https://t.me/learningnets

Code Analysis

- Code Analysis involves analyzing the code
- Allows you to determine functionality that cannot be determined using dynamic analysis
- Code Analysis Tools: Disassemblers and Debuggers

Disassembler and Debuggers

Disassembler: Allow you to examine the code statically

Debugger: Allows you to examine the aspects of the program while it is running

- *Allows you to pause execution by setting breakpoints*
- *Single step through the code*
- *Step over*
- *Continue*

IDA Pro, x64dbg, Ollydbg, Immunity Debugger, Windbg are popular disassemblers and debuggers

https://t.me/learningnets

Code Analysis Tools

IDA Pro

- Disassembles the compiled executable into assembly instructions
- Powerful disassembler and debugger
- Commercial Tool
- Free Versions and demo version available

https://www.hex-rays.com/products/ida/support/download_freeware.shtml

http://out7.hex-rays.com/demo/request

x64dbg

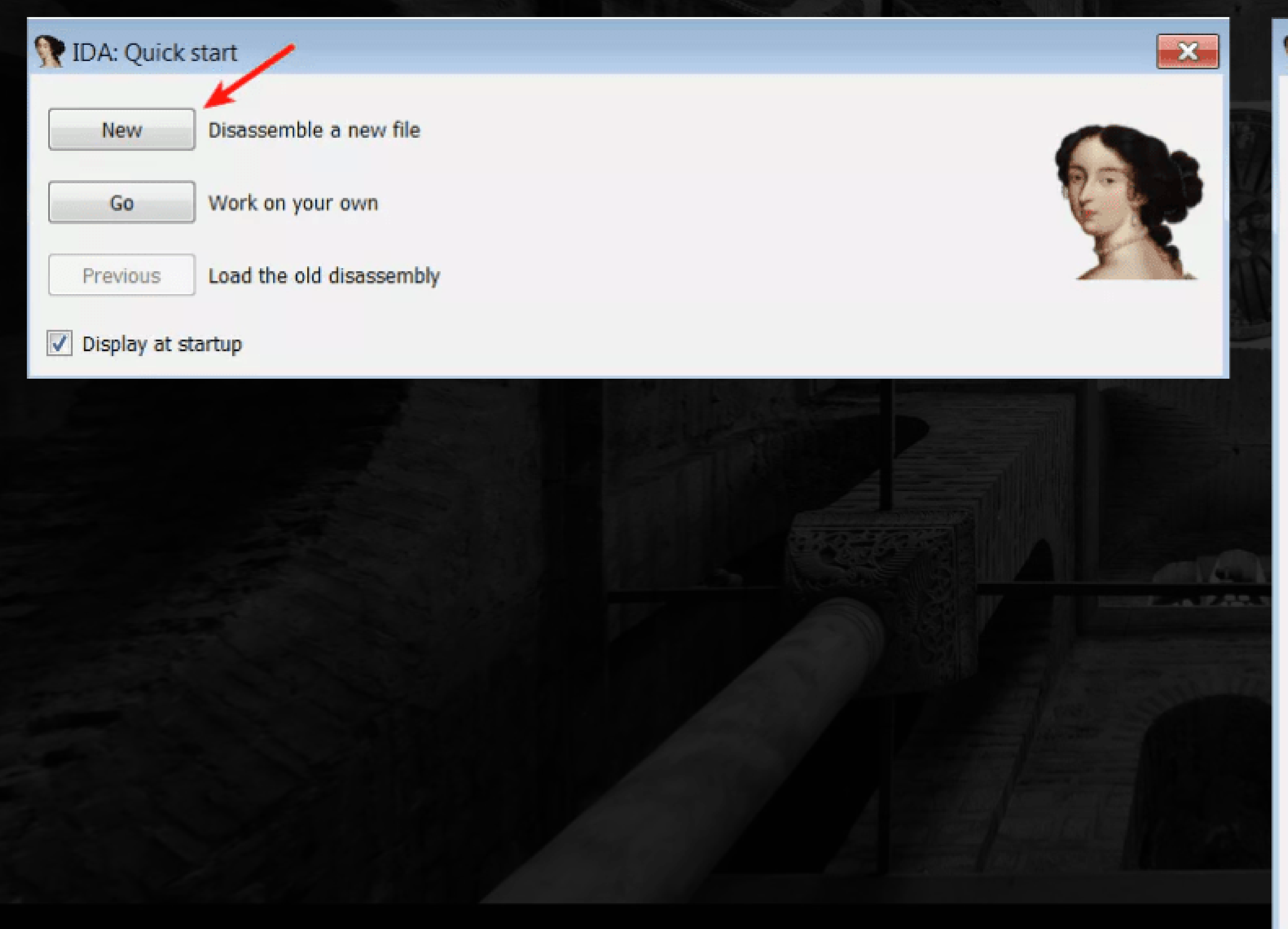
- Open source debugger
- Can debug 32-bit and 64-bit program
- Offers various debugging features

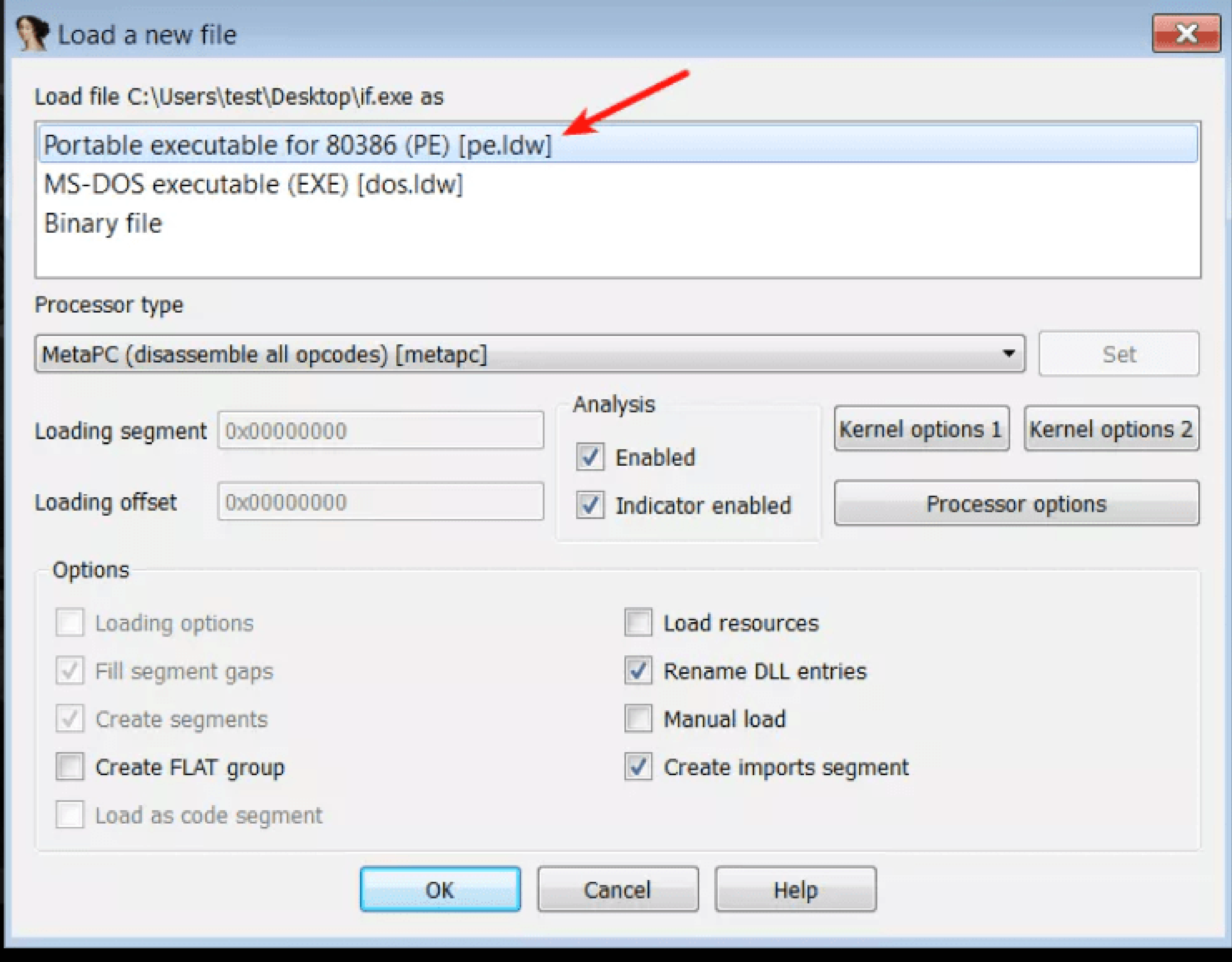
https://x64dbg.com

https://t.me/learningnets

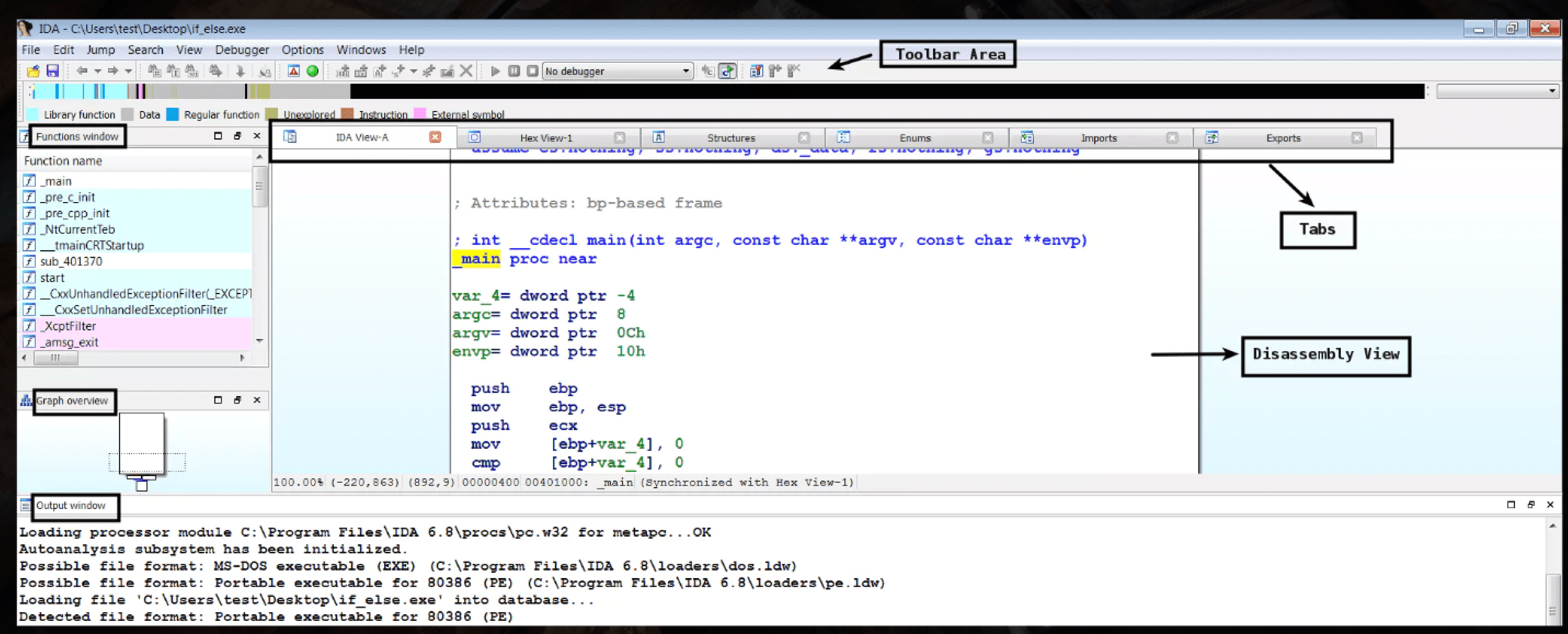
Disassembly using IDA

To load an executable, launch ***IDA pro*** (***right-click | Run as administrator***). Choose ***New*** and select the file you want to analyze. The file will be loaded into the memory and IDA determines the best possible loader to use during the disassembly process



