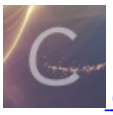
2020年「羊城杯」网络安全大赛 Re部分 WriteUp

原创

Simon菌 于 2020-09-11 19:15:32 发布 1383 收藏 5

分类专栏： CTF 逆向 python

本文链接：https://blog.csdn.net/qq_42436176/article/details/108540232


CTF 同时被 3 个专栏收录

5 篇文章 1 订阅

订阅专栏


逆向

3 篇文章 1 订阅

订阅专栏


python

7 篇文章 6 订阅

订阅专栏

Re

login

使用`PyInstaller Extractor v2.0`把exe进行解包, 在解包指令中发现`Python version: 36`, 切换到python3.6环境中

```
(venv) E:\Python-Project\pyctest\ppyycc>python pyinstxtractor.py 2009085f56dfbbbf571.exe
[+] Processing 2009085f56dfbbbf571.exe
[+] Pyinstaller version: 2.1+
[+] Python version: 36
[+] Length of package: 6021662 bytes
[+] Found 59 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: login.pyc
[+] Found 133 files in PYZ archive
[+] Successfully extracted pyinstaller archive: 2009085f56dfbbbf571.exe

You can now use a python decompiler on the pyc files within the extracted directory
```


可以看见入口文件为`login.pyc`, 使用`uncompyle6 login.pyc`解析pyc文件

```
(venv) E:\Python-Project\pyctest\ppyycc\2009085f56dfbbbf571.exe_extracted>uncompyle6 login.pyc
# uncompyle6 version 3.7.4
# Python bytecode 3.6 (3379)
# Decompiled from: Python 3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 03:37:03) [MSC v.1900 64 bit (AMD64)]
# Embedded file name: login.py
import sys
input1 = input('input something:')
if len(input1) != 14:
    print('Wrong length!')
    sys.exit()
else:
    code = []
    for i in range(13):
        code.append(ord(input1[i]) ^ ord(input1[(i + 1)]))
```


看到一堆数字, 嗯, 又是方程组, 上z3

```
from z3 import *
a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12, a13, a14 = Ints("a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 a12 a13 a
14")
x = Solver()
x.add(a1 * 88 + a2 * 67 + a3 * 65 - a4 * 5 + a5 * 43 + a6 * 89 + a7 * 25 + a8 * 13 - a9 * 36 + a10 * 15 + a11 *
11 + a12 * 47 - a13 * 60 + a14 * 29 == 22748)
x.add(a1 * 89 + a2 * 7 + a3 * 12 - a4 * 25 + a5 * 41 + a6 * 23 + a7 * 20 - a8 * 66 + a9 * 31 + a10 * 8 + a11 * 2
 - a12 * 41 - a13 * 39 + a14 * 17 == 7258)
x.add(a1 * 28 + a2 * 35 + a3 * 16 - a4 * 65 + a5 * 53 + a6 * 39 + a7 * 27 + a8 * 15 - a9 * 33 + a10 * 13 + a11 *
 101 + a12 * 90 - a13 * 34 + a14 * 23 == 26190)
x.add(a1 * 23 + a2 * 34 + a3 * 35 - a4 * 59 + a5 * 49 + a6 * 81 + a7 * 25 + a8 * 128 - a9 * 32 + a10 * 75 + a11
* 81 + a12 * 47 - a13 * 60 + a14 * 29 == 37136)
x.add(a1 * 38 + a2 * 97 + a3 * 35 - a4 * 52 + a5 * 42 + a6 * 79 + a7 * 90 + a8 * 23 - a9 * 36 + a10 * 57 + a11 *
 81 + a12 * 42 - a13 * 62 - a14 * 11 == 27915)
x.add(a1 * 22 + a2 * 27 + a3 * 35 - a4 * 45 + a5 * 47 + a6 * 49 + a7 * 29 + a8 * 18 - a9 * 26 + a10 * 35 + a11 *
 41 + a12 * 40 - a13 * 61 + a14 * 28 == 17298)
x.add(a1 * 12 + a2 * 45 + a3 * 35 - a4 * 9 - a5 * 42 + a6 * 86 + a7 * 23 + a8 * 85 - a9 * 47 + a10 * 34 + a11 *
76 + a12 * 43 - a13 * 44 + a14 * 65 == 19875)
x.add(a1 * 79 + a2 * 62 + a3 * 35 - a4 * 85 + a5 * 33 + a6 * 79 + a7 * 86 + a8 * 14 - a9 * 30 + a10 * 25 + a11 *
 11 + a12 * 57 - a13 * 50 - a14 * 9 == 22784)
x.add(a1 * 8 + a2 * 6 + a3 * 64 - a4 * 85 + a5 * 73 + a6 * 29 + a7 * 2 + a8 * 23 - a9 * 36 + a10 * 5 + a11 * 2 +
 a12 * 47 - a13 * 64 + a14 * 27 == 9710)
x.add(a1 * 67 - a2 * 68 + a3 * 68 - a4 * 51 - a5 * 43 + a6 * 81 + a7 * 22 - a8 * 12 - a9 * 38 + a10 * 75 + a11 *
 41 + a12 * 27 - a13 * 52 + a14 * 31 == 13376)
x.add(a1 * 85 + a2 * 63 + a3 * 5 - a4 * 51 + a5 * 44 + a6 * 36 + a7 * 28 + a8 * 15 - a9 * 6 + a10 * 45 + a11 * 3
1 + a12 * 7 - a13 * 67 + a14 * 78 == 24065)
x.add(a1 * 47 + a2 * 64 + a3 * 66 - a4 * 5 + a5 * 43 + a6 * 112 + a7 * 25 + a8 * 13 - a9 * 35 + a10 * 95 + a11 *
 21 + a12 * 43 - a13 * 61 + a14 * 20 == 27687)
x.add(a1 * 89 + a2 * 67 + a3 * 85 - a4 * 25 + a5 * 49 + a6 * 89 + a7 * 23 + a8 * 56 - a9 * 92 + a10 * 14 + a11 *
 89 + a12 * 47 - a13 * 61 - a14 * 29 == 29250)
x.add(a1 * 95 + a2 * 34 + a3 * 62 - a4 * 9 - a5 * 43 + a6 * 83 + a7 * 25 + a8 * 12 - a9 * 36 + a10 * 16 + a11 *
51 + a12 * 47 - a13 * 60 - a14 * 24 == 15317)

print(x.check())
print(x.model())

>>> [a2 = 24,
 a13 = 88,
 a6 = 43,
 a9 = 52,
 a14 = 33,
 a5 = 104,
 a12 = 74,
 a7 = 28,
 a1 = 119,
 a10 = 108,
 a11 = 88,
 a8 = 91,
 a4 = 7,
 a3 = 10]
```

