

记一次院赛CTF的Crypto和Re题（入门）

原创

CTF小白 于 2019-04-14 14:34:21 发布 5004 收藏 25

分类专栏: [CTF](#) 文章标签: [CTF 小白入门](#)

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

本文链接: https://blog.csdn.net/qq_41429081/article/details/89296213

版权



[CTF 专栏收录该内容](#)

24 篇文章 4 订阅

订阅专栏

目录

Crypto

[easy crypto](#)

[bAcOn](#)

[敌军密报](#)

Re

[easy re](#)

[跳到对的地方](#)

[简单的XOR](#)

[多密码表替换](#)

Crypto

easy crypto

easy crypto

50

阿强盯着这串奇怪的字符，目光锁定在最后两个等号上……

a3RwZ3N7b25mcl82NF9uYXFfZTBnXzEzfQ==
https://blog.csdn.net/qq_41429081

首先，这个很容易的看出这是一个base64加密一串密文，然后用base64解密后是

ktpgs{onfr_64_naq_e0g_13}

可以看出这就是flag的格式，所以一般就是凯撒加密了，但是因为有花括号下划线之类的，所以是凯撒的一个变形rot13

ROT13 ▾



xgctf{base 64 and rot 13}

bAcOn

这题是给了一个字符串baCoNBacoNbaconbACoNbacOnbAconBacOnbacOnbacoNbacaOnbacOnbACOnbACoN可以看出给了我们的提示就是bacon，培根。

我这边的话是改了一个网上找的python代码，然后直接跑出来了。

```
neon1.vscou&gt;extensions\ms-python.vscode-pylance-2019.5.0558\python\files\pyvsu_launcher.py --default --client --no  
    coding by qux  
*****Bacon Encode_Decode System*****  
input should be only lowercase or uppercase,cipher just include a,b(or A,B)  
1.encode  
2.decode  
3.exit  
please input number to choose  
2  
please input string to decode:  
baCoNBacoNbacoNbCoNbacoNbAconBac0nbAconBac0nbacoNbacoNbacoNbAC0nbACoN  
aababbaaabaaaaabbabaaabaabaaabaabaaaabaaaaaaaaaabbbbabbab  
first decode method result is:  
francisbacon  
second decode method result is:  
fsaocijtbacpo
```

因为培根密码是有两张密码表，同时因为大小写的话也有种情况，解出来应该是有四种（我这边另外一种以小写字母为b以大写字母为a的显示结果没有放出来）

```
import re
alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
first_cipher = ["aaaaa", "aaaab", "aaaba", "aaabb", "aabaa", "aabab", "aabba", "aabbb", "abaaa", "abaab", "ababa", "ababb",
```

```

"abbaa","abbab","abbba","abbbb","baaaa","baaab","baaba","baabb","babaa","babab","babba","babbb","bbaaa","bbaab"]
second_cipher = ["aaaaa","aaaab","aaaba","aaabb","aabaa","aabab","aabba","aabbb","abaaa","abaab","ababa",
,"ababb","abbaa","abbab","abbba","abbbb","baaaa","baaab","baaba","baabb","babaa","babab","babba","babbb"]
]
def decode():
    upper_flag = False # 用于判断输入是否为大写
    e_string = input("please input string to decode:\n")
    e_string=change(e_string)
    print(e_string)
    if e_string.isupper():
        upper_flag = True
        e_string = e_string.lower()
    e_array = re.findall(".{5}",e_string)
    d_string1 = ""
    d_string2 = ""
    for index in e_array:
        for i in range(0,26):
            if index == first_cipher[i]:
                d_string1 += alphabet[i]
            if index == second_cipher[i]:
                d_string2 += alphabet[i]
    if upper_flag:
        d_string1 = d_string1.upper()
        d_string2 = d_string2.upper()
    print ("first decode method result is:\n"+d_string1)
    print ("second decode method result is:\n"+d_string2)
    return
def change(s):
    str=""
    for i in s:
        if(i>='a' and i<='z'):
            str+="a"
        if(i>='A' and i<='Z'):
            str+="b"
    return str
def change2(s):
    str=""
    for i in s:
        if(i>='a' and i<='z'):
            str+="b"
        if(i>='A' and i<='Z'):
            str+="a"
    return str
if __name__ == '__main__':
    print ("\t\tcoding by qux")
    while True:
        print ("\t*****Bacon Encode_Decode System*****")
        print ("input should be only lowercase or uppercase,cipher just include a,b(or A,B)")
        print ("1.encode\n2.decode\n3.exit")
        s_number = input("please input number to choose\n")
        if s_number == "1":
            encode()
            input()
        elif s_number == "2":
            decode()
            input()
        elif s_number == "3":
            break
        else:
            continue