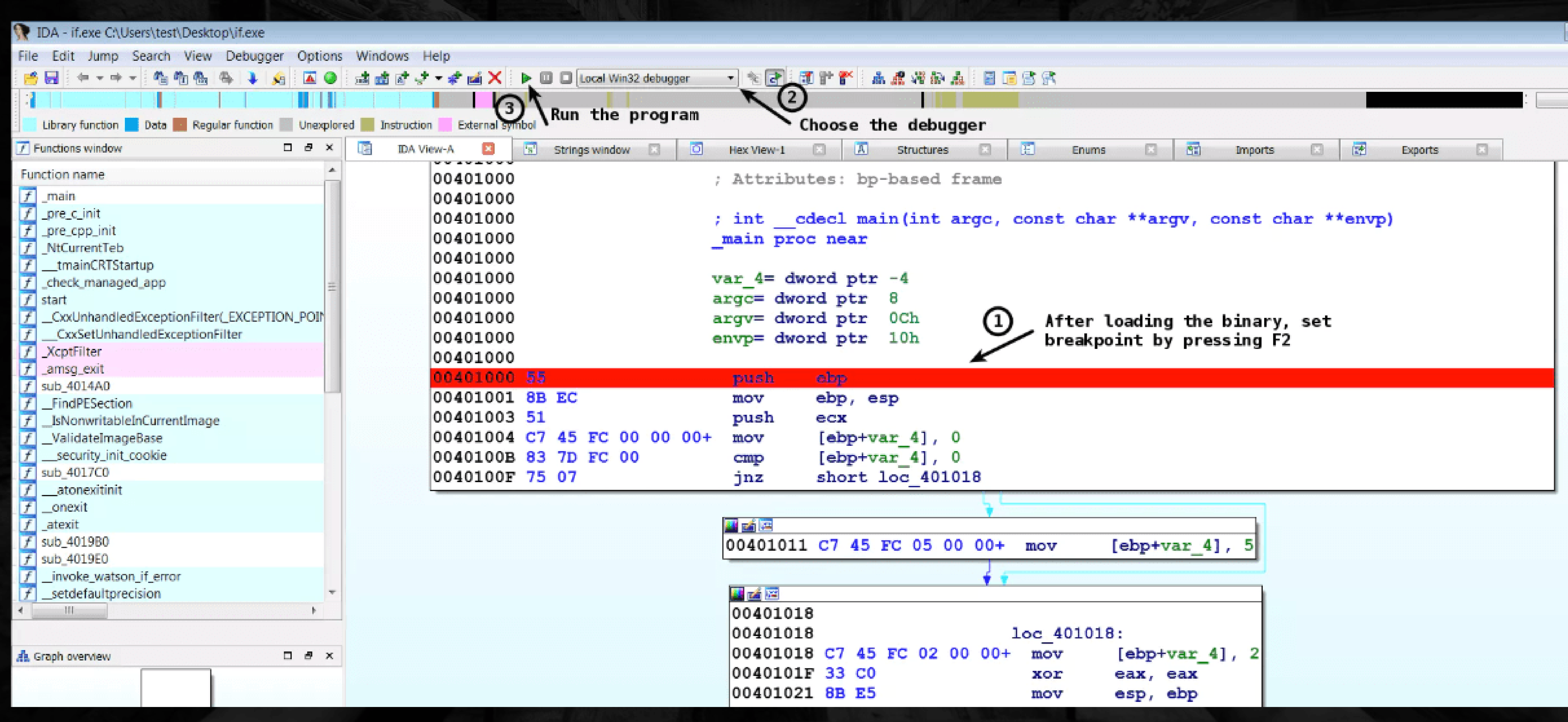


The following screenshot shows the IDA desktop after loading an executable file. The IDA desktop contains multiple tabs, clicking on each tab brings up different windows. Each window contains different information extracted from the binary

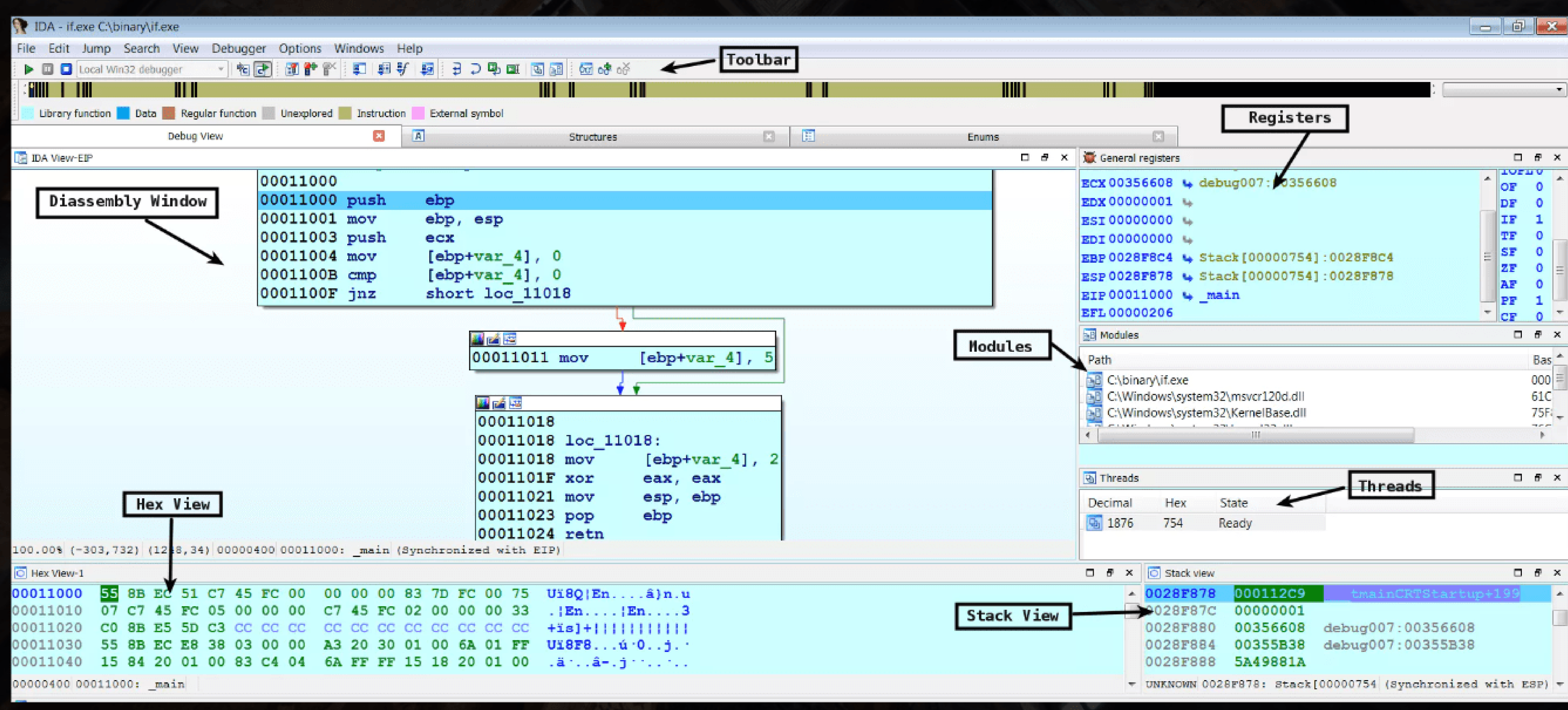


Debugging using IDA

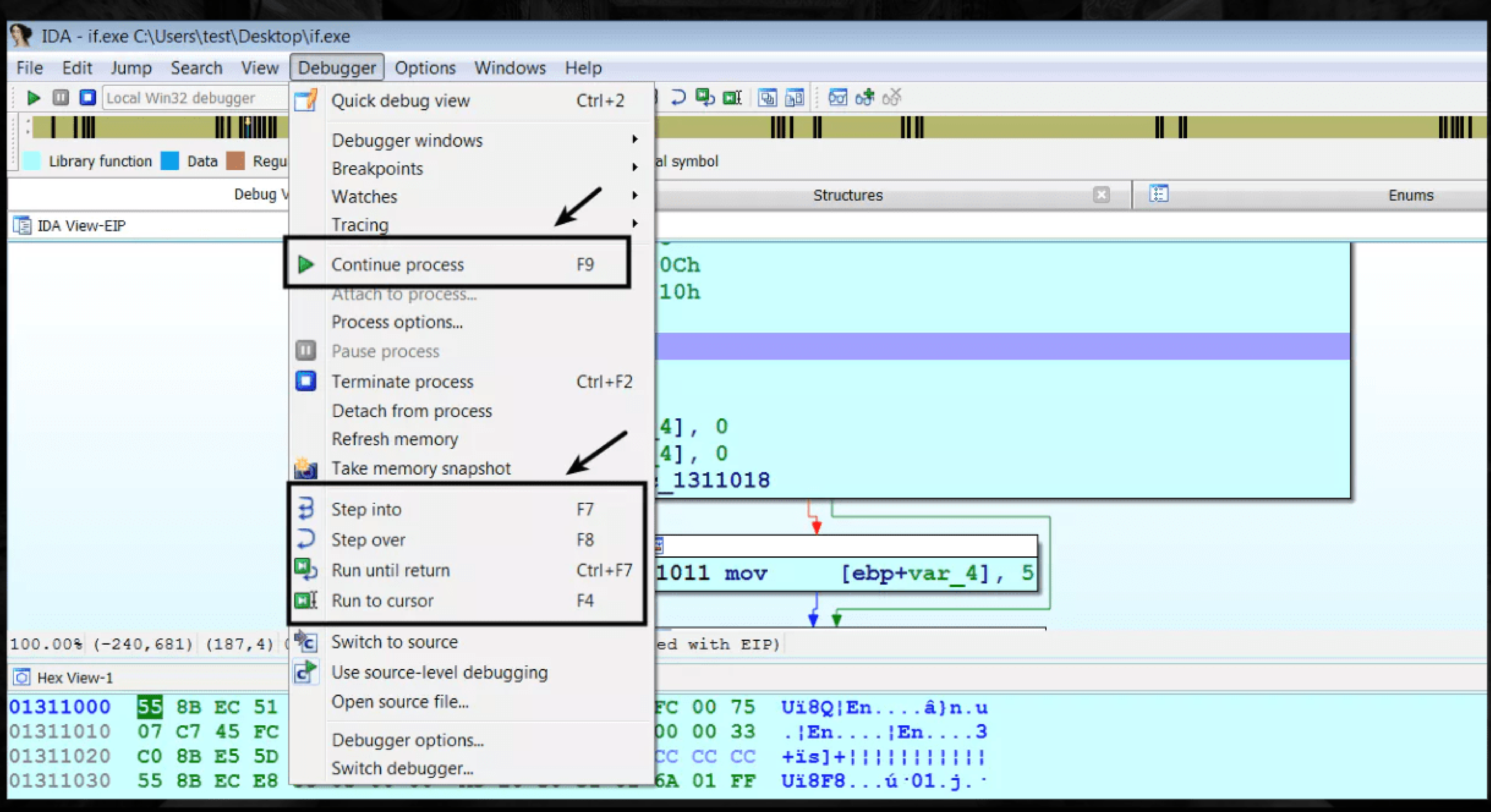
To debug a binary, first load the binary into IDA, set a breakpoint at the first instruction by pressing ***F2*** and then choose the appropriate debugger and Run the program (***F9***)

