REVERSE InnoCTF Writeup

I. Call me

```
reisen1943@ubuntu:~/Documents/CTF/InnoCTF/RE/call_me$ file call_me
call_me: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked,
27993400479149ef55fef49ec50f96f58e, with debug_info, not stripped
```

Try to run it, but we get nothing much

```
reisen1943@ubuntu:~/Documents/CTF/InnoCTF/RE/call_me$ ./call_me

There is nothing...
```

Load it into IDA, we can see the flag() function.

Search for flag(), break the main and run the program

```
pwndbg> p &flag
$2 = (int (*)()) 0x555555555159 <flag>
pwndbg> b*main
Breakpoint 1 at 0x5555555554f3: file var 3.c, line 19.
```

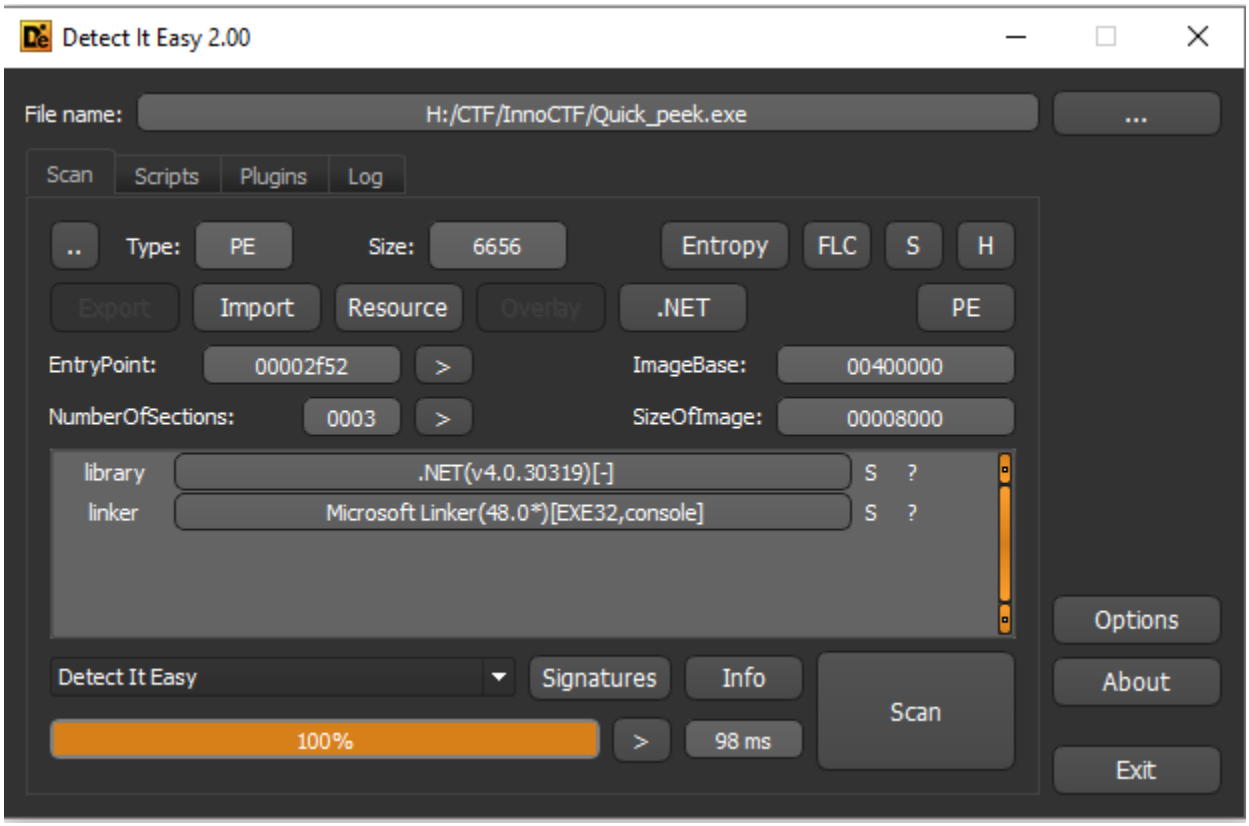
We modify the rip register and let them program continue

```
pwndbg> set $rip = 0x555555555159
```

```
pwndbg> c
Continuing.

Your flag:
InnoCTF{How_d1d_y0u_f1nd_m3_2003bac}ÿ
[Inferior 1 (process 2449) exited with code 047]
pwndbg>
```

