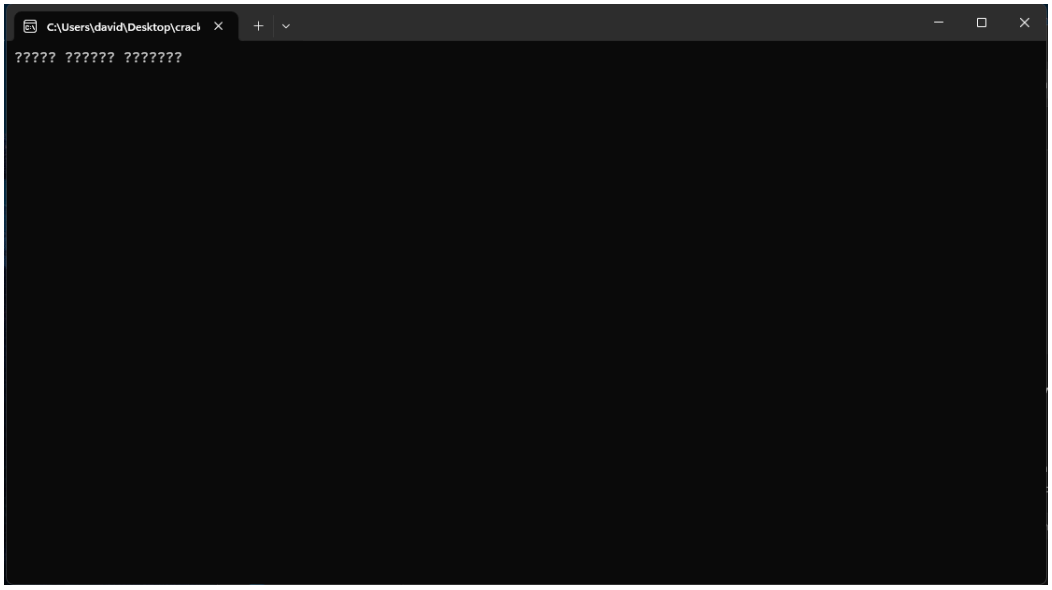
Crackme: https://crackmes.one/crackme/692272032d267f28f69b7ff5

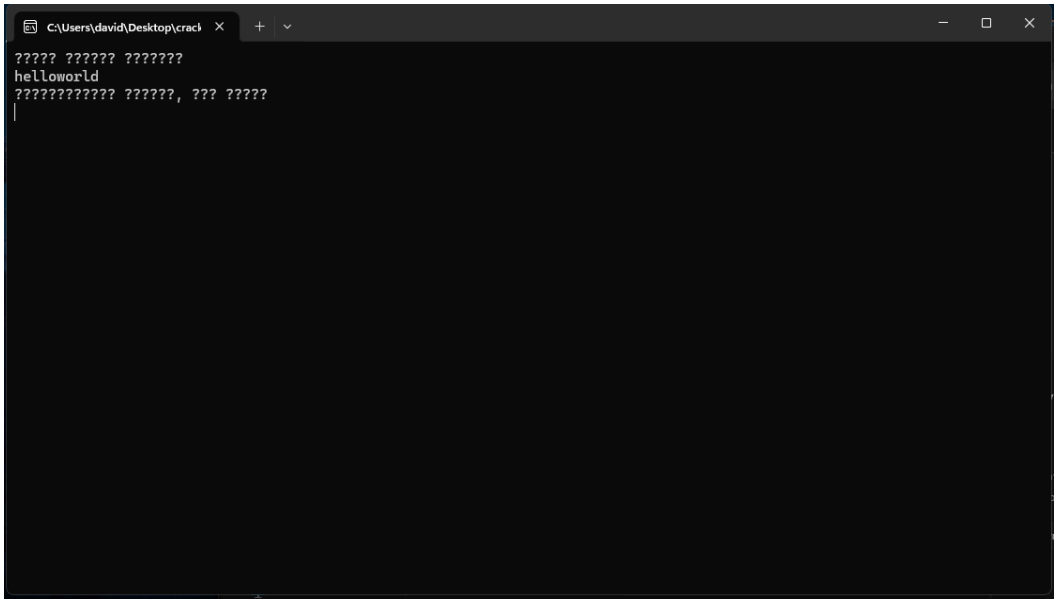
Author: plikan Software Used: x32dbg, dnSpy v6.5.1 Writeup By: SenorGPT

