Lab8. Native Code Reversing

STU ID：20307130044
Your Flag：FLAG{29868659734356183824165126738494685273254614315275839}

Analysis Process Breakdown:

首先 jeb 反编译一下，没什么东西，这里就不放截图了，就是有一个 native 函数，如果真就是 right，如果假就是 wrong，主要去看 native 里面的，用 ida 反编译一下

这是第一部分

```
if ( (*env)->GetStringUTFLength(env, a3) == 59 )
{
  v3 = (*env)->GetStringUTFChars(env, a3, 0);
  if ( *v3 == 70 && v3[1] == 76 && v3[2] == 65 && v3[3] == 71 && v3[4] == 123 && v3[58] == 125 )
  {
    v4 = -53;
    while ( (unsigned __int8)(v3[v4 + 58] - 48) <= 9u )
    {
```

前两个判断表出了**格式是 FLAG{xxxx}**,其中中间有 53 个 **ascii 码小于等于 9** 的字符，这时候大概能猜到是一串数字了（因为连下划线都不在范围内，前面只有一些零散符号）

然后看下面一段，用 x86 在 32 位里反编译的真的很怪，感觉 dest 和最后的 mubiao 好像没关系，mubiao 基本没改过呢？

```
if ( !++v4 )
{
  v19 = 0LL;
  v18 = 0LL;
  v17 = 0LL;
  *(_OWORD *)dest = 0LL;
  strncpy(dest, v3 + 5, 53u);
  v13 = 0LL;
  v12 = 0LL;
  v11 = 0LL;
  v10 = 0LL;
  *(_OWORD *)mubiao = 0LL;
  v14 = 0;
  v5 = 0;
  v6 = -81;
  do
  {
    v8 = *((_BYTE *)&ccc + v6 + 81);
    if ( !v8 )
      v8 = dest[v5++] - 48;
    v15[v6++] = v8;
  }
  while ( v6 );
  return bbb(mubiao);
}
```

也是我运气好，想着换一个版本 x86-64 的反编译看看呢？这里就看懂了

```
*(_OWORD *)dest = 0LL;
strncpy(dest, (const char *)(v5 + 5), 53uLL);
mubiao[4] = 0LL;
mubiao[3] = 0LL;
mubiao[2] = 0LL;
mubiao[1] = 0LL;
mubiao[0] = 0LL;
v15 = 0;
v8 = 0;
for ( j = 1LL; ; j += 2LL )
{
  v10 = ccc[j - 1];
  if ( !v10 )
    v10 = dest[v8++] - 48;
  *(&v13 + j) = v10;
  if ( j == 81 )
    break;
  v11 = ccc[j];
  if ( !v11 )
    v11 = dest[v8++] - 48;
  *((_BYTE *)mubiao + j) = v11;
}
LOBYTE(v12) = bbb((const char *)mubiao);
```

首先，有一个常量 ccc，点进去能看到，是一个 **81 元素长**的数组

```
.data:00002000                  public ccc
.data:00002000 ccc              db    0                 ; DATA XREF: LOAD:00000
.data:00002000                                          ; .got:ccc_ptr↑o
.data:00002001                  db    0
.data:00002002                  db    1
.data:00002003                  db    4
.data:00002004                  db    3
.data:00002005                  db    0
.data:00002006                  db    7
.data:00002007                  db    5
.data:00002008                  db    0
.data:00002009                  db    0
.data:0000200A                  db    0
.data:0000200B                  db    0
.data:0000200C                  db    0
.data:0000200D                  db    1
.data:0000200E                  db    0
.data:0000200F                  db    0
.data:00002010                  db    4
.data:00002011                  db    2
.data:00002012                  db    0
.data:00002013                  db    7
.data:00002014                  db    0
.data:00002015                  db    0
.data:00002016                  db    0
```

这段代码不断拷贝数据，分别**从 ccc 和 flag 里面拷贝数字**，用 python 写一下更清楚一点

```
while True:
    v10 = ccc[j-1]
    if (v10 == 0):
        v10 = ord(dest[v8]) - 48
        v8 += 1
    mubiao.append(v10)
    if j == 81:
        break
    v11 = ccc[j]
    if (v11 == 0):
        v11 = ord(dest[v8]) - 48
        v8 += 1
    mubiao.append(v11)
    j +=2
```

