


# **Angstrom CTF 2021**

# Lockpicking

 Challenge's Description: Clam wanted to have a lockpicking competition but decided against it. So, he settled for a [digital lockpicking competition](#) instead.

Nếu không thể truy cập vào file source của server, bạn có thể tìm thấy nó ở [đây](#)

## TL; DR

1. Cần input để có thể mở khóa: "**The lock opens**" và cùng với thông báo này, file **flag.txt** sẽ được mở để print flag. Đây cũng lại là flag đặt ở server, vì vậy để test( trong trường hợp server down) thì tạo một file flag.txt ở local
2. Các ký tự trong input chỉ thỏa mãn khi thuộc tập hợp{!, ?, @, I, M, O, i, m, o}. Lượng input được nhập là giới hạn và một mảng **a** có 32 phần tử sẽ thay đổi dựa vào mỗi ký tự trong input
3. Sau khi xử lý thì ptu từ **a[14]** đến **a[18]** phải khác 0, thỏa mãn, thì flag được in ra

Theo ý kiến cá nhân của mình, thì bài này thuần về suy luận logic hơn, cũng chính vì lí do đó mà với tư duy vừa gà vừa chậm của mình thì phải mất khá lâu ( hơn một ngày) mình mới hiểu được cách hoạt động, cũng như tìm được một input thỏa mãn

## Các bước giải

### Hiểu work-flow của chương trình

- Dưới đây là 3 hàm chính của chương trình hàm main, hàm **eval\_v8Array**( hàm sub15A0) và **input\_basedFunc**( hàm sub1690):
  - hàm main()

- `eval_v8Array()` - đây là hàm tính toán giá trị các phần tử của mảng( ở đây đặt tên là `v8Array`) dựa vào các ký tự input đầu vào
- `input-basedFunc()` - ở trên ta có nói mảng tính toán dựa vào input đầu vào, thực ra là thông qua hàm này, với mỗi ký tự sẽ có một case thích hợp, đặt biệt các ký tự `i`, `m`, `o`, `l`, `M`, `O` và `!` cũng sẽ tác động trực tiếp đến phần tử của mảng **`v8Array`**(`v8Arr[0]`, `v8Arr[4]`, `v8Arr[8]`, `v8Arr[13]`)

```

1  int main(__int64 a1, char **input_i, char **a3)
2  {
3      char *ptrInput; // rbx
4      __int64 *v4; // rax
5      __int64 result; // rax
6      unsigned __int64 v6; // rt1
7      int len_input; // [rsp+Ch] [rbp-45Ch]
8      __int64 v8Arr; // [rsp+10h] [rbp-458h]
9      char v9; // [rsp+1Ch] [rbp-44Ch]
10     char input; // [rsp+30h] [rbp-438h]
11     unsigned __int64 v11; // [rsp+438h] [rbp-30h]
12
13     v11 = __readfsqword(0x28u);
14
15     sub_55FDF1FE1390((__int64)&v8Arr); // khởi tạo cho v8
16     len_input = 0;
17     while (true)
18     {
19         if (v8[12]) // v10 khởi tạo = 0, v10
20             suspiciousFun1((signed int *)&v8Arr); // CHÚ Ý HÀM NÀY
21
22         printf("> ", input_i);
23
24         if ( !fgets(&input, 1024, stdin) ) // get input thất bại
25         {
26             puts("You walk away from the lock.");
27             goto FAIL_LABEL;
28         }
29
30         int i = 0
31         while(ptrInput[i] == '\0' || ptrInput[i] == '\n')
32         {
33             if (input_basedFunc(&v8Arr, ptrInput[i], &len_input) ) {
34                 puts("Your actions prove to be ineffective against the lock.");
35                 goto FAIL_LABEL;
36             }
37             i++;
38         }
39         // v7 > 164 cũng thất bại
40         if ( len_input > 164 )

```

```

41     break;
42
43     v4 = &v8Arr; // v4 tham chiếu đến địa c
44     while (v4[14]) // check v8[14]
45     {
46         v4 = v4 + 1;
47         if (v8[5] == v4) // vì v9 nằm ở rsp + 0x15, v8(v4 được gán bằng v8) n
48             // thì trong khi v8[14] !=
49         {
50             puts("The lock opens.");
51             getFlag();
52             goto FAIL_LABEL;
53         }
54     }
55 }
56 puts("Your hands grow weary from the stiffness of the lock.");
57 FAIL_LABEL:
58 v6 = __readfsqword(0x28u);
59 result = v11 - v6;
60 if ( v11 == v6 )
61     result = 0LL;
62 return result;
63 }

