

# 攻防世界 easy\_Maze

原创

Bxb0 于 2020-05-04 14:08:41 发布 759 收藏 3

文章标签: [c语言 指针](#)

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/Bsecure/article/details/105916981>

版权

## easy\_Maze

这道题还是挺好玩的。

由题目我们能知道是关于迷宫的题. 下载下来是elf文件, 查壳后无壳, 在linux下运行看看大概流程。

载入IDA, 先分析了接受我们输入的字符的函数. 知道在外面一层的主函数是先生成一个每行7个元素的数组. 然后通过我们输入的字符按控制在矩阵迷宫中走向, 最后按照规定的到达一个目的地。

```
1 __int64 __fastcall Step_2(int (*a1)[7])
2 {
3     int v1; // eax
4     __int64 v2; // rax
5     __int64 v3; // rax
6     __int64 result; // rax
7     __int64 v5; // rax
8     char v6[35]; // [rsp+10h] [rbp-30h]
9     char v7; // [rsp+33h] [rbp-Dh]
10    int v8; // [rsp+34h] [rbp-Ch]
11    int v9; // [rsp+38h] [rbp-8h]
12    int v10; // [rsp+3Ch] [rbp-4h]
13
14    v10 = 0;
15    v9 = 0;
16    v8 = 0;
17    while ( v8 <= 29 && (*a1)[7 * v10 + v9] == 1 ) // 我感觉这里是应该是 a1[v10][v9] 数组指针啊 只不过这样也可以都是先得到首地址
18    { // 到这里, 初步知道了是个矩阵, 通过下面的4个操作, 根据最后的结果, 走规定的路线.
19        std::operator<<<char,std::char_traits<char>>>(std::cin, &v7);
20        v1 = v8++;
21        v6[v1] = v7;
22        if ( v7 == 100 ) // 输入字符 d 的话 v9++ 向右
23        {
24            ++v9;
25        }
26        else if ( v7 > 100 )
27        {
28            if ( v7 == 115 ) // 输入字符 s 的话 v10++ 向下
29            {
30                ++v10;
31            }
32            else
33            {
34                if ( v7 != 119 ) // 输入字符 w 的话 v10-- 向上
35                    goto LABEL_14;
36                --v10;
37            }
38        }
39        else if ( v7 == 97 ) // 输入字符 a 的话 v9-- 向左
40        {
41            --v9;
42        } // 就是我们打游戏熟悉的 WASD 啊, 哈哈
43        else
44        {
45        LABEL_14:
46            v2 = std::operator<<<std::char_traits<char>>(&bss_start, "include illegal words.");
47            std::ostream::operator<<(v2, &std::endl<char,std::char_traits<char>>);
48        }
49    }
50    if ( v10 != 6 || v9 != 6 ) // 由上一层生成矩阵的函数, 我们的矩阵是 7*7, 那这里就表示终点是最后一个元素
51    {
52        v5 = std::operator<<<std::char_traits<char>>(&bss_start, "Oh no!,Please try again~~");
53        std::ostream::operator<<(v5, &std::endl<char,std::char_traits<char>>);
54        result = 0LL;
55    }
```

```

55 }
56 else
57 {
58     v3 = std::operator<<<std::char_traits<char>>(&_bss_start, "Congratulations!");
59     std::ostream::operator<<<(v3, &std::endl<char,std::char_traits<char>>);
60     output(v6, v8);
61     result = 1LL;
62 }
63 return result;
64}

