Native API

Adversaries may interact with the native OS application programming interface (API) to execute behaviors. Native APIs provide a controlled means of calling low-level OS services within the kernel, such as those involving hardware/devices, memory, and processes.[1][2] These native APIs are leveraged by the OS during system boot (when other system components are not yet initialized) as well as carrying out tasks and requests during routine operations.

Adversaries may abuse these OS API functions as a means of executing behaviors. Similar to Command and Scripting Interpreter, the native API and its hierarchy of interfaces provide mechanisms to interact with and utilize various components of a victimized system.

Native API functions (such as `NtCreateProcess`) may be directed invoked via system calls / syscalls, but these features are also often exposed to user-mode applications via interfaces and libraries.[3][4][5] For example, functions such as the Windows API `CreateProcess()` or GNU `fork()` will allow programs and scripts to start other processes.[6][7] This may allow API callers to execute a binary, run a CLI command, load modules, etc. as thousands of similar API functions exist for various system operations.[8][9][10]

Higher level software frameworks, such as Microsoft .NET and macOS Cocoa, are also available to interact with native APIs. These frameworks typically provide language wrappers/abstractions to API functionalities and are designed for ease-of-use/portability of code.[11][12][13][14]

Adversaries may use assembly to directly or in-directly invoke syscalls in an attempt to subvert defensive sensors and detection signatures such as user mode API-hooks.[15] Adversaries may also attempt to tamper with sensors and defensive tools associated with API monitoring, such as unhooking monitored functions via Disable or Modify Tools.

ID: T1106

Sub-techniques: No sub-techniques

ⓘ

Tactic: Execution

ⓘ

Platforms: Linux, Windows, macOS

Contributors: Gordon Long, Box, Inc., @ethicalhax; Stefan Kanthak; Tristan Madani (Cybereason)

Version: 2.2

Created: 31 May 2017

Last Modified: 12 September 2024

Version Permalink

Procedure Examples

ID	Name	Description
S0045	ADVSTORESHELL	ADVSTORESHELL is capable of starting a process using CreateProcess.[16]
S1129	Akira	Akira executes native Windows functions such as `GetFileAttributesW` and `GetSystemInfo`.[17]
S1025	Amadey	Amadey has used a variety of Windows API calls, including `GetComputerNameA`, `GetUserNameA`, and `CreateProcessA`.[18]
S0622	AppleSeed	AppleSeed has the ability to use multiple dynamically resolved API calls.[19]
G0067	APT37	APT37 leverages the Windows API calls: VirtualAlloc(), WriteProcessMemory(), and CreateRemoteThread() for process injection.[20]
G0082	APT38	APT38 has used the Windows API to execute code within a victim's system.[21]
S0456	Aria-body	Aria-body has the ability to launch files using `ShellExecute`.[22]
S1087	AsyncRAT	AsyncRAT has the ability to use OS APIs including `CheckRemoteDebuggerPresent`.[23]
S0438	Attor	Attor's dispatcher has used CreateProcessW API for execution.[24]
S0640	Avaddon	Avaddon has used the Windows Crypto API to generate an AES key.[25]
S1053	AvosLocker	AvosLocker has used a variety of Windows API calls, including `NtCurrentPeb` and `GetLogicalDrives`.[26]
S0638	Babuk	Babuk can use multiple Windows API calls for actions on compromised hosts including discovery and execution.[27][28][29]
S0475	BackConfig	BackConfig can leverage API functions such as `ShellExecuteA` and `HttpOpenRequestA` in the process of downloading and executing files.[30]
S0606	Bad Rabbit	Bad Rabbit has used various Windows API calls.[31]
S1081	BADHATCH	BADHATCH can utilize Native API functions such as, `ToolHelp32` and `RtlAdjustPrivilege` to enable `SeDebugPrivilege` on a compromised machine.[32]
S0128	BADNEWS	BADNEWS has a command to download an .exe and execute it via CreateProcess API. It can also run with ShellExecute.[33][34]
S0234	Bandook	Bandook has used the ShellExecuteW() function call.[35]
S0239	Bankshot	Bankshot creates processes using the Windows API calls: CreateProcessA() and CreateProcessAsUserA().[36]
S0534	Bazar	Bazar can use various APIs to allocate memory and facilitate code execution/injection.[37]
S0470	BBK	BBK has the ability to use the `CreatePipe` API to add a sub-process for execution via cmd.[38]
S0574	BendyBear	BendyBear can load and execute modules and Windows Application Programming (API) calls using standard shellcode API hashing.[39]
S0268	Bisonal	Bisonal has used the Windows API to communicate with the Service Control Manager to execute a thread.[40]
S0570	BitPaymer	BitPaymer has used dynamic API resolution to avoid identifiable strings within the binary, including `RegEnumKeyW`.[41]

