Analyzing Windows Programs - WinAPI, Handles, Networking, COM

ISEA MOOC on Malware Analysis

By
Department of Computer Science & Engineering
Malaviya National Intstitute of Technology Jaipur

Overview

Analyzing Windows Programs

- Windows non-malicious binaries are well-structured by compilers and follow Microsoft guidelines.
- The malicious programs poorly formed and tends to perform unexpected actions.
- To analyze the malware functionality, we need to discuss various windows aspects.
- These aspects include Windows API, Windows registry, user modes, and kernel modes.

Note

- Malware programs modify various host-based artifacts therefore understanding these host-based indicators will help us in analyzing malware programs.

The Windows API

Windows API [1/3]

- The Windows API is a broad set of functionality that governs the way that software programs (malware also) interact with the Microsoft libraries.
- Following are terms and conventions important while dealing with Windows API:
  - **Types:** The `DWORD` and `WORD` types represent 32-bit and 16-bit unsigned integers. Standard C types like int, short, and unsigned int are not normally used.
  - **Hungarian Notation:** Windows generally uses *Hungarian notation* for API function identifiers. This notation uses a prefix naming scheme that makes it easy to identify a variable's type. Variables that contain a 32-bit unsigned integer, or `DWORD`, start with `dw`.
  - **Handles:** *Handles* are items that have been opened or created in the OS, such as a window, process, module, menu, file, and so on.
  - **File System Functions:** Microsoft provides various functions for accessing file systems like `CreateFile`, `ReadFile`, `WriteFile`, etc.
  - **Special Files:** Windows has some special files like regular files but cannot be accessed by their drive paths. Certain special files can provide greater access to system hardware and internal data.

Windows API [2/3]

Figure 1: Common API Types

Type and prefix	Description
WORD (w)	A 16-bit unsigned value.
DWORD (dw)	A double-WORD, 32-bit unsigned value.
Handles (H)	A reference to an object. The information stored in the handle is not documented, and the handle should be manipulated only by the Windows API. Examples include HModule, HInstance, and HKey.
Long Pointer (LP)	A pointer to another type. For example, LPByte is a pointer to a byte, and LPCSTR is a pointer to a character string. Strings are usually prefixed by LP because they are actually pointers. Occasionally, you will see Pointer (P)... prefixing another type instead of LP; in 32-bit systems, this is the same as LP. The difference was meaningful in 16-bit systems.
Callback	Represents a function that will be called by the Windows API. For example, the InternetSetStatusCallback function passes a pointer to a function that is called whenever the system has an update of the Internet status.

Windows API [3/3]

- **Shared Files:** Files are special files with names that start with `\\serverName\share` or `\\?\serverName\share`. They access directories or files in a shared folder stored on a network. The `\\?\` prefix tells the OS to disable all string parsing, and it allows access to longer filenames.

- **Files Accessible via Namespaces:** Additional files are accessible via namespaces within the OS. Namespaces can be thought of as a fixed number of folders, each storing different types of objects. The lowest level namespace is the NT namespace with the prefix `\`. The NT namespace has access to all devices, and all other namespaces exist within the NT namespace.

- **Alternate Data Streams:** The Alternate Data Streams (ADS) feature allows additional data to be added to an existing file within NTFS, essentially adding one file to another. The extra data does not show up in a directory listing, and it is not shown when displaying the contents of the file; it's visible only when you access the stream.

The Windows Registry

Windows Registry

- The Windows registry is used to store OS and program configuration information, such as settings and options.
- Early versions of Windows used `.ini` files to store configuration information.
- The registry was created as a hierarchical database of information to improve performance, and its importance has grown as more applications use it to store information.
- Nearly all Windows configuration information is stored in the registry, including networking, driver, startup, user account, and other information.
- Malware often uses the registry for persistence or configuration data. The malware adds entries into the registry that will allow it to run automatically when the computer boots.

