APPENDIX

LIST OF MOST WINDOWS APIS USED BY MALWARE

Appendix B

eLearn Security
AN INE COMPANY

- **KEYSTROKES Loggers:**
- Keyboard keystrokes can be logged either by polling Keyboard state or by using API hook of keyboard related events, such as:
- GetAsyncKeyState() -> This function polls the state of keys on the keyboard.
- GetKeyState() -> API call (eg: check if the shift key is pressed)
- SetWindowsHookExA() -> Used to retrieve WH_KEYBOARD and WH_MOUSE movement.

- **Network Traffic Monitor APIs:**
- WSASocket() or socket() -> Used to initiate a raw socket
- bind() -> bind a particular socket to a NIC
- WSAIoctl() or ioctlsocket() -> Change the mode of a NIC into Promiscuous mode. (SIO_RECVALL – parameter tells OS to put the n/w card in promiscuous mode).

- **Downloader APIs:**
- A downloader could simply use these two APIs and that's all that is needed to download and execute a file.
- URLDownloadToFile() -> Download and save a file to disk
- ShellExecute(), WinExec(), and others -> Execute new created files

- **HTTP C&C Traffic APIs:**
- If C&C is implemented in malware, it will keep a constant contact (in HTTP) with the C2 server, due to its stateless nature. To initiate an HTTP session, the following APIs can be used:
- InternetOpen()
- Example:

- InternetOpen(USER_AGENT,INTERNET_OPEN_TYPE_PROXY,argv[2],0,0);]
- InternetConnect() – URL Input to build HTTP request, the following APIs are used:
- HttpOpenRequest(), HttpAddRequestHeaders()
- For sending HTTP requests: HTTPSendRequest()
- For reading response, which means that the malware may be reading the response: InternetReadFile().

- **Dropper APIs:**
- Droppers may embed files in its resources section, such as Payloads in the resources section '.rsrc'.
- To manage resources, these APIs are used:
    - FindResource
    - LoadResource
    - SizeOfResource
    - and LockResource

1. **DLL Injection FUCTION calls APIs:**

1.1. Using SetWindowsHookEx():

1.2. LoadLibraryA(): Used to load a malicious DLL into process's address space. One parameter is required, the name of the DLL, which will be loaded.

1.3. GetProcAddress(): Return the address of the filter function of the remote process.

1.4. GetWindowsThreadProcessId(): return the Target thread ID.

1.5. SetWindowsHookEx(): Used for GUI applications to install the filter function into the hook chain of the remote processWorks

1.6. Single call to inject into all applications.

1.7. BroadcastSystemMessage(): Used to transfer messages (internally) from a malicious process to a targeted process.

1.8. Others such as GetMessage() and DispatchMessage()

- **Using CreateRemoteThread():**
- OpenProcess(): enables the malicious process to talk to the targeted process.
- VirtualAllocEx(): enables a process to allocate memory space in another process, malwares use it to store the string name of the malicious DLL.
- WriteProcessMemory(): This enables a process to write into another process memory space. Due to this functionality, a victim process might be injected by a string that later may be loaded to LoadLibraryA() function calls.
- GetModuleHandle(): Helps a process to define the way to access dlls that are loaded into the memory space. Find the kernal32.dll (used to implement loadlibrary function)
- GetProcAddress(): Can be used to find the LoadLibrary() address within the kernel32.dll file.
- CreateRemoteThread(): Enables a malware to create a remote thread in a remote victim process. One of its arguments is the function address, which new thread will run.
- LoadLibraryA will run in the new created thread.
- LocaLibraryA Works on any application and used to load the malicious DLL to the remote address space.

- **API HOOKING: USER-SPACE FUNCTION:**
- The following steps enables to HOOK a user space function
- GetProcAddress: Determine the address of a function to hook
- VirtualProtect: change the memory protection into read or write.

- ReadProcessMemory: or manual copy: used to store the first bytes of a target function.
- Compute new instruction -> Usually it is a JUMP to the rootkit code
- Overwrite first few bytes of target function -> manual copy, WriteProcessMemory()
- VirtualProtect: Restore the original value of memory permission

- **Process Hollowing APIs:**
- CreateProcessA: Create a new process (in a suspended state).
- NtUnmapViewOfSection Remove the contents of a legitimate process from memory. Putting the process into a suspended state and removing its contents makes the process an empty shell.
- VirtualAllocEx: Assigns a new memory address into the hollow process
- WriteProcessMemory: Used to inject a new code in the hollow process
- ResumeThread: Used to resume the process flow.