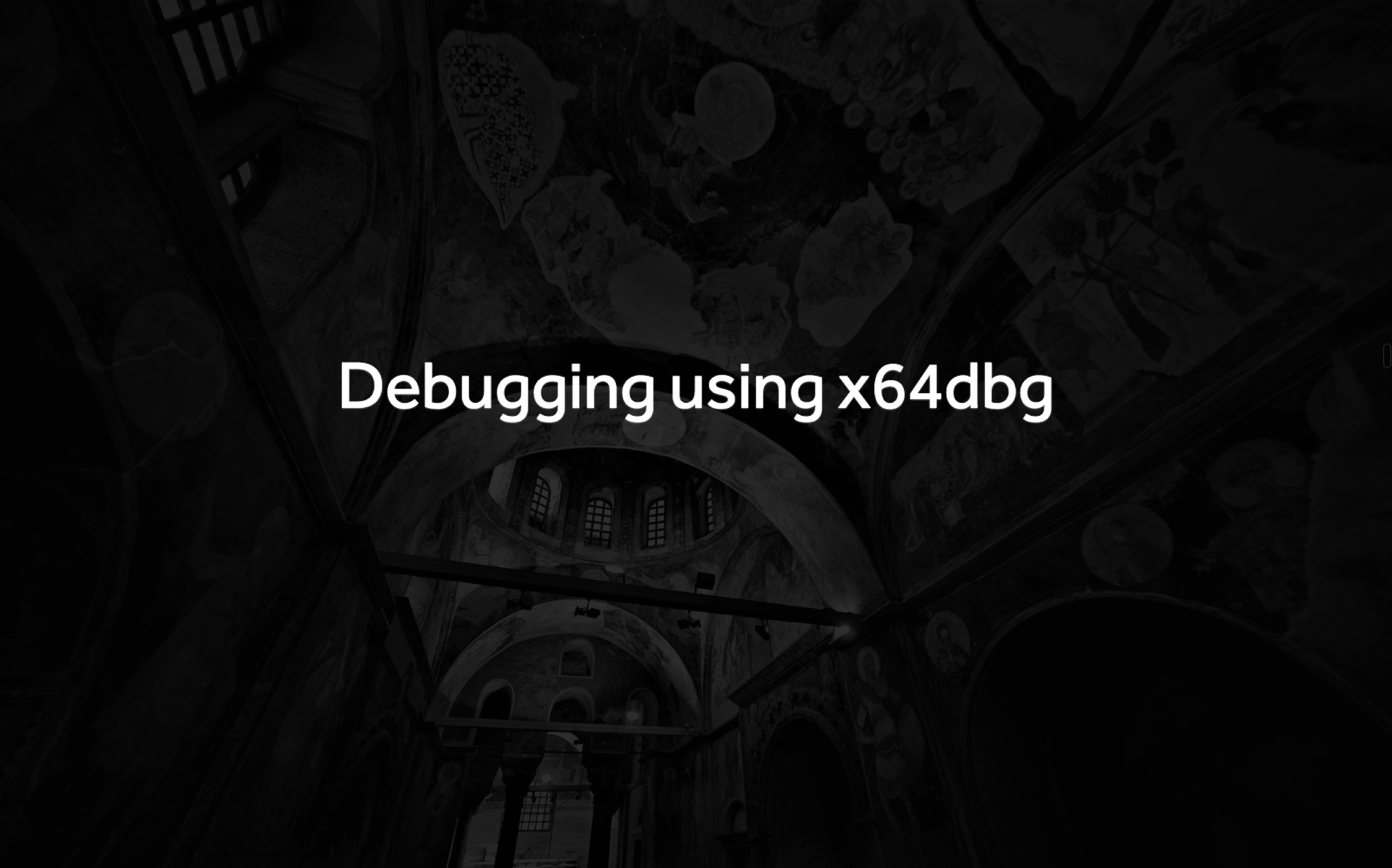


After you launch the program in IDA debugger, the process will be paused and the following debugger display will be presented to you.

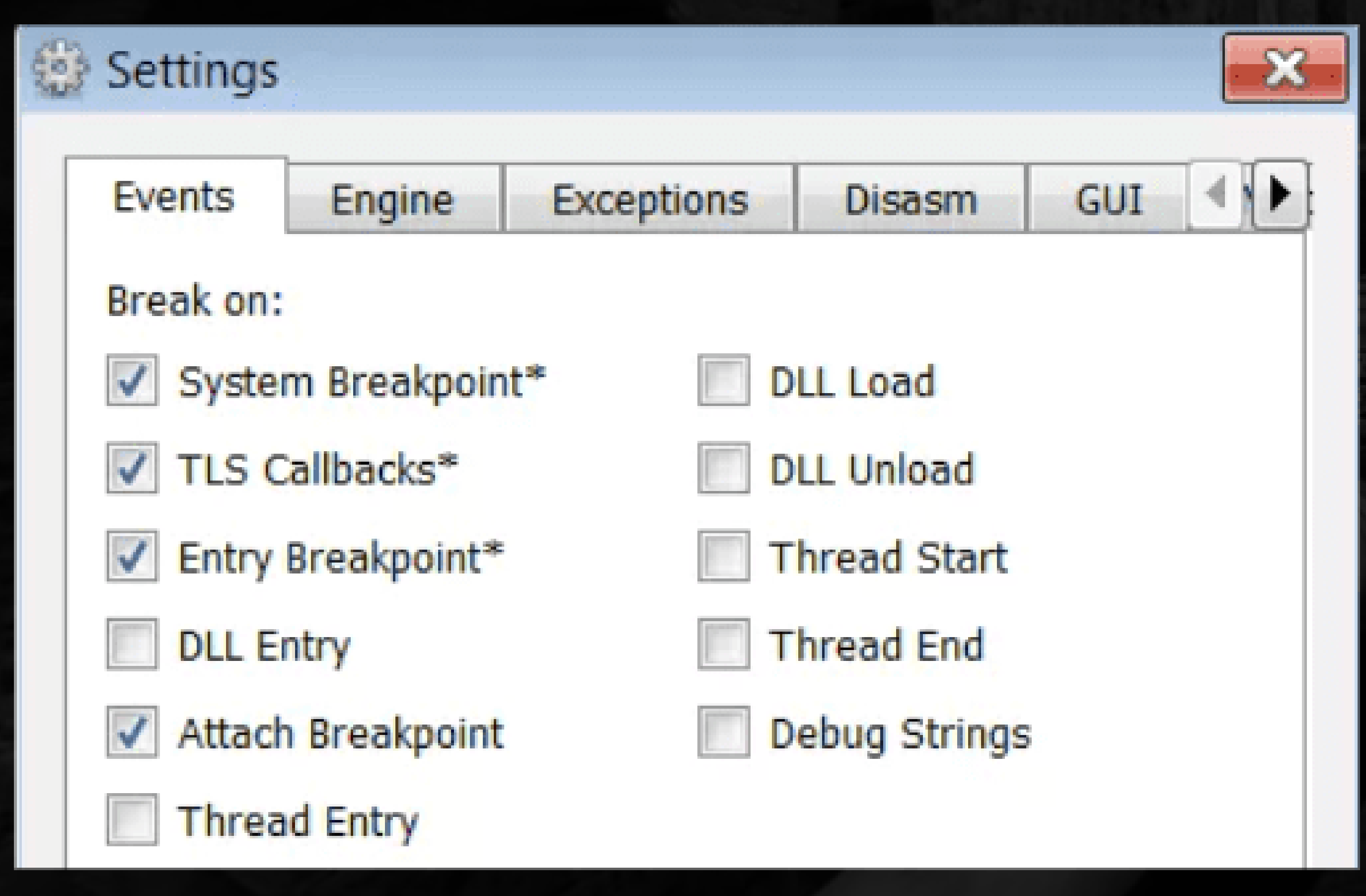


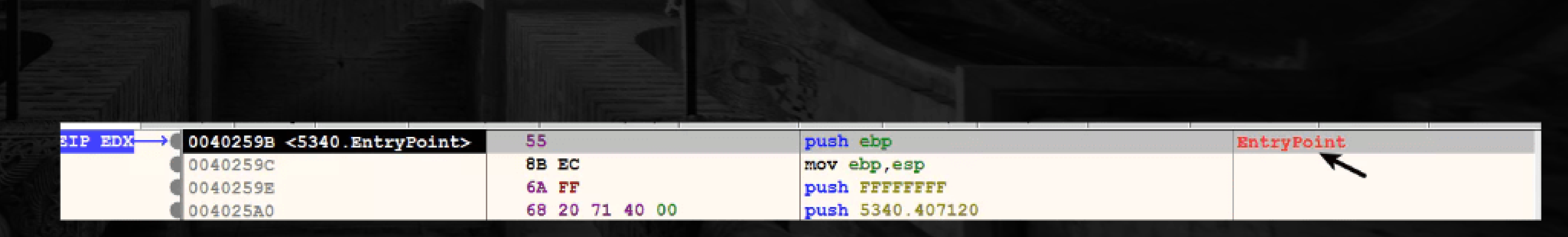
You can access various debugging options from the Debugger menu as shown below



