




CTF题目中的上传绕过姿势

原创

神林、 于 2018-12-03 21:35:36 发布  8795  收藏 5

分类专栏: [CTF](#) 文章标签: [CTF](#)

版权声明: 本文为博主原创文章, 遵循[CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/qq_41079177/article/details/84780056

版权



[CTF 专栏收录该内容](#)

24 篇文章 2 订阅

订阅专栏

常用的上传绕过方式总结

平时网站上有很多可以本地上传的选项, 有一些网站的防护性不好, 如果我们上传了一串恶意的代码的话, 就会对网站造成一些损害, 尤其是上传一句话木马利用菜刀就可以做到链接目标的后台数据库。

0x01: 前端的验证

对于很多页面是利用一段js的脚本实现对后缀名的过滤的, 一般这种bp还没有抓到包的时候前端页面就会发出警告, 对于这样的页面我们可以利用对前端的js代码进行修改从而绕过前端的验证。可以修改成自己的php类型等等。。



0x02: 服务器端MIME类型检测绕过

如果文件上传之后却显示不符合规格, 那么有可能是服务器端的MIME类型检测当用户上传文件到服务器端的时候, 服务器端的程序会获取上传文件的MIME类型, 然后用这个获取到的类型来和期望的MIME类型进行匹配, 如果匹配不上则说明上传的文件不合法。服务端检测MIME类型的代码如下:

```
if (($FILES['upload_file']['type'] == 'image/jpeg') || ($FILES['upload_file']['type'] == 'image/png') || ($FILES['upload_file']['type'] == 'image/gif')){  
    ...//判断过后对文件处理的进一步操作  
}
```

绕过方法:

因为服务端检测的是文件的MIME类型, 而对这个MIME类型的值的获取是通过HTTP请求字段里的Content-Type字段, 所以绕过的方法就是通过修改Content-Type的值, 比如修改为image/jpeg; image/png; image/gif等等允许上传类型对应的MIME值



0x03:黑名单绕过

检测原理

文件类型根据黑名单来检测的原理就是：服务器程序根据一份文件后缀名的名单来判断是否允许当前文件上传到服务器，只要上传的文件类型能够和这个黑名单里面的类型匹配，那么就禁止该文件上传



绕过方法：

黑名单的安全性其实还没白名单的安全性高，至少攻击它的方式比白名单多多了

一般有个专门的 blacklist 文件，里面会包含常见的危险脚本文件 例如 fckeditor 2.4.3 或之前版本的黑名单



1. 找黑名单扩展名的漏网之鱼 - 比如上面就漏掉了asa和cer之类

2. 可能存在大小写绕过漏洞 - 比如 aSp 和 pHp 之类

3. 特别文件名构造 - 比如发送的http包里把文件名改成 help.asp. 或 help.asp_(下划线为空格)，这种命名方式在 windows 系统里是不被允许的，所以需要在 burp 之类里进行修改，然后绕过验证后，会被 windows 系统自动去掉后面的点和空格。

4. IIS 或 nginx 文件名解析漏洞 - 比如 help.asp;.jpg 或http://www.xx.com/help.jpg/2.php

这里注意网上所谓的 nginx 文件名解析漏洞实际上是 php-fpm 文件名解析漏洞 详见 <http://www.cnbeta.com/articles/111752.htm>

5. 0x00 截断绕过 - 这个是基于一个组合逻辑漏洞造成的

给个简单的伪代码

```
name = getname(http request) //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
```

7. 双扩展名解析绕过攻击(1) - 基于 web 服务的解析逻辑 比如在 Apache manual 中有这样一段描述

“Files can have more than one extension, and the order of the extensions is normally irrelevant. For example, if the file [welcome.html.fr](#) maps onto content type text/html and language French then the file welcome.fr.html will map onto exactly the same information. If more than one extension is given which maps onto the same type of meta-information, then the one to the right will be used, except for languages and content encodings. For example, if .gif maps to the MIME-type image/gif and .html maps to the MIME-type text/html, then the file welcome.gif.html will be associated with the MIME-type text/html.”

