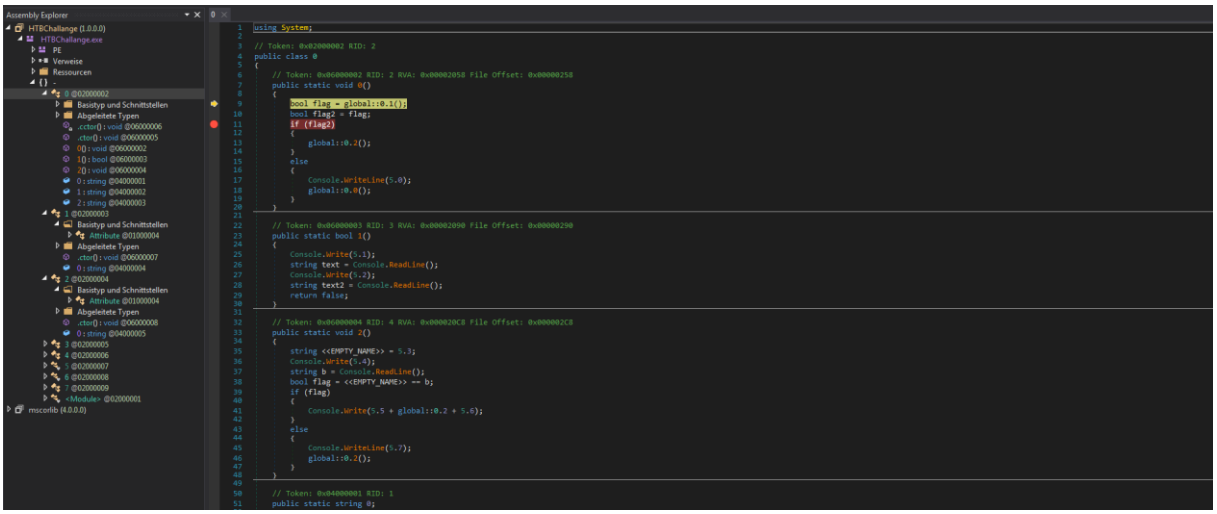
The Challenge has 2 steps which I experienced on a real engagement when I had to reverse a .NET Client.

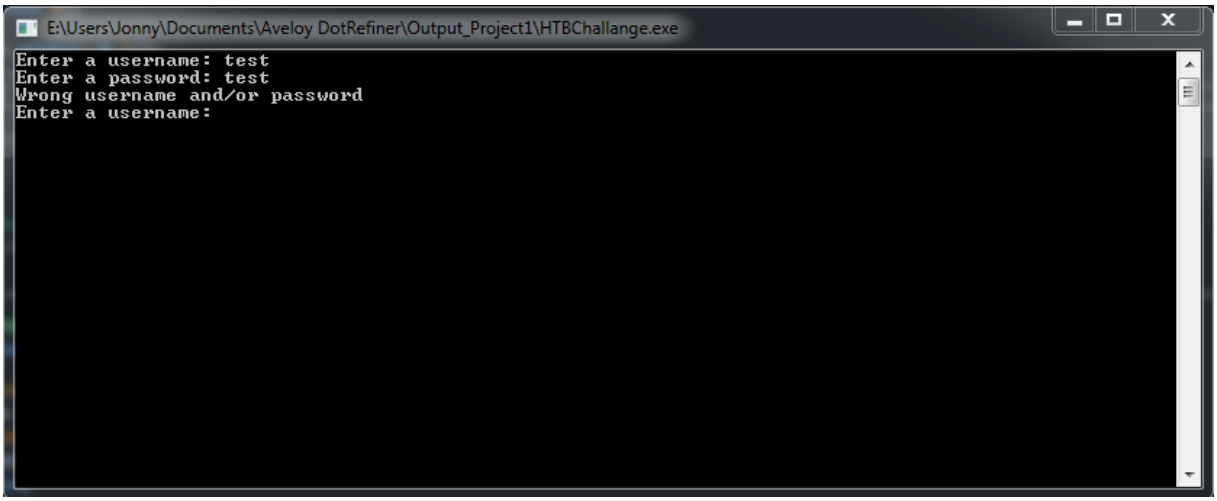
1. Bypass the „Login“
2. Reading a Key from Memory

1.