ID	Name	Description
S1070	Black Basta	Black Basta has the ability to use native APIs for numerous functions including discovery and defense evasion.[42][43][44][45]
G0098	BlackTech	BlackTech has used built-in API functions.[46]
S0521	BloodHound	BloodHound can use .NET API calls in the SharpHound ingestor component to pull Active Directory data.[47]
S0651	BoxCaon	BoxCaon has used Windows API calls to obtain information about the compromised host.[48]
S1063	Brute Ratel C4	Brute Ratel C4 can call multiple Windows APIs for execution, to share memory, and defense evasion.[49][50]
S0471	build_downer	build_downer has the ability to use the `WinExec` API to execute malware on a compromised host.[38]
S1039	Bumblebee	Bumblebee can use multiple Native APIs.[51][52]
S0693	CaddyWiper	CaddyWiper has the ability to dynamically resolve and use APIs, including `SeTakeOwnershipPrivilege`.[53]
S0484	Carberp	Carberp has used the NtQueryDirectoryFile and ZwQueryDirectoryFile functions to hide files and directories.[54]
S0631	Chaes	Chaes used the `CreateFileW()` API function with read permissions to access downloaded payloads.[55]
G0114	Chimera	Chimera has used direct Windows system calls by leveraging Dumpert.[56]
S1149	CHIMNEYSWEEP	CHIMNEYSWEEP can use Windows APIs including `LoadLibrary` and `GetProcAddress`.[57]
S0667	Chrommme	Chrommme can use Windows API including `WinExec` for execution.[58]
S0611	Clop	Clop has used built-in API functions such as WNetOpenEnumW(), WNetEnumResourceW(), WNetCloseEnum(), GetProcAddress(), and VirtualAlloc().[59][60]
S0154	Cobalt Strike	Cobalt Strike's Beacon payload is capable of running shell commands without `cmd.exe` and PowerShell commands without `powershell.exe`[61][62][63]
S0126	ComRAT	ComRAT can load a PE file from memory or the file system and execute it with `CreateProcessW`.[64]
S0575	Conti	Conti has used API calls during execution.[65][66]
S0614	CostaBricks	CostaBricks has used a number of API calls, including `VirtualAlloc`, `VirtualFree`, `LoadLibraryA`, `GetProcAddress`, and `ExitProcess`.[67]
S0625	Cuba	Cuba has used several built-in API functions for discovery like GetIpNetTable and NetShareEnum.[68]
S0687	Cyclops Blink	Cyclops Blink can use various Linux API functions including those for execution and discovery.[69]
S1111	DarkGate	DarkGate uses the native Windows API `CallWindowProc()` to decode and launch encoded shellcode payloads during execution.[70] DarkGate can call kernel mode functions directly to hide the use of process hollowing methods during execution.[71]
S1066	DarkTortilla	DarkTortilla can use a variety of API calls for persistence and defense evasion.[72]
S1033	DCSrv	DCSrv has used various Windows API functions, including `DeviceIoControl`, as part of its encryption process.[73]
S1052	DEADEYE	DEADEYE can execute the `GetComputerNameA` and `GetComputerNameExA` WinAPI functions.[74]