Platform Windows	Difficulty: 1.0	Quality: 4.0	Arch: x86-64	Language: .NET

Regular execution (startup):



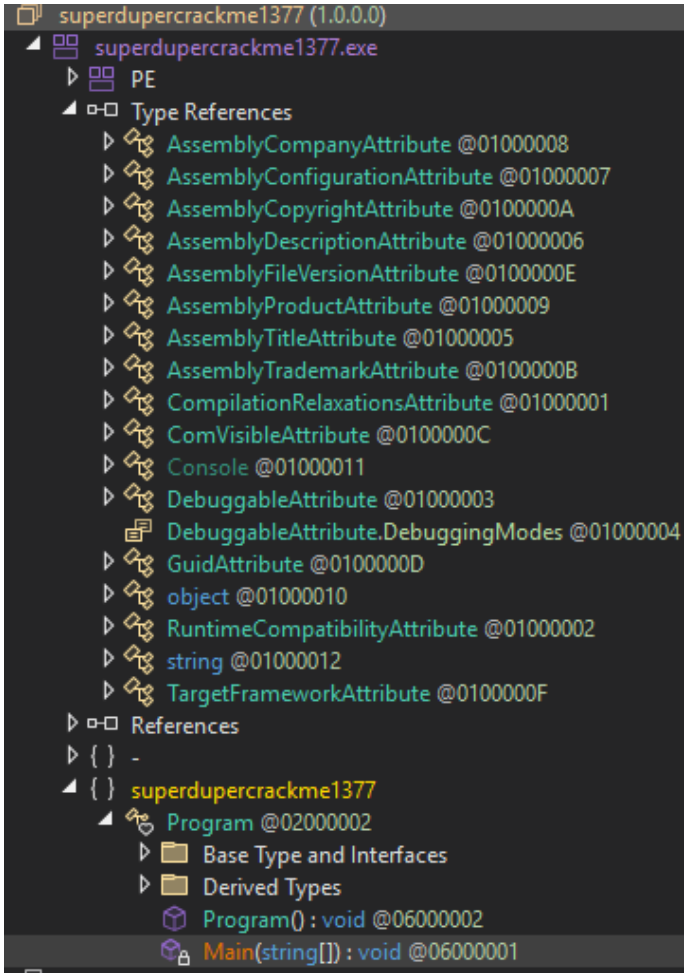
Wrong answer provided - proceeded by termination:



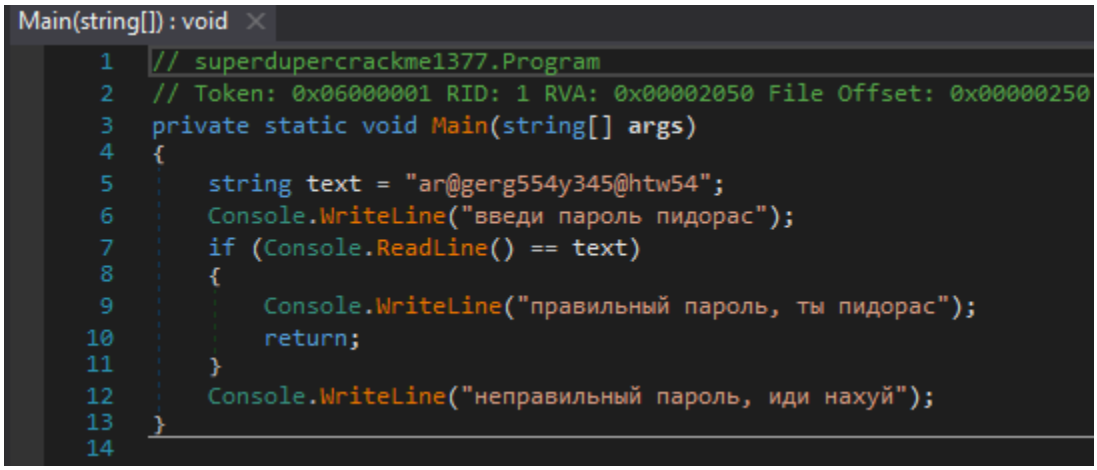
Goal: recover the correct password string (no patching required to solve), using static and/or dynamic analysis.

