A

IMPORTANT WINDOWS FUNCTIONS

This appendix contains a list of Windows functions commonly encountered by malware analysts, along with a short description of each one and how it is likely to be used by malware. Most of these functions are already documented by Microsoft, and this appendix is not intended to rehash that information. The Microsoft documentation is extremely useful and describes almost every function exported by a Microsoft DLL, although it can be lengthy and technical.

You can use this appendix as a reference when performing basic static analysis, whether you're trying to glean information from the import table or just looking for advanced techniques to point you in the right direction. Once you've determined which functions are most relevant for a particular piece of malware, you will need to analyze those functions in disassembly and use the Microsoft documentation to learn the purpose of each parameter.

NOTE *This appendix presents a selective list of functions. We have excluded functions whose purpose should be clear from the function name alone, such as* `ReadFile` *and* `DeleteFile`*.*

`accept`

Used to listen for incoming connections. This function indicates that the program will listen for incoming connections on a socket.

`AdjustTokenPrivileges`

Used to enable or disable specific access privileges. Malware that performs process injection often calls this function to gain additional permissions.

`AttachThreadInput`

Attaches the input processing for one thread to another so that the second thread receives input events such as keyboard and mouse events. Keyloggers and other spyware use this function.

`bind`

Used to associate a local address to a socket in order to listen for incoming connections.

`BitBlt`

Used to copy graphic data from one device to another. Spyware sometimes uses this function to capture screenshots. This function is often added by the compiler as part of library code.

`CallNextHookEx`

Used within code that is hooking an event set by `SetWindowsHookEx`. `CallNextHookEx` calls the next hook in the chain. Analyze the function calling `CallNextHookEx` to determine the purpose of a hook set by `SetWindowsHookEx`.

`CertOpenSystemStore`

Used to access the certificates stored on the local system.

`CheckRemoteDebuggerPresent`

Checks to see if a specific process (including your own) is being debugged. This function is sometimes used as part of an anti-debugging technique.

`CoCreateInstance`

Creates a COM object. COM objects provide a wide variety of functionality. The class identifier (CLSID) will tell you which file contains the code that implements the COM object. See Chapter 7 for an in-depth explanation of COM.

`connect`

Used to connect to a remote socket. Malware often uses low-level functionality to connect to a command-and-control server.

`ConnectNamedPipe`

Used to create a server pipe for interprocess communication that will wait for a client pipe to connect. Backdoors and reverse shells sometimes use `ConnectNamedPipe` to simplify connectivity to a command-and-control server.

`ControlService`

Used to start, stop, modify, or send a signal to a running service. If malware is using its own malicious service, you'll need to analyze the code that implements the service in order to determine the purpose of the call.

`CreateFile`

Creates a new file or opens an existing file.

`CreateFileMapping`

Creates a handle to a file mapping that loads a file into memory and makes it accessible via memory addresses. Launchers, loaders, and injectors use this function to read and modify PE files.

`CreateMutex`

Creates a mutual exclusion object that can be used by malware to ensure that only a single instance of the malware is running on a system at any given time. Malware often uses fixed names for mutexes, which can be good host-based indicators to detect additional installations of the malware.

`CreateProcess`

Creates and launches a new process. If malware creates a new process, you will need to analyze the new process as well.

`CreateRemoteThread`

Used to start a thread in a remote process (one other than the calling process). Launchers and stealth malware use `CreateRemoteThread` to inject code into a different process.

`CreateService`

Creates a service that can be started at boot time. Malware uses `CreateService` for persistence, stealth, or to load kernel drivers.

`CreateToolhelp32Snapshot`

Used to create a snapshot of processes, heaps, threads, and modules. Malware often uses this function as part of code that iterates through processes or threads.

`CryptAcquireContext`

Often the first function used by malware to initialize the use of Windows encryption. There are many other functions associated with encryption, most of which start with `Crypt`.

`DeviceIoControl`

Sends a control message from user space to a device driver. `DeviceIoControl` is popular with kernel malware because it is an easy, flexible way to pass information between user space and kernel space.

DllCanUnloadNow

An exported function that indicates that the program implements a COM server.