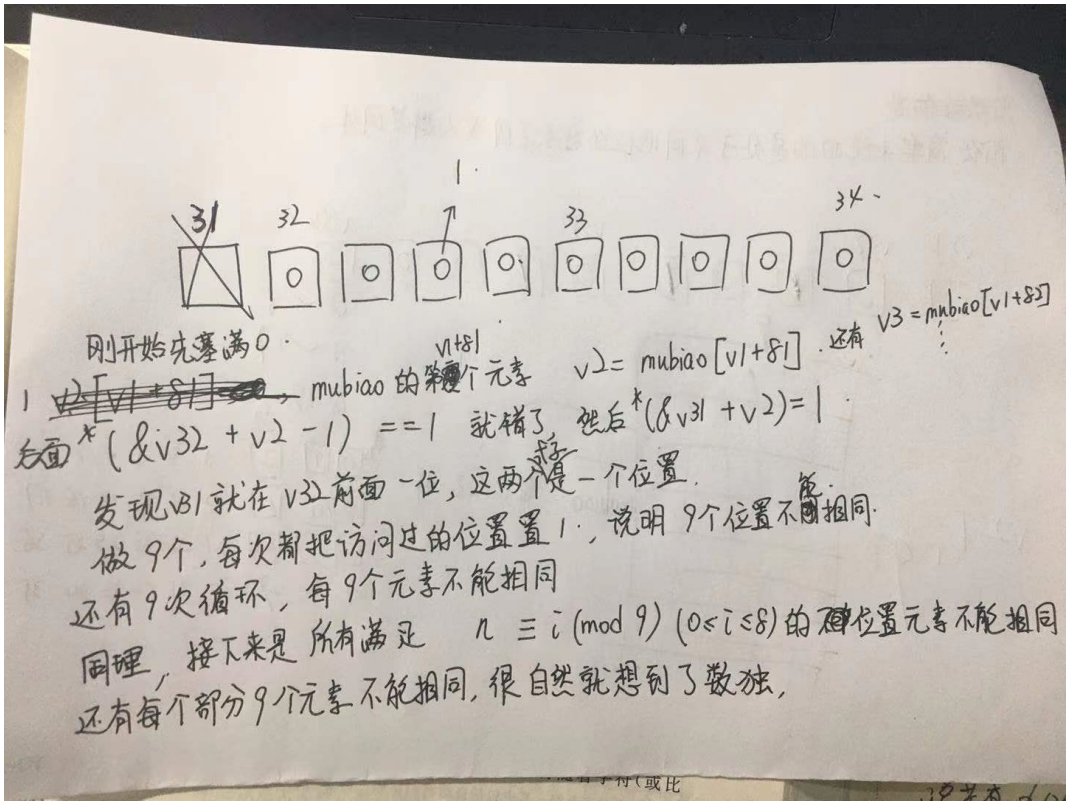
然后就**送到 bbb 函数**了

转战 bbb 函数

这里的分析要用到存放的知识了，我把这一段画了一下：

```
int result; // eax
char v31; // [esp+Bh] [ebp-21h]
int v32; // [esp+Ch] [ebp-20h] BYREF
int v33; // [esp+10h] [ebp-1Ch]
char v34; // [esp+14h] [ebp-18h]
unsigned int v35; // [esp+18h] [ebp-14h]

while ( 1 )
{
  v33 = 0;
  v32 = 0;
  v34 = 0;
  v2 = mubiao[v1 + 81];
  if ( *((_BYTE *)&v32 + v2 - 1) == 1 )
    return 0;
  *(&v31 + v2) = 1;
  v3 = mubiao[v1 + 82];
  if ( *((_BYTE *)&v32 + v3 - 1) == 1 )
    return 0;
  *(&v31 + v3) = 1;
  v4 = mubiao[v1 + 83];
  if ( *((_BYTE *)&v32 + v4 - 1) == 1 )
    return 0;
  *(&v31 + v4) = 1;
  v5 = mubiao[v1 + 84];
  if ( *((_BYTE *)&v32 + v5 - 1) == 1 )
    return 0;
  *(&v31 + v5) = 1;
  v6 = mubiao[v1 + 85];
  if ( *((_BYTE *)&v32 + v6 - 1) == 1 )
    return 0;
  *(&v31 + v6) = 1;
  v7 = mubiao[v1 + 86];
```