```

敌军密报

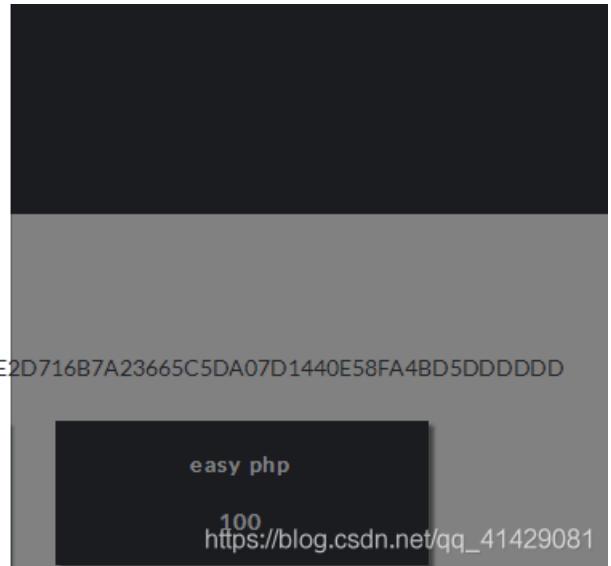
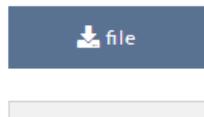
敌军密报

200

报告！我军卧底肖同志截获一份敌军密报，随信还附了一份文件，请李团长过目。

密报：

555554454445554344553544455545554554443333306471EC6E2D716B7A23665C5DA07D1440E58FA4BD5DDDDDD



题目展示就是这样的，一个密文，以及一个文件可以下载。这边这个文件用记事本打开稍微看了一下，有很多类似.pyt、__main__之类的东西，所以应该就是一个python编译后的文件了，这边推荐一个在线反编译python的网站
<http://tools.bugscaner.com/decompyle/>

在线pyc,pyo反编译python反编译

请选择需要反编译的*.pyc或*.pyo文件：

Done

Remove Browse ...

```

1  #! /usr/bin/env python 2.7 (62211)
2  #coding=utf-8
3  # Compiled at: 2019-03-22 10:43:03
4  #Powered by BugScanner
5  #http://tools.bugscaner.com/
6  #如果觉得不错,请分享给你朋友使用吧!
7  import base64
8
9  def convert(c, key, start='a', n=26):
10     a = ord(start)
11     offset = (ord(c) - a + key) % n
12     return chr(a + offset)
13
14
15 def encode_1(s, key):
16     o = ''
17     for c in s:
18         if c.islower():
19             o += convert(c, key, 'a')
20         elif c.isupper():
21             o += convert(c, key, 'A')
22         else:
23             o += c
24
25     return o
26