按照`ord(input1[i]) ^ ord(input1[i + 1])`进行异或, 反推回input`[85, 95, 71, 48, 55, 95, 116, 104, 51, 95, 107, 51, 121, 33]`, 转换为字符串

```
c = [85, 95, 71, 48, 55, 95, 116, 104, 51, 95, 107, 51, 121, 33]
for i in c:
    print(chr(i), end="")
>>> U_G07_th3_k3y!
```

easyre

IDA查看整体逻辑, 发现是flag经过`encode_one` `encode_two` `encode_three`三次编码后变成`EmBmP5Pmn7QcPU4gLYKv5QcMmB3PWHcP5YkPq3=cT6QckkPckoRG`, 首先怀疑有base64

```
14  _main();
15  strcpy(Str2, "EmBmP5Pmn7QcPU4gLYKv5QcMmB3PWHcP5YkPq3=cT6QckkPckoRG");
16  puts("Hello, please input your flag and I will tell you whether it is right or not.");
17  scanf("%38s", &Str);
18  if ( strlen(&Str) == 38
19    && (v3 = strlen(&Str), (unsigned int)encode_one(&Str, v3, &v10, &v12) == 0)
20    && (v4 = strlen(&v10), (unsigned int)encode_two(&v10, v4, &v9, &v12) == 0)
21    && (v5 = strlen(&v9), (unsigned int)encode_three(&v9, v5, &Str1, &v12) == 0)
22    && !strcmp(&Str1, Str2) )
23  {
24    puts("you are right!");
25    result = 0;
26  }
27  else
28  {
29    printf("Something wrong. Keep going.");
30    result = 0;
31  }
32  return result;
33}
```