ID	Name	Description
S0354	Denis	Denis used the `IsDebuggerPresent`, `OutputDebugString`, and `SetLastError` APIs to avoid debugging. Denis used `GetProcAddress` and `LoadLibrary` to dynamically resolve APIs. Denis also used the `Wow64SetThreadContext` API as part of a process hollowing process.[75]
S0659	Diavol	Diavol has used several API calls like `GetLogicalDriveStrings`, `SleepEx`, `SystemParametersInfoAPI`, `CryptEncrypt`, and others to execute parts of its attack.[76]
S0695	Donut	Donut code modules use various API functions to load and inject code.[77]
S0694	DRATzarus	DRATzarus can use various API calls to see if it is running in a sandbox.[78]
S0384	Dridex	Dridex has used the `OutputDebugStringW` function to avoid malware analysis as part of its anti-debugging technique.[79]
S0554	Egregor	Egregor has used the Windows API to make detection more difficult.[80]
S0367	Emotet	Emotet has used `CreateProcess` to create a new process to run its executable and `WNetEnumResourceW` to enumerate non-hidden shares.[81]
S0363	Empire	Empire contains a variety of enumeration modules that have an option to use API calls to carry out tasks.[82]
S0396	EvilBunny	EvilBunny has used various API calls as part of its checks to see if the malware is running in a sandbox.[83]
S0569	Explosive	Explosive has a function to call the OpenClipboard wrapper.[84]
S0512	FatDuke	FatDuke can call `ShellExecuteW` to open the default browser on the URL localhost.[85]
S0696	Flagpro	Flagpro can use Native API to enable obfuscation including `GetLastError` and `GetTickCount`.[86]
S0661	FoggyWeb	FoggyWeb's loader can use API functions to load the FoggyWeb backdoor into the same Application Domain within which the legitimate AD FS managed code is executed.[87]
S1044	FunnyDream	FunnyDream can use Native API for defense evasion, discovery, and collection.[88]
G0047	Gamaredon Group	Gamaredon Group malware has used `CreateProcess` to launch additional malicious components.[89]
S0666	Gelsemium	Gelsemium has the ability to use various Windows API functions to perform tasks.[58]
S0032	gh0st RAT	gh0st RAT has used the `InterlockedExchange`, `SeShutdownPrivilege`, and `ExitWindowsEx` Windows API functions.[90]
S0493	GoldenSpy	GoldenSpy can execute remote commands in the Windows command shell using the `WinExec()` API.[91]
S0477	Goopy	Goopy has the ability to enumerate the infected system's user name via `GetUserNameW`.[75]
G0078	Gorgon Group	Gorgon Group malware can leverage the Windows API call, CreateProcessA(), for execution.[92]
S0531	Grandoreiro	Grandoreiro can execute through the `WinExec` API.[93]
S0632	GrimAgent	GrimAgent can use Native API including `GetProcAddress` and `ShellExecuteW`.[94]
S0561	GuLoader	GuLoader can use a number of different APIs for discovery and execution.[95]
S0499	Hancitor	Hancitor has used `CallWindowProc` and `EnumResourceTypesA` to interpret and execute shellcode.[96]

ID	Name	Description
S0391	HAWKBALL	HAWKBALL has leveraged several Windows API calls to create processes, gather disk information, and detect debugger activity.[97]
S0697	HermeticWiper	HermeticWiper can call multiple Windows API functions used for privilege escalation, service execution, and to overwrite random bites of data.[98][99][100][101]
S0698	HermeticWizard	HermeticWizard can connect to remote shares using `WNetAddConnection2W`.[100]
G0126	Higaisa	Higaisa has called various native OS APIs.[102]
S0431	HotCroissant	HotCroissant can perform dynamic DLL importing and API lookups using `LoadLibrary` and `GetProcAddress` on obfuscated strings.[103]
S0398	HyperBro	HyperBro has the ability to run an application (`CreateProcessW`) or script/file (`ShellExecuteW`) via API.[104]
S0537	HyperStack	HyperStack can use Windows API's `ConnectNamedPipe` and `WNetAddConnection2` to detect incoming connections and connect to remote shares.[105]
S0483	IcedID	IcedID has called `ZwWriteVirtualMemory`, `ZwProtectVirtualMemory`, `ZwQueueApcThread`, and `NtResumeThread` to inject itself into a remote process.[106]
S1152	IMAPLoader	IMAPLoader imports native Windows APIs such as `GetConsoleWindow` and `ShowWindow`.[107]
S0434	Imminent Monitor	Imminent Monitor has leveraged CreateProcessW() call to execute the debugger.[108]
S1139	INC Ransomware	INC Ransomware can use the API `DeviceIoControl` to resize the allocated space for and cause the deletion of volume shadow copy snapshots.[109]
S0259	InnaputRAT	InnaputRAT uses the API call ShellExecuteW for execution.[110]
S0260	InvisiMole	InvisiMole can use winapiexec tool for indirect execution of `ShellExecuteW` and `CreateProcessA`.[111]
S1020	Kevin	Kevin can use the `ShowWindow` API to avoid detection.[112]
S0607	KillDisk	KillDisk has called the Windows API to retrieve the hard disk handle and shut down the machine.[113]
S0669	KOCTOPUS	KOCTOPUS can use the `LoadResource` and `CreateProcessW` APIs for execution.[114]
S0356	KONNI	KONNI has hardcoded API calls within its functions to use on the victim's machine.[115]
S1160	Latrodectus	Latrodectus has used multiple Windows API post exploitation including `GetAdaptersInfo`, `CreateToolhelp32Snapshot`, and `CreateProcessW`.[116][117]
G0032	Lazarus Group	Lazarus Group has used the Windows API `ObtainUserAgentString` to obtain the User-Agent from a compromised host to connect to a C2 server.[118] Lazarus Group has also used various, often lesser known, functions to perform various types of Discovery and Process Injection.[119][120]
S0395	LightNeuron	LightNeuron is capable of starting a process using CreateProcess.[121]
S0680	LitePower	LitePower can use various API calls.[122]
S0681	Lizar	Lizar has used various Windows API functions on a victim's machine.[123]
S0447	Lokibot	Lokibot has used LoadLibrary(), GetProcAddress() and CreateRemoteThread() API functions to execute its shellcode.[124]
S1016	MacMa	MacMa has used macOS API functions to perform tasks.[125][126]