```

https://blog.csdn.net/qq_41429081

可以发现的是有四层加密

第一层凯撒加密

第二层base32

第三层base16

第四层栅栏加密取2

这边分享下我是如何分析这些代码是如何加密的。

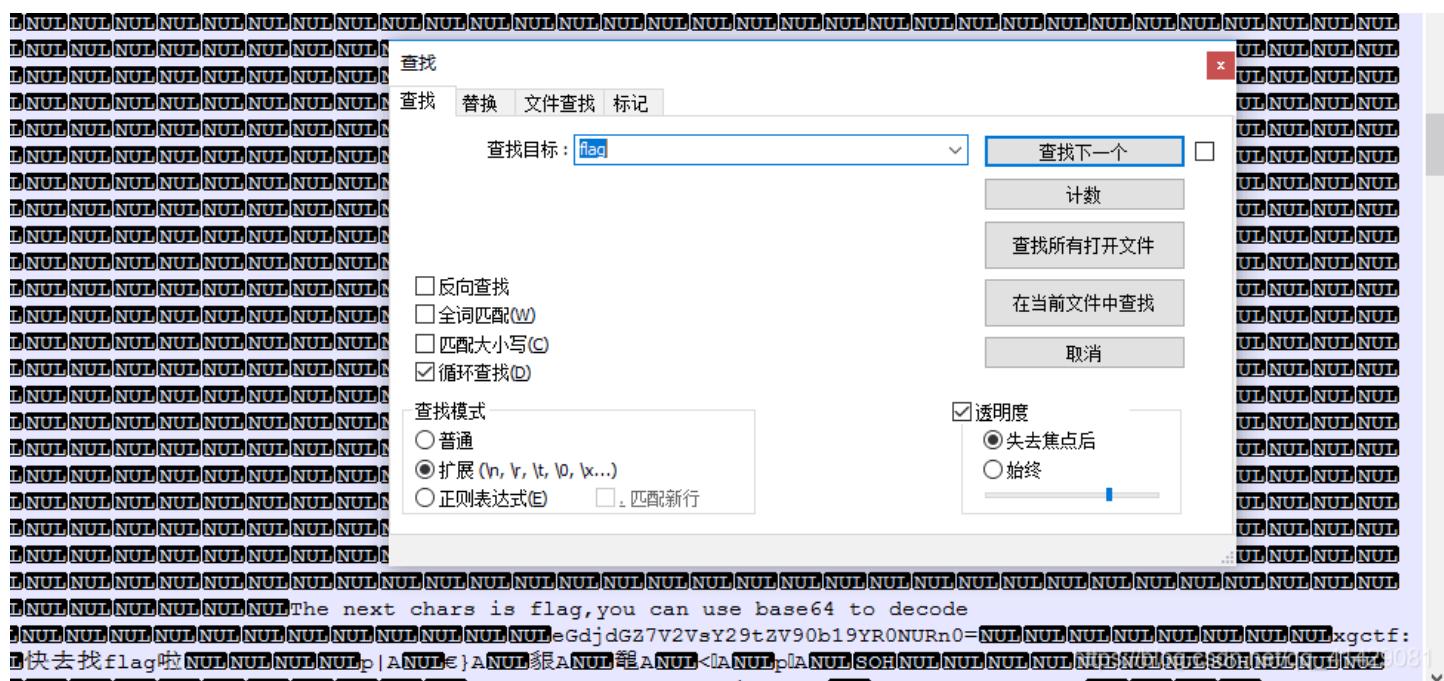
在得到这个python代码后，每一种加密其实都可以单独取出来，然后自己定义一些需要加密的字符，然后可以看到加密后的内容，然后可以判断，如果判断不出来也没关系。

只需要将他的程序逆着写，然后将我们测试的加密后的内容，可以还原出加密前的字符，那么我们逆着的程序也就没有写错了。这样我们就能一步一步逆向出明文了。

Re

easy re

这一题是真的太没有难度了



文本编辑器直接就搜出来了，当然更合适的做法肯定是用ida或od去搜

跳到对的地方

只要代码执行到这里就可以得到Flag哦

这一题的话，就是考察最基本的od动态调试的跳转，使用中文搜索引擎，然后找到相应的内容，双击进去，就可以看到这句话所在的汇编代码段了。然后一般是选择push ebp去跳转就能进去了。

启动程序，这边比较直接的就是直接跳进去，找一个能执行到的地方，改下汇编，让他jmp到我们之前找到的地址。

The screenshot shows the Immunity Debugger interface. The assembly window displays the following code:

```
00AA2500 55          push    ebp  
00AA2501 8BEC        mov     ebp,esp  
00AA2503 E8 78FCFFFF call   0x3.00AA2180  
00AA2508 5D          pop    ebp  
00AA2509 C3          retn  
00AA250A CC          int3  
00AA250B CC          int3  
00AA250C CC          int3
```

A context menu is open over the instruction at address 00AA2503, with the option "汇编于此处 : 00AA2503" (Assembly here : 00AA2503) selected. A dialog box titled "汇编于此处 : 00AA2503" (Assembly here : 00AA2503) is displayed, containing the assembly code "jmp 00AA1990". There is a checked checkbox labeled "使用 NOP 填充" (Use NOP fill) and two buttons: "汇编" (Assemble) and "取消" (Cancel). The right side of the interface shows the寄存器 (Registers) pane.