Upon downloading and reading the description for this crackme, it stated that ilSpy/dnSpy was the right tool for this job. Since I am fairly new to this space, I thought that I would be able to easily solve this with x32dbg.
I first attempted to find the string references for the right and wrong strings but wasn't getting anywhere.
It seems that the CLR (Common Language Runtime) turns the code into CPU instructions at runtime. Using x32dbg would add a whole heap of complexity to the challenge that I am not yet skilled enough to handle.
I saw mscoree.dll in the modules list and dnSpy opened it as a managed assembly, which confirmed it's a .NET/C# crackme.
So I decided it was as good a time as any to try out and learn dnSpy!
Upon opening up the PE (portable executable) within dnSpy, I traversed through the *Assembly Explorer* on the left hand side to get into the main program logic.

PE tree structure:

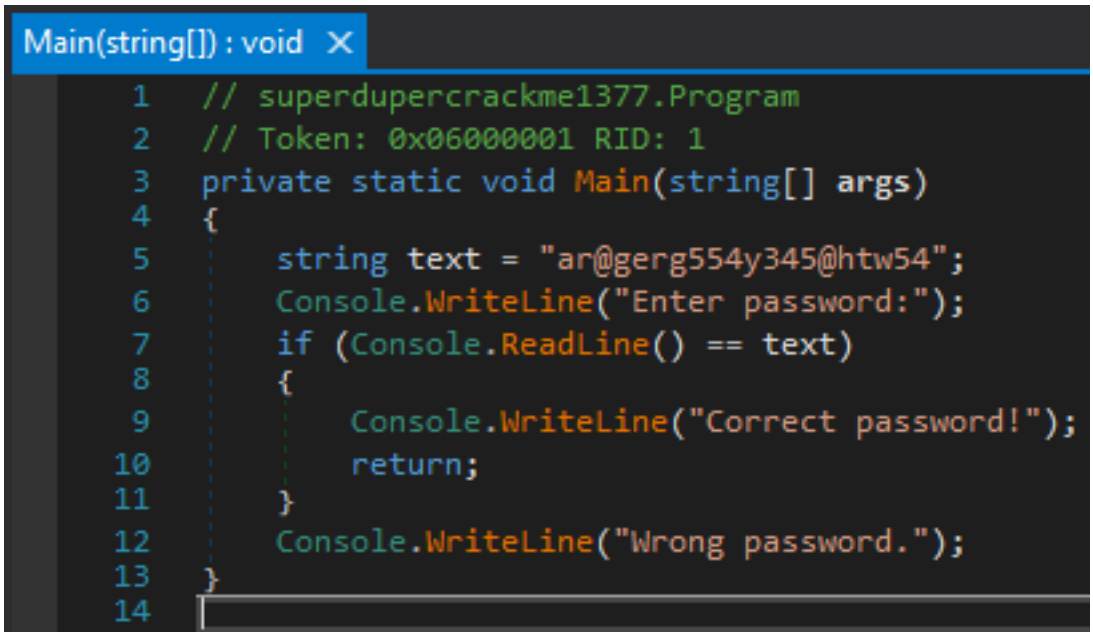


Upon opening up the *Main* function, things became more clear.