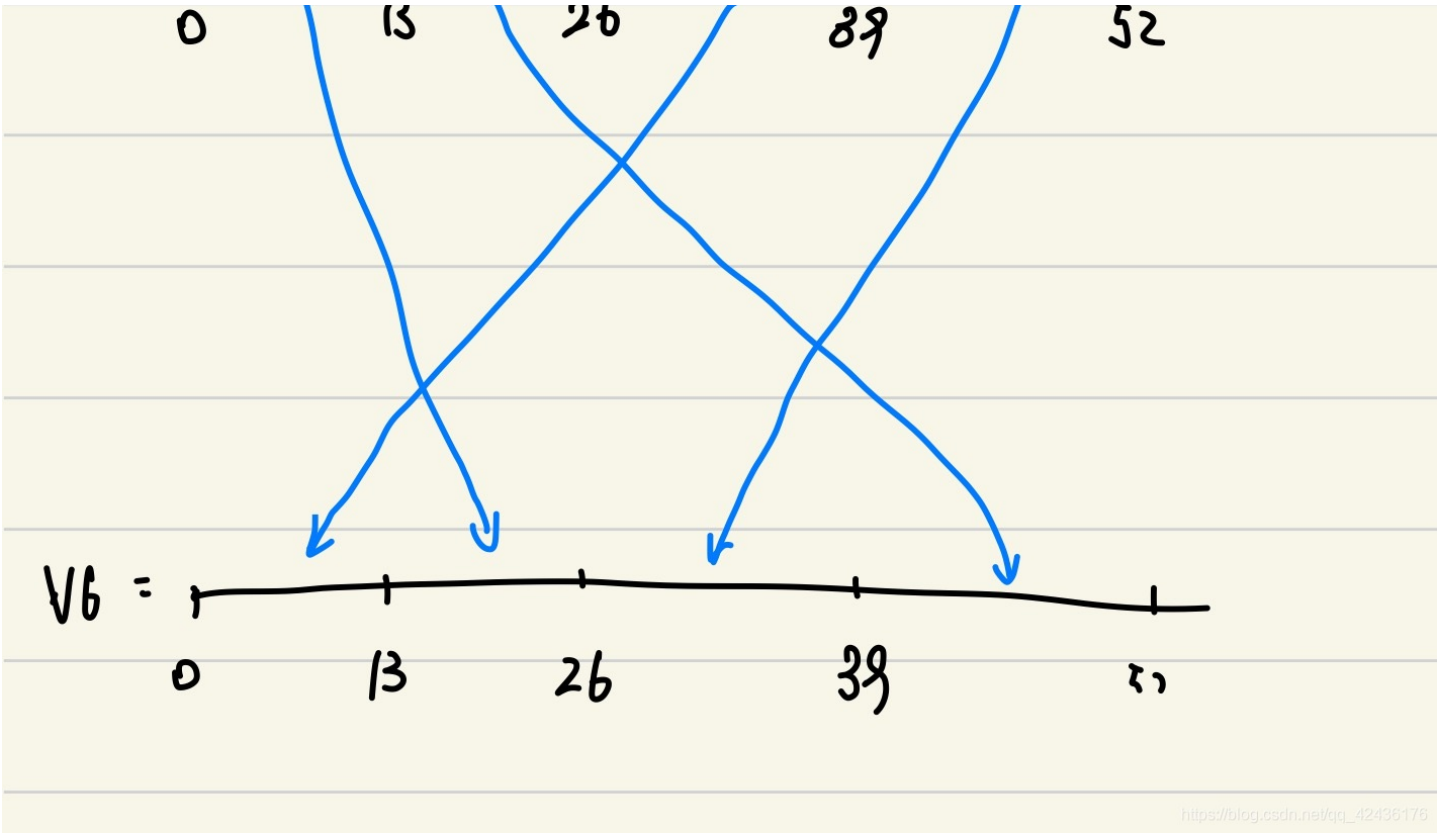

查看`encode_one`, 看到一个变量`alphabet`, 对应的`ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/`, 不用怀疑就是`base64`,

```
*v10 = alphabet[(char)*
if ( v14 + v11 - 3 == i
```

```
; _BYTE alphabet[64]
_ZL8alphabet    db 41h, 42h, 43h, 44h, 45h, 46h, 47h, 48h, 49h, 4Ah, 4Bh
                                        ; DATA XREF: encode_one(char const*,int,char *,int *)+E8↑o
                                        ; encode_one(char const*,int,char *,int *)+177↑o ...
                db 4Ch, 4Dh, 4Eh, 4Fh, 50h, 51h, 52h, 53h, 54h, 55h, 56h
                db 57h, 58h, 59h, 5Ah, 61h, 62h, 63h, 64h, 65h, 66h, 67h
                db 68h, 69h, 6Ah, 6Bh, 6Ch, 6Dh, 6Eh, 6Fh, 70h, 71h, 72h
                db 73h, 74h, 75h, 76h, 77h, 78h, 79h, 7Ah, 30h, 31h, 32h
                db 33h, 34h, 35h, 36h, 37h, 38h, 39h, 2Bh, 2Fh
aArgumentDomain db
```

