Crackmes thellurik

Challenge: https://crackmes.one/crackme/5ab77f5333c5d40ad448c119
Difficulty: Medium (3)
Platform: Unix/linux etc.
Compiled: C++ / C

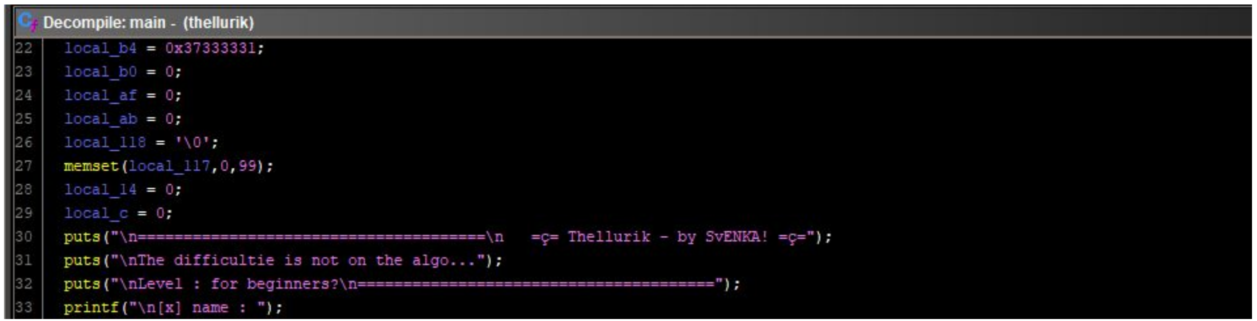
Initial Inspection:

```
file thellurik
        thellurik: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
        dynamically l  inked, interpreter /lib/ld-linux.so.2, missing section headers
```

```
strings thellurik

        =======================================
          =
        = Thellurik - by SvENKA! =
        The difficultie is not on the algo...
        Level : for beginners?
        ========================================
        [x] name :
        [x] c0de :
        ~%s#%s#~
        well done,now try to write a tut if you want !
        try again muahaha
```

Looking Into The Source Code Using Ghidra:



Decompile: main - (thellurik)

```
  local_b4 = 0x37333331;
  local_b0 = 0;
  local_af = 0;
  local_ab = 0;
  local_118 = '\0';
  memset(local_117,0,99);
  local_14 = 0;
  local_c = 0;
  puts("\n========================================\n   =ç= Thellurik - by SvENKA! =ç=");
  puts("\nThe difficultie is not on the algo...");
  puts("\nLevel : for beginners?\n========================================");
  printf("\n[x] name : ");
```