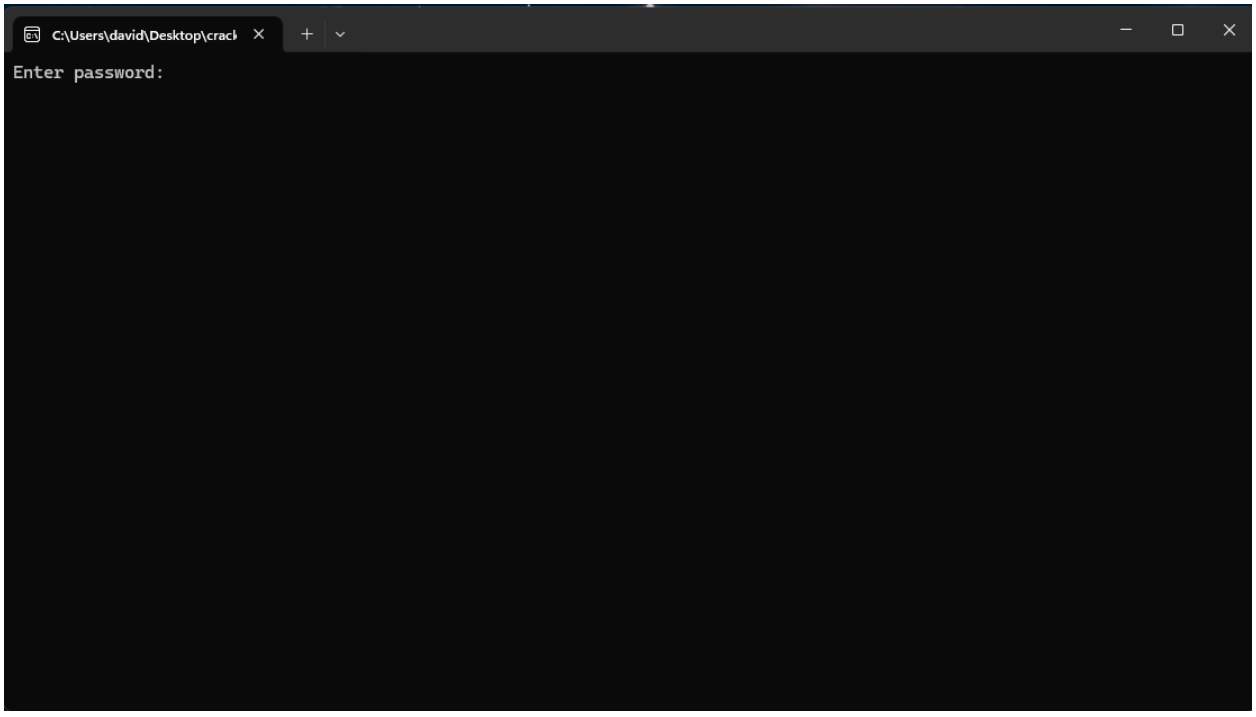
It seems that the reason for the "?" characters are because the program writes Cyrillic text, but my console is using an OEM code page that can't display those characters so they show as "?". It's an encoding/code page issue.

For the sake of convenience and readability I patched the strings within the code to an English version instead of Cyrillic, whilst still maintaining the original code functionality.

