DESCRIPTION

You've probably noticed that all flags are encrypted with some kind of algorithm. Your job is to find out what algorithm it is, by reverse engineering it. Once you've done that, there is another flag in the hidden binary, which you must decrypt.

The decryption key is the name of the algorithm in lower case.

You can start the reversing process by looking at the enc_init function inside the "hidden" binary. Try to write a pseudocode of that function. This will give you a hint about what the algorithm is.

RESOURCES

There were no additional resources given for this task. However, there is a hint given to start from the **enc_init** function inside the **hidden** binary. That being said, the decompilation of the algorithm inside **enc_init** function is in the following picture and the algorithm implemented is RC4.

```
uVar25 = 0;
iVar26 = 0;
do {
  bVar1 = param_1[uVar25];
  iVar26 = (int)((uint)*(byte *)(param_2 +
                                 (int)((long)((ulong)(uint)((int)uVar25 >> 0x1f) << 0x20 |
                                              uVar25 & 0xffffffff) % (long)param_3)) +
                (uint)bVar1 + iVar26) % 0x100;
  param_1[uVar25] = param_1[iVar26];
  uVar25 = uVar25 + 1;
  param_1[iVar26] = bVar1;
} while (uVar25 != 0x100);
```

APPROACH

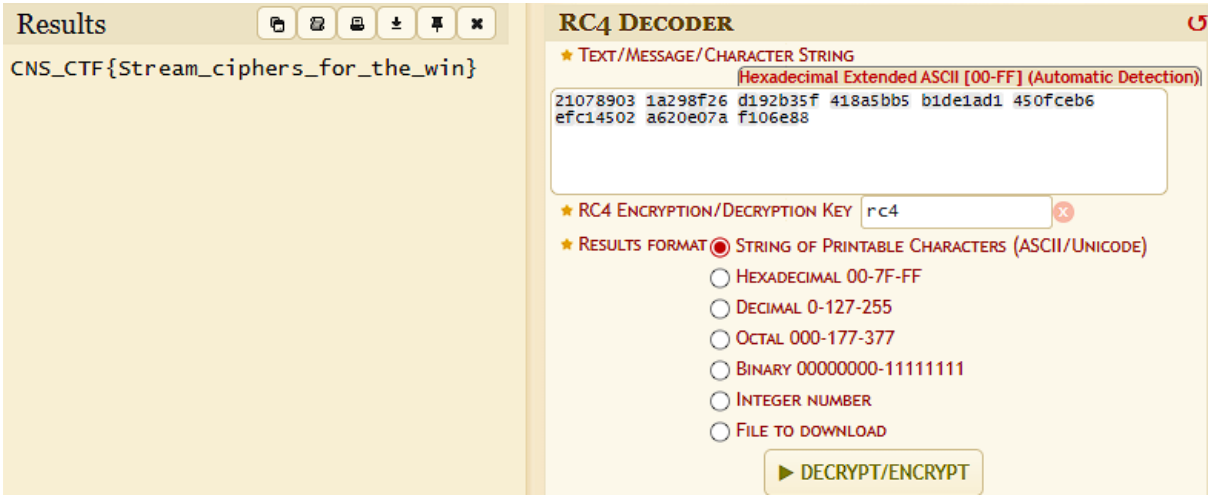
1. After finding out what algorithm it is, I noticed that in the **hidden** binary from the first task there is another flag present inside the binary called **bonus_flag**.

```
mihnea@HOME-PC:/mnt/c/Users/mblot/Desktop/CNS/Tema1_Moodle/hidden$ nm hidden | grep -E " d | D "
0000000000403e08 d _DYNAMIC
0000000000403fe8 d _GLOBAL_OFFSET_TABLE_
0000000000404210 D __TMC_END__
0000000000404020 D __data_start
0000000000403e00 d __do_global_dtors_aux_fini_array_entry
0000000000404028 D __dso_handle
0000000000403df8 d __frame_dummy_init_array_entry
000000000040420b D _edata
00000000004040c0 D bonus_flag
0000000000404140 D msg1
0000000000404124 D t_val
```

2. The bonus_flag is present at the address **0x004040c0 in the .data section.**

```
0x004040c0 21078903 1a298f26 d192b35f 418a5bb5 !....).&..._A.[.
0x004040d0 b1de1ad1 450fceb6 efc14502 a620e07a ....E.....E.. .z
0x004040e0 f106e888 c7c11eed ca952386 4626c80b ..........#.F&..
0x004040f0 1feec116 f54022b1 b8654c65 b94c3eb7 .....@"..eLe.L>.
0x00404100 ebf1507e af406da6 9ad2ee3a 93623254 ..P~.@m....:.b2T
0x00404110 f268698e cc6e99cd 37ded145 540474f8 .hi..n..7..ET.t.
0x00404120 43bf0395 52554c5a 00000000 00000000 C...RULZ........
```

3. Finding the bytes at the specific address, I used an online RC4 decoder with "**rc4**" key to find the flag.



4. The searched flag for this task is:

CNS_CTF{Stream_ciphers_for_the_win}