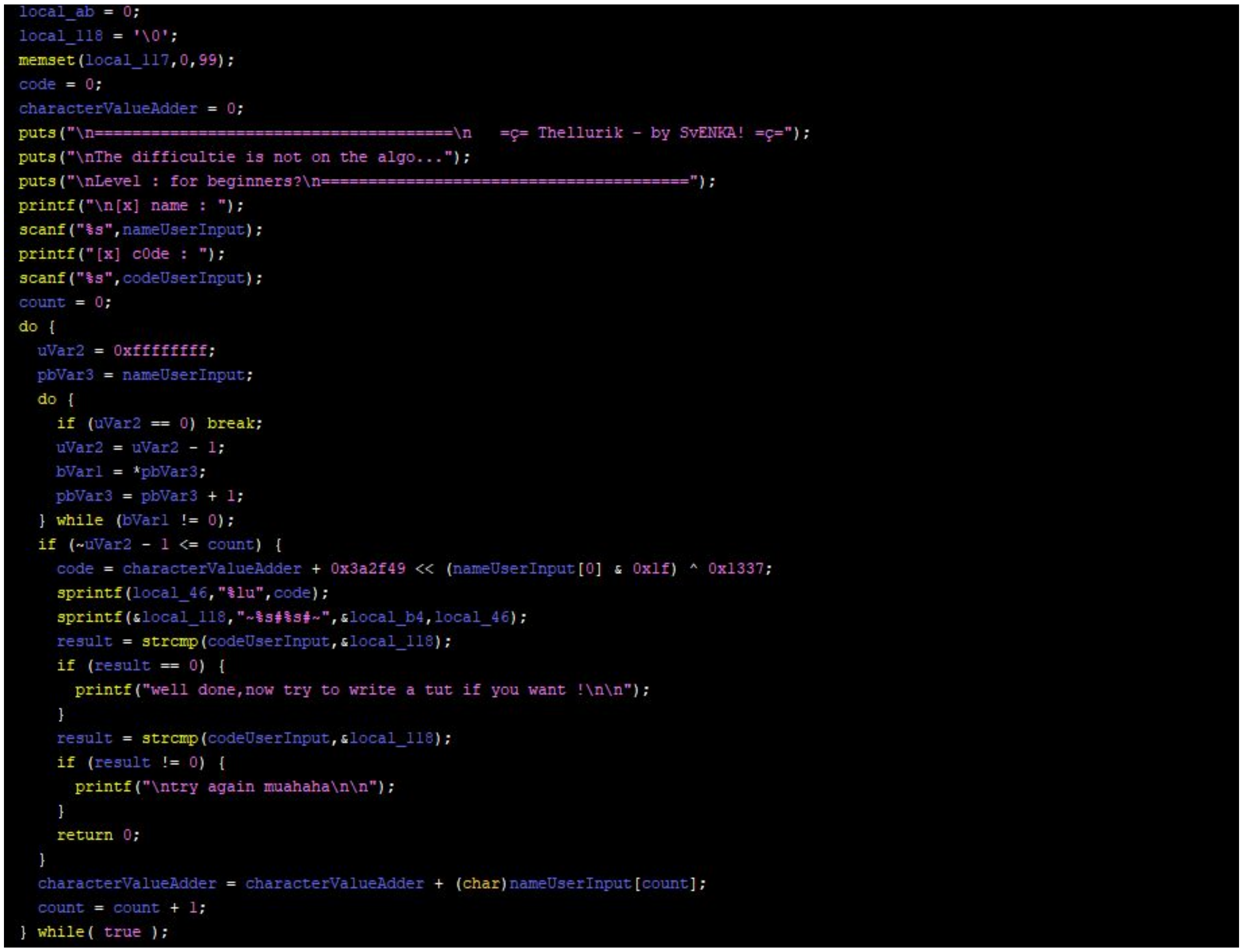
There are a couple of lines in this function that are immediately catching my attention.

```
Line 33:    printf("\n[x] name : ")
Line 34:    scanf("%s",local_78);
Line 35:    printf("[x] c0de : ");
Line 36:    scanf("%s",local_aa);
Line 48:    local_14 = local_c + 0x3a2f49 << (local_78[0] & 0x1f) ^ 0x1337;
Line 51:    result = strcmp(local_aa,&local_118);
Line 61:    local_c = local_c + (char)local_78[local_10];
Line 62:    local_10 = local_10 + 1;
```

1. Looks line 34 takes the users name (user input) and stores it into local_78
2. At line 36: It takes the users code (user input) and stores it into local_aa
3. Line 48 Looks like some type of small level of encryption
4. Looks like on line 51 it compares local_aa which is the user code with an unknown variable.

5. On Line 61 it takes the local_78 variable we saw on line 4 and takes each character value and adds it to variable local_c.
6. Line 62 looks like a counter type variable.

