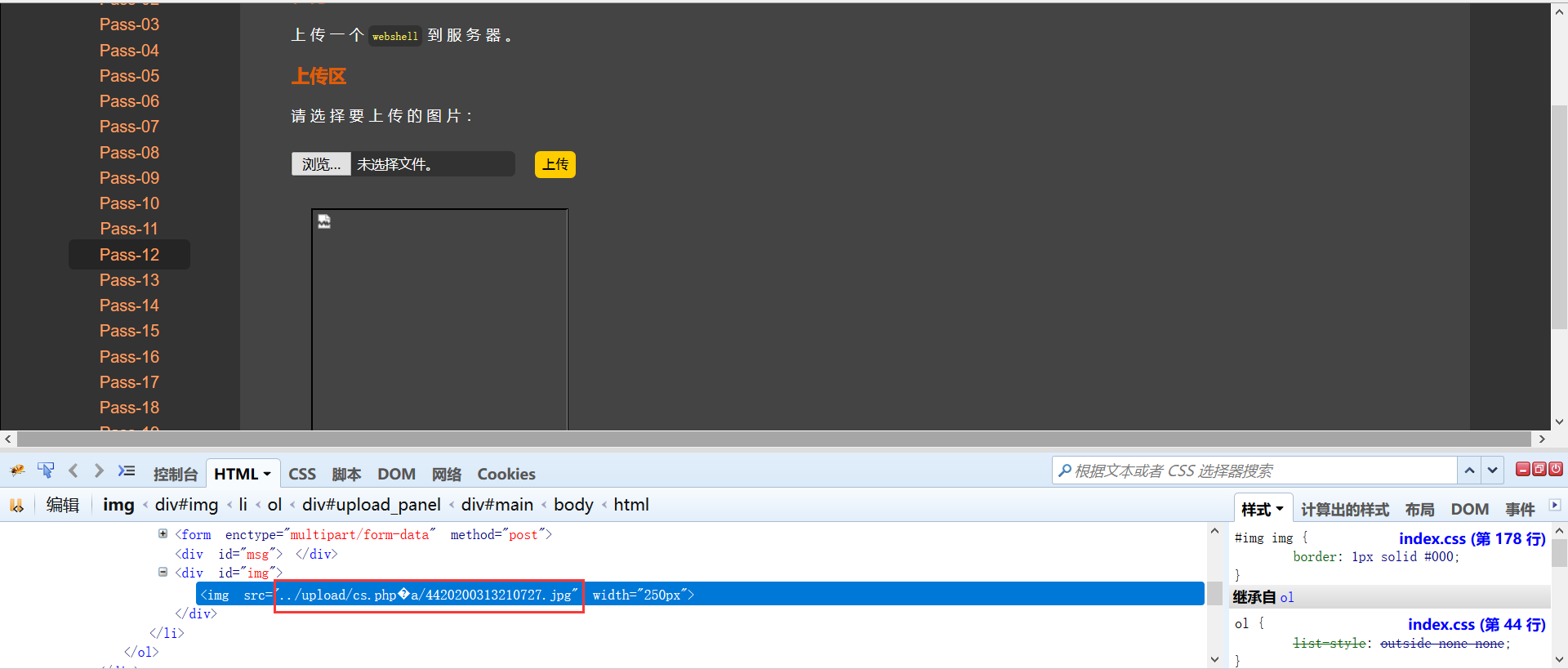


观察数据包很快就定位到了61的位置，前面就是空格的十六进制了，将代表空格的十六进制位数值改为00即可

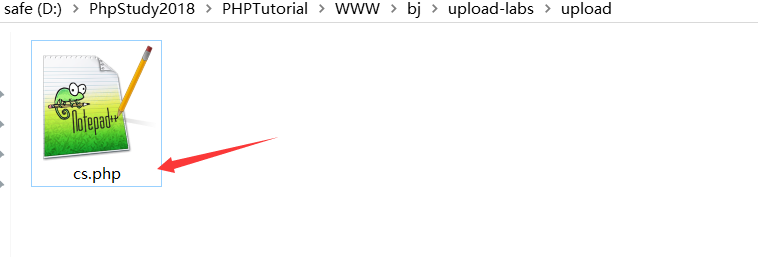


改完之后返回查看中间的空格变成了一个小方块，这样就完成了00截断的初步工作，可以放行数据包了

第二步：查看页面回显和服务器文件格式



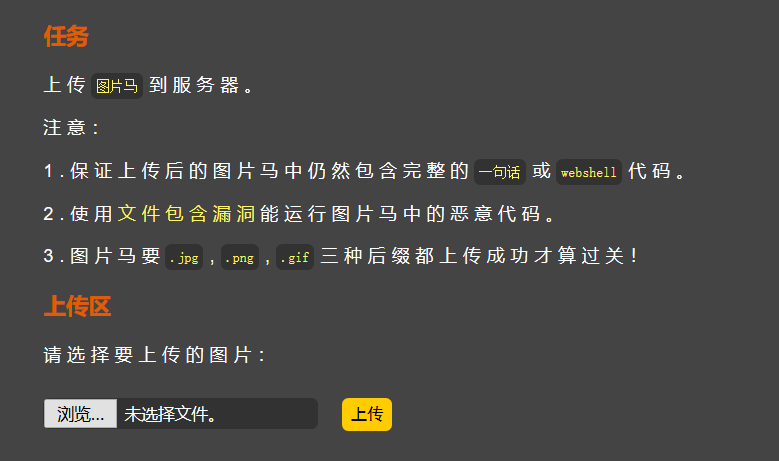
上传成功返回的文件位置和格式是这样的，不过按照我们的预期应该是系统创建文件时读取到cs.php就会被截断，所以服务器上的文件应该是cs.php才对啊，让我们看看服务器上是怎么样的



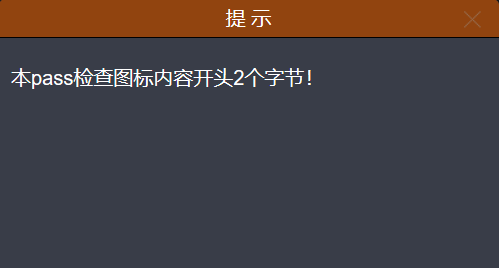
果不其然，00截断还是成功了。

Pass13

打开第十三关，直接页面内容都不一样了。难度再次升级？



看这意思是要上传图片马，还要用到文件包含啊。先看看提示



看来直接改shell后缀是不行了，这里还要检查文件前面的2个字节呐。再看看源码吧

function getReailFileType($filename){

$file = fopen($filename, "rb");

$bin = fread($file, 2); //只读2字节

fclose($file);

$strInfo = @unpack("C2chars", $bin);

$typeCode = intval($strInfo['chars1'].$strInfo['chars2']);

$fileType = '';

switch($typeCode){

case 255216:

$fileType = 'jpg';

break;

case 13780:

$fileType = 'png';

break;

case 7173:

$fileType = 'gif';

break;

default:

$fileType = 'unknown';

}

return $fileType;}

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

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

$file\_type = getReailFileType($temp\_file);

if($file\_type == 'unknown'){

$msg = "文件未知，上传失败！";

}else{

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

if(move\_uploaded\_file($temp\_file,$img\_path)){

$is\_upload = true;

} else {

$msg = "上传出错！";

