# Overlong (FlareOn CTF)

Description:

The secret of this next challenge is cleverly hidden. However, with the right approach, finding the solution will not take an overlong amount of time.

Perhaps this is another cool easy challenge which i really overlooked and stuck for a long time. Thanks to 0xdf for driving me in the right direction.
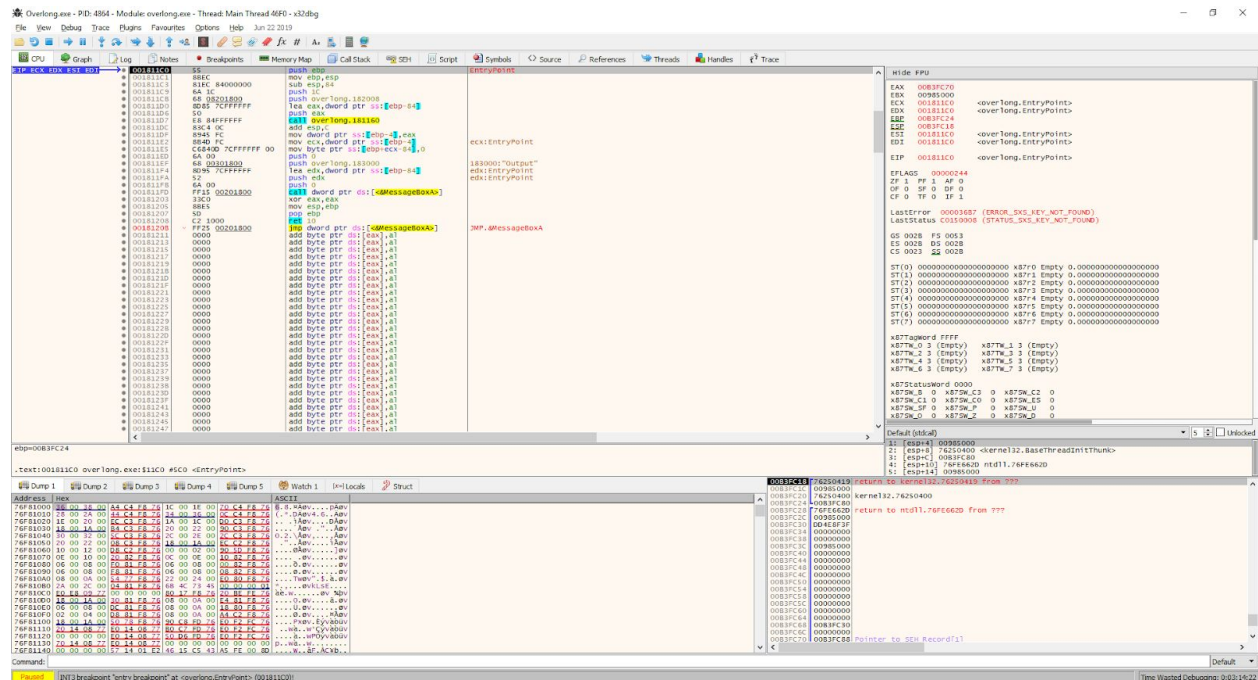
Do you see overlong ? well keep that in mind. Let's start!

Challenge:

Opening overlong.exe result in
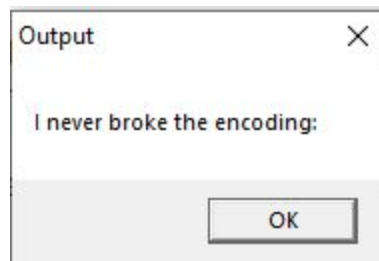
As it was a 32bit binary, I've used x32dbg for this challenge.

Now we are at the entry point of the binary. Stepping through we can see that a function **overlong.181160** being called with certain arguments.

To make it a bit clear i loaded this exe in IDA (Interactive DisAssembler) as well as in Ghidra.