DllGetClassObject

An exported function that indicates that the program implements a COM server.

DllInstall

An exported function that indicates that the program implements a COM server.

DllRegisterServer

An exported function that indicates that the program implements a COM server.

DllUnregisterServer

An exported function that indicates that the program implements a COM server.

EnableExecuteProtectionSupport

An undocumented API function used to modify the Data Execution Protection (DEP) settings of the host, making it more susceptible to attack.

EnumProcesses

Used to enumerate through running processes on the system. Malware often enumerates through processes to find a process to inject into.

EnumProcessModules

Used to enumerate the loaded modules (executables and DLLs) for a given process. Malware enumerates through modules when doing injection.

FindFirstFile/FindNextFile

Used to search through a directory and enumerate the filesystem.

FindResource

Used to find a resource in an executable or loaded DLL. Malware sometimes uses resources to store strings, configuration information, or other malicious files. If you see this function used, check for a `.rsrc` section in the malware's PE header.

FindWindow

Searches for an open window on the desktop. Sometimes this function is used as an anti-debugging technique to search for OllyDbg windows.

FtpPutFile

A high-level function for uploading a file to a remote FTP server.

GetAdaptersInfo

Used to obtain information about the network adapters on the system. Backdoors sometimes call `GetAdaptersInfo` as part of a survey to gather information about infected machines. In some cases, it's used to gather MAC addresses to check for VMware as part of anti-virtual machine techniques.

GetAsyncKeyState

Used to determine whether a particular key is being pressed. Malware sometimes uses this function to implement a keylogger.

GetDC

Returns a handle to a device context for a window or the whole screen. Spyware that takes screen captures often uses this function.

GetForegroundWindow

Returns a handle to the window currently in the foreground of the desktop. Keyloggers commonly use this function to determine in which window the user is entering his keystrokes.

gethostbyname

Used to perform a DNS lookup on a particular hostname prior to making an IP connection to a remote host. Hostnames that serve as command-and-control servers often make good network-based signatures.

gethostname

Retrieves the hostname of the computer. Backdoors sometimes use `gethostname` as part of a survey of the victim machine.

GetKeyState

Used by keyloggers to obtain the status of a particular key on the keyboard.

GetModuleFilename

Returns the filename of a module that is loaded in the current process. Malware can use this function to modify or copy files in the currently running process.

GetModuleHandle

Used to obtain a handle to an already loaded module. Malware may use `GetModuleHandle` to locate and modify code in a loaded module or to search for a good location to inject code.

GetProcAddress

Retrieves the address of a function in a DLL loaded into memory. Used to import functions from other DLLs in addition to the functions imported in the PE file header.

GetStartupInfo

Retrieves a structure containing details about how the current process was configured to run, such as where the standard handles are directed.

GetSystemDefaultLangId

Returns the default language settings for the system. This can be used to customize displays and filenames, as part of a survey of an infected victim, or by "patriotic" malware that affects only systems from certain regions.

GetTempPath

Returns the temporary file path. If you see malware call this function, check whether it reads or writes any files in the temporary file path.

`GetThreadContext`

Returns the context structure of a given thread. The context for a thread stores all the thread information, such as the register values and current state.

`GetTickCount`

Retrieves the number of milliseconds since bootup. This function is sometimes used to gather timing information as an anti-debugging technique. `GetTickCount` is often added by the compiler and is included in many executables, so simply seeing it as an imported function provides little information.

`GetVersionEx`

Returns information about which version of Windows is currently running. This can be used as part of a victim survey or to select between different offsets for undocumented structures that have changed between different versions of Windows.

`GetWindowsDirectory`

Returns the file path to the Windows directory (usually *C:\Windows*). Malware sometimes uses this call to determine into which directory to install additional malicious programs.

`inet_addr`

Converts an IP address string like `127.0.0.1` so that it can be used by functions such as `connect`. The string specified can sometimes be used as a network-based signature.

`InternetOpen`

Initializes the high-level Internet access functions from WinINet, such as `InternetOpenUrl` and `InternetReadFile`. Searching for `InternetOpen` is a good way to find the start of Internet access functionality. One of the parameters to `InternetOpen` is the User-Agent, which can sometimes make a good network-based signature.