Windows Registry: Important Terms

- **Root Key:** The registry is divided into five top-level sections called root keys, each with different purpose. Sometimes, the terms HKEY and hive are also used.
- **Subkey:** A subkey is like a subfolder within a folder.
- **Key:** A key is a folder in the registry that can contain additional folders or values. The root keys and subkeys are both keys.
- **Value Entry:** A value entry is an ordered pair with a name and value.
- **Value or data:** The value or data is the data stored in a registry entry.

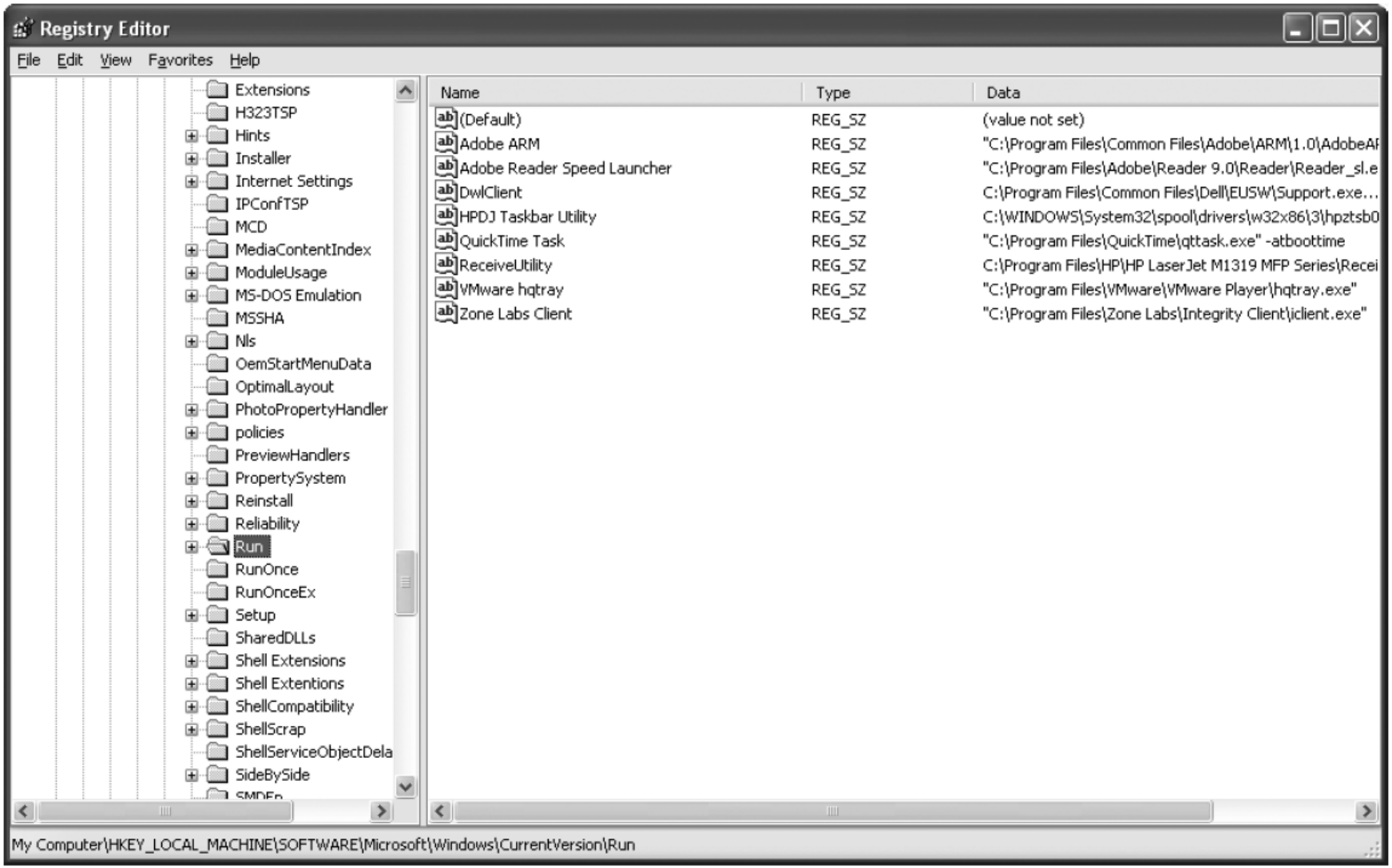
Root Keys

- `HKEY_LOCAL_MACHINE (HKLM)`: Stores settings that are global to the local machine.
- `HKEY_CURRENT_USER (HKCU)`: Stores settings specific to the current user.
- `HKEY_CLASSES_ROOT`: Stores information defining types.
