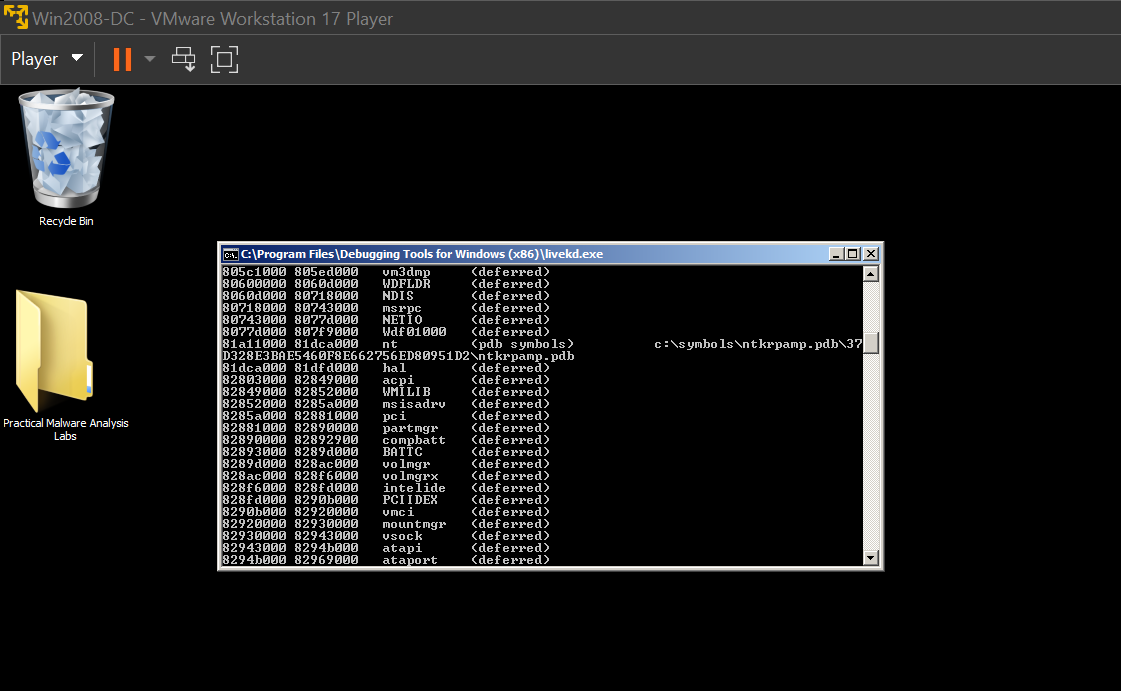
**LAB 17: Using Kernel Debugging Commands with WinDbg**

**Listing Modules with lm**

At the bottom of the Command window, in the command bar, execute this command: **lm**

A long list of loaded modules scrolls by. Scroll back to see the lm command you entered, and the first few loaded kernel modules, as shown below.

Scroll down to find the module named **nt**, as shown below. It's easy to spot because it'e one of the few modules that shows a Symbols path.