To prevent reading the Flag by using a decompiler or trying to read it from source the executable is obscured.

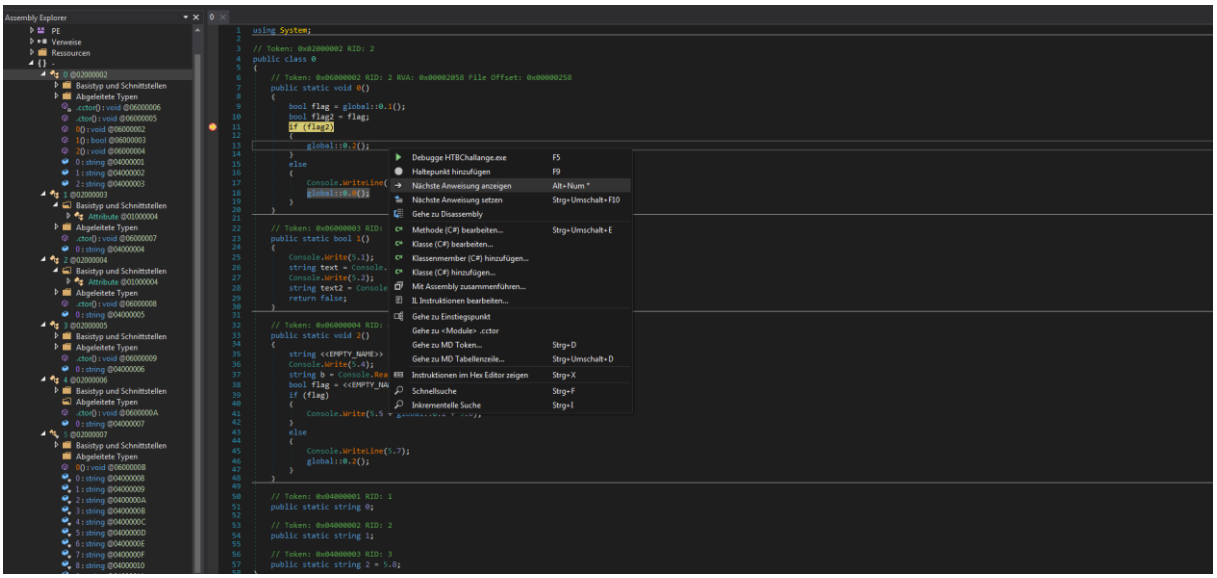


After starting the executable you must enter a username and a password.