```

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

接下来就是找生成那个矩阵了. 通过上面的49个元素知道且下面函数的参数7, 知道规模 7\*7.

```

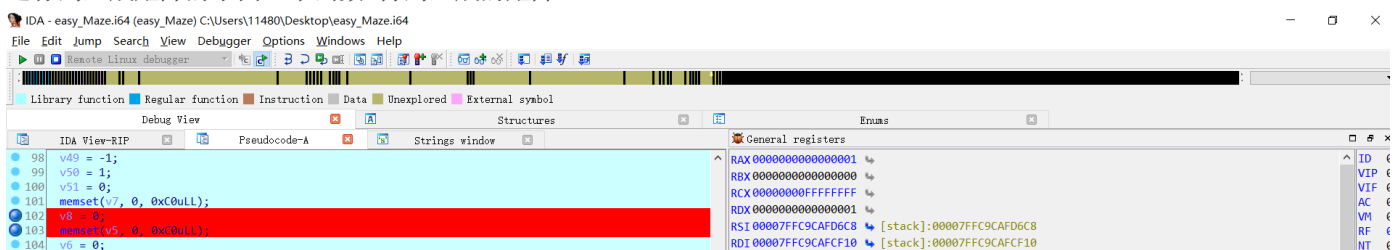
55 v9[3] = 1;
56 v9[4] = -1;
57 v9[5] = 1;
58 v9[6] = -1;
59 v10 = 0;
60 v11 = 0;
61 v12 = 0;
62 v13 = 0;
63 v14 = 1;
64 v15 = -1;
65 v16 = 0;
66 v17 = 0;
67 v18 = 1;
68 v19 = 0;
69 v20 = 0;
70 v21 = 1;
71 v22 = 0;
72 v23 = -1;
73 v24 = -1;
74 v25 = 0;
75 v26 = 1;
76 v27 = 0;
77 v28 = 1;
78 v29 = -1;
79 v30 = 0;
80 v31 = -1;
81 v32 = 0;
82 v33 = 0;
83 v34 = 0;
84 v35 = 0;
85 v36 = 0;
86 v37 = 1;
87 v38 = -1;
88 v39 = -1;
89 v40 = 1;
90 v41 = -1;
91 v42 = 0;
92 v43 = -1;
93 v44 = 2;
94 v45 = 1;
95 v46 = -1;
96 v47 = 0;
97 v48 = 0;
98 v49 = -1;
99 v50 = 1;
100 v51 = 0;
101 memset(v7, 0, 0xC0uLL);
102 v8 = 0;
103 memset(v5, 0, 0xC0uLL);
104 v6 = 0;
105 Step_0((int (*)[7])v9, 7, (int (*)[7])v7); // 生成举证关键函数 1
106 Step_1((int (*)[7])v7, 7, (int (*)[7])v5); // 生成举证关键函数 2
107 v3 = std::operator<<<std::char_traits<char>>(&_bss_start, "Please help me out!");
108 std::ostream::operator<<<(v3, &std::endl<char,std::char_traits<char>>);
109 Step_2((int (*)[7])v5);
110 system("pause");
111 return 0;
112}

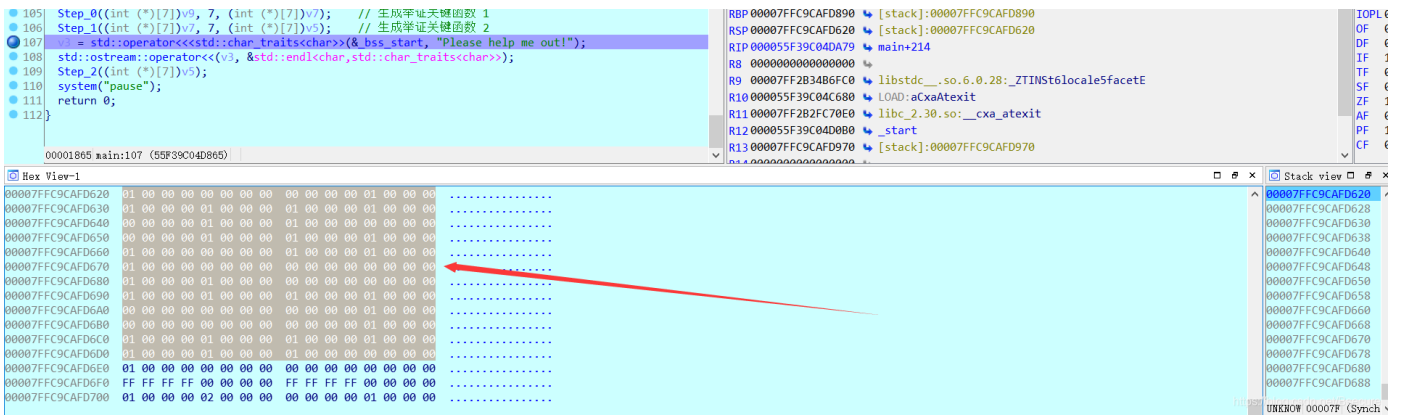
```

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

看了 **Step\_0** 函数, 还是简单, 但是 **step\_1** 中嵌套了2个函数且很复杂. 开始我就把生成矩阵的所有函数及数据都复制到VC再根据栈的特点改下数据的顺序, 然后运行打印出矩阵. 问题是运行后什么也没有打印, 调试发现一个函数中的内存空间和另外一个冲突, 相互覆盖值. 这个函数太多, 改起来也麻烦.

然后转向GDB调试, 但是不熟练. 又转向ida动态调试. 先下断点, 找到储存矩阵的空间的地址, 把这个地址转到数据窗口跟随. 运行到生成矩阵的下面一个函数. 得到生成的矩阵.





把数据复制下来,用C语言打印出来号观察,注意每个数据4个字节(小端).

```
#include <stdio.h>

int main(void)
{
    union
    {
        unsigned char ida_chars[196];
        int a[49];
    }A = {
        1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
        1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
        0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
        1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
        1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
        0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
        1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
        1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
        1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
        1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
        0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
        1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
        0, 0
    };
    int i = 0, j = 0;
    int (*p)[7] = (int (*)[7])A.a;

    for(i = 0; i < 7; i++)
    {
        for(j = 0; j < 7; j++)
        {
            printf("%d ", p[i][j]);
        }
        putchar(10);
    }

    return 0;
}
```

```
1 0 0 1 1 1 1
1 0 1 1 0 0 1
1 1 1 0 1 1 1
0 0 0 1 1 0 0
1 1 1 1 0 0 0
1 0 0 0 1 1 1
1 1 1 1 1 0 0
```

- 根据我们打游戏的熟悉 aswd ,朝着1走到最后. **ssddwdwdddsssaasasaaaassdddwwdds** 最后在linux下输入.

```
b@b-virtual-machine:~/桌面/ida$ ./easy_Maze
Please help me out!
ssddwdwdddsssaasasaaaassdddwwdds
Congratulations!
Thanks! Give you a flag: UNCTF{ssddwdwdddsssaasasaaaassdddwwdds}
sh: 1: pause: not found
b@b-virtual-machine:~/桌面/ida$
```

- 总结: 更加熟悉了linux动态调试的运用,加深了数组指针的理解.