Renaming The Variables In Ghidra



```c
local_ab = 0;
local_118 = '\0';
memset(local_117,0,99);
code = 0;
characterValueAdder = 0;
puts("\n======================================\n   =ç= Thellurik - by SvENKA! =ç=");
puts("\nThe difficultie is not on the algo...");
puts("\nLevel : for beginners?\n=======================================");
printf("\n[x] name : ");
scanf("%s",nameUserInput);
printf("[x] c0de : ");
scanf("%s",codeUserInput);
count = 0;
do {
  uVar2 = 0xffffffff;
  pbVar3 = nameUserInput;
  do {
    if (uVar2 == 0) break;
    uVar2 = uVar2 - 1;
    bVar1 = *pbVar3;
    pbVar3 = pbVar3 + 1;
  } while (bVar1 != 0);
  if (~uVar2 - 1 <= count) {
    code = characterValueAdder + 0x3a2f49 << (nameUserInput[0] & 0x1f) ^ 0x1337;
    sprintf(local_46,"%lu",code);
    sprintf(&local_118,"~%s#%s#~",&local_b4,local_46);
    result = strcmp(codeUserInput,&local_118);
    if (result == 0) {
      printf("well done,now try to write a tut if you want !\n\n");
    }
    result = strcmp(codeUserInput,&local_118);
    if (result != 0) {
      printf("\ntry again muahaha\n\n");
    }
    return 0;
  }
  characterValueAdder = characterValueAdder + (char)nameUserInput[count];
  count = count + 1;
} while( true );
```

Using a Radare2 To Debug The Program:

Let's set a breakpoint at line 51 and check what values are being compared.
[0xf7fc40b0]> db 0x08048601 (SET BREAKPOINT)
[0xf7fc40b0]> dc (START PROGRAM)

```
=====================================
   =ç= Thellurik - by SvENKA! =ç=

The difficultie is not on the algo...

Level : for beginners?
======================================


[x] name : drew
[x] c0de : testcode

hit breakpoint at: 8048601
[0x08048601]> drr (CHECK REGISTERS)

role reg      value    ref
————————————————————————————————————————
A0   eax      ff9b5c94  ([stack]) stack R W 0x3333317e (~1337#61021319#~) --> (invalid31) ascii ('~')
A1   ebx      0        0
A2   ecx      0        0
```

Looks like register EAX has the value of **~1337#61021319#~**. This must be the actual code.

Lets run the code and check!

```
=====================================
   =ç= Thellurik - by SvENKA! =ç=

The difficultie is not on the algo...

Level : for beginners?
======================================


[x] name : drew
[x] c0de : ~1337#61021319#~
well done,now try to write a tut if you want !
```

Understanding The Encryption

code = characterValueAdder + 0x3a2f49 << (nameUserInput[0] & 0x1f) ^ 0x1337;

Lets decode these hex values:
0x3a2f49 = 3813193
0x1f = 31
0x1337 = 4919

code = characterValueAdder + 3813193<< (nameUserInput[0] & 31) ^ 4919;

I am going to find the correct code value for the user "Alex"

Variable nameUserInput[0] would equal the integer value of "A". Which is 65

characterValueAdder = characterValueAdder + (char)nameUserInput[count];
count = count + 1;

This code looks like it takes the value of each character and adds it together for example:

A = 65
l = 108
e = 101
X = 120
Adding all those together equals: 394

code = 394+ 3813193<< (65 & 31) ^ 4919;

"code" would return as: 7631505

Code for "drew"
[x] c0de : **~1337**#61021319**#~**
Lets put the ~1337# on the alex code we created and see if it works!!

====================================
   =ç= Thellurik - by SvENKA! =ç=

The difficultie is not on the algo...

Level : for beginners?
===

[x] name : Alex
[x] c0de : ~1337#7631505#~
well done,now try to write a tut if you want !

Its Works!!!

Creating A Keygen:

Source Code:

```
def createCode(name):
    listName = list(name)
    firstInt = ord(listName[0])

    count = 0
    number = 0

    while count < len(name):
        number = number + ord(listName[count])
        count += 1

    code = number + 3813193 << (firstInt & 31) ^ 4919
    code = "~1337#" + str(code) + "#~"
    return code


code = createCode("asdasfggwgsagagasgasgasagagas")
print(code)
```

==
   =ç= Thellurik - by SvENKA! =ç=

The difficultie is not on the algo...

```
Level : for beginners?
============================================

[x] name : asdasfggwgsagagasgasgasagagas
[x] c0de : ~1337#7628039#~
well done,now try to write a tut if you want !
```

Conclusion:

This was a fun challenge. I was so happy when my keygen actually worked!! This challenge taught me how to reverse a basic encryption and also taught me more about breakpoints.