CRACKMES.ONE

Very Hard 99% you can't do it

Challenge Author: Bobx

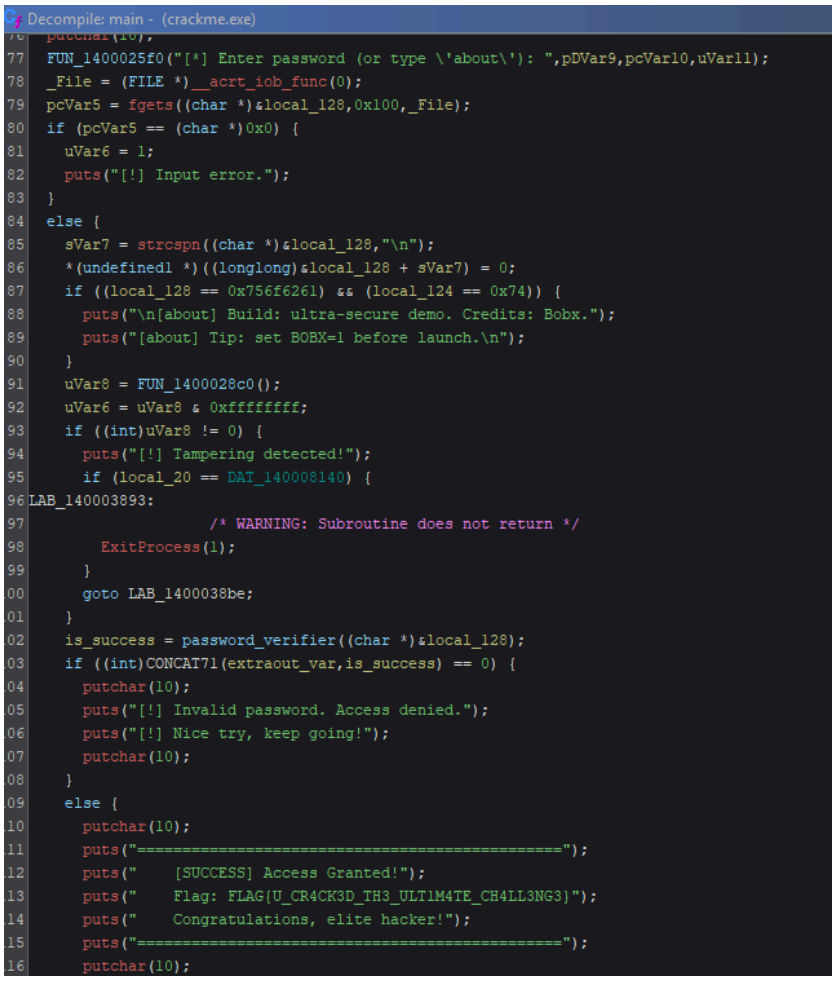
Writeup by scaredandalone

If you do your rudimentary checks like *strings* on the executable, you will find that the flag for the challenge is plain text.

However, that is no fun so let's instead reverse the password checker function.