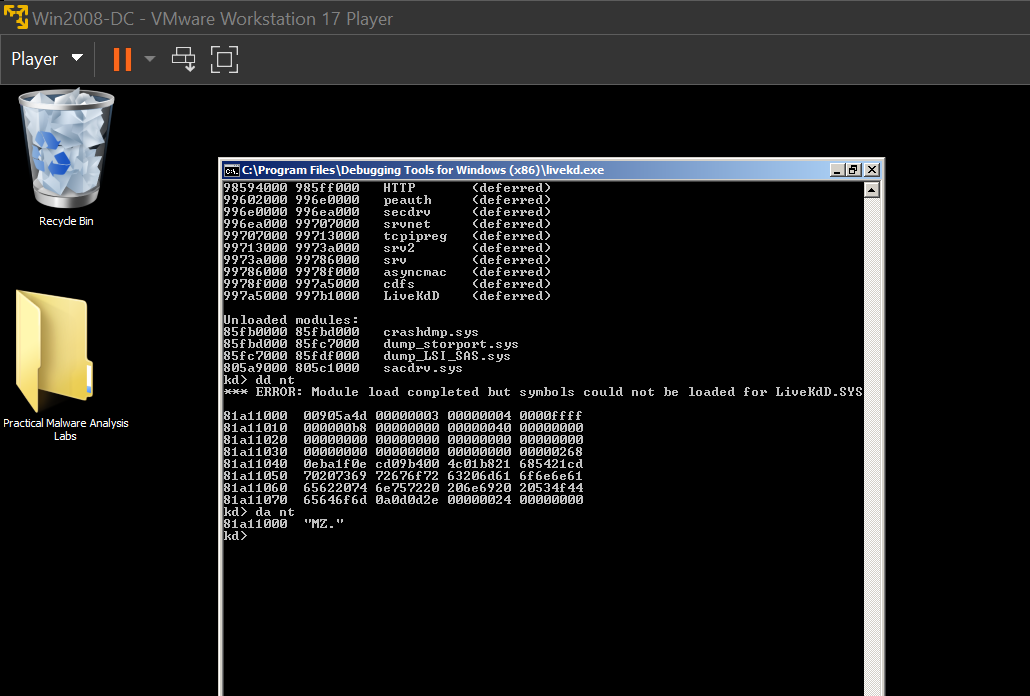
This is Ntoskrnl, the main kernel module.

**Viewing Memory**

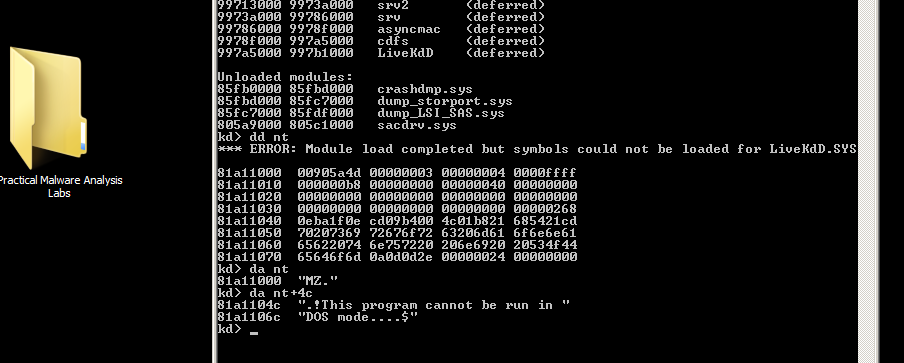
In WinDbg, execute this command: **dd nt**

You see the first several bytes of Ntoskrnl.exe, as shown below. This may be more familiar in ASCII.

In WinDbg, execute this command: **da nt**

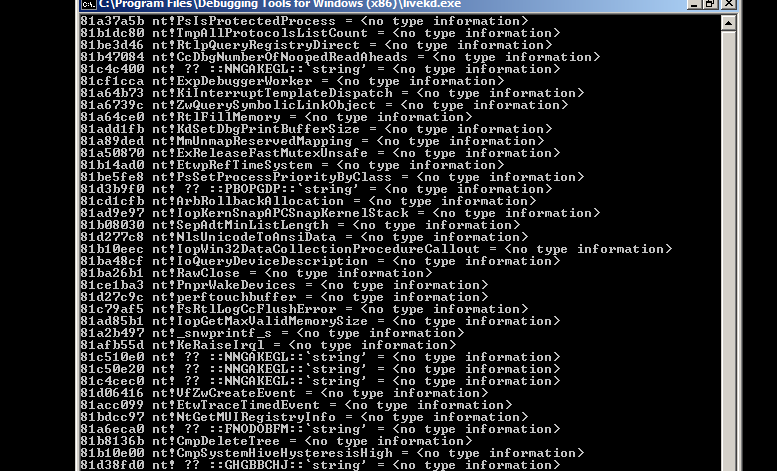
You see the characters "MZ" --they are at the start of every EXE file. 