We can see FUN_00401160 takes three arguments

```
C  Decompile: FUN_00401160 - (Overlong.exe)
1
2  uint __cdecl FUN_00401160(byte *param_1,byte *param_2,uint param_3)
3
4  {
5    byte bVar1;
6    int iVar2;
7    uint local_8;
8
9    local_8 = 0;
10   while( true ) {
11     if (param_3 <= local_8) {
12       return local_8;
13     }
14     iVar2 = FUN_00401000(param_1,param_2);
15     param_2 = param_2 + iVar2;
16     bVar1 = *param_1;
17     param_1 = param_1 + 1;
18     if (bVar1 == 0) break;
19     local_8 = local_8 + 1;
20   }
21   return local_8;
22 }
23
```

```
public start
start proc near

Text= byte ptr -84h
var_4= dword ptr -4

push    ebp
mov     ebp, esp
sub     esp, 84h
push    1ch
push    offset unk_402008
lea     eax, [ebp+Text]
push    eax
call    sub_401160
```

First argument is being ebp which carries the base pointer reference and second argument is 1Ch (28 decimal value) being pushed to stack which is param_3 in our case. As a final argument it does loading effective address of encrypted flag at unk_402008 location.

```
.rdata:00402008 unk_402008       db 0E0h ; à                   ; DATA XREF: start+B↑o
.rdata:00402009                  db  81h
.rdata:0040200A                  db  89h ; ‰
.rdata:0040200B                  db 0C0h ; À
.rdata:0040200C                  db 0A0h ;
.rdata:0040200D                  db 0C1h ; Á
.rdata:0040200E                  db 0AEh ; ®
.rdata:0040200F                  db 0E0h ; à
.rdata:00402010                  db  81h
.rdata:00402011                  db 0A5h ; ¥
.rdata:00402012                  db 0C1h ; Á
.rdata:00402013                  db 0B6h ; ¶
.rdata:00402014                  db 0F0h ; ð
.rdata:00402015                  db  80h ; €
.rdata:00402016                  db  81h
.rdata:00402017                  db 0A5h ; ¥
.rdata:00402018                  db 0E0h ; à
.rdata:00402019                  db  81h
.rdata:0040201A                  db 0B2h ; ²
.rdata:0040201B                  db 0F0h ; ð
.rdata:0040201C                  db  80h ; €
.rdata:0040201D                  db  80h ; €
.rdata:0040201E                  db 0A0h ;
.rdata:0040201F                  db 0E0h ; à
.rdata:00402020                  db  81h
.rdata:00402021                  db 0A2h ; ¢
.rdata:00402022                  db  72h ; r
.rdata:00402023                  db  6Fh ; o
.rdata:00402024                  db 0C1h ; Á
.rdata:00402025                  db 0ABh ; «
.rdata:00402026                  db  65h ; e
.rdata:00402027                  db 0E0h ; à
.rdata:00402028                  db  80h ; €
.rdata:00402029                  db 0A0h ;
.rdata:0040202A                  db 0E0h ; à
.rdata:0040202B                  db  81h
.rdata:0040202C                  db 0B4h ; ´
.rdata:0040202D                  db 0E0h ; à
.rdata:0040202E                  db  81h
.rdata:0040202F                  db 0A8h ; ¨
.rdata:00402030                  db 0C1h ; Á
.rdata:00402031                  db 0A5h ; ¥
.rdata:00402032                  db  20h
.rdata:00402033                  db 0C1h ; Á
.rdata:00402034                  db 0A5h ; ¥
.rdata:00402035                  db 0E0h ; à
```

Then it checks for length of the input. If length is either 0 or less than that it assigns length as 0 and calls the other function FUN_00401000 with pointer to flag being as an argument.

```
while( true ) {
  if (param_3 <= local_8) {
    return local_8;
  }
  iVar2 = FUN_00401000(param_1,param_2);
```

Then i believe it does some logical operations to decrypt the flag. As i only interested in length argument i skipped exploring this decryption part.

As challenge states about **Overlong** we can try increasing the length argument and get the remaining part of the flag.

I just changed **push 1C** to **push 4F**

```
001811C9    6A 4F           push  4F
001811CB    68 08201800     push  overlong  182008
```

Output   ✕

I never broke the encoding:
I_a_M_t_h_e_e_n_C_o_D_i_n_g@flare-on.com

                                              OK