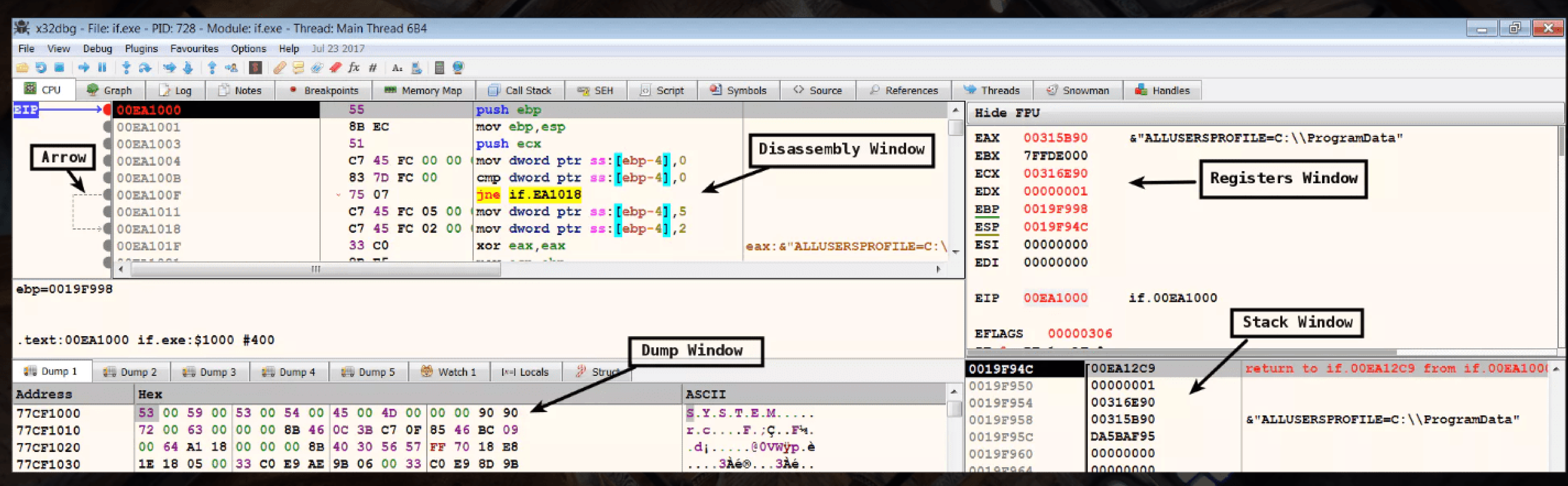

Debugging using x64dbg

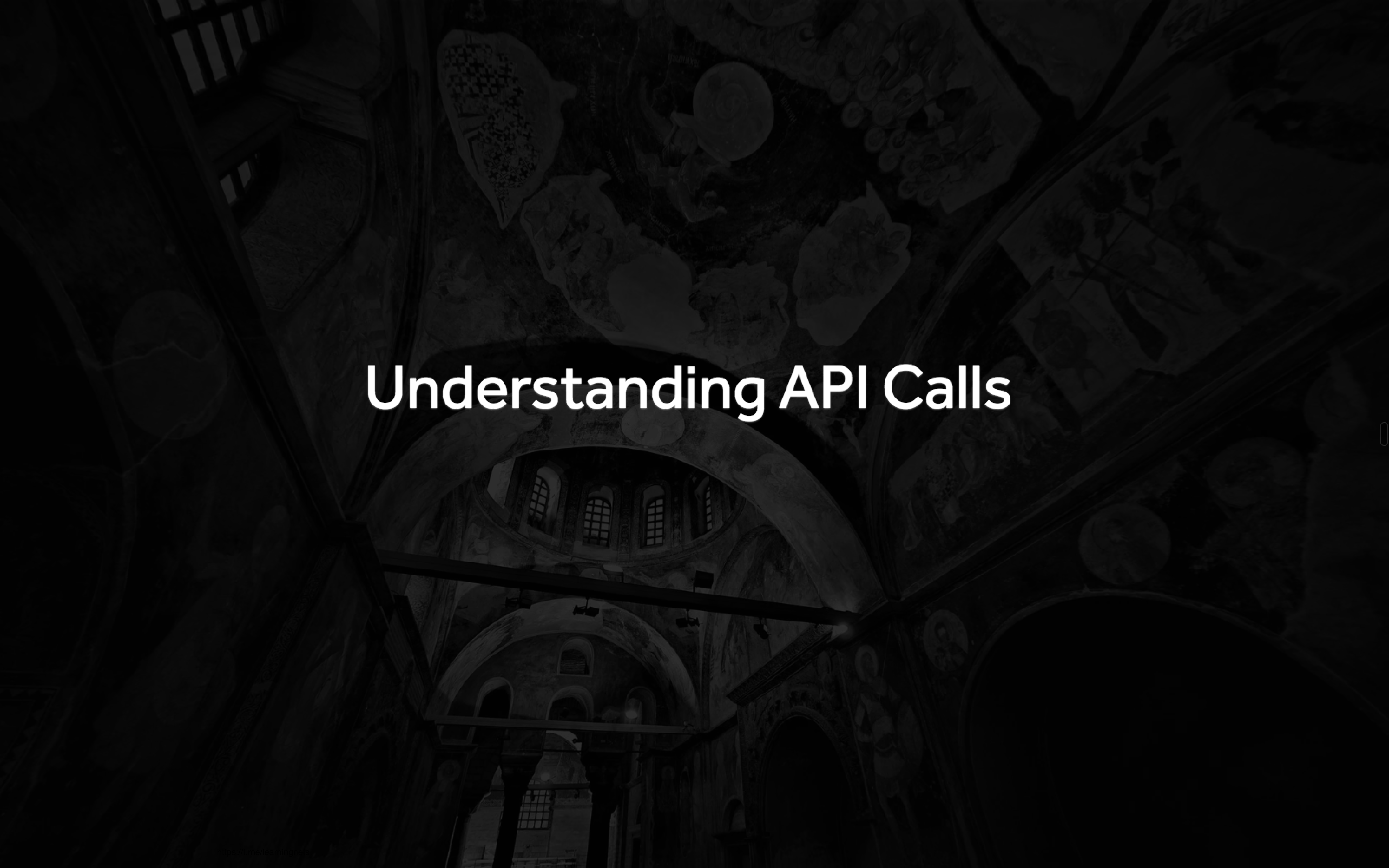
To load an executable select ***File | Open*** and then browse the file you wish to debug, this will start the process and the debugger will pause at the ***System Breakpoint, TLS callback*** or the ***program entry point*** function depending on the configuration settings. You can access the settings dialog by choosing ***Options | Preferences | Events*** as shown below. If you want the execution to pause at the program's entry point directly then uncheck the ***System Breakpoint*** and ***TLS Callbacks**** option.





x64dbg - Below screen shot shows the x64dbg interface





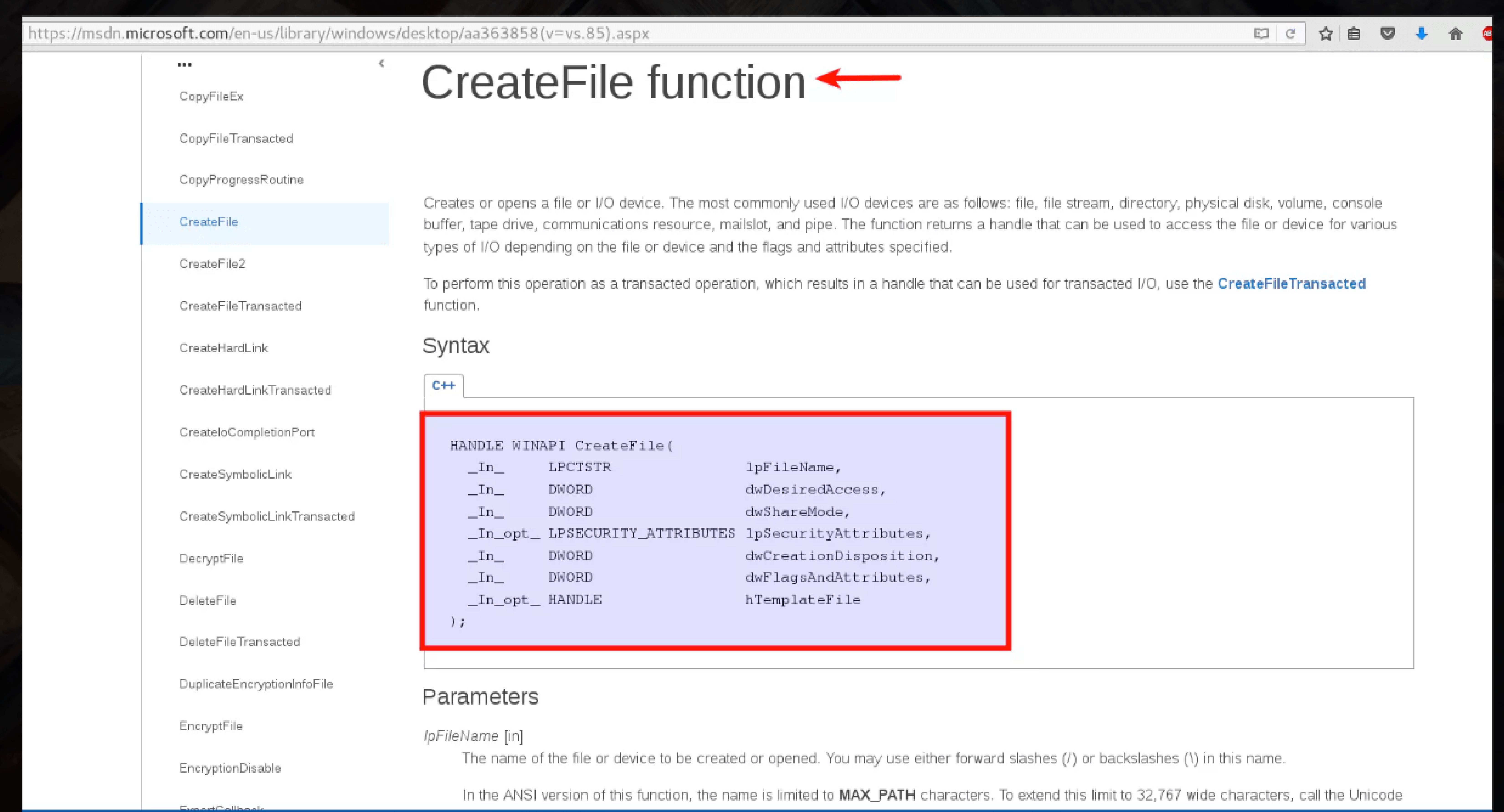
Understanding API Calls

Focus on Function Calls, mainly API calls

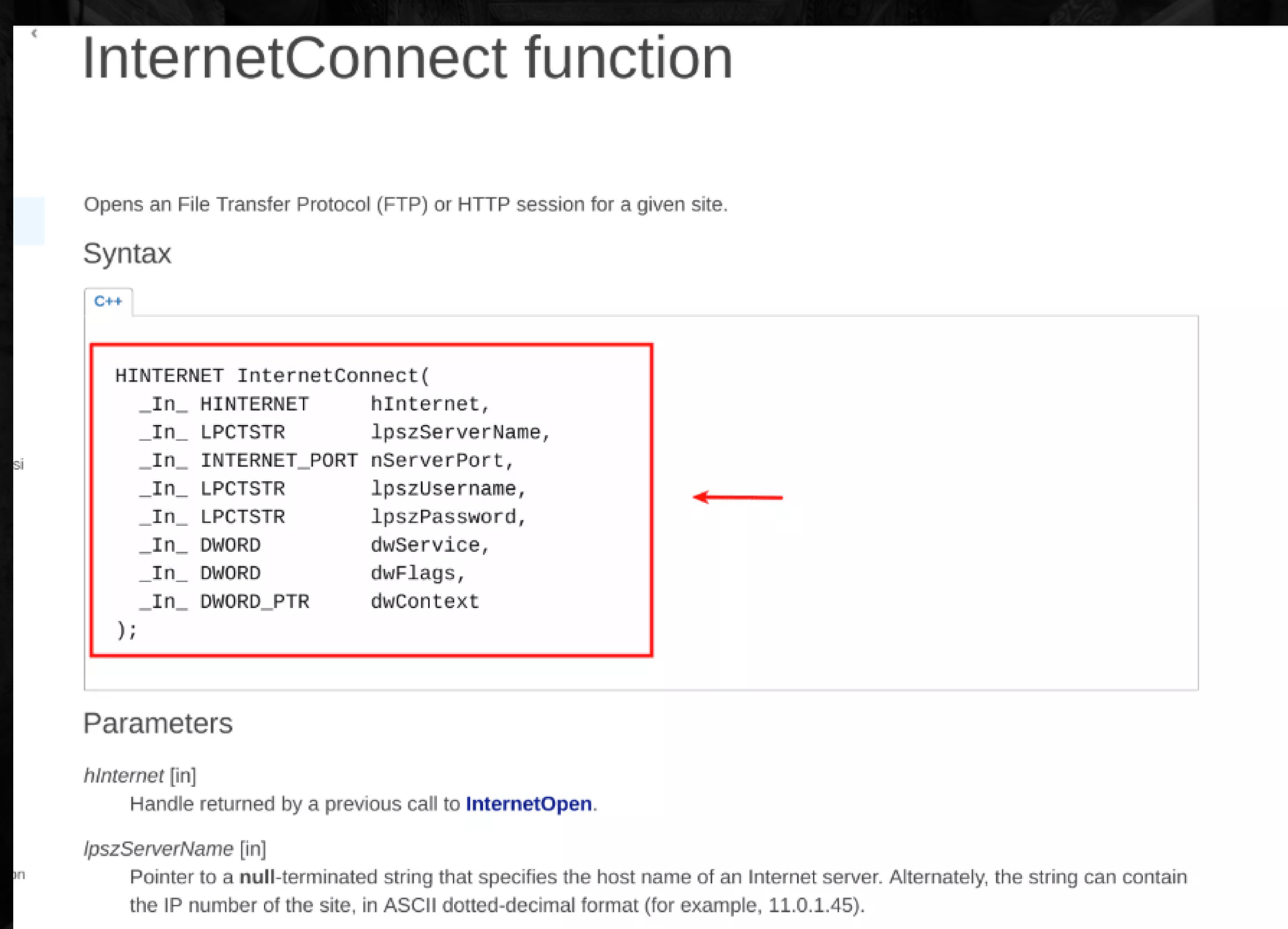
- ***Call*** is the assembly instruction that calls a function
- Functions perform some actions, parameters to functions are pushed on to the stack.
- Function returns value in ***EAX/RAX*** register
- Examining the functions allow you to determine the functionality
- API Calls are external functions that allow malware to interact with its environment. Search MSDN for documentation on API calls

