# Win32 API

## Win32 DLLs

- DLLs provided by Windows
- Found in C:\Windows\System32
- Eg. kernel32.dll
- Try open in CFF Explorer

## Other DLLs provided by Windows

- Ntdll.dll
- Kernel32.dll
- Kernelbase.dll
- Gdi32.dll
- User32.dll
- Comctl32.dll
- Advapi32.dll
- Ws32_32.dll

## DLLs provided by Visual Studio SDK runtime

- Msvcrt.dll
- Msvbvm60.dll
- Vcruntimexx.dll (xx refers to version of the sdk)
- .Net Frameworks (C# and VB.net)

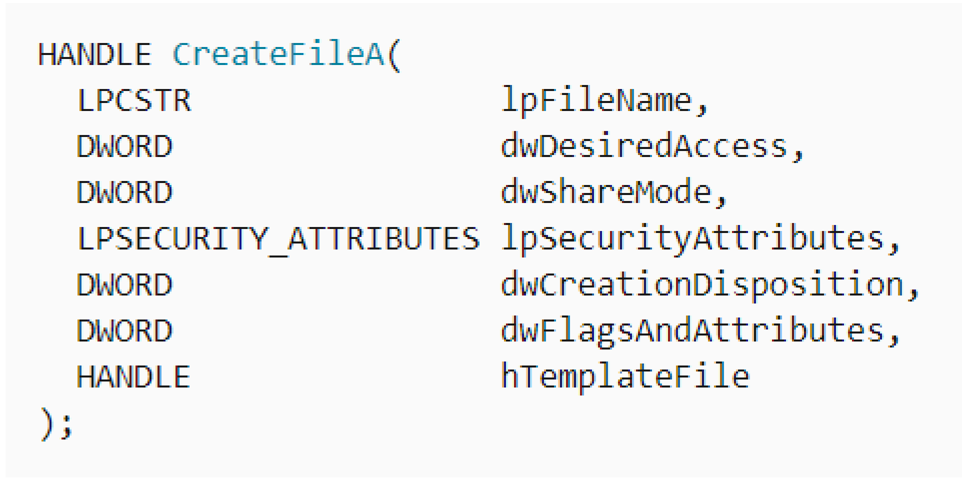
## Studying win32 API from MSDN docs

## Searching for win32 API docs

- Google for API and MSDN
- Try googling CreateFile MSDN
- Not just for creating files
- Can also read files
- Depends on the Parameters passed to the function

# API Parameters govern functionality

- CreateFileA() accepts 7 parameters

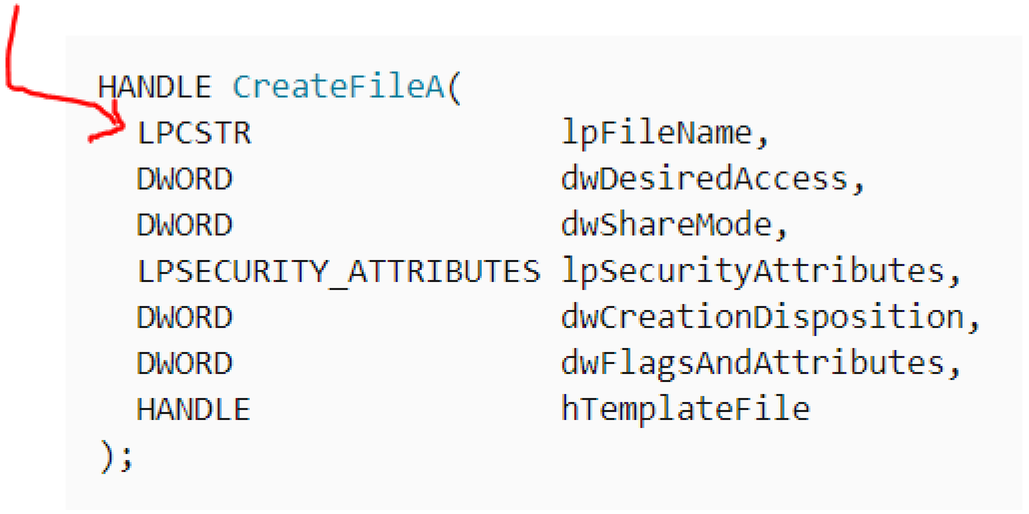


```
HANDLE CreateFileA(
  LPCSTR                lpFileName,
  DWORD                 dwDesiredAccess,
  DWORD                 dwShareMode,
  LPSECURITY_ATTRIBUTES lpSecurityAttributes,
  DWORD                 dwCreationDisposition,
  DWORD                 dwFlagsAndAttributes,
  HANDLE                hTemplateFile
);
```