}

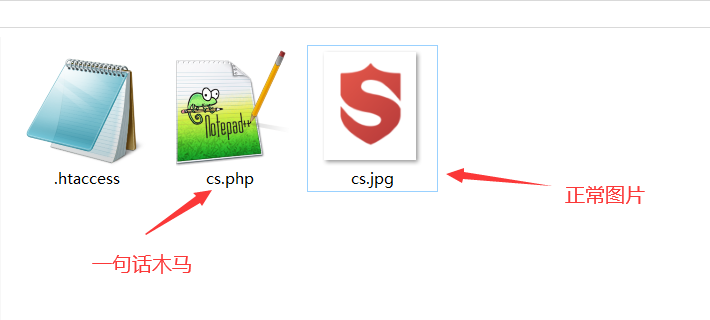
}}

看来确实是检测文件头部2个字节信息啊，一般的文件前两个字节信息都是表明自己文件类型的。这里直接改后缀看来是真不成了，所以要靠图片马了。

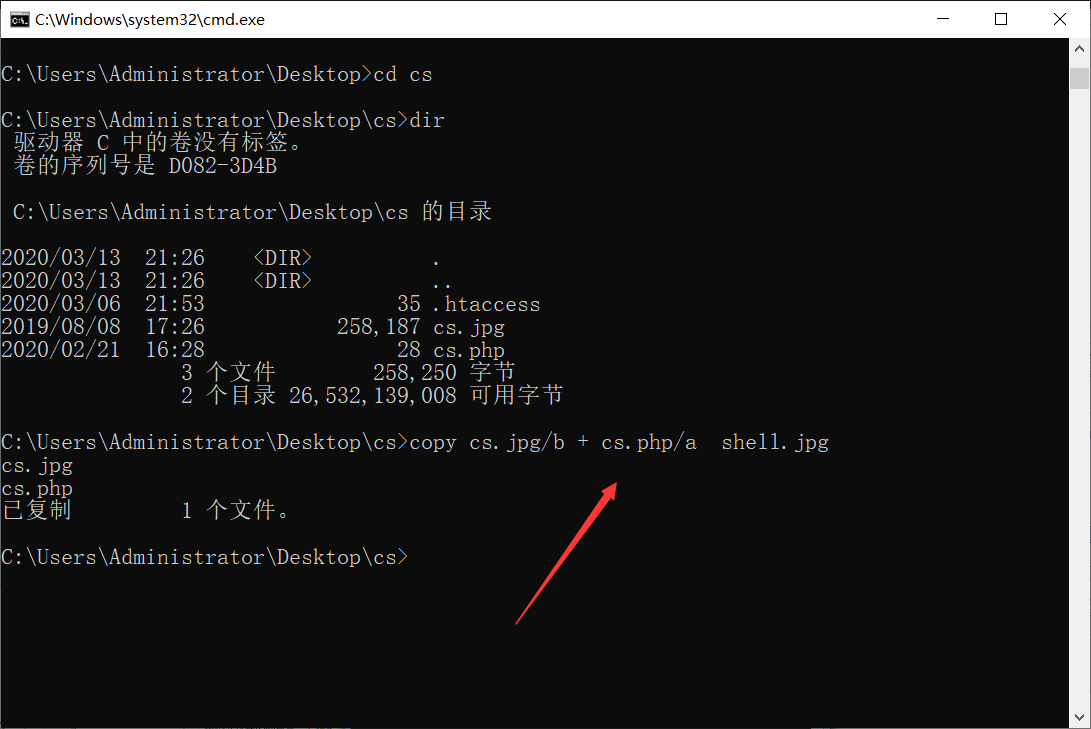
所谓的图片马就是把一张正常的图片和一个shell文件合在一起形成一个新的文件，可以把新的文件定义成图片。