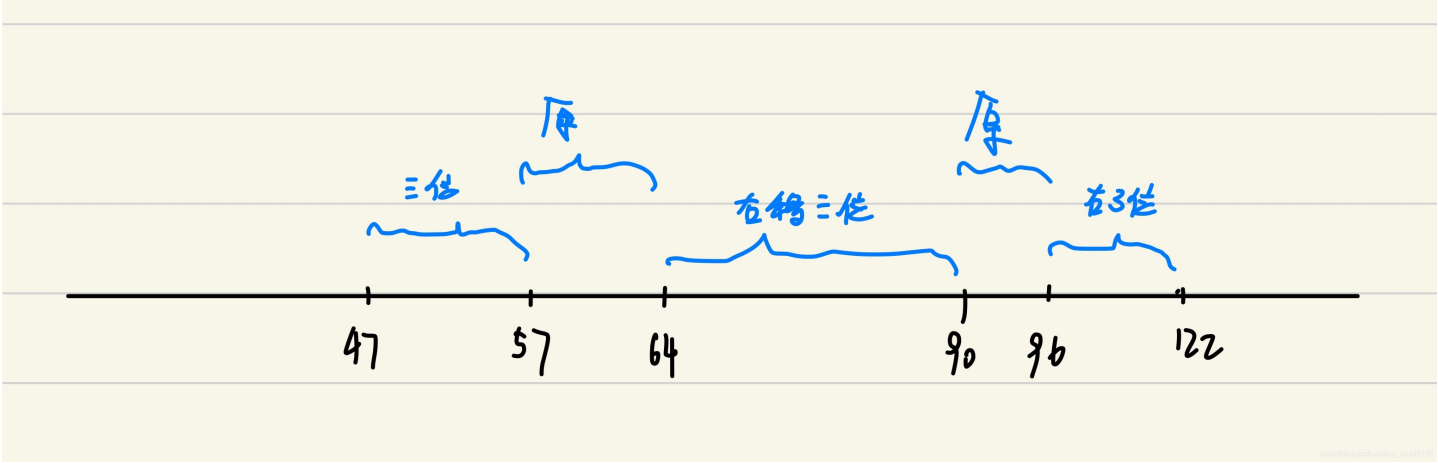

查看`encode_two`, 看到4个`strncpy`用法, 分析后可画出此图

```
6  Source = (char *)a1;
7  v6 = a3;
8  if ( !a1 || !a2 )
9    return 0xFFFFFFFFi64;
10 strncpy(a3, a1 + 26, 0xDui64);
11 strncpy(v6 + 13, Source, 0xDui64);
12 strncpy(v6 + 26, Source + 39, 0xDui64);
13 strncpy(v6 + 39, Source + 13, 0xDui64);
14 return 0i64;
15}
```

查看`encode_three`,貌似很复杂的样子,画出数轴,分析后画出此图

```
11  v7 = a3;
12  for ( i = 0; i < a2; ++i )
13  {
14    v5 = *v8;
15    if ( *v8 <= 64 || v5 > 90 )
16    {
17      if ( v5 <= 96 || v5 > 122 )
18      {
19        if ( v5 <= 47 || v5 > 57 )
20          *v7 = v5;
21        else
22          *v7 = (v5 - 48 + 3) % 10 + 48;
23      }
24      else
25      {
26        *v7 = (v5 - 97 + 3) % 26 + 97;
27      }
28    }
29    else
30    {
31      *v7 = (v5 - 65 + 3) % 26 + 65;
32    }
33    ++v7;
34    ++v8;
35  }
36  return 0i64;
37}
```



先还原`encode_three`,我突然想到这是一一对应的关系,把这个关系写出来然后生成一张map就行了

```
def m(v5):
    if v5 <= 64 or v5 > 90:
        if v5 <= 96 or v5 > 122:
            if v5 <= 47 or v5 > 57:
                return chr(v5)
            else:
                return chr((v5 - 45) % 10 + 48)
        else:
            return chr((v5 - 94) % 26 + 97)
    else:
        return chr((v5 - 62) % 26 + 65)



mm = {}
for i in range(33, 127):
    raw = i
    fin = m(i)
    mm[fin] = chr(raw)

data = "EmBmP5Pmn7QcPU4gLYKv5QcMmB3PWHcP5YkPq3=cT6QckkPckoRG"

print(mm)
fin2 = ""
for c in data:
    fin2 += mm[c]

print(fin2)

>>> {'!': '!', '"': '"', '#': '#', '$': '$', '%': '%', '&': '&', "'": "'", '(': '(', ')': ')', '*': '*', '+': '+
', ',': ',', '-': '-', '.': '.', '/': '/', '3': '0', '4': '1', '5': '2', '6': '3', '7': '4', '8': '5', '9': '6',
 '0': '7', '1': '8', '2': '9', ':': ':', ';': ';', '<': '<', '=': '=', '>': '>', '?': '?', '@': '@', 'D': 'A', '
E': 'B', 'F': 'C', 'G': 'D', 'H': 'E', 'I': 'F', 'J': 'G', 'K': 'H', 'L': 'I', 'M': 'J', 'N': 'K', 'O': 'L', 'P'
: 'M', 'Q': 'N', 'R': 'O', 'S': 'P', 'T': 'Q', 'U': 'R', 'V': 'S', 'W': 'T', 'X': 'U', 'Y': 'V', 'Z': 'W', 'A':
'X', 'B': 'Y', 'C': 'Z', '[': '[', '\\': '\\', ']': ']', '^': '^', '_': '_', '`': '`', 'd': 'a', 'e': 'b', 'f':
'c', 'g': 'd', 'h': 'e', 'i': 'f', 'j': 'g', 'k': 'h', 'l': 'i', 'm': 'j', 'n': 'k', 'o': 'l', 'p': 'm', 'q': 'n
', 'r': 'o', 's': 'p', 't': 'q', 'u': 'r', 'v': 's', 'w': 't', 'x': 'u', 'y': 'v', 'z': 'w', 'a': 'x', 'b': 'y',
 'c': 'z', '{': '{', '|': '|', '}': '}', '~': '~'}
>>> BjYjM2Mjk4NzMR1dIVHs2NzJjY0MTEzM2VhMn0=zQ3NzhhMzhlOD
```