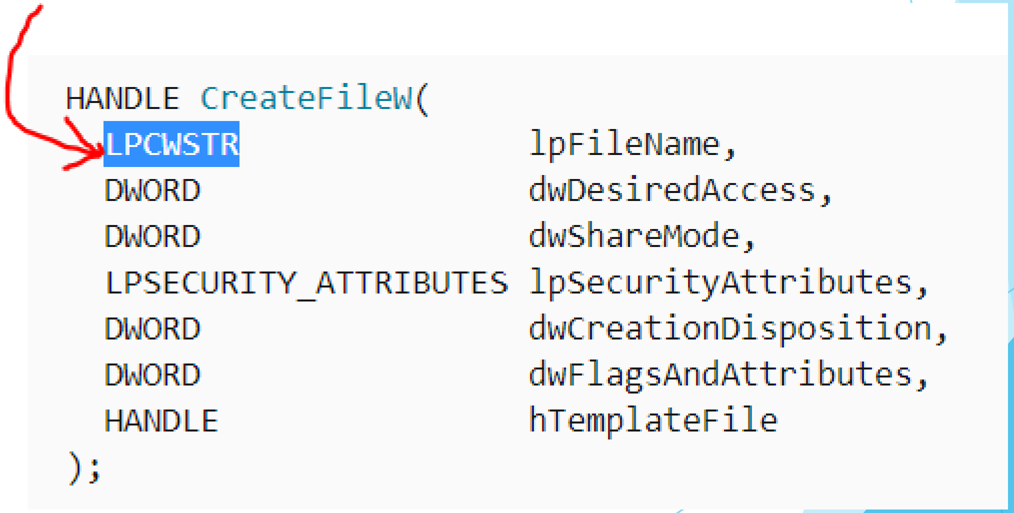
The dwCreationDisposition parameter decides if it is for creating file or, for reading a file

## ASCII and Unicode Versions of API

- CreateFileA accepts ASCII version of the string
- CreateFileW accepts Unicode
- Many other APIs also come in two versions just like this



```
HANDLE CreateFileA(
  LPCSTR                lpFileName,
  DWORD                 dwDesiredAccess,
  DWORD                 dwShareMode,
  LPSECURITY_ATTRIBUTES lpSecurityAttributes,
  DWORD                 dwCreationDisposition,
  DWORD                 dwFlagsAndAttributes,
  HANDLE                hTemplateFile
);
```



```
HANDLE CreateFileW(
  LPCWSTR               lpFileName,
  DWORD                 dwDesiredAccess,
  DWORD                 dwShareMode,
  LPSECURITY_ATTRIBUTES lpSecurityAttributes,
  DWORD                 dwCreationDisposition,
  DWORD                 dwFlagsAndAttributes,
  HANDLE                hTemplateFile
);
```

## Native (NT) Version of the APIs

- CreateFileA and CreateFileW are provided by kernel32.dll
- Another version is NTCreateFile which is provided by ntdll.dll
- It is much low-level because it is closer to the kernel
- Both CreateFileA and CreateFileB calls NTCreateFile internally
- Ntdll.dll then uses system calls (SYSCALLS) to execute the task
- SYSCALLS are kernel level functions
- Kernel Level functions is the heart of the Operating System
- User Level functions (APIs) make use of Kernel Level functions

## Extended Version of an API

- Some APIs has an extended version
- Eg, VirtualAllocEx is the extended version of VirtualAlloc
- They are used to allocate virtual memory
- VirtualAlloc allocates virtual memory for the current running process
- But VirtualAllocEx allocates virtual memory for other running processes
- Malware frequently makes use of them

## The Undocumented APIs

- NT APIs in ntdll.dll are not officially documented by Microsoft
- But hackers have reversed engineered it and put up unofficial docs
- Check out:

  http://undocumented.ntinternals.net/

NtCreateSection is an undocumented API commonly used by malware for a technique called Process Hollowing

## APIs that perform file operations

- CreateFile
- WriteFile
- ReadFile
- SetFilePointer
- DeleteFile
- CloseFile

## APIs that perform Registry operations

- RegCreateKey
- RegDeleteKey
- RegSetValue

## APIs for virtual memory

- VirtualAlloc
- VirtualProtect
- NtCreateSection
- WriteProcessMemory
- NtMapViewOfSection

## APIs on Processes and Threads

- CreateProcess
- ExitProcess
- CreateRemoteThread
- CreateThread
- GetThreadContext
- SetThreadContext
- TerminateProcess
- CreateProcessInternalW

## APIs on DLLs

- LoadLibrary
- GetProcAddress

## APIs on Windows Services

- OpenSCManager
- CreateService
- OpenService
- ChangeServiceConfig2W
- StartService