第一步：生成图片马

我在本地准备了一张正常图片和一个一句话文件

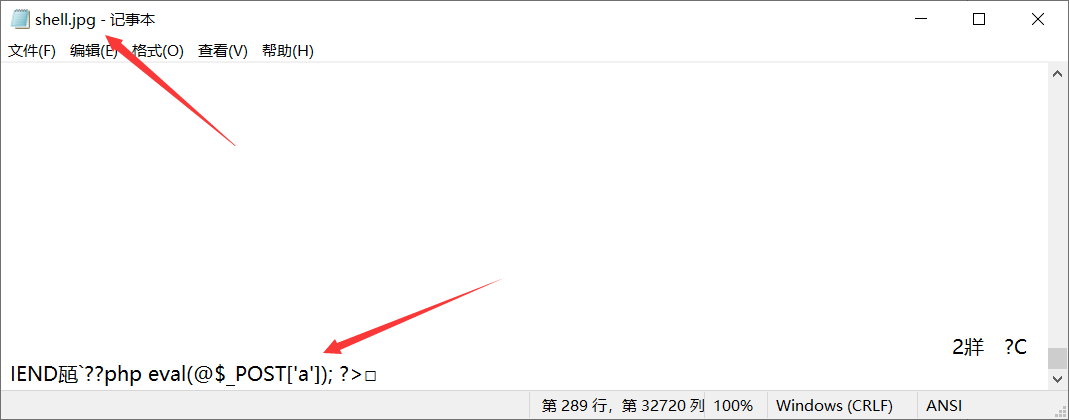


接下来利用cmd生成图片马



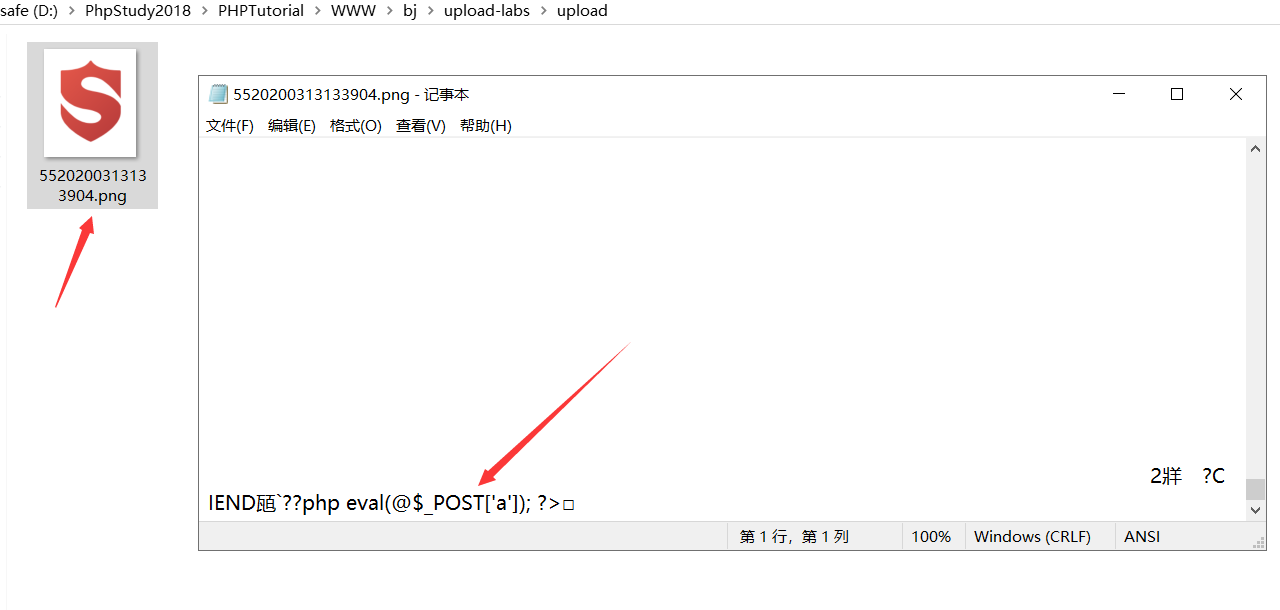
解释:参数/b指定以二进制格式复制、合并文件(图片)，参数/a指定以ASCI1格式复制、合并文件( php文件)

用记事本打开shell.jpg



可以看到一句话确实插入到图片末尾了（这里前面的括号显示乱码是编码问题）。这样直接上传的话，服务器检查文件头查出来就只会是jpg文件。

第二步：直接上传图片马



这里虽然上传了包含一句话的图片马，但是一张图片又不能解析成php执行，除非是配置问题。任务刚开始说可以利用文件包含漏洞，这是个不错的主意。被包含的文件中如果含有php代码则会被解析的。但是靶场自身是没有文件包含漏洞页面的，这里需要我们自己新建一个来模拟真实环境。

第三步：新建文件包含漏洞页面

在上传目录中新建一个include.php文件，写入如下代码

<?php

$file = $\_ GET[ 'page' ] ;

include ($file);