`InternetOpenUrl`

Opens a specific URL for a connection using FTP, HTTP, or HTTPS. URLs, if fixed, can often be good network-based signatures.

`InternetReadFile`

Reads data from a previously opened URL.

`InternetWriteFile`

Writes data to a previously opened URL.

`IsDebuggerPresent`

Checks to see if the current process is being debugged, often as part of an anti-debugging technique. This function is often added by the compiler and is included in many executables, so simply seeing it as an imported function provides little information.

`IsNTAdmin`

Checks if the user has administrator privileges.

`IsWoW64Process`

Used by a 32-bit process to determine if it is running on a 64-bit operating system.

`LdrLoadDll`

Low-level function to load a DLL into a process, just like `LoadLibrary`. Normal programs use `LoadLibrary`, and the presence of this import may indicate a program that is attempting to be stealthy.

`LoadLibrary`

Loads a DLL into a process that may not have been loaded when the program started. Imported by nearly every Win32 program.

`LoadResource`

Loads a resource from a PE file into memory. Malware sometimes uses resources to store strings, configuration information, or other malicious files.

`LsaEnumerateLogonSessions`

Enumerates through logon sessions on the current system, which can be used as part of a credential stealer.

`MapViewOfFile`

Maps a file into memory and makes the contents of the file accessible via memory addresses. Launchers, loaders, and injectors use this function to read and modify PE files. By using `MapViewOfFile`, the malware can avoid using `WriteFile` to modify the contents of a file.

`MapVirtualKey`

Translates a virtual-key code into a character value. It is often used by keylogging malware.

`MmGetSystemRoutineAddress`

Similar to `GetProcAddress` but used by kernel code. This function retrieves the address of a function from another module, but it can only get addresses from *ntoskrnl.exe* and *hal.dll*.

`Module32First`/`Module32Next`

Used to enumerate through modules loaded into a process. Injectors use this function to determine where to inject code.

`NetScheduleJobAdd`

Submits a request for a program to be run at a specified date and time. Malware can use `NetScheduleJobAdd` to run a different program. As a malware analyst, you'll need to locate and analyze the program that will be run in the future.

`NetShareEnum`

Used to enumerate network shares.

`NtQueryDirectoryFile`

Returns information about files in a directory. Rootkits commonly hook this function in order to hide files.

`NtQueryInformationProcess`

Returns various information about a specified process. This function is sometimes used as an anti-debugging technique because it can return the same information as `CheckRemoteDebuggerPresent`.

`NtSetInformationProcess`

Can be used to change the privilege level of a program or to bypass Data Execution Prevention (DEP).

`OleInitialize`

Used to initialize the COM library. Programs that use COM objects must call `OleInitialize` prior to calling any other COM functions.

`OpenMutex`

Opens a handle to a mutual exclusion object that can be used by malware to ensure that only a single instance of malware is running on a system at any given time. Malware often uses fixed names for mutexes, which can be good host-based indicators.

`OpenProcess`

Opens a handle to another process running on the system. This handle can be used to read and write to the other process memory or to inject code into the other process.

`OpenSCManager`

Opens a handle to the service control manager. Any program that installs, modifies, or controls a service must call this function before any other service-manipulation function.

`OutputDebugString`

Outputs a string to a debugger if one is attached. This can be used as an anti-debugging technique.

`PeekNamedPipe`

Used to copy data from a named pipe without removing data from the pipe. This function is popular with reverse shells.

`Process32First`/`Process32Next`

Used to begin enumerating processes from a previous call to `CreateToolhelp32Snapshot`. Malware often enumerates through processes to find a process to inject into.

`QueryPerformanceCounter`

Used to retrieve the value of the hardware-based performance counter. This function is sometimes using to gather timing information as part of an anti-debugging technique. It is often added by the compiler and is included in many executables, so simply seeing it as an imported function provides little information.

`QueueUserAPC`

Used to execute code for a different thread. Malware sometimes uses `QueueUserAPC` to inject code into another process.

`ReadProcessMemory`

Used to read the memory of a remote process.

`recv`

Receives data from a remote machine. Malware often uses this function to receive data from a remote command-and-control server.

