Write-up_picoCTF_2021

Chall: Easy as GDB

You can find out the source of challenge and script here(source, script)

> The name of function, variable in my writeup will called as the way they displays on IDA's disassembly window.

Proceed to solve

1 - Your input will XORed with multiple keys which generated from the loop of function ***sub_82b()*** function

```
char *__cdecl sub_82B(char *src, size_t n)
{
  unsigned int i; // [esp+0h] [ebp-18h]
  char *dest; // [esp+Ch] [ebp-Ch]
  size_t na; // [esp+24h] [ebp+Ch]

  na = (n & 0xFFFFFFFC) + 4;
  dest = (char *)malloc(na + 1);
  strncpy(dest, src, na);
  for ( i = 0xABCF00D; i < 0xDEADBEEF; i += 2075469 )
    sub_6BD((int)dest, na, i);                    // function will xor input with i( "i" as the xor-key)
  return dest;
}
```

Input will xored with multiple xor-key before continuing

> IDA's Decompiler(pseudo-code - F5) sometimes makes some mistakes. Be careful!(i.e it happened in this challenge)

2 - I debugged this program at ***sub_7C2(array, size, option)*** to recognize that this function will shuffle an array with option(1 or -1). Furthermore, two options are inverse. If I take a ***result array*** generated with option -1 as the first argument of ***sub_7C2()***, and select the option 1, the function will return the original array:

+ sub_7C2(original_array, size, 1)--> result_array
+ sub_7C2(result_array, size, -1) --> original_array

3 - After shuffling the encrypted input, it will compared to the target array which be available at ***.data:00002008*** and print correct or no.

Appendix

There are some mistakes of IDA's decompiler I discovered while solving following below:
1.

```
int sub_9AF()
{
  char *s; // ST14_4
  int v1; // ST08_4
  char *v2; // eax
  char *src; // ST1C_4
  size_t n; // ST18_4
  int v6; // [esp-4h] [ebp-1Ch]

  s = (char *)calloc(0x200u, 1u);
  printf("input the flag: ");
  fgets(s, 512, stdin);
  v2 = (char *)strnlen(&unk_2008, 512, v1, v6);
  src = (char *)sub_82B(v2, (size_t)v2);
  sub_7C2((int)src, 1u, 1);
  if ( sub_8C4(src, n) == 1 )
    puts("Correct!");
  else
    puts("Incorrect.");
  return 0;
}
```

- The ***first*** argument of ***sub_82B()*** is not ***v2***, it's ***s*** variable(your input). Let's check again with assembly windows:

```
call    _strnlen
add     esp, 10h
mov     [ebp+n], eax
sub     esp, 8
push    [ebp+n]          ;   2nd arg
push    [ebp+s]          ;  1st arg
call    sub_82B
```

2.

```c
int sub_9AF()
{
  char *s; // ST14_4
  int v1; // ST08_4
  char *v2; // eax
  char *src; // ST1C_4
  size_t n; // ST18_4
  int v6; // [esp-4h] [ebp-1Ch]

  s = (char *)calloc(0x200u, 1u);
  printf("input the flag: ");
  fgets(s, 512, stdin);
  v2 = (char *)strnlen(&unk_2008, 512, v1, v6);
  src = (char *)sub_82B(v2, (size_t)v2);
  sub_7C2((int)src, 1u, 1);
  if ( sub_8C4(src, n) == 1 )
    puts("Correct!");
  else
    puts("Incorrect.");
  return 0;
}
```

- The second argument of sub_7c2() isn't 1, it's n(length of your input), check again with assembly window:

```
call    sub_82B
add     esp, 10h
mov     [ebp+src], eax
sub     esp, 4
push    1                3rd arg
push    [ebp+n]          2nd arg
push    [ebp+src]        1st arg
call    sub_7C2
```