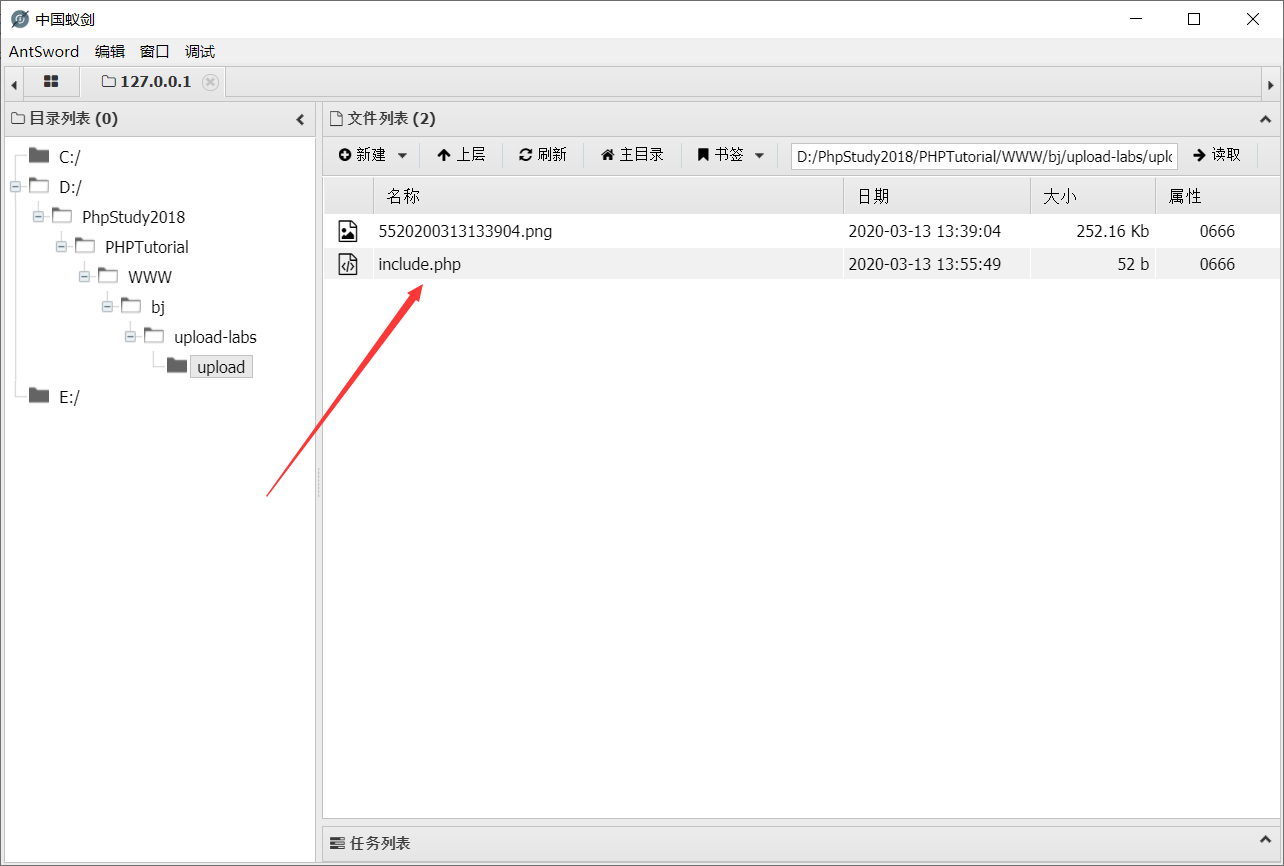
?>

Tip:这里环境要php5.3以上，不然运行文件包含漏洞会出错。

第四步：利用文件包含解析shell

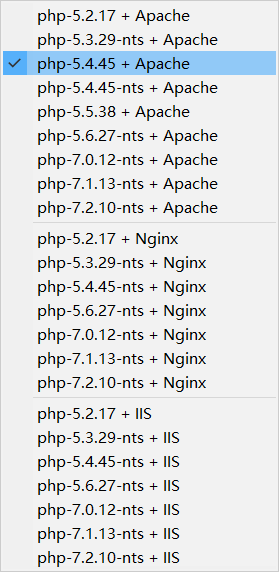
如果图片马中包含的是phpinfo()之类的可以直接浏览器访问，如果是一句话可以直接用蚁剑连接