`RegisterHotKey`

Used to register a handler to be notified anytime a user enters a particular key combination (like CTRL-ALT-J), regardless of which window is active when the user presses the key combination. This function is sometimes used by spyware that remains hidden from the user until the key combination is pressed.

`RegOpenKey`

Opens a handle to a registry key for reading and editing. Registry keys are sometimes written as a way for software to achieve persistence on a host. The registry also contains a whole host of operating system and application setting information.

`ResumeThread`

Resumes a previously suspended thread. `ResumeThread` is used as part of several injection techniques.

`RtlCreateRegistryKey`

Used to create a registry from kernel-mode code.

`RtlWriteRegistryValue`

Used to write a value to the registry from kernel-mode code.

`SamIConnect`

Connects to the Security Account Manager (SAM) in order to make future calls that access credential information. Hash-dumping programs access the SAM database in order to retrieve the hash of users' login passwords.

`SamIGetPrivateData`

Queries the private information about a specific user from the Security Account Manager (SAM) database. Hash-dumping programs access the SAM database in order to retrieve the hash of users' login passwords.

`SamQueryInformationUse`

Queries information about a specific user in the Security Account Manager (SAM) database. Hash-dumping programs access the SAM database in order to retrieve the hash of users' login passwords.

`send`

Sends data to a remote machine. Malware often uses this function to send data to a remote command-and-control server.

`SetFileTime`

Modifies the creation, access, or last modified time of a file. Malware often uses this function to conceal malicious activity.

`SetThreadContext`

Used to modify the context of a given thread. Some injection techniques use `SetThreadContext`.

`SetWindowsHookEx`
Sets a hook function to be called whenever a certain event is called. Commonly used with keyloggers and spyware, this function also provides an easy way to load a DLL into all GUI processes on the system. This function is sometimes added by the compiler.

`SfcTerminateWatcherThread`
Used to disable Windows file protection and modify files that otherwise would be protected. `SfcFileException` can also be used in this capacity.

`ShellExecute`
Used to execute another program. If malware creates a new process, you will need to analyze the new process as well.

`StartServiceCtrlDispatcher`
Used by a service to connect the main thread of the process to the service control manager. Any process that runs as a service must call this function within 30 seconds of startup. Locating this function in malware tells you that the function should be run as a service.

`SuspendThread`
Suspends a thread so that it stops running. Malware will sometimes suspend a thread in order to modify it by performing code injection.

`system`
Function to run another program provided by some C runtime libraries. On Windows, this function serves as a wrapper function to `CreateProcess`.

`Thread32First`/`Thread32Next`
Used to iterate through the threads of a process. Injectors use these functions to find an appropriate thread to inject into.

`Toolhelp32ReadProcessMemory`
Used to read the memory of a remote process.

`URLDownloadToFile`
A high-level call to download a file from a web server and save it to disk. This function is popular with downloaders because it implements all the functionality of a downloader in one function call.

`VirtualAllocEx`
A memory-allocation routine that can allocate memory in a remote process. Malware sometimes uses `VirtualAllocEx` as part of process injection.

`VirtualProtectEx`
Changes the protection on a region of memory. Malware may use this function to change a read-only section of memory to an executable.

`WideCharToMultiByte`
Used to convert a Unicode string into an ASCII string.

`WinExec`
Used to execute another program. If malware creates a new process, you will need to analyze the new process as well.

WlxLoggedOnSAS (and other Wlx* functions)

A function that must be exported by DLLs that will act as authentication modules. Malware that exports many Wlx* functions might be performing Graphical Identification and Authentication (GINA) replacement, as discussed in Chapter 11.

Wow64DisableWow64FsRedirection

Disables file redirection that occurs in 32-bit files loaded on a 64-bit system. If a 32-bit application writes to *C:\Windows\System32* after calling this function, then it will write to the real *C:\Windows\System32* instead of being redirected to *C:\Windows\SysWOW64*.

WriteProcessMemory

Used to write data to a remote process. Malware uses WriteProcessMemory as part of process injection.

WSAStartup

Used to initialize low-level network functionality. Finding calls to WSAStartup can often be an easy way to locate the start of network-related functionality.