II. Quick peek



Hmm .Net so we need a .Net decompiler here. I use ILspy

Try to run it

Load into the ILspy and read the code

Flag: InnoCTF{1337_SPAgh377i_CoD3}

III. Crack_me

```
PS H:\CTF\InnoCTF> .\crack_me.exe
Enter the key: 123123123
Invalid key!
PS H:\CTF\InnoCTF>
```

We need to find the key. This binary is 32-bit so we use IDA for decompiler

We didn’t see main function so maybe the binary is stripped. Let find for “Invalid key” string. So we can see the function that use this string to display. I rename it to “Fail_Function”

```
setlocale(0, "Russian");
sub_4022C0(&v26);
v27 = 0;
sub_401440(std::cout, "Enter the key: ");
sub_4013F0(std::cin, &v26);
i = sub_403310("_", 0);
if ( i == -1 || sub_403310("_", i + 1) == -1 )
  Fail_Function();
if ( sub_4033F0(&v26) != 15 )
  Fail_Function();
if ( *(_BYTE *)sub_402850(0) != 114 )
  Fail_Function();
v21 = sub_4035A0((int)&v15, 1, 6u);
v20 = v21;
LOBYTE(v27) = 1;
v25 = sub_4017A0(v21, "everse");
LOBYTE(v27) = 0;
sub_4025E0(&v15);
if ( v25 )
  Fail_Function();
if ( *(_BYTE *)sub_402850(8) != 105 || (v0 = *(char *)sub_402850(8), *(char *)sub_402850(9) + v0 != 220) )
```

We can see the cout and cin function. The sub_403310() is finding for “_” char. We can see 2 function will be called. So we can surely say that the string must contain 2 char “_”.

Sub_4033F0() is strlen. So the key must have 15 character.

Sub_402850() is check for the first char in the key (0 index). The first char is “r”

Next the program split the key from 1 – 6 index and compare with “everse”.

Now we know the key have first part is “reverse”.

```
if ( v25 )
  Fail_Function();
if ( *(_BYTE *)sub_402850(8) != 105 || (v0 = *(char *)sub_402850(8), *(char *)sub_402850(9) + v0 != 220) )
  Fail_Function();
v13 = "fine";
v12 = (char *)-6;
v16 = &v6;
sub_4035A0((int)&v6, 11, 4u);
v19 = sub_403090(&v14, v6, v7, v8, v9, v10, v11, v12);
v18 = v19;
LOBYTE(v27) = 2;
v24 = sub_4017A0(v19, v13);
v23 = v24;
LOBYTE(v27) = 0;
sub_4025E0(&v14);
if ( v23 )
  Fail_Function();
for ( i = sub_403310("e", 0); i != -1; i = sub_403310("e", 0) )
  *(_BYTE *)sub_402850(i) = 51;
v13 = "}\n";
v12 = &v26;
```

Next the program compare the 9th char with ‘i’. And the next char is ‘s’ because ‘i’ + ‘s’ = 220. After that, we know the flag is “reverse_is_XXXX”

```
.text:0040311A                call    sub_402850
.text:0040311F                movsx   ecx, byte ptr [eax]
.text:00403122                mov     edx, [ebp+arg_1C]
.text:00403125                lea     eax, [ecx+edx-41h]
.text:00403129                cdq
.text:0040312A                mov     ecx, 1Ah
.text:0040312F                idiv    ecx
.text:00403131                add     edx, 41h
.text:00403134                push    edx
.text:00403135                lea     ecx, [ebp+var_28]
.text:00403138                call    sub_402980
.text:0040313D                jmp     short loc_403169
```

We see that the program move value of the char in ecx, load 0xFFFFFFFA in edx and calculate it. In short: char – 6 – 0x41. And then it add the result with 0x41.

After the program compare the result with “fine”. So we can write a little script to calculate the final part.

```
1  final = "fine"
2  result = ""
3  for i in final:
4      for j in range(33,128):
5          if j - 6 - 0x41 + 0x41 == ord(i):
6              result += chr(j)
7  print result
```

Result: lotk

So we have the key: “reverse_is_lotk”

```
PS H:\CTF\InnoCTF> .\crack_me.exe
Enter the key: reverse_is_lotk
Success!!!
InnoCTF{r3v3rs3_is_lotk}
Press any key to continue . . .
```