如果上传一个文件名为 help.asp.123

首先扩展名123并没有在扩展名 blacklist 里，然后扩展名 123 也没在 Apache 可解析扩展名 list 里，这个时候它会向前搜寻下一个可解析扩展名，或搜寻到.php，最后会以 php 执行

8. 双扩展名解析绕过攻击(2) - 基于web服务的解析方式 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



9. 危险解析绕过攻击 - 基于 web 服务的解析方式 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



白名单检测

白名单相对来说比黑名单安全一些，但也不见得就绝对安全了

1. 特别文件名构造

2. IIS或nginx文件名解析漏洞

3. 0x00 截断绕过

.htaccess 文件攻击

无论是黑名单还是白名单 再直接点就是直接攻击.htaccess 文件 在 PHP manual 中提到了下面一段话

move_uploaded_file section, there is a warning which states

'If the destination file already exists, it will be overwritten.' 如果 PHP 安全没配置好

就可以通过 move_uploaded_file 函数把自己写的.htaccess 文件覆盖掉服务上的

这样就能任意定义解析名单了

0x04 服务端验证绕过(文件完整性检测)

文件头检测

主要是在文件内容开始设置好图片文件的幻数

要绕过 jpg 文件检测就要在文件开头写上下图的值

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	FF	D8	FF	E0	00	10	4A	46	49	46	00	01	01	00	00	01	ÿØÿà JFIF

要绕过 gif 文件检测就要在文件开头写上下图的值

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	FF	D8	FF	E0	00	10	4A	46	49	46	00	01	01	00	00	01	ÿØÿà JFIF

要绕过 png 文件检测就要在文件开头写上下面的值

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	PNG

然后在文件头后面加上自己的一句话木马就行了

图像大小及相关信息检测

常用的就是 `getimagesize()`函数

只需要把文件头部分伪造好就 ok 了，就是在幻数的基础上还加了一些文件信息

有点像下面的结构

```
GIF89a(...some binary data...)<?php phpinfo(); ?>(... skipping the rest of binary data ...)
```

文件加载检测

这个是最变态的检测了，一般是调用的 API 或函数去进行文件加载测试常见的是图像渲染测试，再变态点的甚至是进行二次渲染(后面会提到)对它的攻击一般就两种方式，一个是渲染测试绕过，另一个是攻击文件加载器自身

渲染测试绕过

先用 GIMP 对一张图片进行代码注入

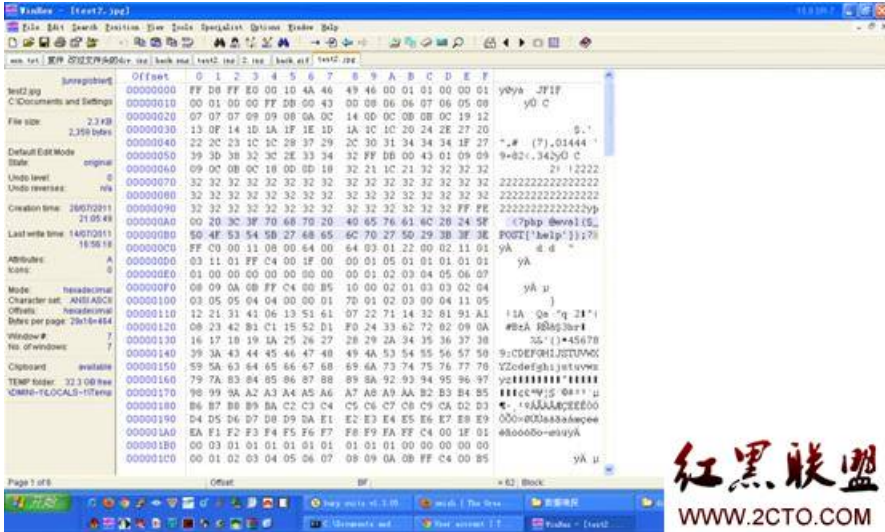
用 winhex 看数据可以分析出这类工具的原理是在不破坏文件本身的渲染情况下找一个空白区进行填充代码 一般是图片的注释区

