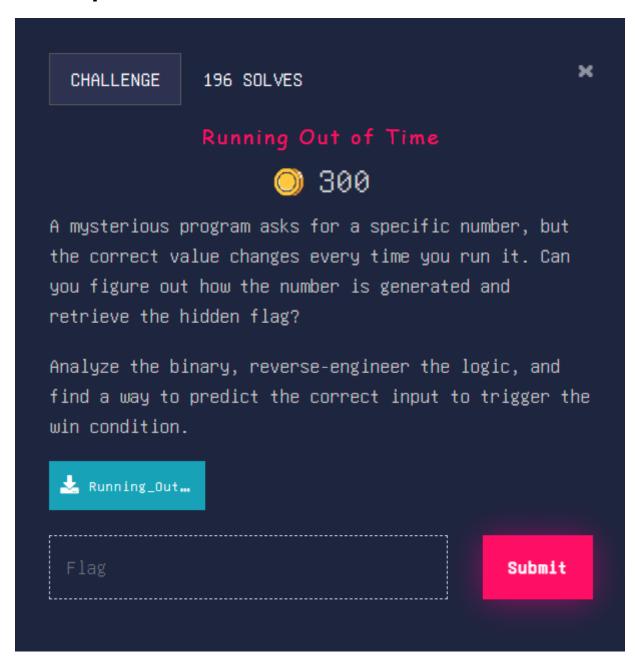
Running_out_of_time

Description



Category

#binary

Solution

The obvious thing to do when I see an elf file is to check out in Ghidra. I find a main function comparing the string with a local_c value, if corrects it calls a function.

```
2 int __cdecl main(int _Argc,char **_Argv,char **_Env)
3
4 {
5
    int iVarl;
     __time64_t _Var2;
    FILE * File;
    char local 38 [44];
8
    int local_c;
.0
.1
.2
.3
.4
.5
    __main();
_Var2 = time((__time64_t *)0x0);
    srand((uint)_Var2);
    local c = rand();
    local_c = local_c % 100;
    printf("Provide the correct value: ");
    _File = __iob_func();
.8
    fgets(local_38,0x20,_File);
    iVarl = atoi(local_38);
20
    if (iVarl == local_c) {
21
      p3xr9q_t1zz();
22
    }
23
    else {
24
      puts("Incorrect. Please try again.");
25
26
    return 0;
27 }
28
```

So I check what that function does:

```
Decompile: p3xr9q t1zz - (Running Out Of Time.exe)
 2 void p3xr9q_t1zz(void)
 3
 4 {
    byte local 28 [27];
   byte local_d;
 7
    uint local_c;
 8
   local 28[0] = 0xld;
10
   local 28[1] = 0x1b;
11
    local_28[2] = 0x47;
12
   local 28[3] = 0x19;
13
   local_28[4] = 0x75;
   local 28[5] = 0x1f;
14
15
    local_28[6] = 0x1d;
16
   local 28[7] = 0xla;
17
    local_28[8] = 0x5a;
18
    local_28[9] = 0x5a;
19
   local_28[10] = 0x19;
20
   local 28[11] = 0x4e;
    local d = 0x2a;
21
    printf("Success! Here is your output: ");
22
23
    for (local_c = 0; local_c < 0xc; local_c = local_c + 1) {</pre>
24
       putchar((int)(char)(local_28[(int)local_c] ^ local_d));
25
26
    putchar(10);
27
    return:
28 }
29
```

The loop performs a XOR operation for each character in the local_28 array So technically, this problem is solved by understanding the output encryption and not even the input sanitization.

Which is literally the flag.