- **Few AntiDebugger/VM Detection:**
- GetTickCount() identifies the time to detect the presence of a debugger.
- CountClipboardFormats() This API call is used to define if a process's clipboard is empty
- GetForeGroundWindow() API call that checks whether the foreground color of a window was changing, we assume here that automated sandbox tools do not switch active windows around.
- Isdebuggerpresent() → Detect debugger

- **General APIs – Part #1:**
- Fopen(): Open a file to read
- Sscanf(): Parse a string.
- ReadFile(): Locate which file will be read by inspecting the hFile handle in the Kernel32.dll.
- CreateFIleA(): Creates a new file and reads existing one.
- CreateToolhelp32Snapshot(): Retrieves a list of current running processes.
- Process32First(): Used to iterate the list retrieved from createToolhelp32Snapshort API call.

- **General APIs – Part #2:**
- FindWindowsA(): Locate a particular window using call or a title name. This API returns a handle on the top-level window. It searches for the window, using its name or class name, which matches the specified strings (Search is case insensitive).
- CreateThread(): Used to launch a thread that runs within the current process.
- GetEIP(): Used by SHELLCODE of methods often to locate where it will be loaded in memory.

- **General APIs – Part #3:**
- RegSetValueEx(): Disable the popup of User Account Control.
- GetFIleSize(): Generally SHELLCODEs use it to determine its own document by iterating through handle values to document files that are loaded.
- malloc(): Used to allocate memory on the heap.
- Free(): If the OS calls this function, the shell code will execute using the HEAP buffer exploit.
- GetTempPathA(): locates the temporary folder location to download the file.

- **DLL Injection – Classic Method**
- Below is a list of APIs to look for that are often used for classical DLL Injections:
- OpenProcess
- VirtualAllocEx
- WriteProcessMemory
- CreateRemoteThread

- **DLL Injection – SetWindowsHookEx Method**
- Below is a list of APIs to look for that are used for DLL Injections using the SetWindowsHookEx method:
- LoadLibrary or LoadLibraryEx
- GetProcAddress
- SetWindowsHookEx

- **DLL Injection – APC Method**
- Below is a list of APIs to look for that are used for DLL Injection using the APC method:
- OpenProcess and CreateTool32Snapshot
- Process32First and Thread32First
- Process32Next and Thread32Next
- VirtualAllocEx, VirtualFreeEx, and CloseHandle
- WriteProcessMemory
- QueueUserAPC/NtQueueApcThread

- **DLL Injection – ALPC Method**
- Below is a list of APIs to look for that are used for DLL Injection using the APC method:
- NtQuerySystemlnformation, NtQueryObject, and NtClose
- NtDuplicateObject or ZwDuplicateObject
- GetCurrentProcess, GetSystemlnfo, and RtllnitUnicodeString
- NtConnectPort, OpenProcess, CloseHandle,.
- VirtualAllocEx, WriteProcessMemory, CopyMemory,and ReadProcessMemory,
- VirtualFreeEx, VirtualQueryEx, and GetMappedFileName

- **Process Hollowing**
- Below is a list of APIs to look for that are used:
- CreateProcess("CREATE_SUSPENDED"),
- NtQueryProcesslnformation, ReadProcessMemory
- GetModuleHandle and GetProcAddress,
- ZwUnmapViewOfSection or NtUnmapViewOfSection
- VirtualAllocEx, WriteProcessMemory, and VirtualProtectEx
- SetThreadContext and ResumeThread

- **Process Doppelgänging Method**
- Below is a list of APIs to look for that are used:
- CreateFileTransacted, WriteFlle, and NtCreateSection
- RollbackTransaction and NtCreateProcessEx,
- RtlCreateProcessParametersEx, VirtualAllocEx,
- WriteProcessMemory, NtCreateThreadEx, and NtResumeThread

- Check URL for more info.

- **Reflective PE Injection Method**
- Below is a list of APIs to look for that are often used:
- OpenProcess and OpenProcessToken
- LoadRemoteLibraryR or LoadLibrary
- CreateFileA, VirtualAlloc, GetProcAddress
- HeapAlloc, HeapFree and CloseHandle

- **Thread Execution Hijacking Method**
- Below is a list of APIs to look for that are often used:
- OpenProcess and OpenProcessToken
- LoadRemoteLibraryR or LoadLibrary
- CreateFileA, VirtualAlloc, GetProcAddress
- HeapAlloc, HeapFree and CloseHandle

2. **Common APIs**
3. **Process & Threads**

Common APIs used for Process and Threads…

- **Process and Thread APIs**
- AttachThreadInput: This function is used to send the input processing of a thread to another thread. So, the second thread can receive input events such as keyboard events. Malware such as Keyloggers and spyware utilize this function.