## APIs on Mutexes

- CreateMutex
- OpenMutex

## Behaviour Identification with APIs

## Behaviour Identifcation with APIs

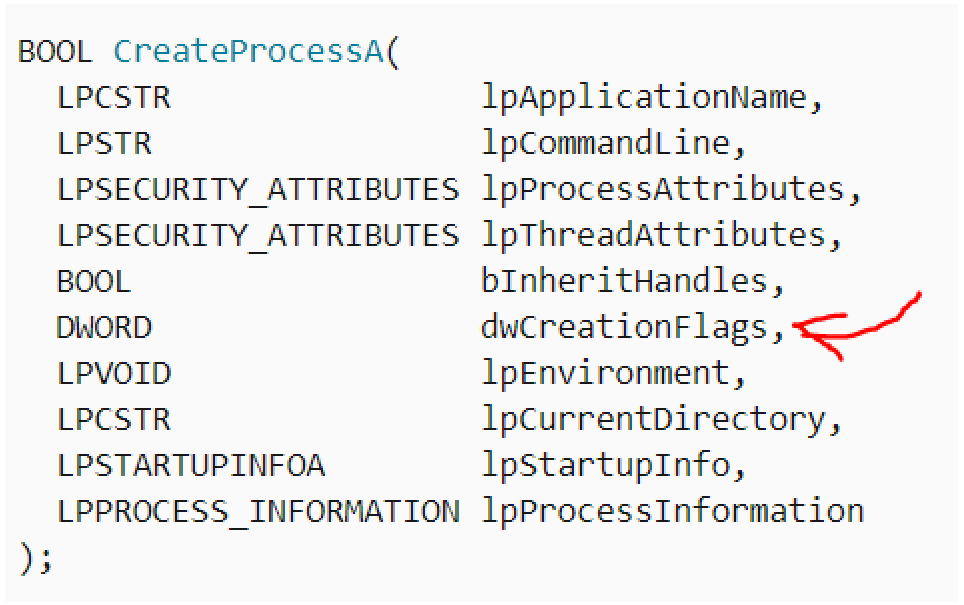
- Usage of APIs per se is not necessarily malware
- You need to analyze:

1. Context
2. Parameters supplied to APIs
3. Sets of APIs used in sequence

Take the case of Process Hollowing…

## Example 1: Process Hollowing

- It is a popular technique used by malware
- It uses CreateProcess API to create a brand-new process in suspended mode
- To do that, it sets dwCreationFlag = CREATE_SUSPENDED
- Normal programs do not do that



```
BOOL CreateProcessA(
  LPCSTR                lpApplicationName,
  LPSTR                 lpCommandLine,
  LPSECURITY_ATTRIBUTES lpProcessAttributes,
  LPSECURITY_ATTRIBUTES lpThreadAttributes,
  BOOL                  bInheritHandles,
  DWORD                 dwCreationFlags,
  LPVOID                lpEnvironment,
  LPCSTR                lpCurrentDirectory,
  LPSTARTUPINFOA        lpStartupInfo,
  LPPROCESS_INFORMATION lpProcessInformation
);
```

## Example 2: WriteProcessMemory

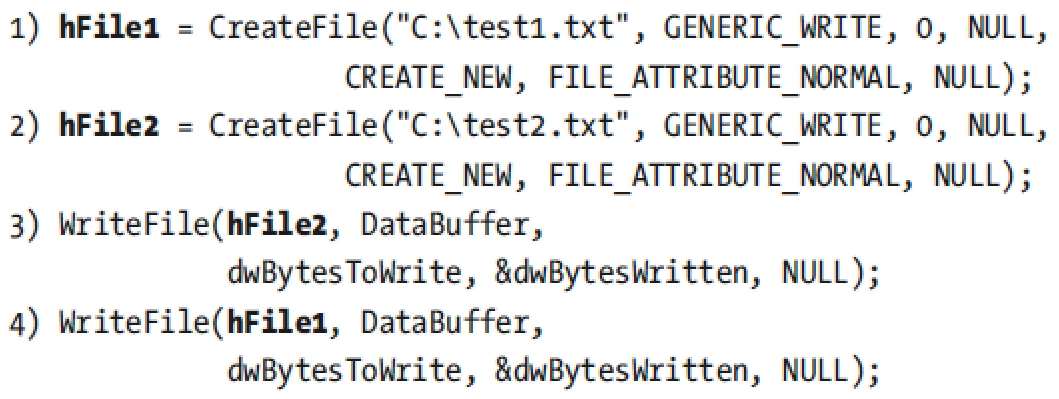
- It writes into the memory of another process
- Debuggers use this – so by itself it is not malicious
- But if a process also uses VirtualAllocEx and CreateRemoteThread then it is malware

So, the set of APIs used in sequence make it malicious

## Using Handle to Identify Sequences

- Handle is a reference to files, registry, memory and processes
- Processes makes use of handles to perform operations on the object it refers
- These handles are parameters passed to processes
- Tracking these handles help us identify sequence of APIs for any process
- These sequences help us confirm if a process is malware
- take case of CreateFile…

## Example of using handles: CreateFile



```
1) hFile1 = CreateFile("C:\test1.txt", GENERIC_WRITE, 0, NULL,
                    CREATE_NEW, FILE_ATTRIBUTE_NORMAL, NULL);
2) hFile2 = CreateFile("C:\test2.txt", GENERIC_WRITE, 0, NULL,
                    CREATE_NEW, FILE_ATTRIBUTE_NORMAL, NULL);
3) WriteFile(hFile2, DataBuffer,
            dwBytesToWrite, &dwBytesWritten, NULL);
4) WriteFile(hFile1, DataBuffer,
            dwBytesToWrite, &dwBytesWritten, NULL);
```

Can you identify the sequences? Tip: Trace the handles

Thank you