In WinDbg, execute this command: **da nt+4c**

You see the message "This program cannot be run in DOS mode", as shown below: 

**Searching for Functions**

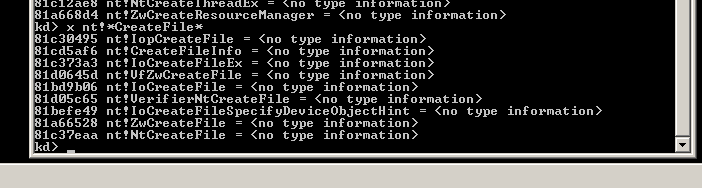
In WinDbg, execute this command: **x nt!\***

This finds all the functions in Ntoskrnl. There are a lot of them, as shown below: 

In WinDbg, execute this command: **x nt!\*Create\***

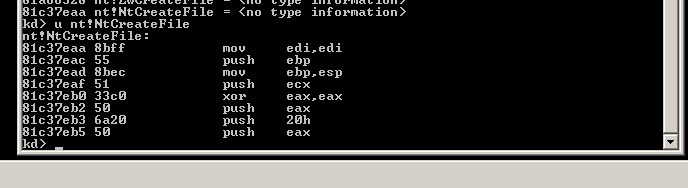
This finds all the functions in Ntoskrnl that contain the word "Create". There are a lot of them, too. 

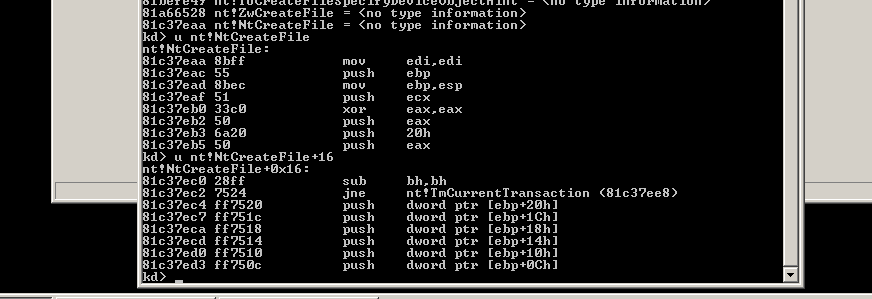
In WinDbg, execute this command: **x nt!\*CreateFile\***

This finds all the functions in Ntoskrnl that contain the word "CreateFile". There are only about ten of those, including "nt!NtCreateFile", as shown below: 

**Unassembling a Function**

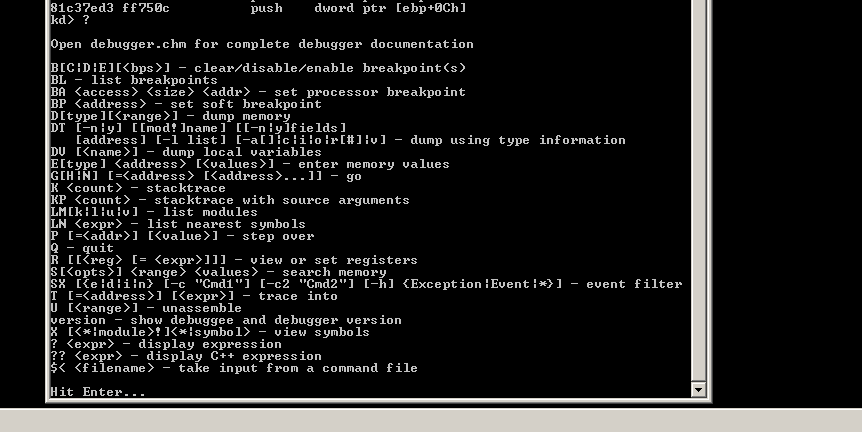
In WinDbg, execute this command: **u nt!NtCreateFile**

This shows the first few bytes of the function, disassembled, as shown below: 