```

```

1  int eval_v8Array(__int64 v8)
2  {
3      int ele_4; // ecx
4      int ele_8; // edx
5      char ele_15; // al
6      char ele_13; // si
7      signed __int64 result; // rax
8
9      ele_4 = v8[4];
10     ele_8 = v8[8];
11     if ( ele_4 == 11 && v8[0] == 22 )
12         v8[18] = ele_8 == 7; // *
13     ele_15 = v8[15];
14     ele_13 = v8[13];
15     if ( ele_8 == 7 )
16     {
17         if (!ele_15 || (ele_15 = v8[18] == 0 )
18         {
19             ele_15 = 0;
20             if ( ele_4 == 8 && v8[0] == 9 )
21             {
22                 ele_15 = v8[16];
23                 if ( ele_15 )
24                     ele_15 = ele_13 ^ 1;

```

```

25     }
26 }
27 v8[15] = ele_15;           // *
28 }
29 else if ( ele_8 == 13 && ele_4 == 37 && v8[0] == 6 )
30 {
31     if ( !ele_13 )
32         v8[16] = 1;         // *
33     if ( !ele_15 )
34         goto LABEL_8;
35 }
36
37 if ( ele_15 == ele_13 )
38     v8[16] = 0;             // *
39
40
41 if ( ele_15 )
42 {
43 LABEL_8:
44     if ( v8[18] && !v8[16] )
45         v8[17] = 1;         // *
46 }
47
48 if ( ele_8 | ele_4 || (result = 1LL, v8[0]) )
49 {
50 LABEL_12:
51     result = 0LL;
52     v8[14] = 0;             // *
53     return result;
54 }
55 v8[14] = 1;                 // *
56 return result;
57 }

```

```

1  int input_basedFunc(__int64 v8, char input_i, _DWORD *len_input)
2  {
3      int v3; // ecx
4      int result; // rax
5      int tmp_o;
6      int tmp_i;
7      int tmp_m; // edx
8
9      v3 = (*len_input)++;
10     if ( input_i == '!' )
11     {
12         (v8[13]) ^= 1u;
13
14     GOOD:

```

```

15     eval_v8Array(v8);
16     result = 0;
17 }
18 else {
19     switch ( input_i )
20     {
21         case '?':
22             suspiciousFun1(v8);
23             --len_input;
24             goto GOOD
25         ;
26
27         case '@':
28             v8[12] ^= 1;
29             *len_input = v3;
30             goto GOOD
31         ;
32         case 'i':case 'I':
33         {
34             if(input_i == 'i')
35                 v8[0] = (v8[0] + 1) % 28;
36             else v8[0] = (v8[0] + 27) %28
37             goto GOOD
38         };
39         case 'm':case 'M':
40         {
41             if(input_i == 'm')
42                 v8[4] = (v8[4] + 1) % 44;
43             else v8[4] = (v8[4] + 43) % 44;
44             goto GOOD
45         };
46         case 'o':case 'O':
47         {
48             if (input_i == 'o')
49                 v8[8] = (v8[8] + 1) % 56;
50             else v8[8] = (v8[8] + 55) %56;
51             goto GOOD
52         };
53         default:
54             result = 1LL;
55             break;
56     }
57 } //kết thúc câu lệnh else
58
59 return result;
60 }

```

**Sinh một input thỏa mãn**

- Sau khi đã hiểu được hoạt động của file thực thi, mục tiêu của chúng ta là sinh ra input để v14 - v18 phải bằng 1:

```

1 Set theo thứ tự:
2
3 + v8Arr[18] = 1( viết tắt là v18): "IIIIIImmmmmmmmmmmooooooooo"
4     Lúc này v0 = 22, v4 = 11, v8 = 7
5
6 + v16 = 1: "iiiiiiiiiimmmmmmmmmmmmmmmmmmmoooooooo"
7     Lúc này v0 = 6, v4 = 37, v8 = 13
8
9 + "!"
10    input này để v13 = 1
11
12 + "iiimmmmmmmmmmmmmmmmmmm000000"
13    Lúc này v0 = 9, v4 = 8, v8 = 7
14
15 + input: "!" để v13 = 0,
16    vì v0, v4, v8 giữ nguyên nên v15 = 1
17
18 + input: "!" sinh ra v13 = 1, và vì v13 = v15 nên v16 = 0,
19    cùng lúc đó v16 = 0 và v18 = 1, nên v17 = 1. Lúc này v0 = 9, v4 = 8, v8
20 + input: "IIIMMMMMMMMMMMMMMMoooooooo!" sinh ra v0 = 6, v4 = 37, v8 = 13
21    v13 = 0 và (v0 = 6, v4 = 37, v8 = 13) nên v16 = 1
22 + "IIIIIImmmmmmm00000000000000"
23    v0 = v4 = v8 = 0 nên v14 = 1
24 ==> như vậy lúc này v14- v18 đã được set bằng 1, sinh ra flag

```

[illegible]