  https://msdn.microsoft.com/en-us/default.aspx

Below example shows the MSDN documentation on ***CreateFile*** API call



Below example shows the MSDN documentation for *InternetConnect* API function


InternetConnect function

Opens an File Transfer Protocol (FTP) or HTTP session for a given site.

Syntax

C++

```
HINTERNET InternetConnect(
  _In_ HINTERNET     hInternet,
  _In_ LPCTSTR       lpszServerName,
  _In_ INTERNET_PORT nServerPort,
  _In_ LPCTSTR       lpszUsername,
  _In_ LPCTSTR       lpszPassword,
  _In_ DWORD         dwService,
  _In_ DWORD         dwFlags,
  _In_ DWORD_PTR     dwContext
);
```

Parameters

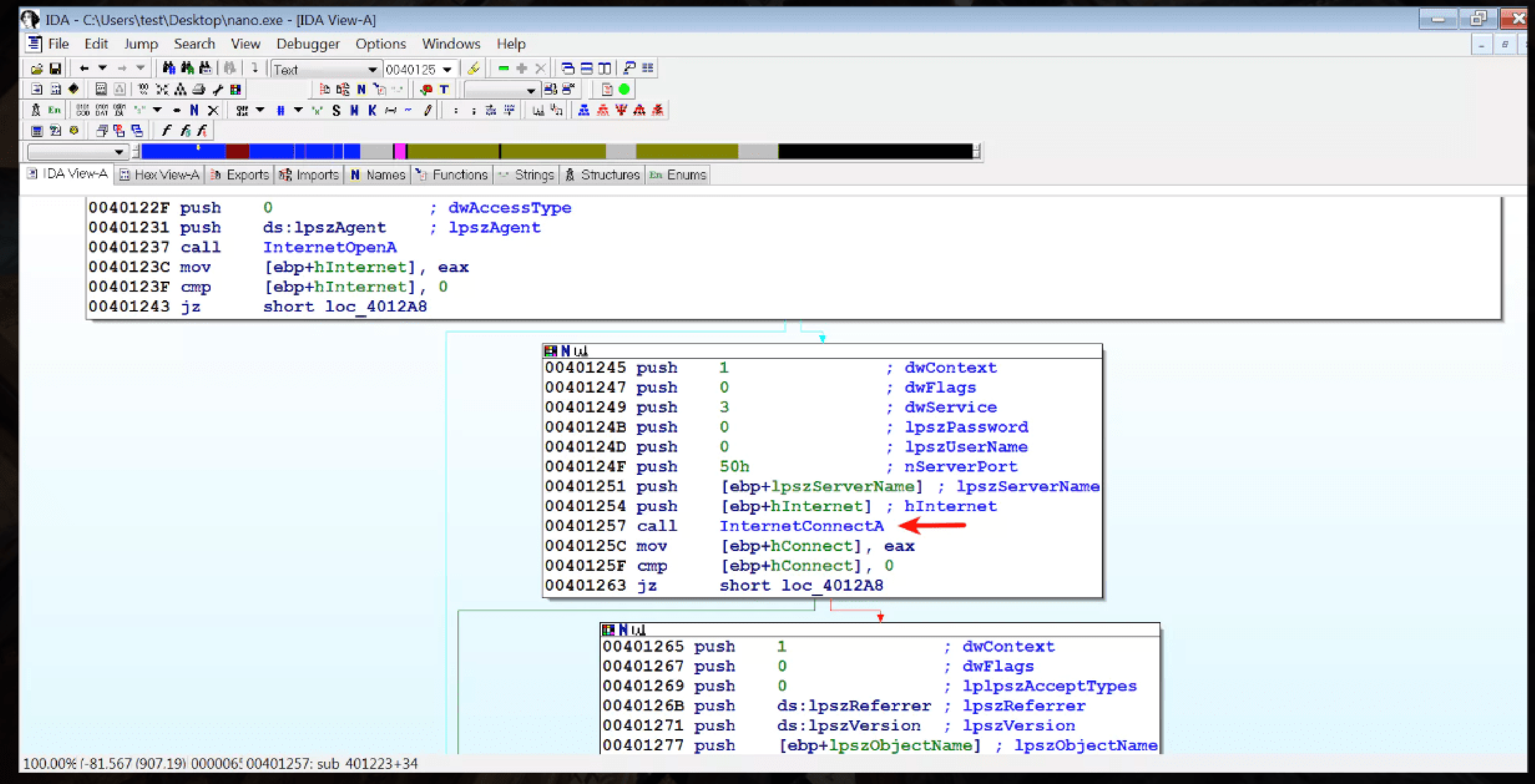
hInternet [in]

Handle returned by a previous call to **InternetOpen**.

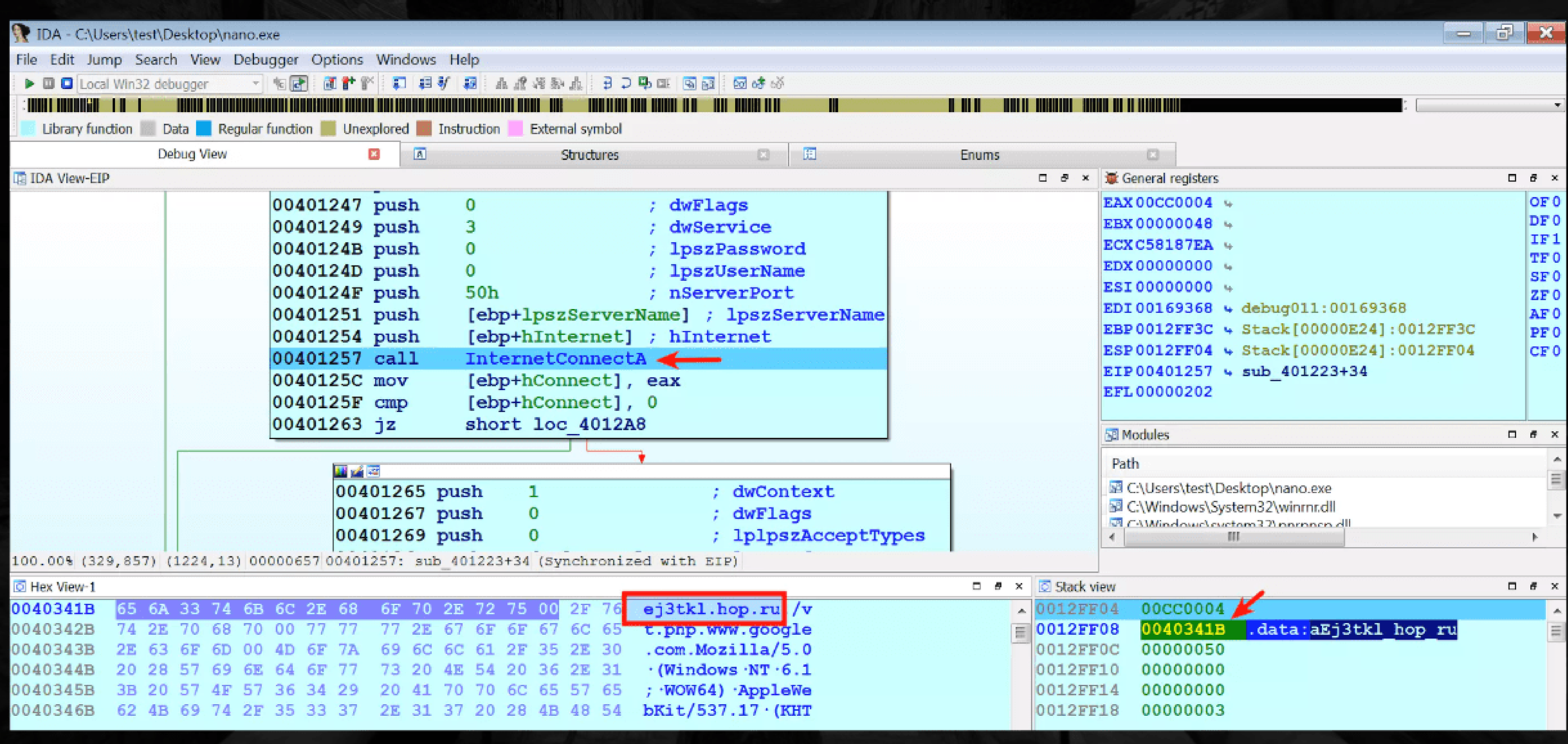
lpszServerName [in]