- `HKEY_CURRENT_CONFIG`: Stores settings about the current hardware configuration, specifically differences between the current and the standard configuration.
- `HKEY_USERS`: Defines settings for the default user, new users, and current users.

Registry Editor [1/2]

- The Registry Editor (Regedit) is a built-in Windows tool used to view and edit the registry.
- The window on the left shows the open subkeys. The window on the right shows the value entries in the subkey.
- Each value entry has a name, type, and value. The full path for the subkey currently being viewed is shown at the bottom of the window.
- Writing entries to the *Run subkey* is a well-known way to set up software to run automatically. While not a very stealthy technique, it is often used by malware to launch itself automatically.

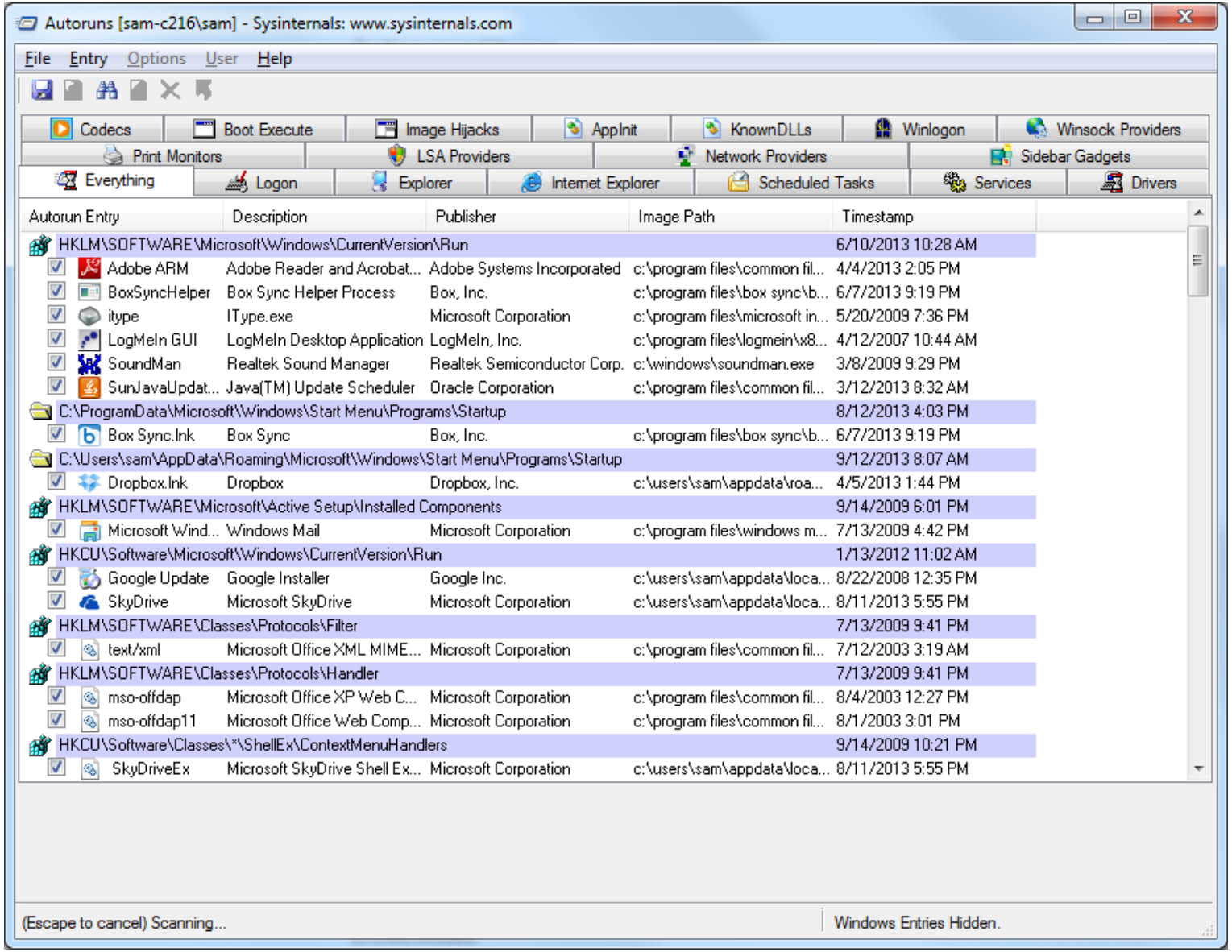
Registry Editor [2/2]