还原`encode_two`,参照画的图对原来的字符串进行还原,最后base64解码

```
import base64
data = "BjYjM2Mjk4NzMR1dIVHs2NzJjY0MTEzM2VhMn0=zQ3NzhhMzhlOD"

raw = ["0" for _ in range(38)]

raw[0: 13] = data[13: 26]
raw[13: 26] = data[39: 52]
raw[26: 39] = data[0: 13]
raw[39: 52] = data[26: 39]

print(base64.b64decode("".join(raw).encode()).decode())

>>> GWHT{672cc4778a38e80cb362987341133ea2}
```

Bytecode

第一次遇到这种题目, Bytecode为python的字节码转换成人类可读的形式, 当smail来读就行, 还原成python脚本

```
  4           0 LOAD_CONST               0 (3)
              3 LOAD_CONST               1 (37)
              6 LOAD_CONST               2 (72)
              9 LOAD_CONST               3 (9)
             12 LOAD_CONST               4 (6)
             15 LOAD_CONST               5 (132)
             18 BUILD_LIST               6
             21 STORE_NAME               0 (en)

  5          24 LOAD_CONST               6 (101)
             27 LOAD_CONST               7 (96)
             30 LOAD_CONST               8 (23)
             33 LOAD_CONST               9 (68)
             36 LOAD_CONST              10 (112)
             39 LOAD_CONST              11 (42)
             42 LOAD_CONST              12 (107)
             45 LOAD_CONST              13 (62)
             48 LOAD_CONST               7 (96)
             51 LOAD_CONST              14 (53)
             54 LOAD_CONST              15 (176)
             57 LOAD_CONST              16 (179)
             60 LOAD_CONST              17 (98)
             63 LOAD_CONST              14 (53)
             66 LOAD_CONST              18 (67)
             69 LOAD_CONST              19 (29)
             72 LOAD_CONST              20 (41)
             75 LOAD_CONST              21 (120)
             78 LOAD_CONST              22 (60)
             81 LOAD_CONST              23 (106)
             84 LOAD_CONST              24 (51)
             87 LOAD_CONST               6 (101)
             90 LOAD_CONST              25 (178)
             93 LOAD_CONST              26 (189)
             96 LOAD_CONST               6 (101)
             99 LOAD_CONST              27 (48)
            102 BUILD_LIST              26
            105 STORE_NAME               1 (output)

  7         108 LOAD_CONST              28 ('welcome to GWHT2020')
            111 PRINT_ITEM
            112 PRINT_NEWLINE
```

```
 9          113 LOAD_NAME                2 (raw_input)
            116 LOAD_CONST              29 ('please input your flag:')
            119 CALL_FUNCTION            1
            122 STORE_NAME               3 (flag)

10          125 LOAD_NAME                3 (flag)
            128 STORE_NAME               4 (str)

12          131 LOAD_NAME                5 (len)
            134 LOAD_NAME                4 (str)
            137 CALL_FUNCTION            1
            140 STORE_NAME               6 (a)

13          143 LOAD_NAME                6 (a)
            146 LOAD_CONST              30 (38)
            149 COMPARE_OP               0 (<)
            152 POP_JUMP_IF_FALSE      173

14          155 LOAD_CONST              31 ('lenth wrong!')
            158 PRINT_ITEM
            159 PRINT_NEWLINE

15          160 LOAD_NAME                7 (exit)
            163 LOAD_CONST              32 (0)
            166 CALL_FUNCTION            1
            169 POP_TOP
            170 JUMP_FORWARD             0 (to 173)

17      >>  173 LOAD_NAME                8 (ord)
            176 LOAD_NAME                4 (str)
            179 LOAD_CONST              32 (0)
            182 BINARY_SUBSCR
            183 CALL_FUNCTION            1
            186 LOAD_CONST              33 (2020)
            189 BINARY_MULTIPLY
            190 LOAD_NAME                8 (ord)
            193 LOAD_NAME                4 (str)
            196 LOAD_CONST              34 (1)
            199 BINARY_SUBSCR
            200 CALL_FUNCTION            1
            203 BINARY_ADD
            204 LOAD_CONST              33 (2020)
            207 BINARY_MULTIPLY
            208 LOAD_NAME                8 (ord)
            211 LOAD_NAME                4 (str)
            214 LOAD_CONST              35 (2)
            217 BINARY_SUBSCR
            218 CALL_FUNCTION            1
            221 BINARY_ADD
            222 LOAD_CONST              33 (2020)
            225 BINARY_MULTIPLY
            226 LOAD_NAME                8 (ord)
            229 LOAD_NAME                4 (str)
            232 LOAD_CONST               0 (3)
            235 BINARY_SUBSCR
            236 CALL_FUNCTION            1
            239 BINARY_ADD
            240 LOAD_CONST              33 (2020)
            243 BINARY_MULTIPLY
```

```
          244 LOAD_NAME                8 (ord)
          247 LOAD_NAME                4 (str)
          250 LOAD_CONST              36 (4)
          253 BINARY_SUBSCR
          254 CALL_FUNCTION            1
          257 BINARY_ADD
          258 LOAD_CONST              37 (1182843538814603)
          261 COMPARE_OP               2 (==)
          264 POP_JUMP_IF_FALSE      275

18        267 LOAD_CONST              38 ('good!continue\xe2\x80\xa6\xe2\x80\xa6')
          270 PRINT_ITEM
          271 PRINT_NEWLINE