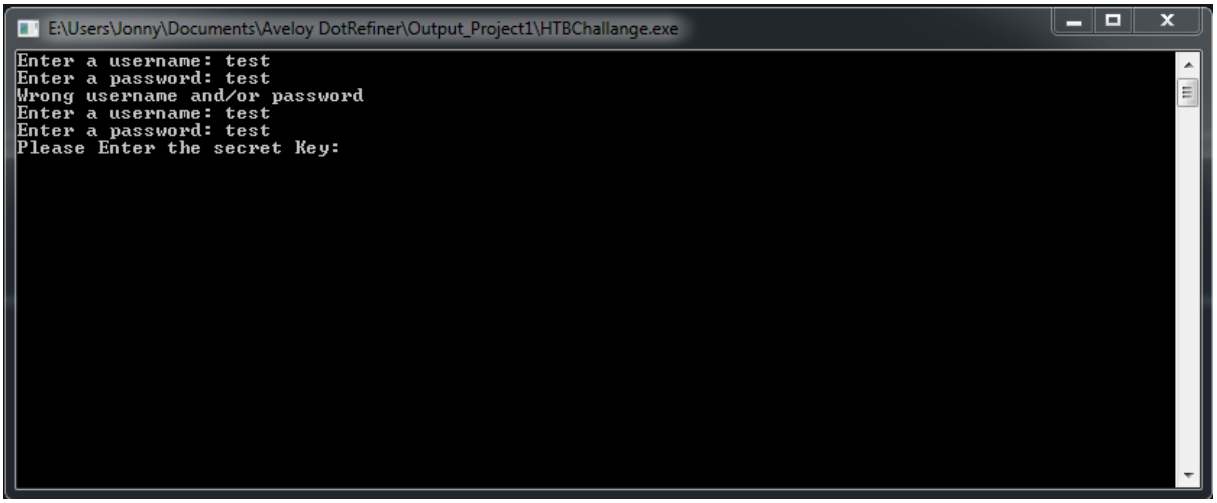


On a false login you must reenter username and password. There is actually no username and password which is correct implemented in the binary.

You must bypass this check() function to get to the next step. Since this is just a Client which has no server side validation the user has full control over the binary and can change the binary execution.

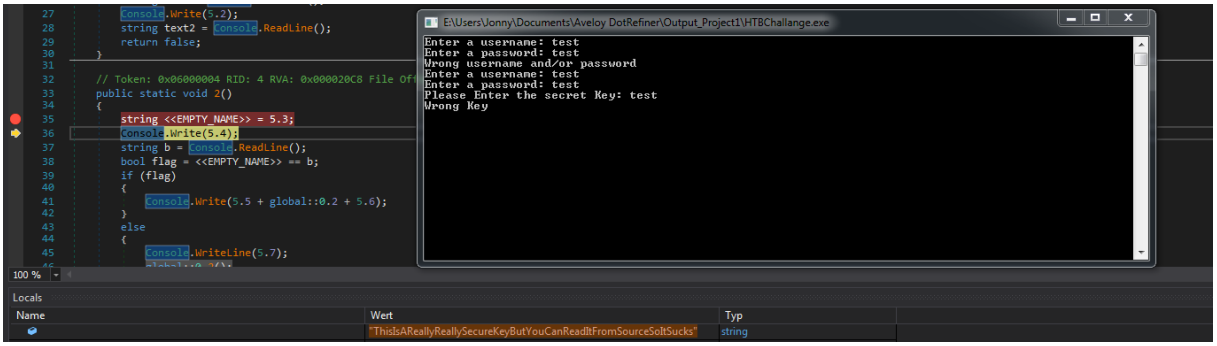


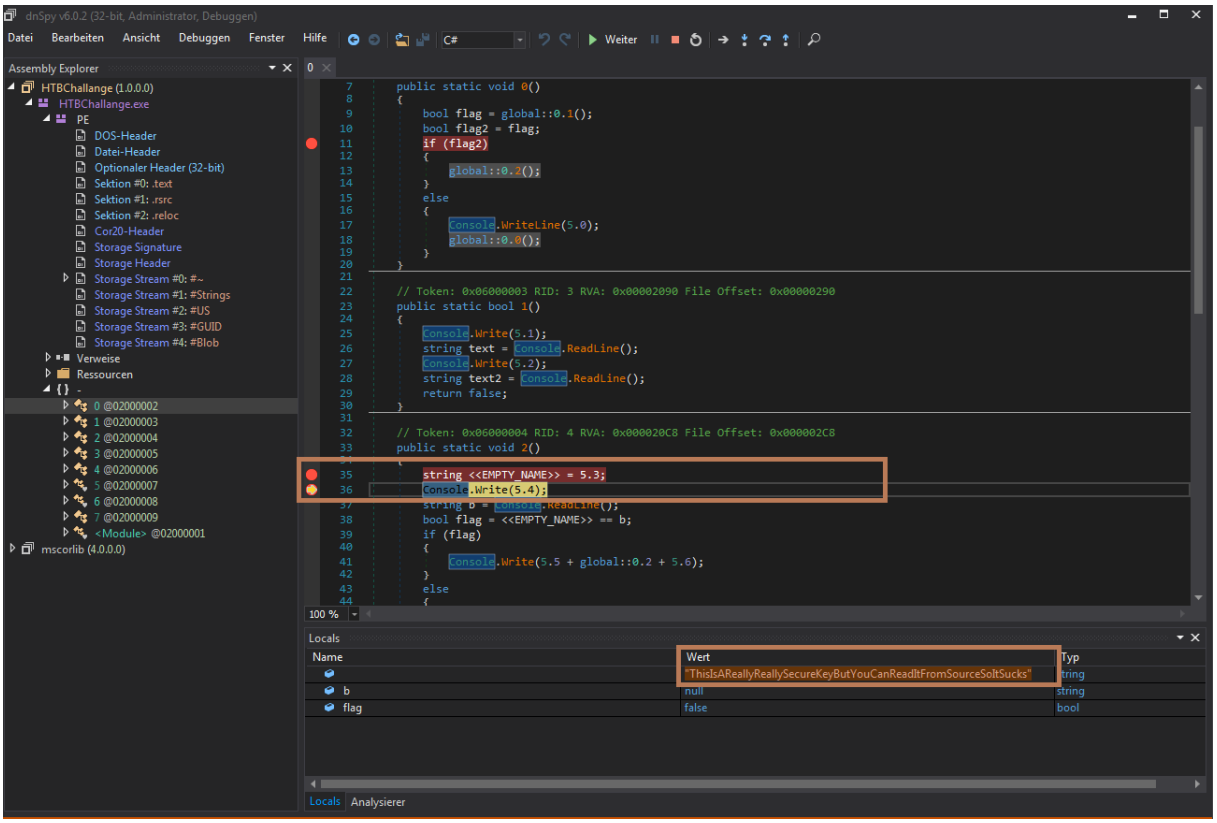
If (flag2) will check if the check() function returned **true** which is no possible without changing the value from **false** to **true** or use a decompiler such as dnspy https://github.com/0xd4d/dnSpy which lets you easily jump into the Instruction even without passing the check. In the figure above you can use “Nächste Anweisung setzen” which means “Set next Instruction”. Sorry that it’s in german.