Autoruns [1/2]

- Sysinternals tool
- Lists code that will run automatically when system starts
  - Executables
  - DLLs loaded into IE and other programs
  - Drivers loaded into Kernel
  - It checks 25 to 30 registry locations
  - Won't necessarily find all automatically running code

Autoruns [2/2]


Autoruns [sam-c216\sam] - Sysinternals: www.sysinternals.com
File Entry Options User Help
Codecs Boot Execute Image Hijacks AppInit KnownDLLs Winlogon Winsock Providers
Print Monitors LSA Providers Network Providers Sidebar Gadgets
Everything Logon Explorer Internet Explorer Scheduled Tasks Services Drivers
Autorun Entry Description Publisher Image Path Timestamp
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run 6/10/2013 10:28 AM
Adobe ARM Adobe Reader and Acrobat... Adobe Systems Incorporated c:\program files\common fil... 4/4/2013 2:05 PM
BoxSyncHelper Box Sync Helper Process Box, Inc. c:\program files\box sync\b... 6/7/2013 9:19 PM
itype IType.exe Microsoft Corporation c:\program files\microsoft in... 5/20/2009 7:36 PM
LogMeIn GUI LogMeIn Desktop Application LogMeIn, Inc. c:\program files\logmein\x8... 4/12/2007 10:44 AM
SoundMan Realtek Sound Manager Realtek Semiconductor Corp. c:\windows\soundman.exe 3/8/2009 9:29 PM
SunJavaUpdat... Java(TM) Update Scheduler Oracle Corporation c:\program files\common fil... 3/12/2013 8:32 AM
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup 8/12/2013 4:03 PM
Box Sync.lnk Box Sync Box, Inc. c:\program files\box sync\b... 6/7/2013 9:19 PM
C:\Users\sam\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup 9/12/2013 8:07 AM
Dropbox.lnk Dropbox Dropbox, Inc. c:\users\sam\appdata\roa... 4/5/2013 1:44 PM
HKLM\SOFTWARE\Microsoft\Active Setup\Installed Components 9/14/2009 6:01 PM
Microsoft Wind... Windows Mail Microsoft Corporation c:\program files\windows m... 7/13/2009 4:42 PM
HKCU\Software\Microsoft\Windows\CurrentVersion\Run 1/13/2012 11:02 AM
Google Update Google Installer Google Inc. c:\users\sam\appdata\loca... 8/22/2008 12:35 PM
SkyDrive Microsoft SkyDrive Microsoft Corporation c:\users\sam\appdata\loca... 8/11/2013 5:55 PM
HKLM\SOFTWARE\Classes\Protocols\Filter 7/13/2009 9:41 PM
text/xml Microsoft Office XML MIME... Microsoft Corporation c:\program files\common fil... 7/12/2003 3:19 AM
HKLM\SOFTWARE\Classes\Protocols\Handler 7/13/2009 9:41 PM
mso-offdap Microsoft Office XP Web C... Microsoft Corporation c:\program files\common fil... 8/4/2003 12:27 PM
mso-offdap11 Microsoft Office Web Comp... Microsoft Corporation c:\program files\common fil... 8/1/2003 3:01 PM
HKCU\Software\Classes*\ShellEx\ContextMenuHandlers 9/14/2009 10:21 PM
SkyDriveEx Microsoft SkyDrive Shell Ex... Microsoft Corporation c:\users\sam\appdata\loca... 8/11/2013 5:55 PM
(Escape to cancel) Scanning...
Windows Entries Hidden.