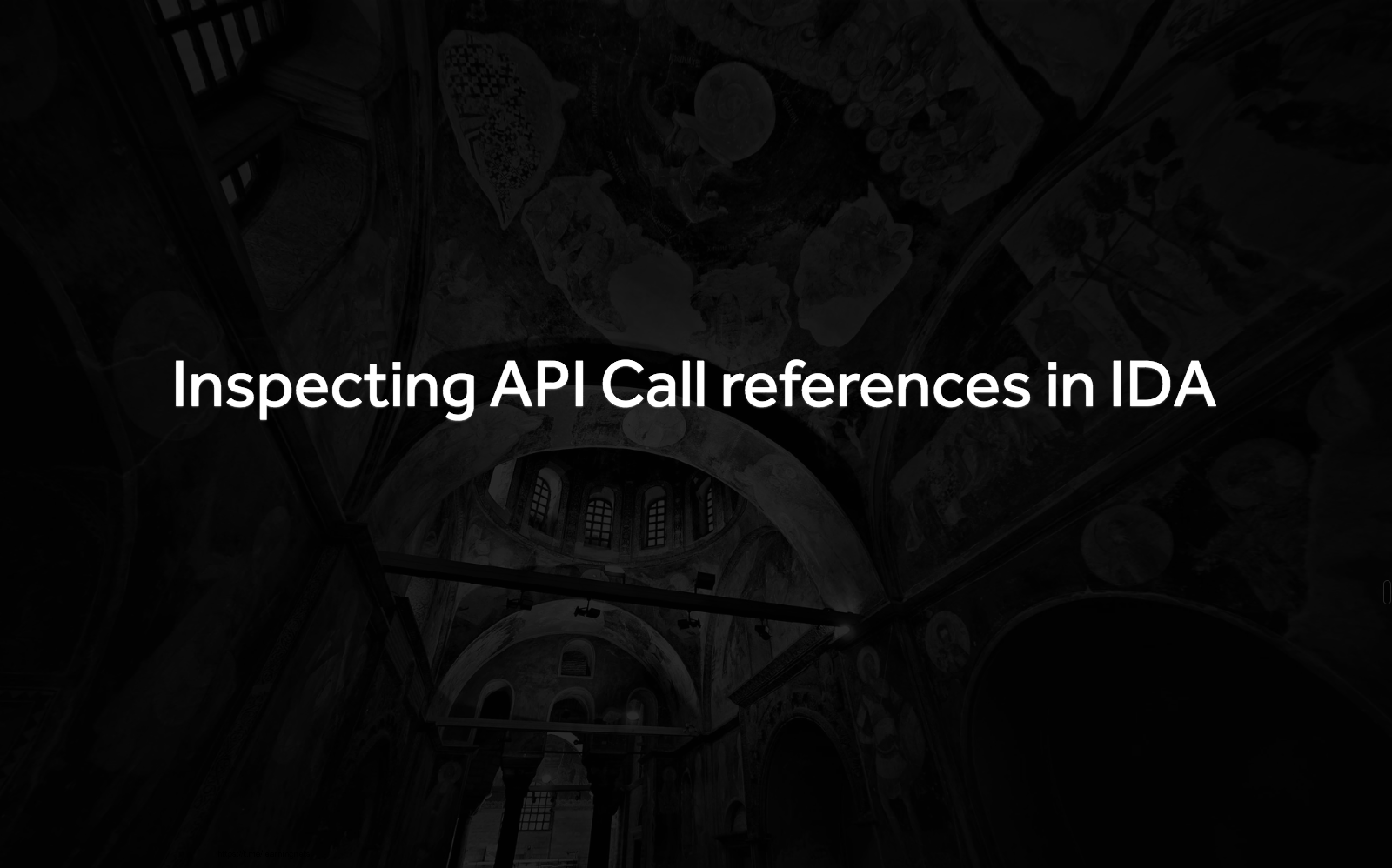
Pointer to a **null**-terminated string that specifies the host name of an Internet server. Alternately, the string can contain the IP number of the site, in ASCII dotted-decimal format (for example, 11.0.1.45).

Below screenshot shows the disassembled code of a malware sample using IDA, the malware uses ***InternetConnectA*** API call to open an HTTP session with C2 server which is the 2nd parameter to the API call but the server name is not clear just by looking at the disassembled code

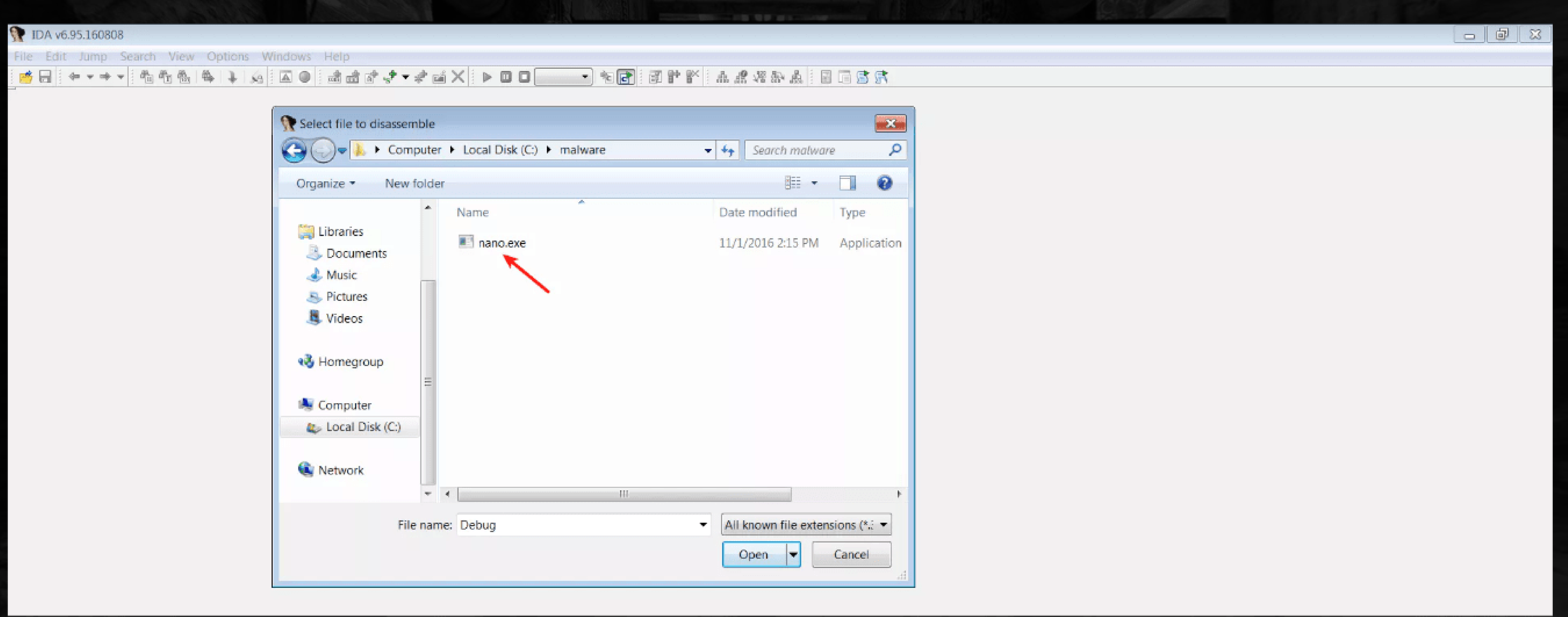


Debugging the malware sample and inspecting the *2nd* parameter on the stack displays the name of the C2 server that malware will connect to

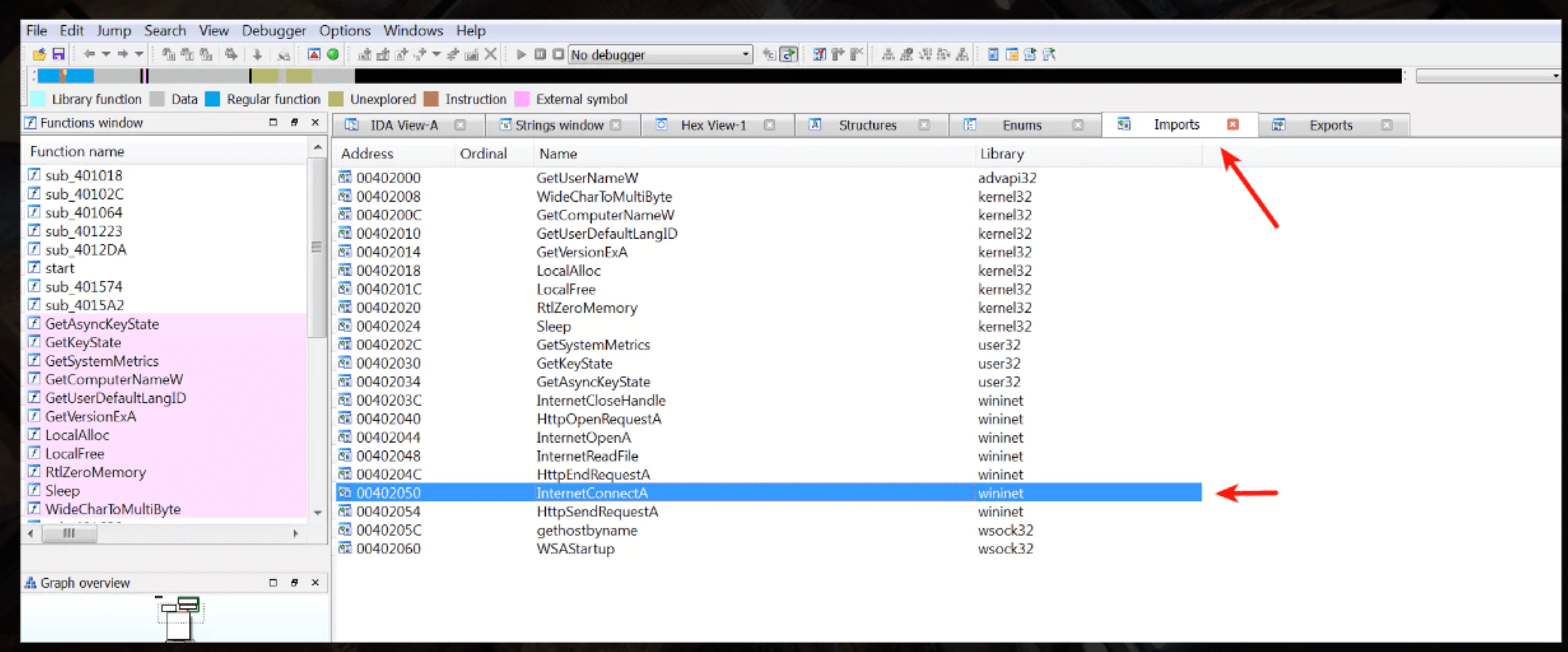



Inspecting API Call references in IDA

Load the executable in IDA and click on '*Open*"



Click on the ***imports*** tab, this will display all the imported functions as shown below. In this example, Let's try to find where ***InternetConnectA()*** api call is used in the code