**nt!NtCreateFile+16**

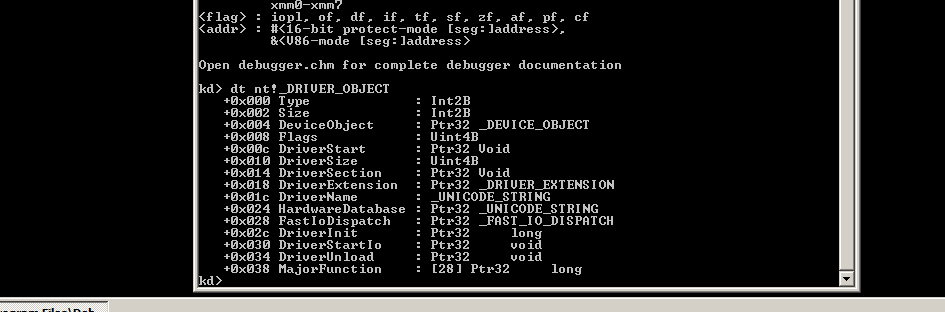
**Online Help**

Close the Disassembly window. In WinDbg, execute this command: **?**

You see the first page of the online help, as shown below: 

**Viewing Type Information for a Structure**

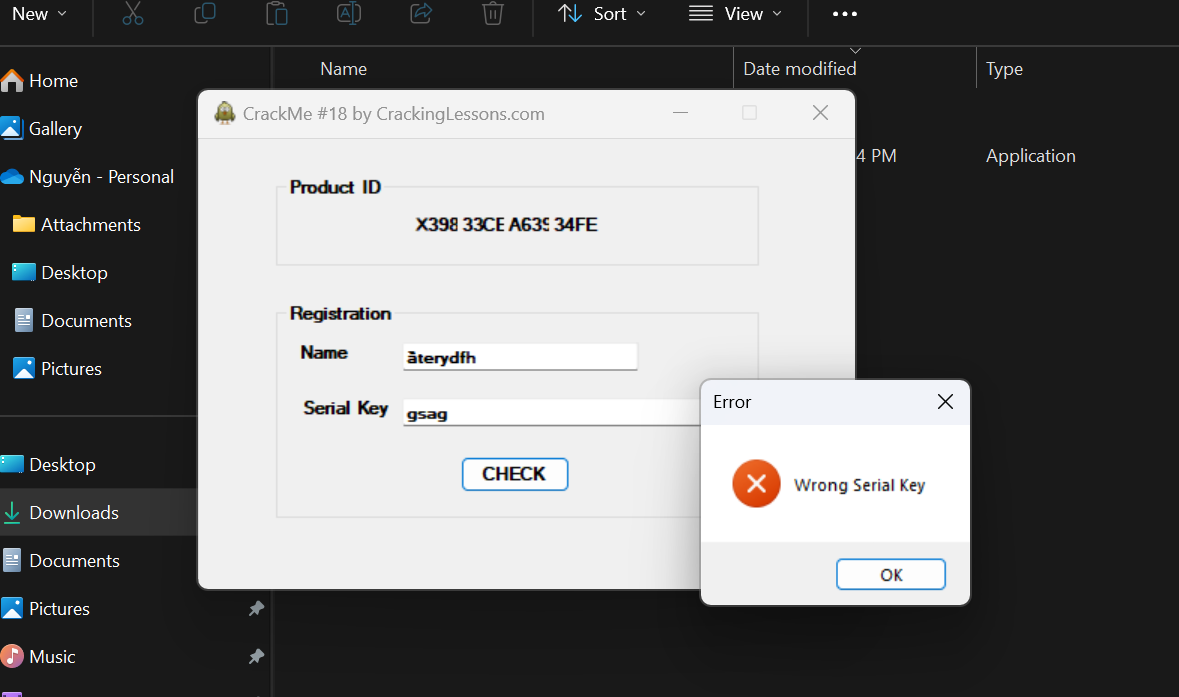
In WinDbg, execute this command: **dt nt!\_DRIVER\_OBJECT**