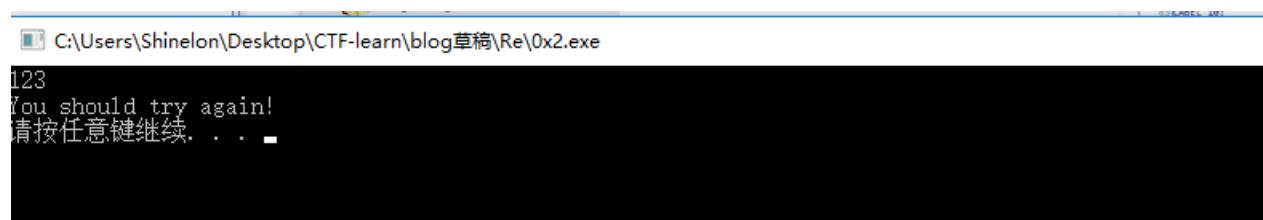
The screenshot shows the Immunity Debugger interface with the following details:

- Title Bar:** C:\Users\Shinelon\Desktop\CTF\xgctf {Jump_And_Get_Flag}
- Menu Bar:** 文件(F)、查看(V)、调试(D)、插件(P)、选项(T)、窗口(W)、帮助(H)、[+]、快捷菜单、Tools、BreakPoint->
- Toolbar:** Includes icons for Run, Stop, Break, Step, and various memory dump options.
- Registers:** Shows CPU registers: le, mt, tw, hc, pk, br, ... s.
- Stack:** Shows the stack contents: 00AA2500, 55, 8BEC, E9 88F4FFFF, 5D, C3, CC, CC, CC, CC, CC, CC.
- Registers:** Shows CPU registers: le, mt, tw, hc, pk, br, ... s.
- Call Stack:** Shows the call stack: kernel32!76C062C4.
- Bottom Status Bar:** https://blog.csdn.net/qq_41429081

当然，这样可能会导致我们程序崩溃，但是我们只是要这个flag，只要让他显示出来就好了。

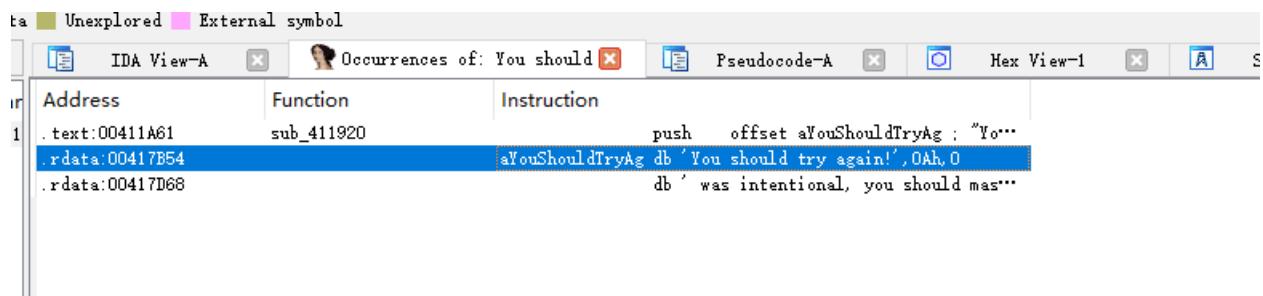
简单的XOR

这题我当时是没有做出来的，只要当时没有找到真正的main函数，后来我发现一个找main函数的技巧。因为你直接运行那个需要你逆向的程序，是会出现一些信息的



```
C:\Users\Shinelon\Desktop\CTF-learn\blog草稿\Re\0x2.exe
123
You should try again!
请按任意键继续. . .
```

输入123，他提示我try again。于是可以在ida中搜索这个字符串，ALT+T。



Address	Function	Instruction
.text:00411A61	sub_411920	push offset aYouShouldTryAg ; "Yo..."
.rdata:00417854		aYouShouldTryAg db 'You should try again!',0Ah,0
.rdata:00417D68		db ' was intentional, you should mas...'