ID	Name	Description
S1060	Mafalda	Mafalda can use a variety of API calls.[127]
S0652	MarkiRAT	MarkiRAT can run the ShellExecuteW API via the Windows Command Shell.[128]
S0449	Maze	Maze has used several Windows API functions throughout the encryption process including IsDebuggerPresent, TerminateProcess, Process32FirstW, among others.[129]
S0576	MegaCortex	After escalating privileges, MegaCortex calls `TerminateProcess()`, `CreateRemoteThread`, and other Win32 APIs.[130]
G0045	menuPass	menuPass has used native APIs including `GetModuleFileName`, `lstrcat`, `CreateFile`, and `ReadFile`.[131]
S1059	metaMain	metaMain can execute an operator-provided Windows command by leveraging functions such as `WinExec`, `WriteFile`, and `ReadFile`.[127][132]
S0455	Metamorfo	Metamorfo has used native WINAPI calls.[133][134]
S0688	Meteor	Meteor can use `WinAPI` to remove a victim machine from an Active Directory domain.[135]
S1015	Milan	Milan can use the API `DnsQuery_A` for DNS resolution.[112]
S0084	Mis-Type	Mis-Type has used Windows API calls, including `NetUserAdd` and `NetUserDel`.[136]
S0083	Misdat	Misdat has used Windows APIs, including `ExitWindowsEx` and `GetKeyboardType`.[136]
S1122	Mispadu	Mispadu has used a variety of Windows API calls, including ShellExecute and WriteProcessMemory.[137][138]
S0256	Mosquito	Mosquito leverages the CreateProcess() and LoadLibrary() calls to execute files with the .dll and .exe extensions.[139]
S0630	Nebulae	Nebulae has the ability to use `CreateProcess` to execute a process.[140]
S0457	Netwalker	Netwalker can use Windows API functions to inject the ransomware DLL.[141]
S0198	NETWIRE	NETWIRE can use Native API including `CreateProcess` `GetProcessById`, and `WriteProcessMemory`.[142]
S1090	NightClub	NightClub can use multiple native APIs including `GetKeyState`, `GetForegroundWindow`, `GetWindowThreadProcessId`, and `GetKeyboardLayout`.[143]
S1100	Ninja	The Ninja loader can call Windows APIs for discovery, process injection, and payload decryption.[144][145]
S0385	njRAT	njRAT has used the ShellExecute() function within a script.[146]
C0022	Operation Dream Job	During Operation Dream Job, Lazarus Group used Windows API `ObtainUserAgentString` to obtain the victim's User-Agent and used the value to connect to their C2 server.[118]
C0006	Operation Honeybee	During Operation Honeybee, the threat actors deployed malware that used API calls, including `CreateProcessAsUser`.[147]
C0013	Operation Sharpshooter	During Operation Sharpshooter, the first stage downloader resolved various Windows libraries and APIs, including `LoadLibraryA()`, `GetProcAddress()`, and `CreateProcessA()`.[148]
C0014	Operation Wocao	During Operation Wocao, threat actors used the `CreateProcessA` and `ShellExecute` API functions to launch commands after being injected into a selected process.[149]