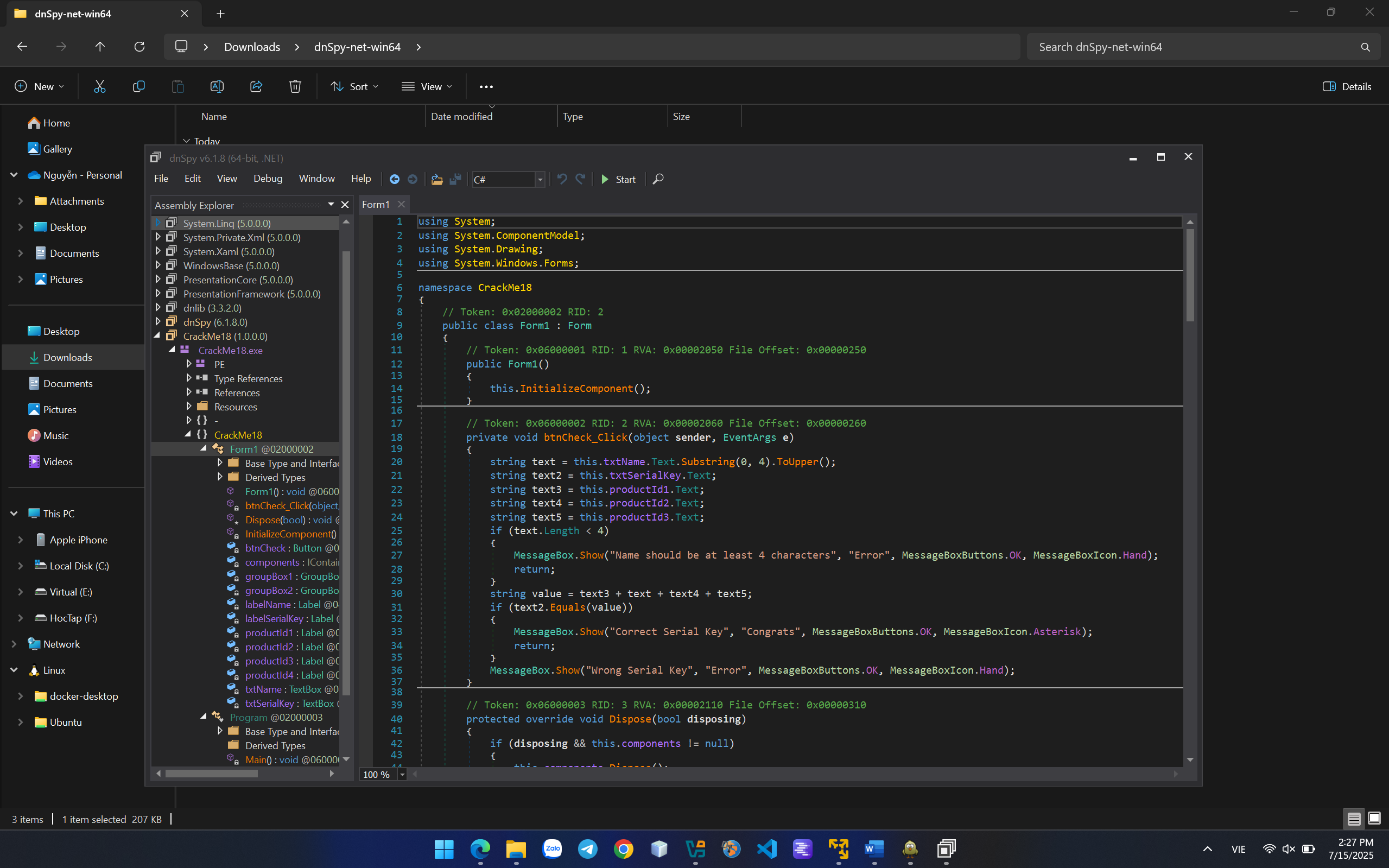
This shows the first few lines of a driver object structure, which stores information about a kernel driver, as shown below. Notice the DriverStart pointer--this contains the location of the driver in memory. 