- CreateThread: Creates a thread to run in the virtual address space of the calling process. The parameters include: the thread attributes, the initial stack size (in bytes), the thread starting address, a pointer to the variables that will be passed to the thread, and creation flags. Upon successful, this function returns a handle to the created thread (and optionally, the thread ID), otherwise it returns NULL.
- CreateRemoteThread: Allows to create a thread in the virtual address space of a remote/another process. Stealth malware and launchers use this function to inject code into processes. (more explanation).
- CreateRemoteThreadEx: If succeed, it will call NtCreateThreadEx in Ntdll.dll., which cause the usual transition into the kernel mode.

- **Process and Thread APIs**
- EnumProcesses: Retrieve the set of running processes in the system. Commonly used by malware to locate a target process for code injection.
- CreateToolhelp32Snapshot: Used to obtain a snapshot of running processes, threads, modules and heaps. Often used by malware within the code that is used to enumerate running processes and threads
- ControlService: used to modify, start, stop, or send a signal into a running service. In case the malware is using its own service, you need to analyze the code that implements the malicious service to identify the call purpose.
- CreateProcess: Used to create and starts a new process with the same access token of the creating process. You should analyze any new process that is created by a malware.

- **Process and Thread APIs**
- CreateProcessAsUser: The created process will run in the same security context of the user that is determined by the specified token.
- CreateProcessA: Creates a process and its main thread. The created process will run in the same security context of the creating process. If the calling process is impersonating another user, the new process uses the token for the calling process, not the impersonation token.

- **Process and Thread APIs**

- CreateProcessWithTokenW and CreateProcessWithLogonW: CreateProcessWithTokenW is the same as CreateProcessAsUser, but differs in the required caller privileges.

- CreateProcessWithLogonW is used to log on using the credentials of the given user and then create a process with the obtained token in one stroke.

- Both CreateProcessWithTokenW and CreateProcessWithLogonW are part of advapi32.Dll. Both functions makes a Remote Procedure Call (RPC) call to perform the actual process creation by calling a secondary logon service, the seclogon.dll, which is hosted in the SvcHost.exe.

- **Process and Thread APIs**

- ExitProcess: Terminates the process and all its threads. Does not force child processes to be end. Exiting a process does not imply removing the process object from the system, it is only removed if the last handle of this process is closed.

- CreateService: Used to create services to start at boot time. CreateService is used by malwares to load kernel drivers or to attain persistence or stealth.

- DeviceIoControl: This function passes control messages to device driver from the user space. Used by Kernel malware to send information from the user space to the kernel space.

- **Process and Thread APIs**

- EnumProcessModules: Used to list loaded modules (DLLs and executables) of a particular process. In process injection, malwares use this function to enumerates loaded modules.

- FindResource: Used to retrieve resources in loaded DLL or an executable. Resources may be used by malwares to store configuration information, strings, or their malicious files. If you find this function, then you should examine the .rsrc section in the PE header of the malware.

- GetModuleHandle: Used to get a handle to a loaded module. GetModuleHandle may be used by malwares to determine a suitable location for code injection or to find and alter code in a loaded module.

- **Process and Thread APIs**
- GetProcAddress: Used to obtain a particular function address in a loaded DLL into memory. Used by processes to import functions from different DLLs besides the ones that are imported in the file header of the PE.
- GetStartupInfo: Used to return a structure that contains information regarding how the current process was configured to run, such as where the standard handles are directed.
- GetThreadContext: Used to retrieve the context structure of a particular thread. The context stores information about the thread, such as the current state and register values.

- **Process and Thread APIs**
- IsNTAdmin: Used to check if the current user has privileges of administrator.
- IsWoW64Process: 32-bit processes used this function to check if it is running on a 64-bit operating system.
- LdrLoadDll: Same as LoadLibrary, used to load a DLL into a process. Usually LoadLibrary function is used. If the low-level function LdrLoadDll is used it indicates that the program that is trying to be stealthy.

- **Process and Thread APIs**
- NtQueryInformationProcess: Used to retrieve information about a particular process. Sometimes malware use it as an anti-debugging mechanism since it can retrieve the same information that CheckRemoteDebuggerPresent return.
- NtSetInformationProcess: Used to adjust the access level of a program, thus allows the program to bypass Data Execution Prevention (DEP).

- **Process and Thread APIs**
- OpenMutex: Used to open a handle to a mutual exclusion object, malware use this object to guarantee that only one instance of it is running on a system at any time. Often, malwares use fixed names for these mutexes. Hence mutex names can be used as host-based signatures.
- OpenProcess: Used to open a handle to another process running on the system. Malwares use this function to read or write to the memory of another process or to inject malicious code into another process.
- PeekNamedPipe: This function is used to copy data from a named pipe, this is common with reverse shells.