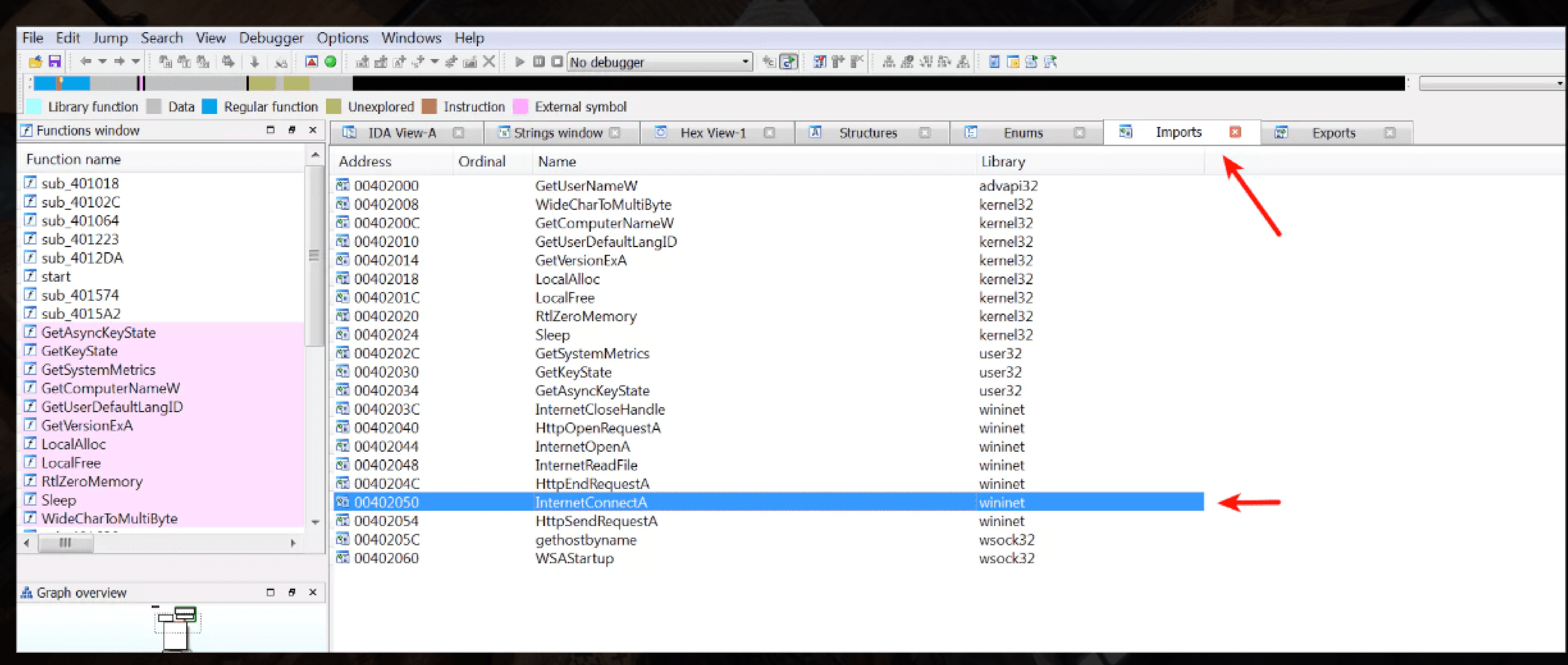


Double clicking on the desired API call will bring up the window shown below

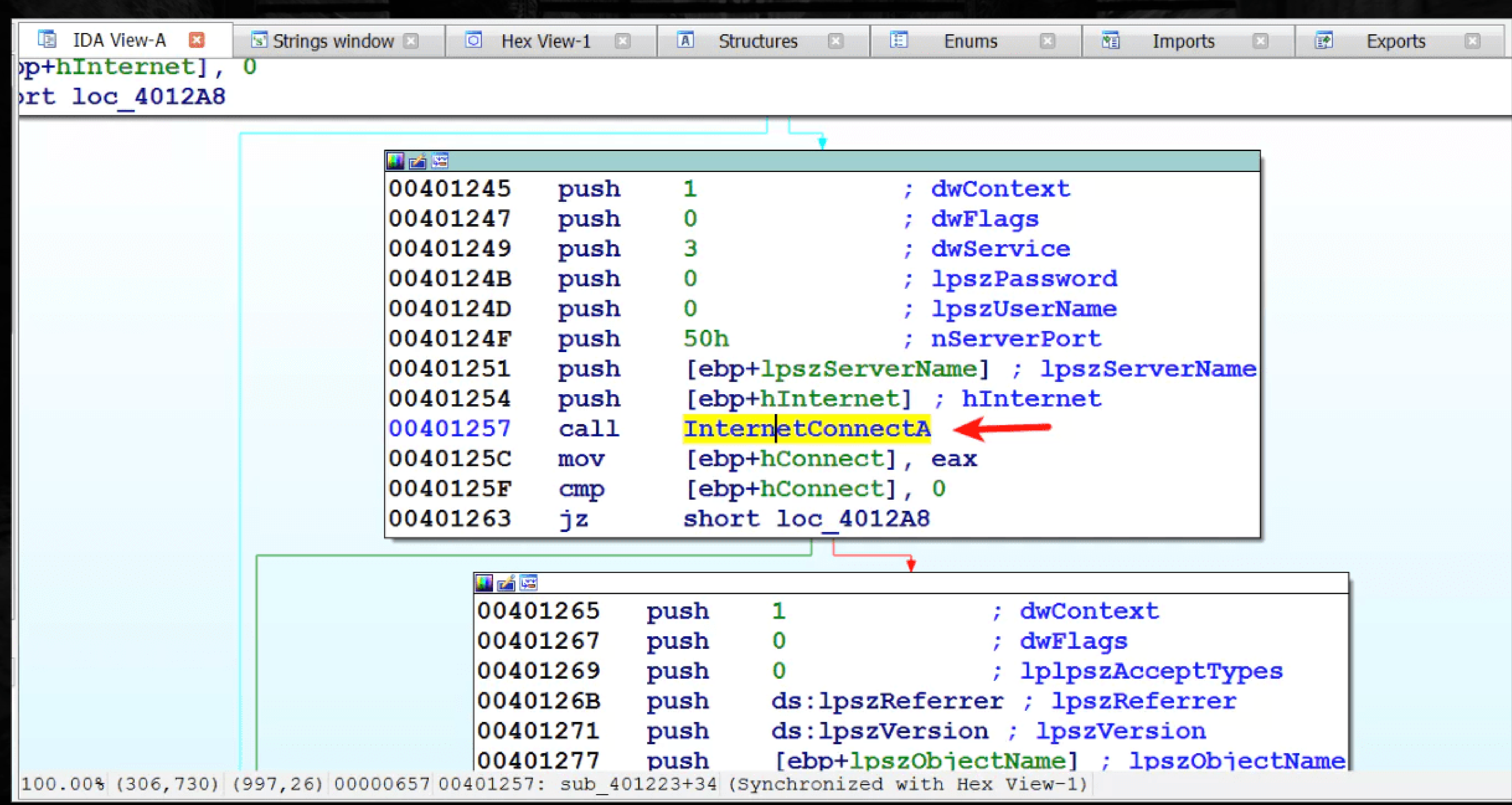
IDA View-A | Strings window | Hex View-1 | Structures | Enums | Imports | Exports

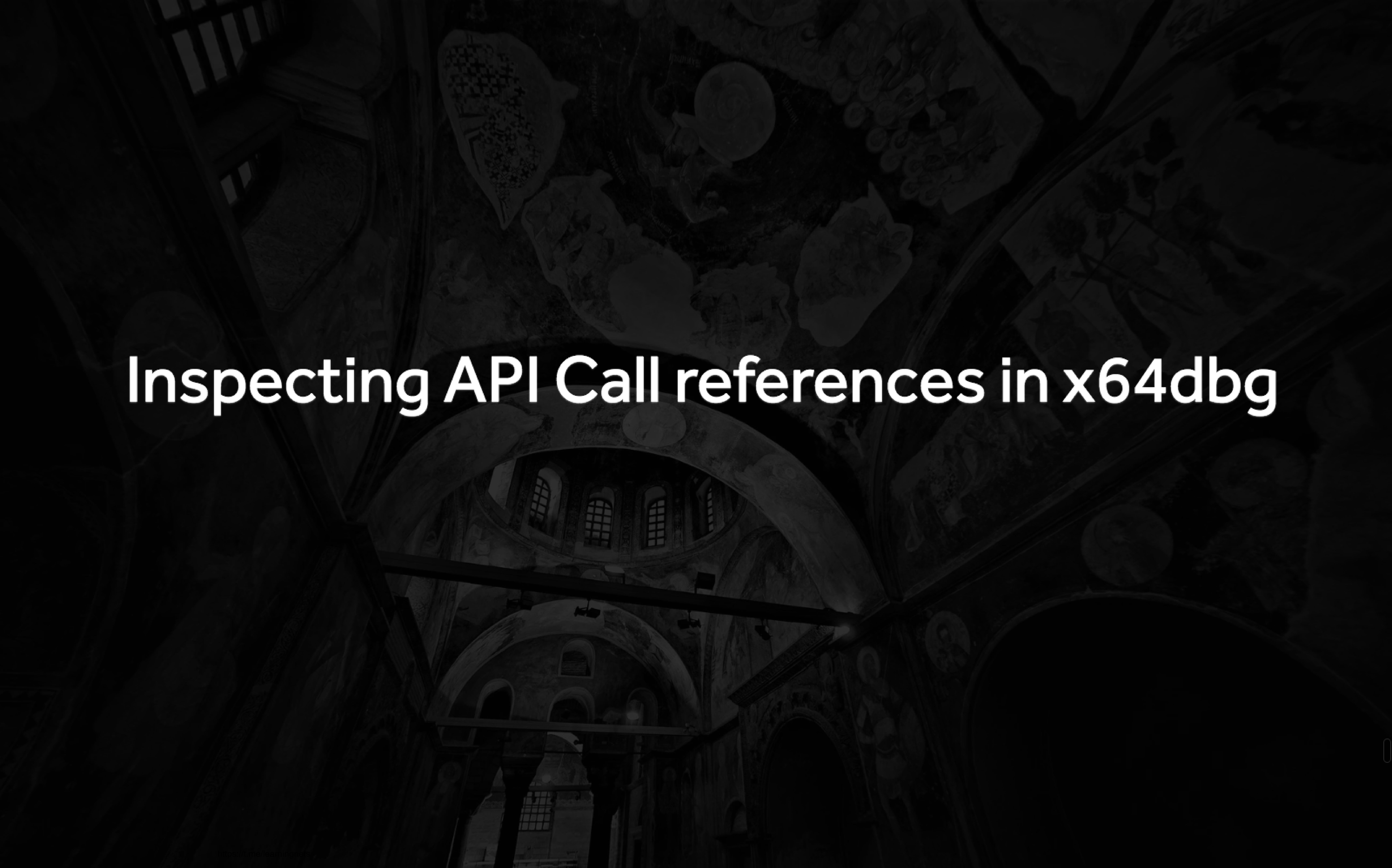
```
.idata:00402048                             ; DATA XREF: InternetReadFile↑r
.idata:0040204C ; BOOL __stdcall HttpEndRequestA(HINTERNET hRequest, LPINTERNET_BUFFERSA lpBuffersOut,
.idata:0040204C                 extrn __imp_HttpEndRequestA:dword
.idata:0040204C                             ; DATA XREF: HttpEndRequestA↑r
.idata:00402050 ; HINTERNET __stdcall InternetConnectA(HINTERNET hInternet, LPCSTR lpszServerName, INT
.idata:00402050                 extrn __imp_InternetConnectA:dword
.idata:00402050                             ; DATA XREF: InternetConnectA↑r
.idata:00402054 ; BOOL __stdcall HttpSendRequestA(HINTERNET hRequest, LPCSTR lpszHeaders, DWORD dwHead
.idata:00402054                 extrn __imp_HttpSendRequestA:dword
.idata:00402054                             ; DATA XREF: HttpSendRequestA↑r
.idata:00402058
.idata:0040205C ;
.idata:0040205C ; Imports from wsock32.dll
.idata:0040205C ;
.idata:0040205C ; struct hostent *__stdcall gethostbyname(const char *name)
.idata:0040205C                 extrn __imp_gethostbyname:dword ; DATA XREF: gethostbyname↑r
.idata:0040205C                             ; .rdata:004020C8↓o
.idata:00402060 ; int __stdcall WSAStartup(WORD wVersionRequested, LPWSADATA lpWSAData)
.idata:00402060                 extrn __imp_WSAStartup:dword ; DATA XREF: WSAStartup↑r
.idata:00402064
.idata:00402064
```

Highlight the API call and press ***'x'*** key that will display all the cross-references to the API. In this case, highlighting ***InternetConnectA()*** api call and pressing ***x*** key shows reference to the code where the api call is used as shown below