          272 JUMP_FORWARD            15 (to 290)

20    >>  275 LOAD_CONST              39 ('bye~')
          278 PRINT_ITEM
          279 PRINT_NEWLINE

21        280 LOAD_NAME                7 (exit)
          283 LOAD_CONST              32 (0)
          286 CALL_FUNCTION            1
          289 POP_TOP

23    >>  290 BUILD_LIST               0
          293 STORE_NAME               9 (x)

24        296 LOAD_CONST              40 (5)
          299 STORE_NAME              10 (k)

25        302 SETUP_LOOP             128 (to 433)
          305 LOAD_NAME               11 (range)
          308 LOAD_CONST              41 (13)
          311 CALL_FUNCTION            1
          314 GET_ITER
      >>  315 FOR_ITER               114 (to 432)
          318 STORE_NAME              12 (i)

26        321 LOAD_NAME                8 (ord)
          324 LOAD_NAME                4 (str)
          327 LOAD_NAME               10 (k)
          330 BINARY_SUBSCR
          331 CALL_FUNCTION            1
          334 STORE_NAME              13 (b)

27        337 LOAD_NAME                8 (ord)
          340 LOAD_NAME                4 (str)
          343 LOAD_NAME               10 (k)
          346 LOAD_CONST              34 (1)
          349 BINARY_ADD
          350 BINARY_SUBSCR
          351 CALL_FUNCTION            1
          354 STORE_NAME              14 (c)

28        357 LOAD_NAME               14 (c)
          360 LOAD_NAME                0 (en)
          363 LOAD_NAME               12 (i)
          366 LOAD_CONST               4 (6)
          369 BINARY_MODULO
          370 BINARY_SUBSCR
```

```
          370 BINARY_SUBSCR
          371 BINARY_XOR
          372 STORE_NAME              15 (a11)

29        375 LOAD_NAME               13 (b)
          378 LOAD_NAME                0 (en)
          381 LOAD_NAME               12 (i)
          384 LOAD_CONST               4 (6)
          387 BINARY_MODULO
          388 BINARY_SUBSCR
          389 BINARY_XOR
          390 STORE_NAME              16 (a22)

30        393 LOAD_NAME                9 (x)
          396 LOAD_ATTR               17 (append)
          399 LOAD_NAME               15 (a11)
          402 CALL_FUNCTION            1
          405 POP_TOP

31        406 LOAD_NAME                9 (x)
          409 LOAD_ATTR               17 (append)
          412 LOAD_NAME               16 (a22)
          415 CALL_FUNCTION            1
          418 POP_TOP

32        419 LOAD_NAME               10 (k)
          422 LOAD_CONST              35 (2)
          425 INPLACE_ADD
          426 STORE_NAME              10 (k)
          429 JUMP_ABSOLUTE          315
      >>  432 POP_BLOCK

33    >>  433 LOAD_NAME                9 (x)
          436 LOAD_NAME                1 (output)
          439 COMPARE_OP               2 (==)
          442 POP_JUMP_IF_FALSE      453

34        445 LOAD_CONST              38 ('good!continue\xe2\x80\xa6\xe2\x80\xa6')
          448 PRINT_ITEM
          449 PRINT_NEWLINE
          450 JUMP_FORWARD            15 (to 468)

36    >>  453 LOAD_CONST              42 ('oh,you are wrong!')
          456 PRINT_ITEM
          457 PRINT_NEWLINE

37        458 LOAD_NAME                7 (exit)
          461 LOAD_CONST              32 (0)
          464 CALL_FUNCTION            1
          467 POP_TOP

39    >>  468 LOAD_NAME                5 (len)
          471 LOAD_NAME                4 (str)
          474 CALL_FUNCTION            1
          477 STORE_NAME              18 (l)

40        480 LOAD_NAME                8 (ord)
          483 LOAD_NAME                4 (str)
          486 LOAD_NAME               18 (l)
          489 LOAD_CONST              43 (7)
```

```
          492 BINARY_SUBTRACT
          493 BINARY_SUBSCR
          494 CALL_FUNCTION           1
          497 STORE_NAME             19 (a1)

41        500 LOAD_NAME               8 (ord)
          503 LOAD_NAME               4 (str)
          506 LOAD_NAME              18 (l)
          509 LOAD_CONST              4 (6)
          512 BINARY_SUBTRACT
          513 BINARY_SUBSCR
          514 CALL_FUNCTION           1
          517 STORE_NAME             20 (a2)

42        520 LOAD_NAME               8 (ord)
          523 LOAD_NAME               4 (str)
          526 LOAD_NAME              18 (l)
          529 LOAD_CONST             40 (5)
          532 BINARY_SUBTRACT
          533 BINARY_SUBSCR
          534 CALL_FUNCTION           1
          537 STORE_NAME             21 (a3)

43        540 LOAD_NAME               8 (ord)
          543 LOAD_NAME               4 (str)
          546 LOAD_NAME              18 (l)
          549 LOAD_CONST             36 (4)
          552 BINARY_SUBTRACT
          553 BINARY_SUBSCR
          554 CALL_FUNCTION           1
          557 STORE_NAME             22 (a4)

44        560 LOAD_NAME               8 (ord)
          563 LOAD_NAME               4 (str)
          566 LOAD_NAME              18 (l)
          569 LOAD_CONST              0 (3)
          572 BINARY_SUBTRACT
          573 BINARY_SUBSCR
          574 CALL_FUNCTION           1
          577 STORE_NAME             23 (a5)

45        580 LOAD_NAME               8 (ord)