对于渲染测试基本上都能绕过



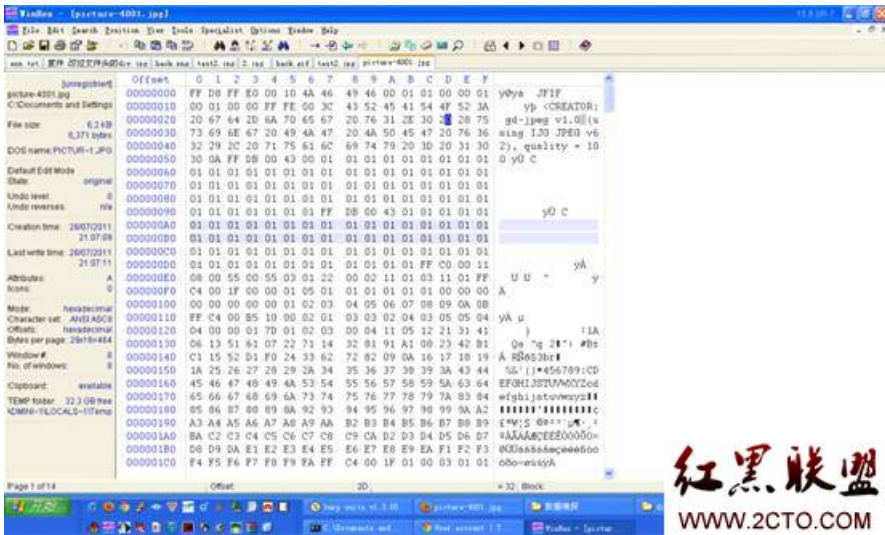
但如果碰到变态的二次渲染 基本上就没法绕过了，估计就只能对文件加载器进行攻击了

比如上传文件前，文件的数据如下



红黑联盟
WWW.2CTO.COM

然后上传这个 jpg 但把它重新下载回本地发现了奇怪的地方



红黑联盟
WWW.2CTO.COM

上传后图片被二次渲染过

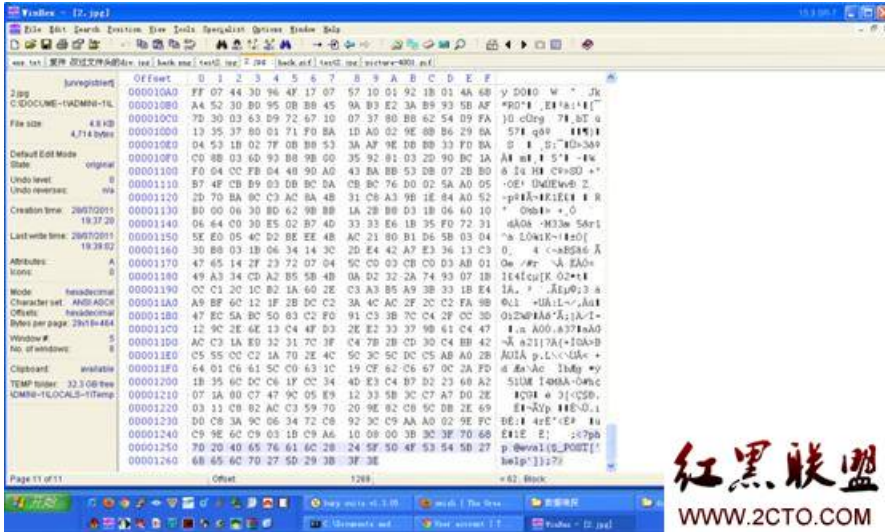
新的 JPG 图片内容里含有这个

CREATOR: gd-jpeg v1.0 (using IJG JPEG v62)

貌似是调用的 GD php 的 gd 库

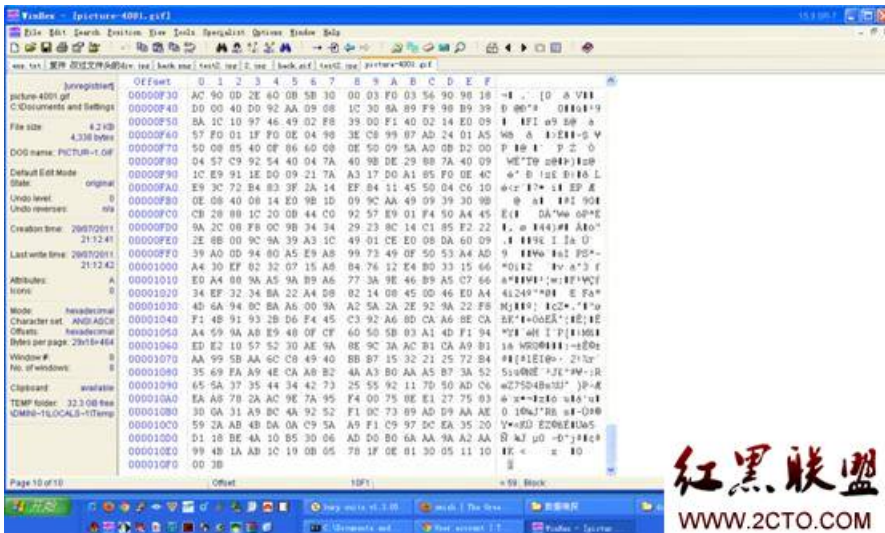
测试了 gif 文件也一样

原文件内容是（虽然文件名是 2.jpg，实际文件格式是 gif 哈）



红黑联盟
WWW.2CTO.COM

上传后下载回来对比 可以发现文件被重新渲染过 一句话代码也不见了



红黑联盟
WWW.2CTO.COM

然后是进行报错触发 看下是被用什么 API 或函数进行的二次渲染

上传文件数据不完整的 gif 文件 触发报错后, 知道后台用的是 imagecreatefromgif() 这个函数



上传文件数据不完整的 png 文件 触发报错后, 知道后台用的是 imagecreatefrompng() 这个函数



一般进行过二次渲染 再想绕过个人经验是几乎不可能了 它相当于是把原本属于图像数据的部分抓了出来, 再用自己的 API 或函数进行重新渲染 在这个过程中非图像数据的部分直接就隔离开了