2.

After bypassing the check() function you must enter the secret key. The key will be initialized in the flag() function which is also checks If the key is valid. So you must read the key from memory. You can set up a break point at line 35 where the key is initialized and watch the local variables.

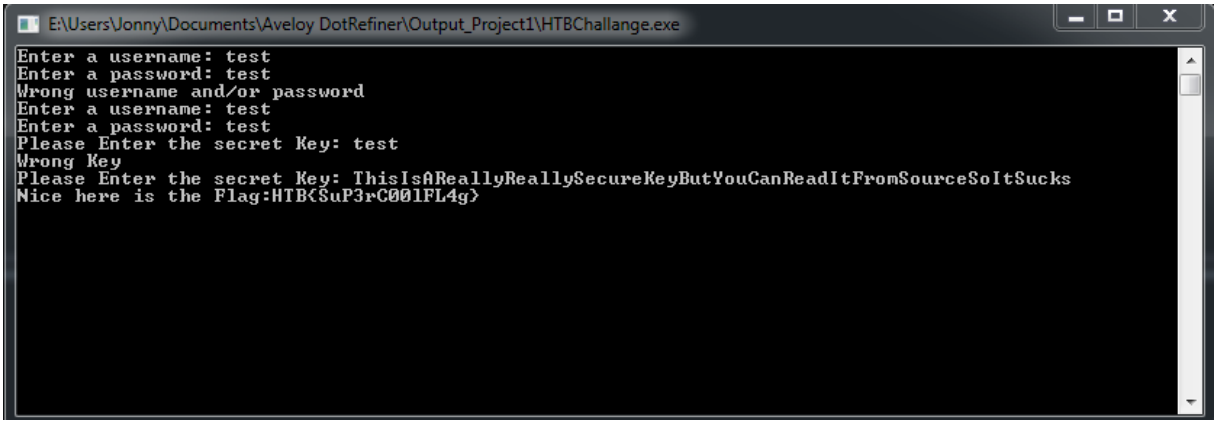




After entering a false key you can read the right key from memory as shown in this figure.

The right Key is: ThisIsAReallyReallySecureKeyButYouCanReadItFromSourceSoItSucks

It could be also possible to just jump into the if() which validates, if the key is correct.



After passing the correct key you get the Flag.