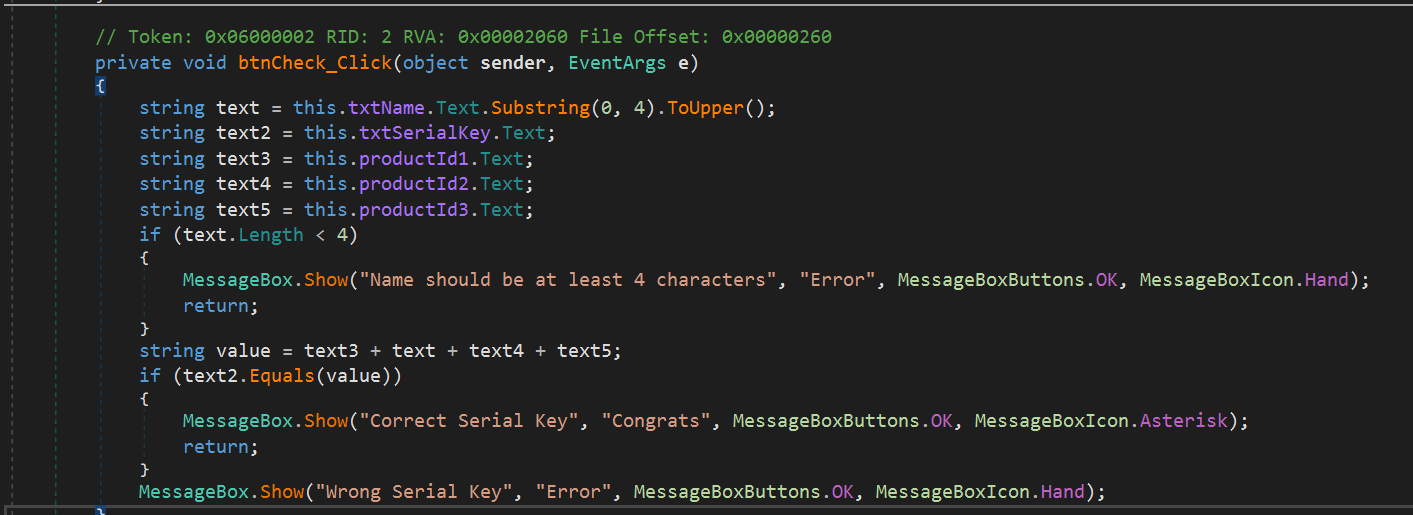
**CRACK ME 18**

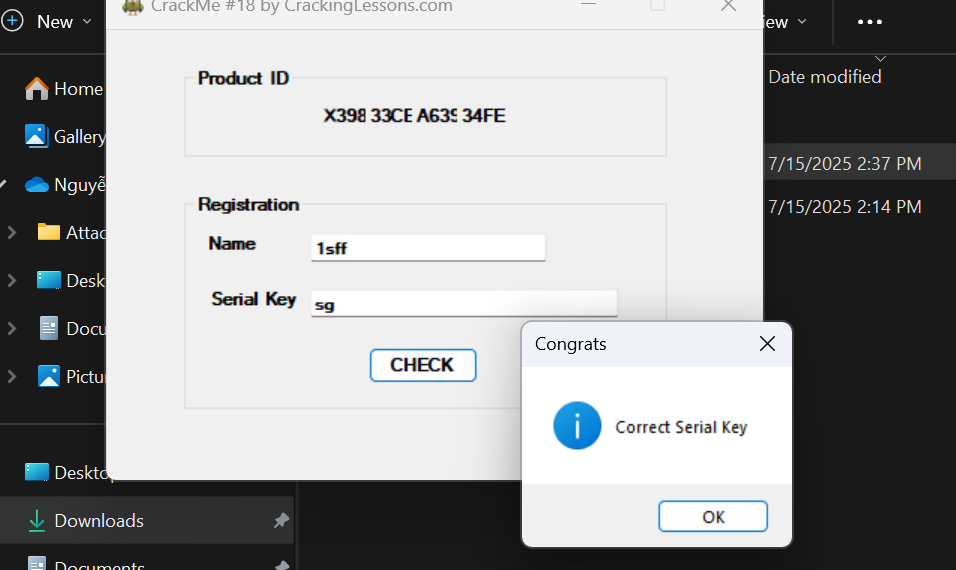
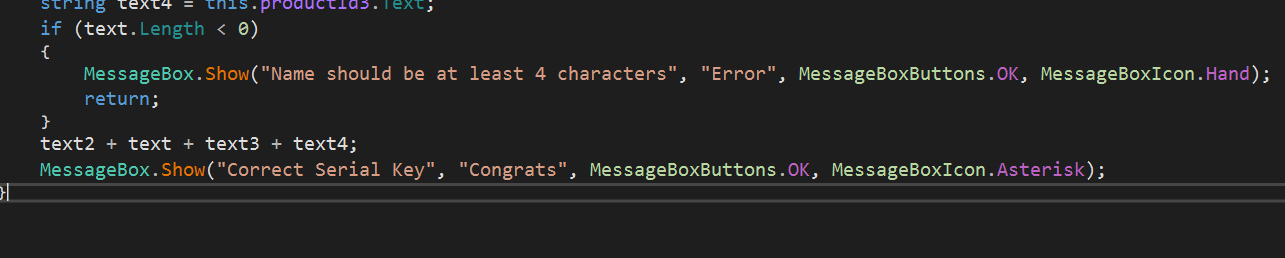
There are 3 tasks:

