

i春秋网鼎杯网络安全大赛Beijing题目writeup

原创

iqiqiya 于 2018-08-20 17:31:37 发布 5869 收藏 4

分类专栏: [我的CTF之路](#) [我的逆向之路](#) -----2018春秋网鼎杯 [我的CTF进阶之路](#) 文章标签: [i春秋网鼎杯网络安全大赛Beijing](#) [题目writeup](#) [Beijing writeup](#) [reverse](#)

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/xiangshangbashaonian/article/details/81874856>

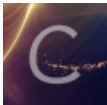
版权



[我的CTF之路](#) 同时被 3 个专栏收录

92 篇文章 5 订阅

订阅专栏



[我的逆向之路](#)

108 篇文章 10 订阅

订阅专栏



[-----2018春秋网鼎杯](#)

6 篇文章 0 订阅

订阅专栏

题目信息:



下载链接: <https://pan.baidu.com/s/1jZ36MzyiHsILxkJ0LwsOXg> 密码: 37k3

IDA载入 发现函数很少 也没有啥可以字符串 那就直分析

先看main函数 可以直接F5变成c伪代码

```

Function name | Segn | 13 | char v10; // al
              |      | 14 | char v11; // al
              |      | 15 | char v12; // al
              |      | 16 | char v13; // al
              |      | 17 | char v14; // al
              |      | 18 | char v15; // al
              |      | 19 | char v16; // al
              |      | 20 | char v17; // al
              |      | 21 | char v18; // al
              |      | 22 | char v19; // al
              |      | 23 | char v20; // al
              |      | 24 |
              |      | 25 | v0 = sub_8048460(dword_804A03C); // 每次都是给sub_8048460传入不同参数进行处理
              |      | 26 | printf("%c", v0);
              |      | 27 | fflush(stdout);
              |      | 28 | v1 = sub_8048460(dword_804A044);
              |      | 29 | printf("%c", v1);
              |      | 30 | fflush(stdout);
              |      | 31 | v2 = sub_8048460(dword_804A0E0);
              |      | 32 | printf("%c", v2);
              |      | 33 | fflush(stdout);
              |      | 34 | v3 = sub_8048460(dword_804A050);
              |      | 35 | printf("%c", v3);
              |      | 36 | fflush(stdout);
              |      | 37 | v4 = sub_8048460(dword_804A058);
              |      | 38 | printf("%c", v4);
              |      | 39 | fflush(stdout);
              |      | 40 | v5 = sub_8048460(dword_804A0E4);
              |      | 41 | printf("%c", v5);
              |      | 42 | fflush(stdout);
              |      | 43 | v6 = sub_8048460(dword_804A064);
  
```

<https://blog.csdn.net/xiangshangbashaonian>

双击任意dword_804Axxx

接着向下翻到dword_段

```

Segn | .data:0804A038 | byte_804A038 | db 7Dh | ; DATA XREF: sub_8048460:loc_80485C4↑r
     | .data:0804A039 | byte_804A039 | db 0ECh | ; DATA XREF: sub_8048460+16B↑r
     | .data:0804A03A | byte_804A03A | db 5Fh | ; DATA XREF: sub_8048460:loc_80485DE↑r
     | .data:0804A03B | byte_804A03B | db 57h | ; DATA XREF: sub_8048460+185↑r
     | .data:0804A03C | dword_804A03C | dd 6 | ; DATA XREF: main+10↑r
     | .data:0804A040 | | | db 4Ch | ; L
     | .data:0804A041 | | | db 0 |
     | .data:0804A042 | | | db 0 |
     | .data:0804A043 | | | db 0 |
     | .data:0804A044 | dword_804A044 | dd 9 | ; DATA XREF: main+43↑r
     | .data:0804A048 | | | db 0D3h |
     | .data:0804A049 | | | db 0 |
     | .data:0804A04A | | | db 0 |
     | .data:0804A04B | | | db 0 |
     | .data:0804A04C | | | db 3Ah | ; 前面没有dword_xxxxxxx的话就记为0
     | .data:0804A04D | | | db 0 |
     | .data:0804A04E | | | db 0 |
     | .data:0804A04F | | | db 0 |
     | .data:0804A050 | dword_804A050 | dd 1 | ; DATA XREF: main+B1↑r
     | .data:0804A054 | | | db 94h |
     | .data:0804A055 | | | db 0 |
     | .data:0804A056 | | | db 0 |
     | .data:0804A057 | | | db 0 |
     | .data:0804A058 | dword_804A058 | dd 0Ah | ; DATA XREF: main+E8↑r
     | .data:0804A05C | | | db 29h | ; )
     | .data:0804A05D | | | db 0 |
     | .data:0804A05E | | | db 0 |
     | .data:0804A05F | | | db 0 |
     | .data:0804A060 | | | db 3 |
     | .data:0804A061 | | | db 0 |
     | .data:0804A062 | | | db 0 |
     | .data:0804A063 | | | db 0 |
     | .data:0804A064 | dword_804A064 | dd 8 | ; DATA XREF: main+156↑r
     | .data:0804A068 | | | db 0FDh |
  