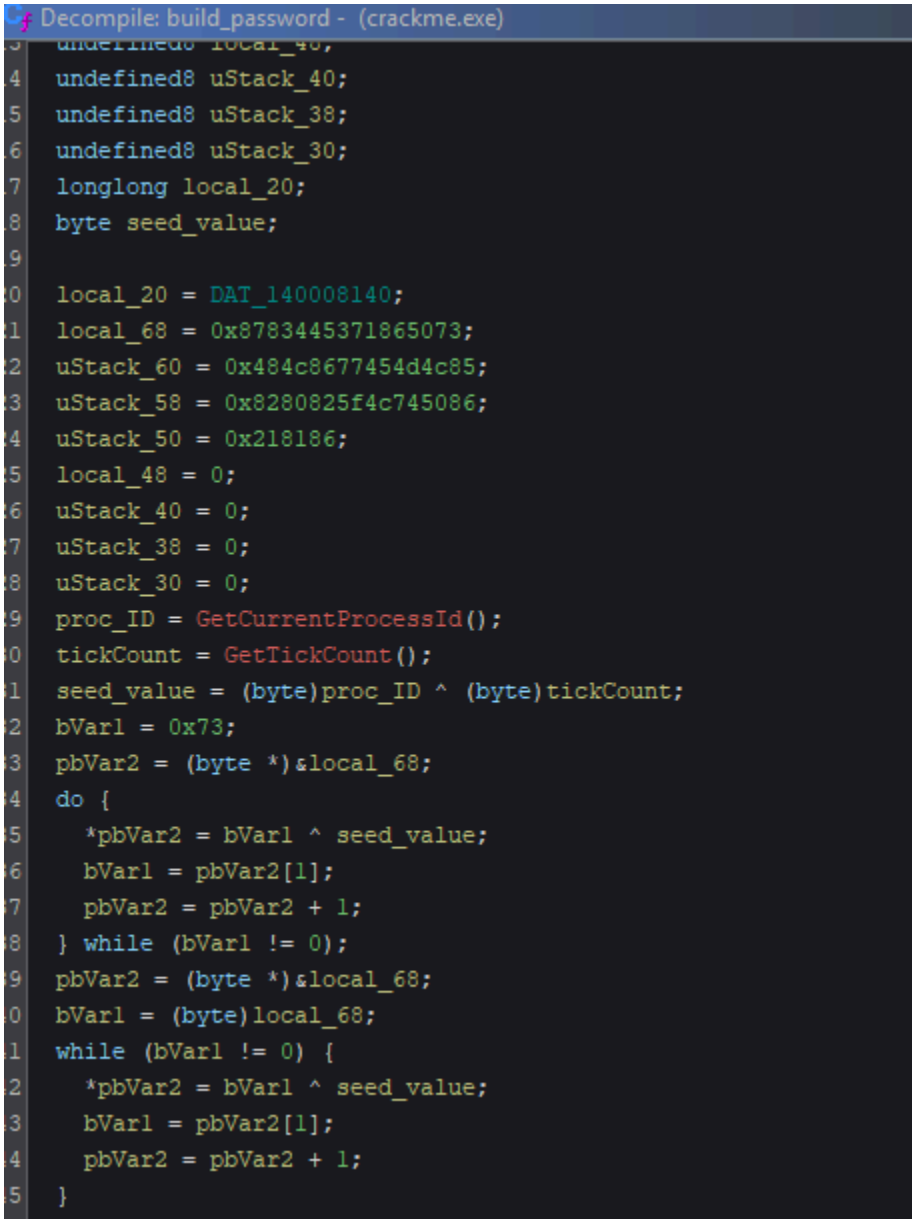
Looking through the entry function and following function calls, we can find the “main” function which contains all of the logic.

Figure 1: Main re-compiled view



We can see the flag that I was talking about earlier and the password verification function that leads to it upon success.

Figure 2: Build password function



We can see something very odd going on here.

The function is trying to hide a hard coded password by using some “dynamic” logic.

First the four 8-byte constants are placed into a local buffer (local_68, uStack_60, uStack_58, uStack_50), which together form the raw encoded bytes of the secret.

Then it reads the current process ID and the tick count since (system) startup, XORs their lower byte to get a single “seed” byte (seed_value).

This key is then used as a key in two passes over the buffer.

In the first loop, it walks forward through the bytes starting at local_68, repeatedly setting the byte at *pbVar2 = bVar1 ^ seed_value, where bVar1 is initially 0x73 and then updated to the next byte each time.

This continues until it reaches a null terminator.

However, immediately after the first pass, the second loop resets the pointer to the start of the buffer and does another walk.

Because XOR is its own inverse, and because of how bVar1 is updated from the transformed data, the second pass effectively undoes the seed-dependent scrambling from the first pass.

As a result, the final contents of the buffer converge to a fixed byte sequence that no longer depends on the process ID or tick count.

The double-XOR logic is constructed so that the final string copied out of local_68, is actually constant across all environments. Meaning, that the PID and tickcount do not actually contribute to the generation.

Password (Raw hex bytes):

73 50 86 71 53 44 83 87 85 4C 4D 45 77 86 4C 48 86 50 74 4C 5F 82 80 82 86 81 21

Flag:

FLAG{U_CR4CK3D_TH3_ULT1M4TE_CH4LL3NG3}