Common Registry Functions

- Malware often uses registry functions that are part of the Windows API in order to modify the registry to run automatically when the system boots.
- The following are the most common registry functions:
  - `RegOpenKeyEx` opens a registry for editing and querying. There are functions that allow you to query and edit a registry key without opening it first, but most programs use `RegOpenKeyEx` anyway.
  - `RegSetValueEx` adds a new value to the registry and sets its data.
  - `RegGetValue` returns the data for a value entry in the registry.

Note

- The registry keys and functions can be used by the malware programs to autorun their code.

Analyzing Registry Code [1/2]

- Following code shows real malware opening the Run key from the registry and adding a value so that the program runs each time Windows starts.

```
0040286F   push    2                 ; samDesired
00402871   push    eax               ; ulOptions
00402872   push    offset SubKey     ; "Software\\Microsoft\\Windows\\CurrentVersion\\Run"
00402877   push    HKEY_LOCAL_MACHINE ; hKey
0040287C ❶call     esi ; RegOpenKeyExW
0040287E   test    eax, eax
00402880   jnz     short loc_4028C5
00402882
00402882 loc_402882:
00402882   lea     ecx, [esp+424h+Data]
00402886   push    ecx               ; lpString
00402887   mov     bl, 1
00402889 ❷call     ds:lstrlenW
0040288F   lea     edx, [eax+eax+2]
00402893 ❸push     edx               ; cbData
00402894   mov     edx, [esp+428h+hKey]
00402898 ❹lea      eax, [esp+428h+Data]
0040289C   push    eax               ; lpData
0040289D   push    1                 ; dwType
0040289F   push    0                 ; Reserved
004028A1 ❺lea      ecx, [esp+434h+ValueName]
004028A8   push    ecx               ; lpValueName
004028A9   push    edx               ; hKey
004028AA   call    ds:RegSetValueExW
```

Analyzing Registry Code [2/2]

- The code calls the `RegOpenKeyEx` function at ① with the parameters needed to open a handle to the registry key `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`.
- The value name at ⑤ and data at ④ are stored on the stack as parameters to this function, and are shown here as having been labeled by IDA Pro.
- The call to `lstrlenW` at ② is needed in order to get the size of the data, which is given as a parameter to the `RegSetValueEx` function at ③.

Comments in the code

- The comments give information about the meanings of the values being pushed.
- The `samDesired` value indicates the type of security access requested, the `ulOptions` field is an unsigned long integer representing the options for the call, and the `hKey` is the handle to the root key being accessed.

Networking APIs

Networking APIs

- Malware commonly relies on network functions to do its dirty work, and there are many Windows API functions for network communication.
- **Berkeley Compatible Sockets:** Malware most commonly uses Berkeley compatible sockets, functionality that is almost identical on Windows and UNIX systems.

Function	Description
`socket`	Creates a socket
`bind`	Attaches a socket to a particular port, prior to the accept call
`listen`	Indicates that a socket will be listening for incoming connections
`accept`	Opens a connection to a remote socket and accepts the connection
`connect`	Opens a connection to a remote socket; the remote socket must be waiting for the connection
`recv`	Receives data from the remote socket
`send`	Sends data to the remote socket

Server and Client Sides

- Malware can use any of the sides (Server or client) while implementing networking programming for spreading infection.
- Client-side applications that connect to a remote socket, you will see the `socket` call, `connect` call, `send` and `recv` call as necessary.
- A service application that listens for incoming connections, follows the sequence of calls as - the `socket`, `bind`, `listen`, and `accept` functions followed by `send` and `recv`, as necessary.