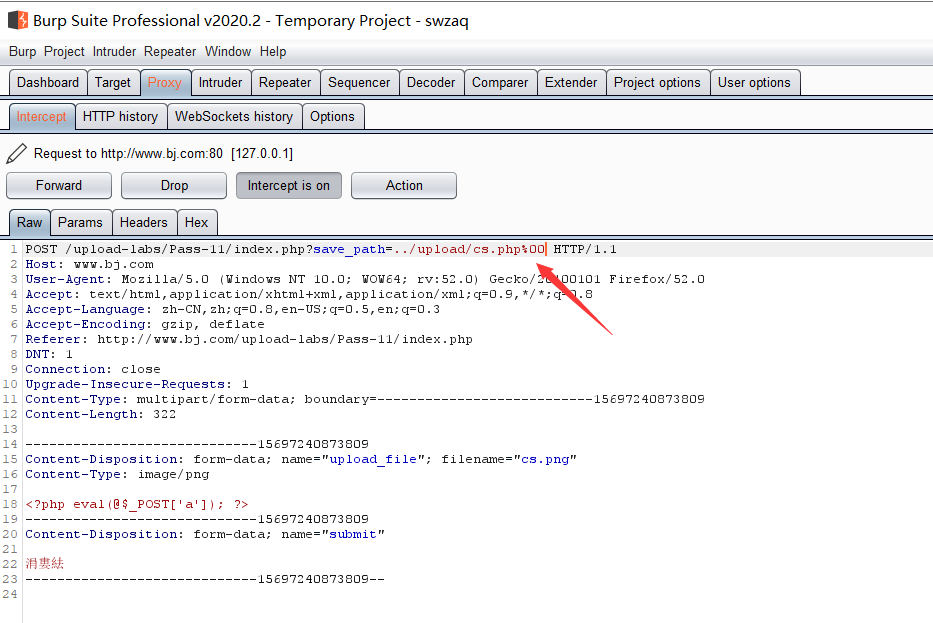


文章的最后来演示一下开头提到的00截断条件问题

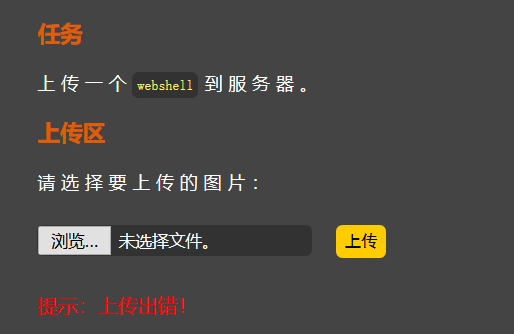
1. php版本>5.3.4



我选择了5.4.45版本。

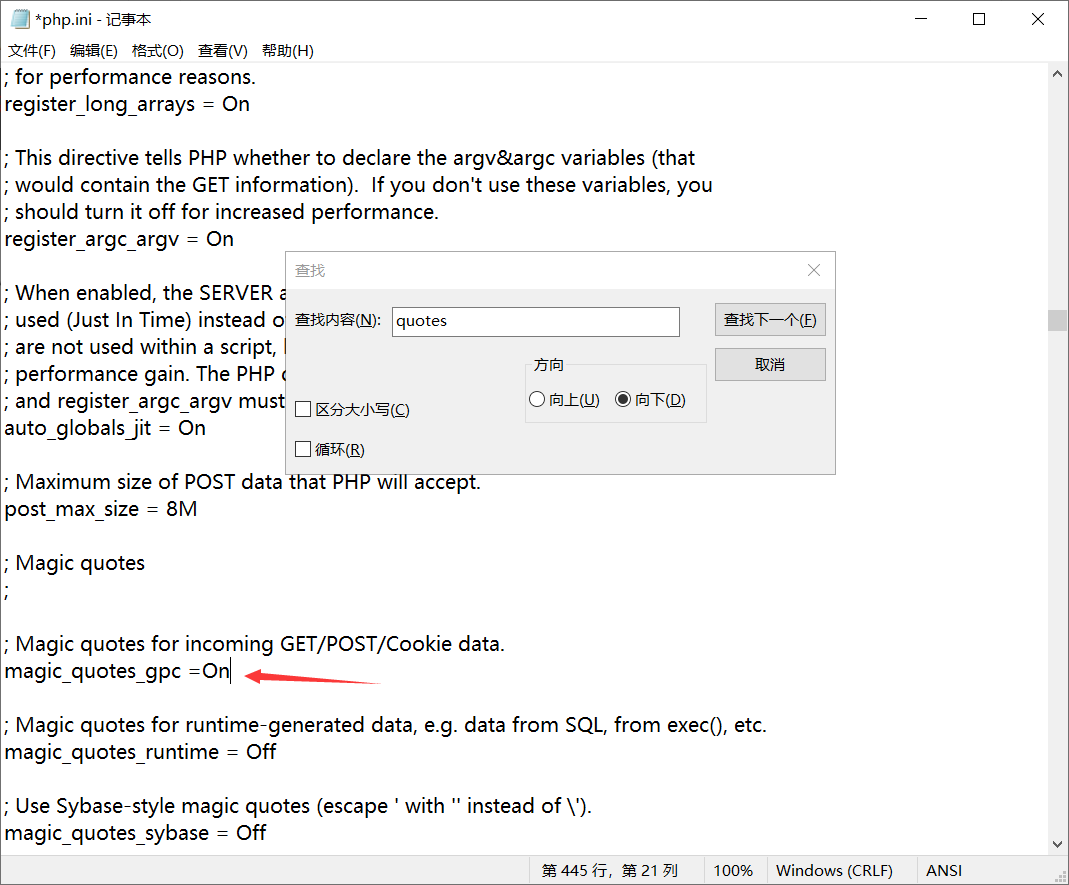


虽然截断操作正常，但是上传是失败的



1. magic\_quotes\_gpc=on

将php版本换成5.3.4之前的，再查看magic\_quotes\_gpc的状态，如果关闭了就将其开启。



但是截断之后仍然是上传失败的。

原因是：当打开magic\_quotes\_gpc时，所有的 '（单引号），"（双引号），\（反斜线）和 NULL字符（%00）都会被自动加上一个反斜杠进行转义。