ID	Name	Description
S1050	PcShare	PcShare has used a variety of Windows API functions.[88]
S1145	Pikabot	Pikabot uses native Windows APIs to determine if the process is being debugged and analyzed, such as `CheckRemoteDebuggerPresent`, `NtQueryInformationProcess`, `ProcessDebugPort`, and `ProcessDebugFlags`.[150] Other Pikabot variants populate a global list of Windows API addresses from the `NTDLL` and `KERNEL32` libraries, and references these items instead of calling the API items to obfuscate execution.[151]
S0517	Pillowmint	Pillowmint has used multiple native Windows APIs to execute and conduct process injections.[152]
S0501	PipeMon	PipeMon's first stage has been executed by a call to `CreateProcess` with the decryption password in an argument. PipeMon has used a call to `LoadLibrary` to load its installer.[153]
S0435	PLEAD	PLEAD can use `ShellExecute` to execute applications.[154]
S0013	PlugX	PlugX can use the Windows API functions `GetProcAddress`, `LoadLibrary`, and `CreateProcess` to execute another process.[155][156]
S0518	PolyglotDuke	PolyglotDuke can use `LoadLibraryW` and `CreateProcess` to load and execute code.[85]
S0453	Pony	Pony has used several Windows functions for various purposes.[157]
S1058	Prestige	Prestige has used the `Wow64DisableWow64FsRedirection()` and `Wow64RevertWow64FsRedirection()` functions to disable and restore file system redirection.[158]
S0147	Pteranodon	Pteranodon has used various API calls.[159]
S0650	QakBot	QakBot can use `GetProcAddress` to help delete malicious strings from memory.[160]
S1076	QUIETCANARY	QUIETCANARY can call `System.Net.HttpWebRequest` to identify the default proxy configured on the victim computer.[161]
S0629	RainyDay	The file collection tool used by RainyDay can utilize native API including `ReadDirectoryChangeW` for folder monitoring.[140]
S0458	Ramsay	Ramsay can use Windows API functions such as `WriteFile`, `CloseHandle`, and `GetCurrentHwProfile` during its collection and file storage operations. Ramsay can execute its embedded components via `CreateProcessA` and `ShellExecute`.[162]
S0662	RCSession	RCSession can use WinSock API for communication including `WSASend` and `WSARecv`.[163]
S0416	RDFSNIFFER	RDFSNIFFER has used several Win32 API functions to interact with the victim machine.[164]
S0496	REvil	REvil can use Native API for execution and to retrieve active services.[165][166]
S0448	Rising Sun	Rising Sun used dynamic API resolutions to various Windows APIs by leveraging `LoadLibrary()` and `GetProcAddress()`.[148]
S0240	ROKRAT	ROKRAT can use a variety of API calls to execute shellcode.[167]
S1078	RotaJakiro	When executing with non-root permissions, RotaJakiro uses the the `shmget` API to create shared memory between other known RotaJakiro processes. RotaJakiro also uses the `execvp` API to help its dead process "resurrect".[168]
S1073	Royal	Royal can use multiple APIs for discovery, communication, and execution.[169]
S0148	RTM	RTM can use the `FindNextUrlCacheEntryA` and `FindFirstUrlCacheEntryA` functions to search for specific strings within browser history.[170]