The WinINet API

- In addition to the Winsock API, there is a higher-level API called the WinINet.
- To use the functions of this API, the application imports the functions form DLL `Wininet.dll`.
- The WinINet API implements protocols, such as HTTP and FTP, at the application layer. We can gain an understanding of what malware is doing based on the connections that it opens:
  - `InternetOpen` is used to initialize a connection to the Internet.
  - `InternetOpenUrl` is used to connect to a URL (which can be an HTTP page or an FTP resource).
  - `InternetReadFile` works much like the ReadFile function, allowing the program to read the data from a file downloaded from the Internet.

Note

- Malware can use the WinINet API to connect to a remote server and get further instructions for execution.

Following Running Malware

Transferring Execution

- There are many ways that malware can transfer execution in addition to the `jump` and `call` instructions.
  - **DLLs:** Dynamic link libraries (DLLs) are the current Windows way to use libraries to share code among multiple applications.
  - **Processes:** A process is a program being executed by Windows. Each process manages its own resources, such as open handles and memory.
  - **Threads:** Processes are the container for execution, but threads are what the Windows OS executes. Threads are independent sequences of instructions that are executed by the CPU without waiting for other threads.
  - **Mutexes:** Mutexes are global objects that coordinate multiple processes and threads.
  - **Services:** Windows allows tasks to run without their own processes or threads by using services that run as background applications.
  - **Component Object Model (COM):** The Microsoft Component Object Model (COM) is an interface standard that makes it possible for different software components to call each other's code without knowledge of specifics about each other.
  - **Exceptions:** Exceptions allow a program to handle events outside the flow of normal execution. Most of the time, exceptions are caused by errors, such as division by zero.

DLLs

- A DLL is an executable file that does not run alone, but exports functions that can be used by other applications.
- The main advantage of using DLLs over static libraries is that the memory used by the DLLs can be shared among running processes.
- When distributing an executable, we can use DLLs that are known to be on the host Windows system without needing to redistribute them. This helps software developers and malware writers minimize the size of their software distributions.
- DLLs are also a useful code-reuse mechanism.

How Malware Authors Use DLLs

- **To store malicious code:** Malware authors find it more advantageous to store malicious code in a DLL, rather than in an `.exe` file. Malware sometimes uses DLLs to load itself into another process.

- **By using Windows DLLs:** The Windows DLLs contain the functionality needed to interact with the OS. The way that a malicious program uses the Windows DLLs often offers tremendous insight to the malware analyst.

- **By using third-party DLLs:** Malware can also use third–party DLLs to interact with other programs. When you see malware that imports functions from a third-party DLL, you can infer that it is interacting with that program to accomplish its goals.

Processes

- Malware can also execute code outside the current program by creating a new process or modifying an existing one.
- A process contains one or more threads that are executed by the CPU.
- Traditionally, malware has consisted of its own independent process, but newer malware more commonly executes its code as part of another process.
- Processes also contribute to stability by preventing errors or crashes in one program from affecting other programs.
- **Creating a New Process:** The function most commonly used by malware to create a new process is `CreateProcess`. This function has many parameters, and the caller has a lot of control over how it will be created.

Threads

- Processes are the container for execution, but threads are what the Windows OS executes.
- Threads are independent sequences of instructions that are executed by the CPU without waiting for other threads.
- A process contains one or more threads, which execute part of the code within a process.
- Threads within a process all share the same memory space, but each has its own processor registers and stack.
- **Thread Context:** When one thread is running, it has complete control of the CPU, or the CPU core, and other threads cannot affect the state of the CPU or core. When a thread changes the value of a register in a CPU, it does not affect any other threads. Before an OS switches between threads, all values in the CPU are saved in a structure called the thread context. The OS then loads the thread context of a new thread into the CPU and executes the new thread.