          583 LOAD_NAME               4 (str)
          586 LOAD_NAME              18 (l)
          589 LOAD_CONST             35 (2)
          592 BINARY_SUBTRACT
          593 BINARY_SUBSCR
          594 CALL_FUNCTION           1
          597 STORE_NAME             24 (a6)

46        600 LOAD_NAME              19 (a1)
          603 LOAD_CONST              0 (3)
          606 BINARY_MULTIPLY
          607 LOAD_NAME              20 (a2)
          610 LOAD_CONST             35 (2)
          613 BINARY_MULTIPLY
          614 BINARY_ADD
          615 LOAD_NAME              21 (a3)
          618 LOAD_CONST             40 (5)
```

```
           621 BINARY_MULTIPLY
           622 BINARY_ADD
           623 LOAD_CONST              44 (1003)
           626 COMPARE_OP               2 (==)
           629 POP_JUMP_IF_FALSE      807

47         632 LOAD_NAME               19 (a1)
           635 LOAD_CONST              36 (4)
           638 BINARY_MULTIPLY
           639 LOAD_NAME               20 (a2)
           642 LOAD_CONST              43 (7)
           645 BINARY_MULTIPLY
           646 BINARY_ADD
           647 LOAD_NAME               21 (a3)
           650 LOAD_CONST               3 (9)
           653 BINARY_MULTIPLY
           654 BINARY_ADD
           655 LOAD_CONST              45 (2013)
           658 COMPARE_OP               2 (==)
           661 POP_JUMP_IF_FALSE      807

48         664 LOAD_NAME               19 (a1)
           667 LOAD_NAME               20 (a2)
           670 LOAD_CONST              46 (8)
           673 BINARY_MULTIPLY
           674 BINARY_ADD
           675 LOAD_NAME               21 (a3)
           678 LOAD_CONST              35 (2)
           681 BINARY_MULTIPLY
           682 BINARY_ADD
           683 LOAD_CONST              47 (1109)
           686 COMPARE_OP               2 (==)
           689 POP_JUMP_IF_FALSE      804

49         692 LOAD_NAME               22 (a4)
           695 LOAD_CONST               0 (3)
           698 BINARY_MULTIPLY
           699 LOAD_NAME               23 (a5)
           702 LOAD_CONST              35 (2)
           705 BINARY_MULTIPLY
           706 BINARY_ADD
           707 LOAD_NAME               24 (a6)
           710 LOAD_CONST              40 (5)
           713 BINARY_MULTIPLY
           714 BINARY_ADD
           715 LOAD_CONST              48 (671)
           718 COMPARE_OP               2 (==)
           721 POP_JUMP_IF_FALSE      801

50         724 LOAD_NAME               22 (a4)
           727 LOAD_CONST              36 (4)
           730 BINARY_MULTIPLY
           731 LOAD_NAME               23 (a5)
           734 LOAD_CONST              43 (7)
           737 BINARY_MULTIPLY
           738 BINARY_ADD
           739 LOAD_NAME               24 (a6)
           742 LOAD_CONST               3 (9)
           745 BINARY_MULTIPLY
           746 BINARY_ADD
```

```
           747 LOAD_CONST              49 (1252)
           750 COMPARE_OP               2 (==)
           753 POP_JUMP_IF_FALSE      798

51         756 LOAD_NAME               22 (a4)
           759 LOAD_NAME               23 (a5)
           762 LOAD_CONST              46 (8)
           765 BINARY_MULTIPLY
           766 BINARY_ADD
           767 LOAD_NAME               24 (a6)
           770 LOAD_CONST              35 (2)
           773 BINARY_MULTIPLY
           774 BINARY_ADD
           775 LOAD_CONST              50 (644)
           778 COMPARE_OP               2 (==)
           781 POP_JUMP_IF_FALSE      795

52         784 LOAD_CONST              51 ('congraduation!you get the right flag!')
           787 PRINT_ITEM
           788 PRINT_NEWLINE
           789 JUMP_ABSOLUTE          795
           792 JUMP_ABSOLUTE          798
      >>   795 JUMP_ABSOLUTE          801
      >>   798 JUMP_ABSOLUTE          804
      >>   801 JUMP_ABSOLUTE          807
      >>   804 JUMP_FORWARD            0 (to 807)
      >>   807 LOAD_CONST              52 (None)
           810 RETURN_VALUE
```

```
en = [3, 37, 72, 9, 6, 132]
output = [101, 96, 23, 68, 112, 42, 107, 62, 96, 53, 176, 179, 98, 53, 67, 29, 41, 120, 60, 106, 51, 101, 178, 1
89, 101,
          48]
flag = input()

str1 = flag
if len(flag) < 38:
    print("lenth wrong!")
    exit(0)

s = ord(str1[0]) * 2020
s += ord(str1[1])
s *= 2020
s += ord(str1[2])
s *= 2020
s += ord(str1[3])
s *= 2020
s += ord(str1[4])
if s == 1182843538814603:
    print('good!continue')
else:
    exit()

x = []
k = 5
for i in range(13):
    b = ord(str1[k])
    c = ord(str1[k + 1])
    a11 = c ^ en[i % 6]
    a22 = b ^ en[i % 6]
    x.append(a11)
    x.append(a22)
    k = k + 2

if x == output:
    print('good!continue')
else:
    exit()

l = len(str1)
a1 = ord(str1[l - 7])
a2 = ord(str1[l - 6])
a3 = ord(str1[l - 5])
a4 = ord(str1[l - 4])
a5 = ord(str1[l - 3])
a6 = ord(str1[l - 2])


if a1 * 3 + a2 * 2 + a3 * 5 == 1003:
    if a1 * 4 + a2 * 7 + a3 * 9 == 2013:
        if a1 + a2 * 8 + a3 * 2 == 1109:
            if a4 * 3 + a5 * 2 + a6 * 5 == 671:
                if a4 * 4 + a5 * 7 + a6 * 9 == 1252:
                    if a4 + a5 * 8 + a6 * 2 == 644:
                        print('congraduation!you get the right flag!')
```