- **Process and Thread APIs**
- Process32First/Process32Next: Used to start processes enumeration from a prior call to CreateToolhelp32Snapshot function. Often used by malwares to enumerate running processes to determine a target process for code injection.
- QueueUserAPC: Used to execute another thread code. Sometimes used by malware to inject code into a target process.
- ReadProcessMemory: Used to retrieve the memory contents of a remote process.
- ResumeThread: Used to resume the running of a suspended thread. This function is used by different injection techniques.

- **Process and Thread APIs**
- SetThreadContext: Used to change a thread context. Some malwares use this function for code injection.
- SetWindowsHookEx: Used to set a hook function that is called if a particular event is called. Malwares such as spyware and keyloggers use SetWindowsHookEx. Also, it allows to load a DLL into all GUI processes on the system. Sometimes, this function is added by the compiler.
- SfcTerminateWatcherThread: Used to modify protected files by disabling Windows file protection.

- **Process and Thread APIs**
- ShellExecute: Used to execute other processes.
- StartServiceCtrlDispatcher: Services use this function to connect the process main thread to the service control manager. All processes that run as a service should call StartServiceCtrlDispatcher in the first 30 seconds of its startup. Finding StartServiceCtrlDispatcher in a malware function indicates that the function must run as a service.
- SuspendThread: Suspend a running thread. Sometimes, malwares suspend the target thread for code injection.

- **Process and Thread APIs**
- System: Used to start running programs provided by C runtime libraries. On Windows operating system, System function serves as a wrapper to CreateProcess function.
- Thread32First/Thread32Next: Used to enumerates the threads of a given process. Used by injectors to locate a victim thread for code injection.
- Toolhelp32ReadProcessMemory: Used to retrieve the memory contents of a remote process.

- **Process and Thread APIs**
- VirtualAllocEx: Used to allocate memory space in a remote process. Sometimes, used by malwares as a part of code injection.
- VirtualProtectEx: Used to modify the protection of specific region in the memory. This function may be used by malwares to update the state of memory region from read-only to executable.
- WideCharToMultiByte: Converts a Unicode string to an ASCII string.
- WinExec: Used to run another program.
- WriteProcessMemory: Used to write data into a remote process. Used by malwares as a part of code injection.

- **Process and Thread APIs**
- ExitThread: Calling this function, explicitly or after finishing a thread procedure, will deallocate the current thread's stack, all pending I/O that was created by the thread will be cancelled, and the thread is terminated. If the thread is the last thread in the process when this function is called, the thread's process is also terminated.
- Process.Start: Starts resource of process and connect it with the Process component.
- ShellExecuteA, ShellExecuteExW, ShellExecuteW, and ShellExecuteExA: All performs an operation on a particular file.
- LoadImageA*:Loads an icon, cursor, animated cursor, or bitmap.

4. **File System**

Common APIs used for File Systems…

- **Process and Thread APIs**
- This function is used to send the input processing of a thread to another thread. So, the second thread can receive input events such as keyboard events. Malware such as Keyloggers and spyware utilize this function.
- CreateThread: Creates a thread to run in the virtual address space of the calling process. The parameters include: the thread attributes, the initial stack size (in bytes), the thread starting address, a pointer to the variables that will be passed to the thread, and creation flags. Upon successful, this function returns a handle to the created thread (and optionally, the thread ID), otherwise it returns NULL.
- CreateRemoteThread: Allows to create a thread in the virtual address space of a remote/another process. Stealth malware and launchers use this function to inject code into processes. (more explanation).
- CreateRemoteThreadEx: If succeed, it will call NtCreateThreadEx in Ntdll.dll., which cause the usual transition into the kernel mode.

- **Process and Thread APIs**
- This function is used to send the input processing of a thread to another thread. So, the second thread can receive input events such as keyboard events. Malware such as Keyloggers and spyware utilize this function.
- CreateThread: Creates a thread to run in the virtual address space of the calling process. The parameters include: the thread attributes, the initial stack size (in bytes), the thread starting address, a pointer to the variables that will be passed to the thread, and creation flags. Upon successful, this function returns a handle to the created thread (and optionally, the thread ID), otherwise it returns NULL.
- CreateRemoteThread: Allows to create a thread in the virtual address space of a remote/another process. Stealth malware and launchers use this function to inject code into processes. (more explanation).
- CreateRemoteThreadEx: If succeed, it will call NtCreateThreadEx in Ntdll.dll., which cause the usual transition into the kernel mode.