然后我就找到了他真正的main函数，很容易有木有

```

141
3 int v0; // edx
4 int v1; // ecx
5 int v2; // edx
6 int v3; // edx
7 int v4; // ecx
8 int v5; // ST08_4
9 signed int j; // [esp+D0h] [ebp-A4h]
10 signed int i; // [esp+DCh] [ebp-98h]
11 char v9[36]; // [esp+E8h] [ebp-8Ch]
12 char v10[36]; // [esp+10Ch] [ebp-68h]
13 char v11; // [esp+130h] [ebp-44h]
14 char v12; // [esp+131h] [ebp-43h]
15 char v13; // [esp+132h] [ebp-42h]
16 char v14; // [esp+133h] [ebp-41h]
17 char v15; // [esp+134h] [ebp-40h]
18 char v16; // [esp+135h] [ebp-3Fh]
19 char v17; // [esp+136h] [ebp-3Eh]
20 char v18; // [esp+137h] [ebp-3Dh]
21 char v19; // [esp+138h] [ebp-3Ch]
22 char v20; // [esp+139h] [ebp-3Bh]
23 char v21; // [esp+13Ah] [ebp-3Ah]
24 char v22; // [esp+13Bh] [ebp-39h]
25 char v23; // [esp+13Ch] [ebp-38h]
26 char v24; // [esp+13Dh] [ebp-37h]
27 char v25; // [esp+13Eh] [ebp-36h]
28 char v26; // [esp+13Fh] [ebp-35h]
29 char v27; // [esp+140h] [ebp-34h]
30 char v28; // [esp+141h] [ebp-33h]
31 char v29; // [esp+142h] [ebp-32h]
32 char v30; // [esp+143h] [ebp-31h]
33 char v31; // [esp+144h] [ebp-30h]
34 char v32; // [esp+145h] [ebp-2Fh]
35 char v33; // [esp+146h] [ebp-2Eh]
36 char v34; // [esp+147h] [ebp-2Dh]
37 char v35[32]; // [esp+150h] [ebp-24h]
38 int v36; // [esp+170h] [ebp-4h]
39 int savedregs; // [esp+174h] [ebp+0h]
40

```

https://blog.csdn.net/qq_41429081

```

● 42 strcpy(v35, "hang_dian_xin_gong_ctf!!");
● 43 v11 = 16;
● 44 v12 = 6;
● 45 v13 = 13;
● 46 v14 = 19;
● 47 v15 = 57;
● 48 v16 = 31;
● 49 v17 = 61;
● 50 v18 = 9;
● 51 v19 = 7;
● 52 v20 = 44;
● 53 v21 = 39;
● 54 v22 = 0;
● 55 v23 = 29;
● 56 v24 = 0;
● 57 v25 = 2;
● 58 v26 = 14;
● 59 v27 = 29;
● 60 v28 = 30;
● 61 v29 = 0;
● 62 v30 = 27;
● 63 v31 = 27;
● 64 v32 = 20;
● 65 v33 = 0;
● 66 v34 = 92;
● 67 sub_411154("%s", (unsigned int)v9, 25);
● 68 for ( i = 0; i < 24; ++i )
● 69     v10[i] = *(&v11 + i) ^ v9[i];
● 70 for ( j = 0; j < 24; ++j )
● 71 {
● 72     if ( v10[j] != v35[j] )
● 73     {
● 74         sub_411048("You should try again!\n");
● 75         system("pause");
● 76         sub_411228(v1, v0);
● 77         goto LABEL_10;
● 78     }
● 79 }
● 80 sub_411048("The flag is your input!\n");
● 81 system("pause");
● 82 sub_411228(v4, v3);
● 83 LABEL_10:
● 84 v5 = v2.

```

https://blog.csdn.net/qq_41429081

找到了main函数，就可以看的出来，他就是要将我们输入的东西，和v11到v34的值进行异或，最后的结果要等于v35就是"hang_dian_xin_gong_ctf!"。这边需要的知识点就是，抑或的时候 $A \wedge B = C$ 那么 $A \wedge C = B$ 、 $B \wedge C = A$ 就都成立了

于是写了个脚本

```
mb.py
1 s1="hang_dian_xin_gong_ctf!!"
2 arr1=[]
3 for i in s1:
4     arr1.append(i)
5 print(len(arr1))
6 arr2=[16,6,13,19,57,31,61,9,7,44,39,0,29,0,2,14,29,30,0,27,27,20,0,92]
7 print(len(arr2))
8 str=""
9 for i in range(0,24):
10    str+=chr(ord(arr1[i])^arr2[i])
11
12 print(str)
```

https://blog.csdn.net/qq_41429081

就获得了flag

```
n\.vscode\extensions\ms-python.python-2019.3.6558\pythonFiles\ptvsd_1
24
24
xgctf{This_is_easy_xor!}
```