1. Patch it to always succeed no matter what name and serial key you enter.
2. Do serial fishing to extract the serial key based on a given name of your choice.
3. Create a keygen



Because this program using .Net C#, so I using this dnSpy tool to debugging this file .exe

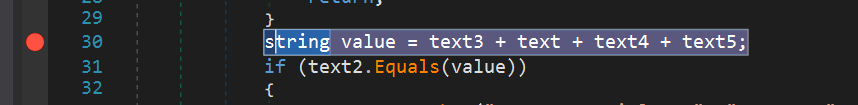
This is the code part of this check program

I change the text.Length < 0 and delete the IF block of Correct status with Incorrect satus like this and patch to check: 

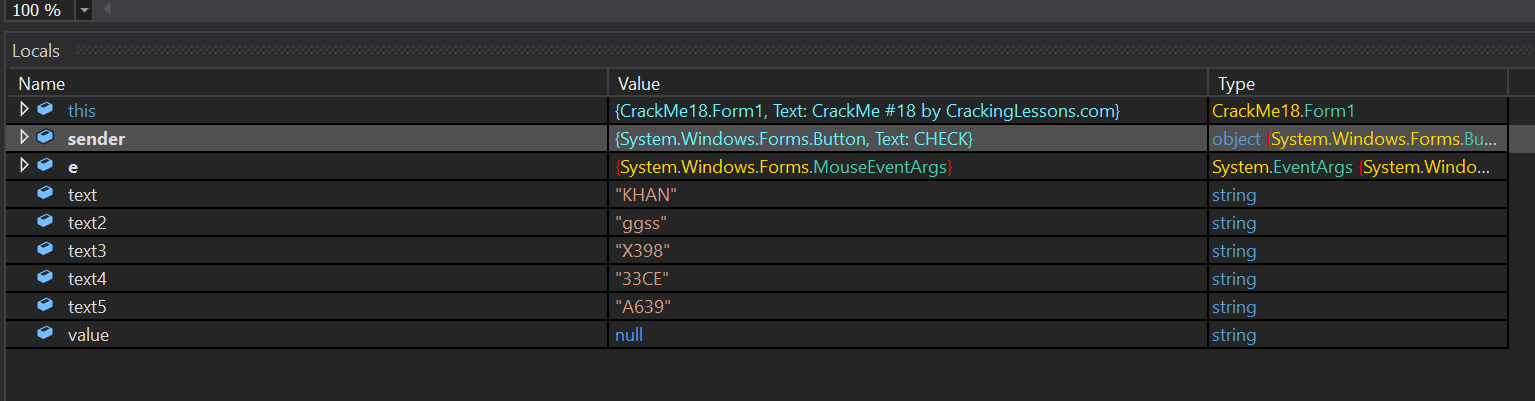
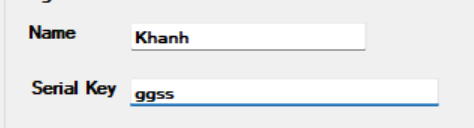
Done the mission 1.