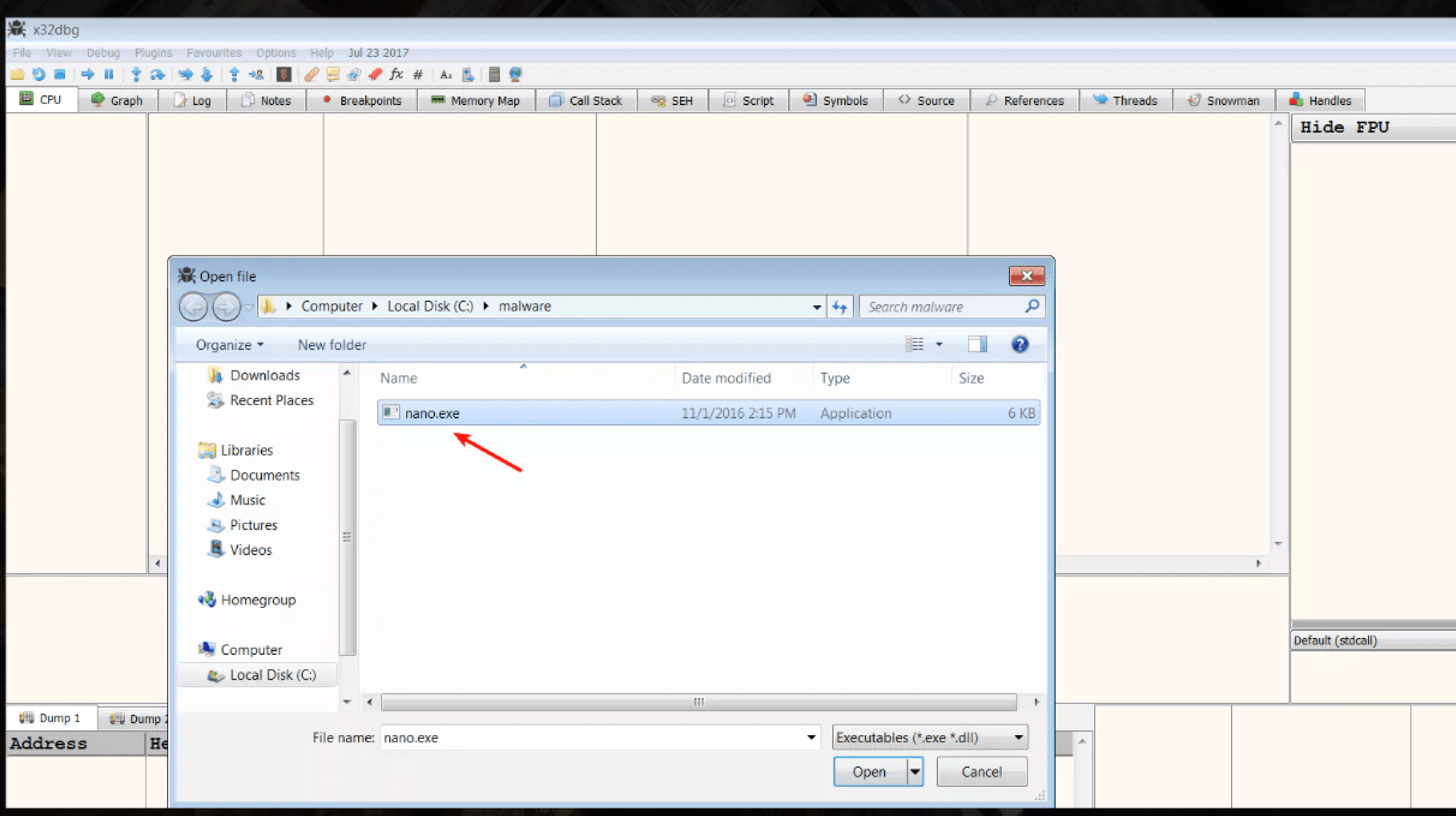


Highlight the entry that you want to examine and click ***ok***. The below screenshot shows the code where ***InternetConnectA()*** api is used now you can examine the code.

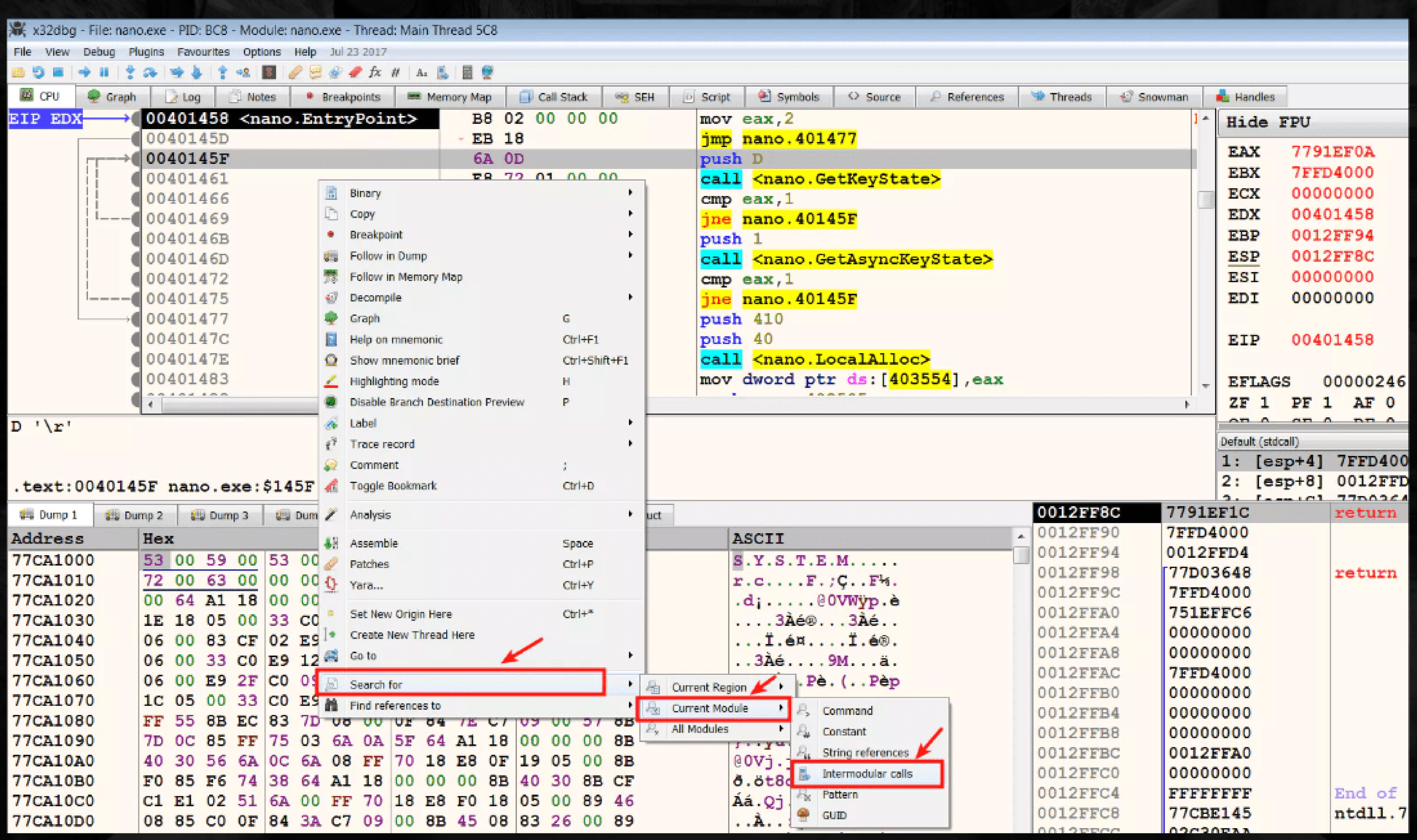



Inspecting API Call references in x64dbg

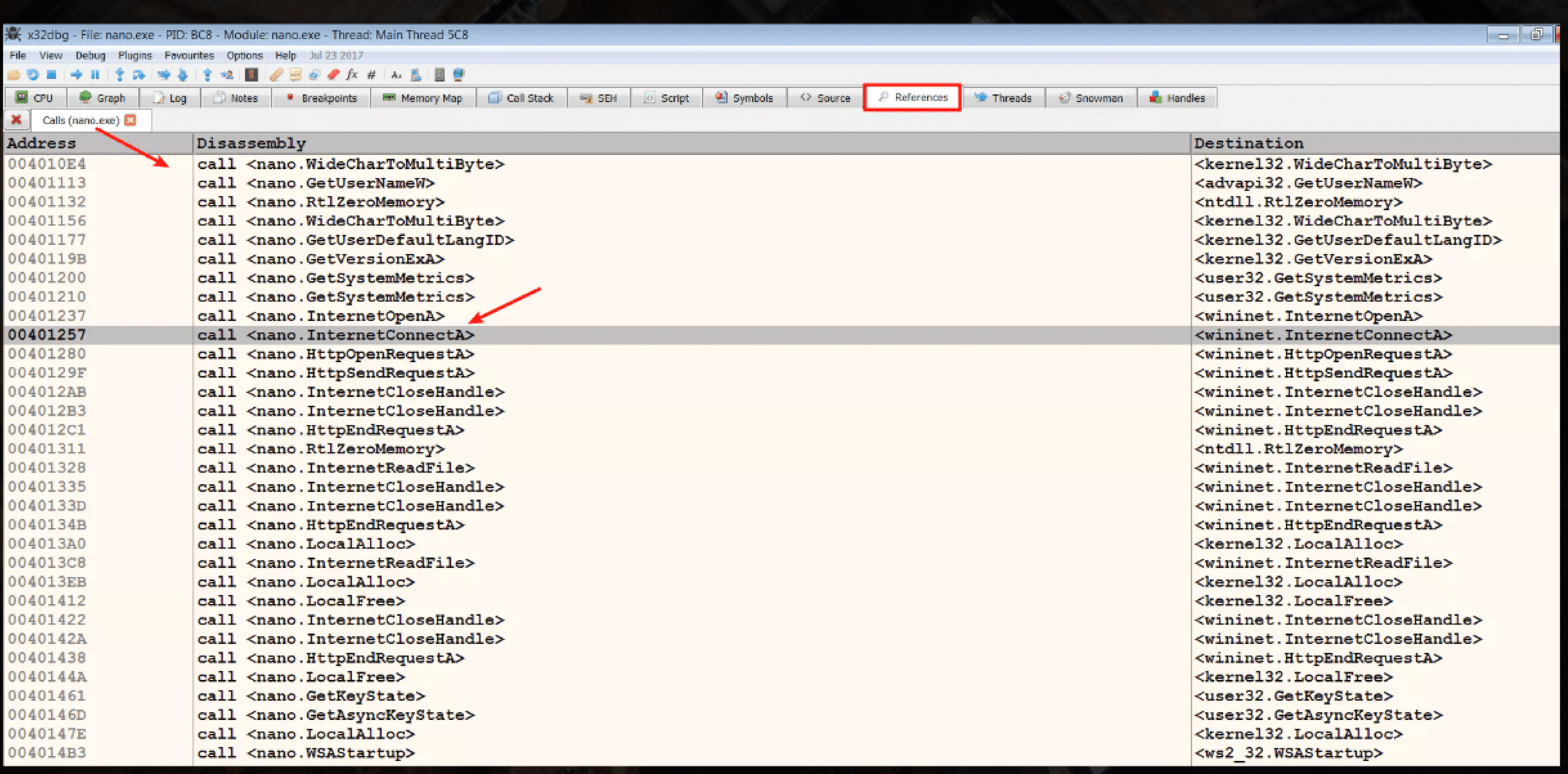
Load the executable in x64dbg by selecting ***File / Open***



Once the binary is loaded and paused at the entrypoint, right click on Disassembly window and choose ***Search For | Current Module | Intermodular calls*** as shown below



The following screenshot shows references to the API calls in the references Windows. You can double click on any entry to navigate to the code where the API is called



Double-clicking the entry in the ***references*** window will bring up the code where the API is referenced as shown below. At this point, you can set a breakpoint and run the program which pauses the execution before the call to API function and then you inspect the parameters passed to the API