再回去看 ccc，81 个元素，有些有数字，其余都是零，太明显了就是数独，然后找解数独的网站解了一下

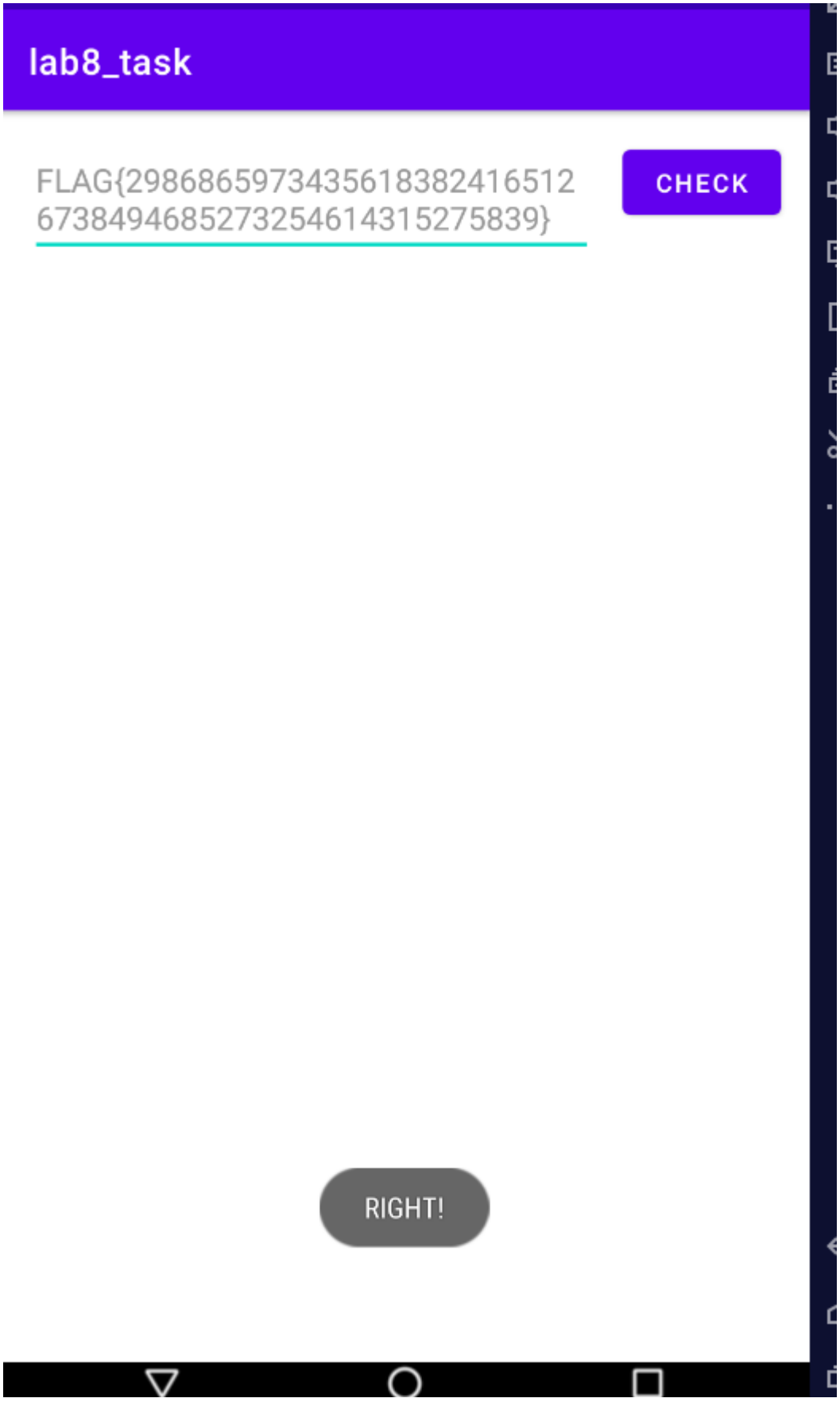
数独求解器

		1	4	3		7	5	
				1			4	2
	7				2	9		
7					9			
	5						9	
			1					3
		9	7				8	
6	8			9				
	1	7		2	6	4		

数独求解器

2	9	1	4	3	8	7	5	6
8	6	5	9	1	7	3	4	2
4	7	3	5	6	2	9	1	8
7	3	8	2	4	9	1	6	5
1	5	2	6	7	3	8	9	4
9	4	6	1	8	5	2	7	3
3	2	9	7	5	4	6	8	1
6	8	4	3	9	1	5	2	7
5	1	7	8	2	6	4	3	9

把结果直接输入


lab8_task
FLAG{29868659734356183824165126738494685273254614315275839}
CHECK
RIGHT!