多密码表替换

找到main函数如下

```
26 v25 = __readfsqword(0x28u);
27 v15 = 'swczdaeq';
28 v16 = 'tnvhfyrx';
29 v17 = 'iujkklmgb';
30 v18 = 'po';
31 v19 = 0;
32 v20 = 'AZXCVBNM';
33 v21 = 'LKJHGFDS';
34 v22 = 'EWQYUIOP';
35 v23 = 'TR';
36 v24 = 0;
37 v10 = 'w!k9R!F9';
38 v11 = 'W88Mt!U3';
39 v12 = 'd09';
40 v7 = '24837159';
41 v8 = '60';
42 v9 = 0;
43 makeEmpty(v14, 20LL);
44 makeEmpty(buf, 20LL);
45 puts(&s);
46 read(0, buf, 0x13uLL);

https://blog.csdn.net/qq_41429081
```

```
40 i=buf, buf, 0x13uLL,
41 for ( i = 0; i <= 18; ++i )
42 {
43     if ( !buf[i] )
44     {
45         puts(&byte_B00);
46         return 0;
47     }
48     for ( j = 0; j <= 18; ++j )
49     {
50         if ( buf[j] <= '@' || buf[j] > 'z' )
51         {
52             if ( buf[j] <= ' ' || buf[j] > 'z' )
```

```

60
61     if ( buf[j] <= '/' || buf[j] > '9' )
62     {
63         if ( buf[j] == 95 )
64             v14[j] = '!';
65         else
66         {
67             v14[j] = *((_BYTE *)&v7 + buf[j] - 48);
68         }
69     }
70     else
71     {
72         v14[j] = *((_BYTE *)&v20 + buf[j] - 97);
73     }
74 }
75 else
76 {
77     v14[j] = *((_BYTE *)&v15 + buf[j] - 65);
78 }
79
80 for ( k = 0; k <= 18; ++k )
81 {
82     if ( *((_BYTE *)&v10 + k) != v14[k] )
83     {
84         puts(&byte_B28);
85         return 0;
86     }
87 }
88 puts(&byte_B60);

```

https://blog.csdn.net/qq_41429081

我把它改写成python代码大概是这个意思

```

1 buf=[]
2 v14=[]
3 v10="9F!R9k!w3U!tM88w90d"
4 v7="9517384206"
5 v20="MNBCXZASDFGHJKLMNOPUIYQWERT"
6 v15="qeadzcwsxryfhvntgbmlkjuiop"
7

```

```

30 ...
31 for i in range(0,19):
32     if(buf[i]>='A' and buf[i]<='Z'):
33         v14.append(v15[ord(buf[i])-65])
34     elif(buf[i]>='a' and buf[i]<='z'):
35         v14.append(v20[ord(buf[i])-97])
36     elif(buf[i]>='0' and buf[i]<='9'):
37         v14.append(v7[ord(buf[i])-48])
38     else:
39         v14.append('!')
40 ...
41 ...

```

https://blog.csdn.net/qq_41429081

然后最后这个v14等于v10就是正确的

这样我们就知道v14，于是上面的代码中我们不知道的就只有需要我们输入的buf[]。所以写一个求buf的代码

```
5     v20="mnbvcxzabdfrjklpqoiuytqwerty"
6     v15="qeadzcwsxryfhvntgbmlkjuiop"
7
8     for i in range(0,19):
9         v14.append(v10[i])
10
11    for i in range(0,19):
12        if(v14[i]=='!'):
13            buf.append(95)
14        for j in range(0,len(v15)):
15            if(v14[i]==v15[j]):
16                buf.append([j+65])
17                continue
18        for j in range(0,len(v20)):
19            if(v14[i]==v20[j]):
20                buf.append(j+97)
21                continue
22        for j in range(0,len(v7)):
23            if(v14[i]==v7[j]):
24                buf.append(j+48)
25                continue
26    str=""
27    for i in buf:
28        str+=chr(i)
29    print(str)
30    ...
31    for i in range(0,19):
```

https://blog.csdn.net/qq_41429081

运行就能得到flag

```
ions\ms-python.python-2019.3.6558\pythonFiles\ptvsd_laur
0k_y0U_G4t_Pa55w0rD
```