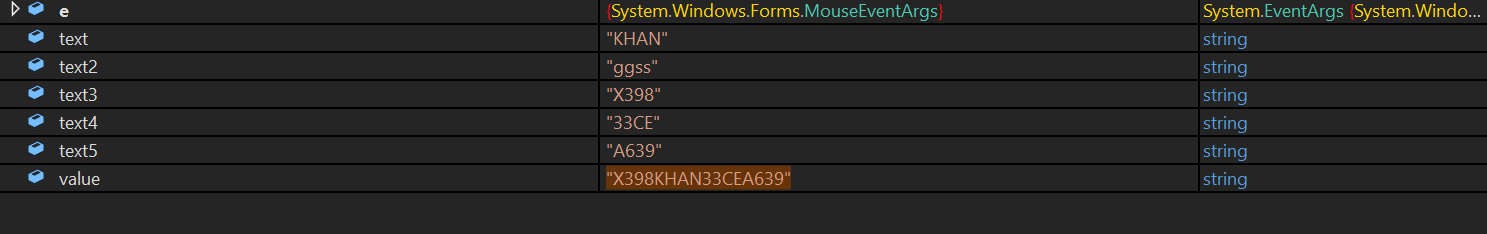
Next to challenge 2: serial fishing

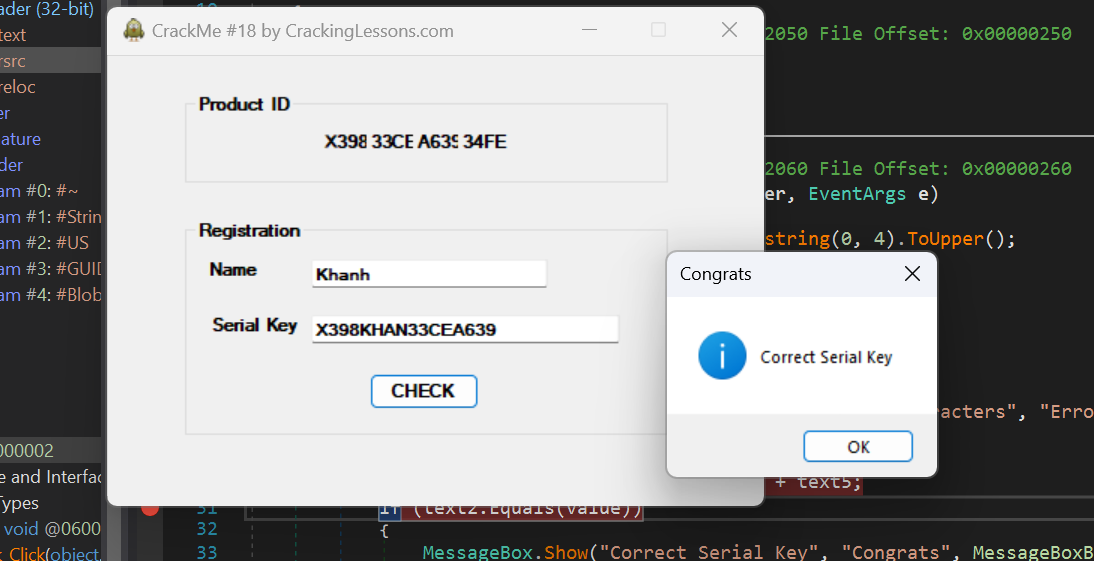
Add breakpoint to this line to view the change of String compare after this.



Run and see that:



Run to the IF compare to see the Value variable. This is the exact fishsing key we need: 

That is the fishing code: 

Next the final challenge:

Create keygen means display keygen to the box, so i change the box to display keygen like this: 