首先分析第一段, 其实根据常识即可知道应该是`GWHT{`, 带入后正确

```
s = ord(str1[0]) * 2020
s += ord(str1[1])
s *= 2020
s += ord(str1[2])
s *= 2020
s += ord(str1[3])
s *= 2020
s += ord(str1[4])
if s == 1182843538814603:
    print('good!continue')
else:
    exit()
```

第二段分析, 可知为从第5位开始, 每两位进行某种异或运算, 而且其取值仅和output有关, 和前面和后面无关, 只有两位进行爆破即可

```
en = [3, 37, 72, 9, 6, 132]
output = [101, 96, 23, 68, 112, 42, 107, 62, 96, 53, 176, 179, 98, 53, 67, 29, 41, 120, 60, 106, 51, 101, 178, 1
89, 101,
          48]
final = []

for zz in range(0, 26, 2):
    def foo():
        for xx in range(42, 127):
            for yy in range(42, 127):
                str1 = 'GWHT{' + f"{chr(xx)}{chr(yy)}" * 13
                x = []
                k = 5
                for i in range(13):
                    b = ord(str1[k])
                    c = ord(str1[k + 1])
                    a11 = c ^ en[i % 6]
                    a22 = b ^ en[i % 6]
                    x.append(a11)
                    x.append(a22)
                    k += 2
                if x[zz] == output[zz] and x[zz + 1] == output[zz + 1]:
                    final.append(xx)
                    final.append(yy)
                    return
        print("fail", zz)
    foo()

print("".join([chr(i) for i in final]))

>>> cfa2b87b3f746a8f0ac5c5963f
```

第三段, 又双叒叕是方程组, ~~这次怎么这么喜欢出方程组~~, z3直接解

```
from z3 import *

a1, a2, a3, a4, a5, a6 = Ints("a1 a2 a3 a4 a5 a6")

x = Solver()

x.add(a1 * 3 + a2 * 2 + a3 * 5 == 1003)
x.add(a1 * 4 + a2 * 7 + a3 * 9 == 2013)
x.add(a1 + a2 * 8 + a3 * 2 == 1109)
x.add(a4 * 3 + a5 * 2 + a6 * 5 == 671)
x.add(a4 * 4 + a5 * 7 + a6 * 9 == 1252)
x.add(a4 + a5 * 8 + a6 * 2 == 644)

x.check()
print(x.model())

for i in [97, 101, 102, 102, 55, 51]:
   print(chr(i), end="")

>>> [a5 = 55, a2 = 101, a6 = 51, a3 = 102, a4 = 102, a1 = 97]
>>> aeff73
```

最后拼接一下

`cfa2b87b3f746a8f0ac5c5963faeff73`