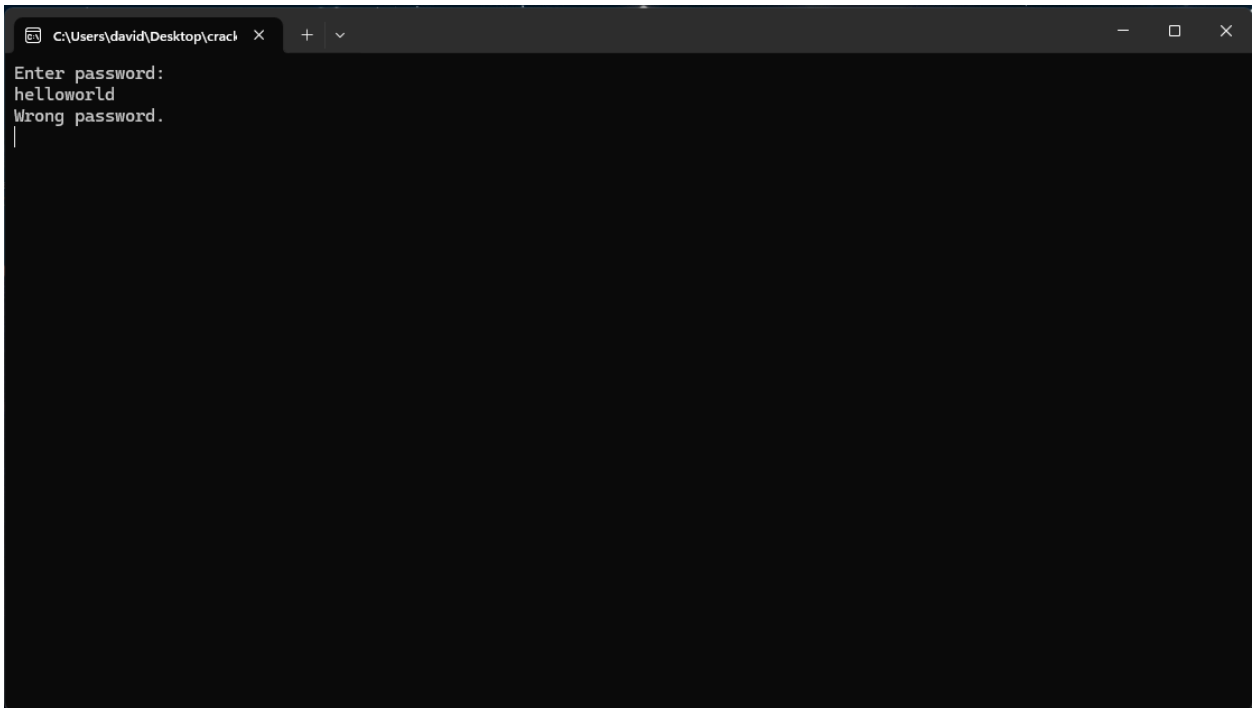


Now, the program execution looks like:

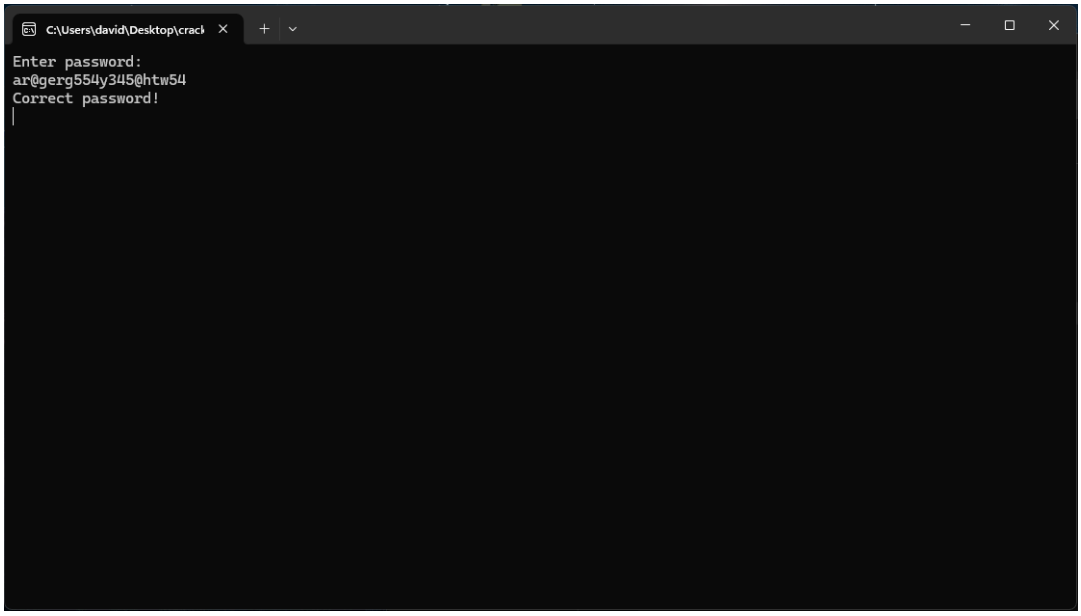
Regular execution (startup):



Wrong answer provided - proceeded by termination:



Great, now we can read and understand it! Taking a look back at the source code, the flag/key seems to be in plaintext being assigned to the variable text. Which is then being used to compare against the user input. Let's go ahead and enter in that value "ar@gerg554y345@htw54" and see what happens.



We did it!