```

<https://blog.csdn.net/xiangshangbashaonian>

最后得到顺序6 9 0 1 10 0 8 0 11 2 3 1 13 4 5 2 7 2 3 1 12

从v0到v20

```

• .data:0804A020 byte_804A020 db 61h ; DATA XREF: sub_8048460:loc_804848C↑
• .data:0804A021 byte_804A021 db 4Ch ; DATA XREF: sub_8048460:33↑
• .data:0804A022 byte_804A022 db 67h ; DATA XREF: sub_8048460:loc_80484A6↑
• .data:0804A023 byte_804A023 db 59h ; DATA XREF: sub_8048460+4D↑
• .data:0804A024 byte_804A024 db 69h ; DATA XREF: sub_8048460:loc_80484C0↑
• .data:0804A025 byte_804A025 db 29h ; DATA XREF: sub_8048460+67↑
• .data:0804A026 byte_804A026 db 6Eh ; DATA XREF: sub_8048460:loc_80484DA↑
• .data:0804A027 byte_804A027 db 42h ; DATA XREF: sub_8048460+81↑
• .data:0804A028 byte_804A028 db 62h ; DATA XREF: sub_8048460:loc_80484F4↑
• .data:0804A029 byte_804A029 db 0Dh ; DATA XREF: sub_8048460+9B↑
• .data:0804A02A byte_804A02A db 65h ; DATA XREF: sub_8048460:loc_804850E↑
; DATA XREF: sub_8048460:loc_804850E↑

• .data:0804A020 byte_804A020 db 'a' 0 ; DATA XREF: sub_8048460:loc_804848C↑
• .data:0804A021 byte_804A021 db 4Ch ; DATA XREF: sub_8048460+33↑
• .data:0804A022 byte_804A022 db 'g' 1 ; DATA XREF: sub_8048460:loc_80484A6↑
• .data:0804A023 byte_804A023 db 59h ; DATA XREF: sub_8048460+4D↑
• .data:0804A024 byte_804A024 db 'i' 2 ; DATA XREF: sub_8048460:loc_80484C0↑
• .data:0804A025 byte_804A025 db 29h ; DATA XREF: sub_8048460+67↑
• .data:0804A026 byte_804A026 db 'n' 3 ; DATA XREF: sub_8048460:loc_80484DA↑
• .data:0804A027 byte_804A027 db 42h ; DATA XREF: sub_8048460+81↑
• .data:0804A028 byte_804A028 db 'b' 4 ; DATA XREF: sub_8048460:loc_80484F4↑
• .data:0804A029 byte_804A029 db 0Dh ; DATA XREF: sub_8048460+9B↑
• .data:0804A02A byte_804A02A db 'e' 5 ; DATA XREF: sub_8048460:loc_804850E↑
• .data:0804A02B byte_804A02B db 71h ; DATA XREF: sub_8048460+B5↑
• .data:0804A02C byte_804A02C db 'f' 6 ; DATA XREF: sub_8048460:loc_8048528↑
• .data:0804A02D byte_804A02D db 34h ; DATA XREF: sub_8048460+CF↑
• .data:0804A02E byte_804A02E db 'j' 7 ; DATA XREF: sub_8048460:loc_8048542↑
• .data:0804A02F byte_804A02F db 0C6h ; DATA XREF: sub_8048460+E9↑
• .data:0804A030 byte_804A030 db 'm' 8 ; DATA XREF: sub_8048460:loc_804855C↑
• .data:0804A031 byte_804A031 db 8Ah ; DATA XREF: sub_8048460+103↑
• .data:0804A032 byte_804A032 db 'l' 9 ; DATA XREF: sub_8048460:loc_8048576↑
• .data:0804A033 byte_804A033 db 7Fh ; DATA XREF: sub_8048460+11D↑
• .data:0804A034 byte_804A034 db '{' 10 ; DATA XREF: sub_8048460:loc_8048590↑
• .data:0804A035 byte_804A035 db 0AEh ; DATA XREF: sub_8048460+137↑
• .data:0804A036 byte_804A036 db 'z' 11 ; DATA XREF: sub_8048460:loc_80485AA↑
• .data:0804A037 byte_804A037 db 92h ; DATA XREF: sub_8048460+151↑
• .data:0804A038 byte_804A038 db '}' 12 ; DATA XREF: sub_8048460:loc_80485C4↑
• .data:0804A039 byte_804A039 db 0ECh ; DATA XREF: sub_8048460+16B↑
• .data:0804A03A byte_804A03A db '_' 13 ; DATA XREF: sub_8048460:loc_80485DE↑
• .data:0804A03B byte_804A03B db 57h ; DATA XREF: sub_8048460+185↑
• .data:0804A03C dword_804A03C dd 6 ; DATA XREF: main+10↑

```

把光标移到上边按r就可以转换成字符

得到对应关系

0 1 2 3 4 5 6 7 8 9 10 11 12 13

a g i n b e f j m l { z } _

然后就是按照上边得到的顺序进行一一对应

```

1 6 9 0 1 10 0 8 0 11 2 3 1 13 4 5 2 7 2 3 1 12
2 f l a g { a m a z i n g _ b e i j i n g }
3
4 0 1 2 3 4 5 6 7 8 9 10 11 12 13
5 a g i n b e f j m l { z } _

```

最后得到flag{amazing_beijing}