如果要对文件加载器进行攻击，常见的就是溢出攻击，上传自己的恶意文件后，服务上的文件加载器进行加载测试时，被触发攻击执行 shellcode 比如 access/mdb 溢出

大家可以参考下 <http://lcx.cc/?FoxNews=1542.html>

0x05 各种情况下的检测绕过分析

A 客户端验证绕过(javascript 扩展名检测)

用反向代理工具(burp 之类)或禁用 js 便可以绕过客户端验证

B 服务端验证绕过(http request 包检测)

- Content-type (Mime type) 检测 用反向代理工具(burp 之类)进行 Content-type 伪造

C 服务端验证绕过(扩展名检测)

- 黑名单检测

- 找黑名单扩展名的漏网之鱼 - 比如上面就漏掉了 asa 和 cer 之类 可能存在大小写绕过漏洞 - 比如 aSp 和 pHp 之类
- 特别文件名构造 - 比如发送的 http 包里把文件名改成 help.asp. 或 help.asp_(下划线为空格)
- IIS或nginx文件名解析漏洞 - 比如 help.asp;.jpg 或 <http://www.xx.com/help.jpg/2.php>
- 0x00 截断绕过 - 这个是基于一个组合逻辑漏洞造成的 双扩展名解析绕过攻击(1) - 基于 web 服务的解析逻辑
- 双扩展名解析绕过攻击(2) - 基于 web 服务的解析方式 危险解析绕过攻击 - 基于 web 服务的解析方式

- 白名单检测

- 特别文件名构造 (同黑名单攻击第 3 条)
- IIS 或 nginx 文件名解析漏洞 (同黑名单攻击第 4 条)
- 0x00 截断绕过 (同黑名单攻击第 5 条)

- .htaccess 文件攻击

在 PHP 安全没配置好的情况下，用自己的.htaccess 覆盖服务上原文件

D 服务端验证绕过(文件完整性检测)

- - 文件头检测 在文件开始伪装文件的幻数
- - 图像分辨率检测
- 在文件开始伪装图像大小数据
- - 文件加载检测 用工具对文件空白数据区或注释区进行代码注入绕过 (图像仅能绕过渲染测试，而不能绕过二次渲染)用恶意文件去攻击加载器本身

E 相互关系与组合情况

首先客户端验证和服务端验证是相互独立的，所以分开绕过就行了 主要难点是在服务端验证的组合上 文件完整性检测已经包含文件头检测和图像大小及相关信息检测，但不包含文件扩展名检测 它是以加载来作为检测的方式，比如用图像渲染函数去渲染一张图片 文件扩展名检测和文件头检测都是同级的，相互独立 所以如果是文件扩展名+文件头检测可以同时分开绕过

0x06 关于图像代码注入后的解析简答

我在自己本机搭环境测试过 环境为 Apache+PHP 发现其无法解析图像中的一句话

可能很多人都遇到过这类情况 并不真正清楚把代码注入图片之类后，怎么访问才能连接上一句话 这里就来解释下原理
其实就算对图片进行代码注入后，还需要调用对应的解析器来解析才能让代码得以解析执行 好比你把一个 exe 扩展改成 jpg，在桌面上打开，图像查看器会报错，说无法打开该文件
而你在 cmdshell 下，无论这个 exe 扩展名是什么，哪怕是 jpg，也能执行 因为只要调用到了正确的文件加载器或解析器，它是
以文件头幻数来判断文件格式的，而不是文件扩展名，所以我们如果只是简单把图片代码注入后，上传访问，这个时候，还并不能

解析里面的一句话代码

大家可以参考下:<http://www.2cto.com/Article/201106/93084.html>

常见的是结合 LocalFileInclude 漏洞来解析我们的图片
(RemoteFileInclude 和 RemoteCodeExecution 在这里就有点大才小用了哈)

比如某个站有这样一个 URL

www.2cto.com/view.php?page=contact.php

我们替换 contact.php 为.../

www.2cto.com/view.php?page=.../

得到一个报错

```
Warning: include(.../)[function.include]: failed to open stream: No such file or directory in  
/home/sirgod/public_html/2cto.com/view.php on line 1337
```

就说明存在 LFI 漏洞，这个时候找到我们的图片文件路径

用一句话 client 去连接 www.2cto.com/view.php?page=.../upload/help.jpg

就可以成功的得到 shell 了

还有像 nginx/php-fpm 解析漏洞，也可以直接解析图片里的代码 所以大家要了解哪些环境下才能对图片里的代码进行解析
这样才能结合上传绕过最后得到 webshell ?