Windows API

When reversing software you're going to encounter a lot of functionality introduced by the OS. Ultimately just about everything boils down to a call to the Windows API. For this reason it's invaluable to recognize and understand the Windows API. I will introduce you to some of the API, how it works, naming conventions, and common themes that will help strengthen you're understanding of the API.

It's good to note that understanding the Windows API can be incredibly powerful, especially if you want to do low level work with the OS. I write many of my own tools and programs that deal with OS level concepts such as reading/writing memory, process enumeration, system enumeration, registry manipulation, networking, and much more.

WOW64

WOW64 is a subsystem of Windows which stands for Windows 32 on Windows 64. It's a vital part of 64 bit Windows.

64 bit versions of Windows have two critical OS directories which contain various OS DLLs and programs shipped with the OS. These folders are C:\Windows\System32 and C:\Windows\SysWOW64. If you were to look in these directories you'd see their contents are almost exactly the same. The reason for these two directories existing is for backward compatibility and running software. When a program references a system DLL, it's going to pull it from one of those two locations. The way in which it determines which directory to look in is based on whether it's a 32 bit or 64 bit program. So this is where it gets a bit odd.

On 64 bit windows:

- 32 bit DLLs are located in C:\Windows\SysWOW64
- 64 bit DLLs are located in C:\Windows\System32

When a program wants to load a system DLL the default location is System32. This goes for both 64-bit and 32-bit. This is where a possible issue arises for 32-bit programs since 32 bit DLLs are in SysWOW64. WOW64 redirects the 32-bit program's request from System32 to SysWOW64. As you could probably guess this is only one of many things WOW64 does. For our purposes, knowing WOW64 exists and generally what it does is good enough. However, it is quite an interesting topic and the knowledge of it's inner workings can be helpful. For now though, here's a place to start if you want to look further into it.

https://docs.microsoft.com/en-us/archive/msdn-magazine/2006/may/x64-starting-out-in-64-bit-windows-systems-with-visual-c

API Usage

Now that we have WOW64 out of the way, let's get back to the Windows API (WinAPI). WinAPI has functionality for pretty much everything you can think of. As mentioned earlier, it's eventually required for anything you want to do with/to the OS. Because of the vastness that is the Windows API, I will only cover what's the most important. I strongly recommend you skim through various components of the API yourself.

Some API topics I will not cover, but may be useful to learn more about on your own:

- Graphics (Such as DWM, GDI, etc.)
- User Input
- Hooking (Could be very useful to know for process exploitation/manipulation)
- PnP

What I WILL cover:

- General Terminology (Naming conventions, trends, data types, etc.)
- File I/O
- Registry
- Sockets (Networking)
- Processes
- Threads
- Memory
- Services

Here is a great place to look for further research: https://docs.microsoft.com/en-us/windows/win32/apiindex/windows-api-list

General Terminology

- Function Naming Conventions:
    - A - ASCII, utilizes ASCII for character encoding which is 1 byte per character.
    - W - Wide, the function utilizes wide character encoding, aka 2 byte Unicode.
    - Ex (Such as CopyFileEx) - Extended, it's a reworked/redeveloped version of a previous function. Think of it as an updated function renamed to keep compatibility with other functions. If they were to just change the function without renaming it the changes may adversely affect current programs.
    -