Creating a Thread

- The `CreateThread` function is used to create new threads. The function's caller specifies a start address, which is often called the start function.
- Malware can use `CreateThread` in multiple ways, such as the following:
  - Malware can use `CreateThread` to load a new malicious library into a process, with `CreateThread` called and the address of `LoadLibrary` specified as the start address.
  - Malware can create two new threads for input and output: one to listen on a socket or pipe and then output that to standard input of a process, and the other to read from standard output and send that to a socket or pipe. The malware's goal is to send all information to a single socket or pipe in order to communicate seamlessly with the running application.

Interprocess Coordination with Mutexes

- Mutexes are global objects that coordinate multiple processes and threads.
- Mutexes are mainly used to control access to shared resources (like memory), and are often used by malware.
- Mutexes are important to malware analysis because they often use hard-coded names, which make good host-based indicators.
- The thread gains access to the mutex with a call to `WaitForSingleObject`, and any subsequent threads attempting to gain access to it must wait. When a thread is finished using a mutex, it uses the `ReleaseMutex` function.
- Malware will commonly create a mutex (`CreateMutex` function) and attempt to open an existing mutex with the same name to ensure that only one version of the malware is running at a time.

Services [1/2]

- Another way for malware to execute additional code is by installing it as a service.
- Windows allows tasks to run without their own processes or threads by using services that run as background applications.
- Using services has many advantages for the malware writer.
  - Services are normally run as `SYSTEM` or another privileged account. Malware writers use services as the `SYSTEM` account has more access than administrator or user accounts.
  - Services also provide another way to maintain persistence on a system, because they can be set to run automatically when the OS starts, and may not even show up in the Task Manager as a process.

Services [2/2]

- Services can be installed and manipulated via a few Windows API functions, which are prime targets for malware. There are several key functions to look for:
  - `OpenSCManager` Returns a handle to the service control manager, which is used for all subsequent service-related function calls. All code that will interact with services will call this function.
  - `CreateService` Adds a new service to the service control manager, and allows the caller to specify whether the service will start automatically at boot time or must be started manually.
  - `StartService` Starts a service, and is used only if the service is set to be started manually.
- The Windows OS supports several different service types, the most commonly used by malware is the `WIN32_SHARE_PROCESS` type, which stores the code for the service in a DLL, and combines several different services in a single, shared process.
- In Task Manager, you can find several instances of a process called svchost.exe, which are running `WIN32_SHARE_PROCESS`-type services.

Component Object Model (COM)

- The Microsoft Component Object Model (COM) is an interface standard that makes it possible for different software components to call each other's code without knowledge of specifics about each other.
- When analyzing malware that uses COM, you'll need to be able to determine which code will be run as a result of a COM function call.
- COM works with any programming language and was designed to support reusable software components that could be utilized by all programs.
- Malware that uses COM functionality can be difficult to analyze.
- Each thread that uses COM must call the `OleInitialize` or `CoInitializeEx` function at least once prior to calling any other COM library functions.
- For analyzing malicious Windows programs, a malware analyst can search for these calls to determine whether a program is using COM functionality.

Exceptions

- Exceptions allow a program to handle events outside the flow of normal execution.
- Mostly, exceptions are caused by errors, such as division by zero, are raised by hardware; others, such as an invalid memory access, are raised by the OS.
- We can also raise an exception explicitly in code with the `RaiseException` call.
- Structured Exception Handling (SEH) is the Windows mechanism for handling exceptions.
- Exception handlers can be used in exploit code to gain execution.
- A pointer to exception-handling information is stored on the stack, and during a stack overflow, an attacker can overwrite the pointer. By specifying a new exception handler, the attacker gains execution when an exception occurs.

Summary

- We discussed various Windows concepts that are important to malware analysis.
- The concepts such as Windows APIs, processes, threads, and network functionality will be very useful if you are analyzing malware.
- In order to understand malware functionality, we must follow these techniques with different Windows internals.