ID	Name	Description
S0446	Ryuk	Ryuk has used multiple native APIs including `ShellExecuteW` to run executables, `GetWindowsDirectoryW` to create folders, and `VirtualAlloc`, `WriteProcessMemory`, and `CreateRemoteThread` for process injection.[171]
S0085	S-Type	S-Type has used Windows APIs, including `GetKeyboardType`, `NetUserAdd`, and `NetUserDel`.[136]
S1018	Saint Bot	Saint Bot has used different API calls, including `GetProcAddress`, `VirtualAllocEx`, `WriteProcessMemory`, `CreateProcessA`, and `SetThreadContext`.[172][173]
S1099	Samurai	Samurai has the ability to call Windows APIs.[144]
G0034	Sandworm Team	Sandworm Team uses Prestige to disable and restore file system redirection by using the following functions: `Wow64DisableWow64FsRedirection()` and `Wow64RevertWow64FsRedirection()`.[158]
S1085	Sardonic	Sardonic has the ability to call Win32 API functions to determine if `powershell.exe` is running.[174]
S1089	SharpDisco	SharpDisco can leverage Native APIs through plugins including `GetLogicalDrives`.[143]
S0444	ShimRat	ShimRat has used Windows API functions to install the service and shim.[175]
S0445	ShimRatReporter	ShimRatReporter used several Windows API functions to gather information from the infected system.[175]
G1008	SideCopy	SideCopy has executed malware by calling the API function `CreateProcessW`.[176]
S0610	SideTwist	SideTwist can use `GetUserNameW`, `GetComputerNameW`, and `GetComputerNameExW` to gather information.[177]
G0091	Silence	Silence has leveraged the Windows API, including using CreateProcess() or ShellExecute(), to perform a variety of tasks.[178][179]
S0692	SILENTTRINITY	SILENTTRINITY has the ability to leverage API including `GetProcAddress` and `LoadLibrary`.[180]
S0623	Siloscape	Siloscape makes various native API calls.[181]
S0627	SodaMaster	SodaMaster can use `RegOpenKeyW` to access the Registry.[182]
S0615	SombRAT	SombRAT has the ability to respawn itself using `ShellExecuteW` and `CreateProcessW`.[67]
S1034	StrifeWater	StrifeWater can use a variety of APIs for execution.[183]
S0603	Stuxnet	Stuxnet uses the SetSecurityDescriptorDacl API to reduce object integrity levels.[184]
S0562	SUNSPOT	SUNSPOT used Windows API functions such as `MoveFileEx` and `NtQueryInformationProcess` as part of the SUNBURST injection process.[185]
S1064	SVCReady	SVCReady can use Windows API calls to gather information from an infected host.[186]
S0242	SynAck	SynAck parses the export tables of system DLLs to locate and call various Windows API functions.[187][188]
S0663	SysUpdate	SysUpdate can call the `GetNetworkParams` API as part of its C2 establishment process.[189]
G0092	TA505	TA505 has deployed payloads that use Windows API calls on a compromised host.[190]
S0011	Taidoor	Taidoor has the ability to use native APIs for execution including `GetProcessHeap`, `GetProcAddress`, and `LoadLibrary`.[191][192]

ID	Name	Description
S0595	ThiefQuest	ThiefQuest uses various API to perform behaviors such as executing payloads and performing local enumeration.[193]
S0668	TinyTurla	TinyTurla has used `WinHTTP`, `CreateProcess`, and other APIs for C2 communications and other functions.[194]
G1022	ToddyCat	ToddyCat has used `WinExec` to execute commands received from C2 on compromised hosts.[145]
S0678	Torisma	Torisma has used various Windows API calls.[195]
S0266	TrickBot	TrickBot uses the Windows API call, CreateProcessW(), to manage execution flow.[196] TrickBot has also used `Nt*` API functions to perform Process Injection.[197]
G0081	Tropic Trooper	Tropic Trooper has used multiple Windows APIs including HttpInitialize, HttpCreateHttpHandle, and HttpAddUrl.[198]
G0010	Turla	Turla and its RPC backdoors have used APIs calls for various tasks related to subverting AMSI and accessing then executing commands through RPC and/or named pipes.[199]
S0022	Uroburos	Uroburos can use native Windows APIs including `GetHostByName`.[200]
S0386	Ursnif	Ursnif has used `CreateProcessW` to create child processes.[201]
S0180	Volgmer	Volgmer executes payloads using the Windows API call CreateProcessW().[202]
S0670	WarzoneRAT	WarzoneRAT can use a variety of API calls on a compromised host.[203]
S0612	WastedLocker	WastedLocker's custom crypter, CryptOne, leveraged the VirtualAlloc() API function to help execute the payload.[204]
S0579	Waterbear	Waterbear can leverage API functions for execution.[205]
S0689	WhisperGate	WhisperGate has used the `ExitWindowsEx` to flush file buffers to disk and stop running processes and other API calls.[206][207]
S0466	WindTail	WindTail can invoke Apple APIs `contentsOfDirectoryAtPath`, `pathExtension`, and (string) `compare`.[208]
S0141	Winnti for Windows	Winnti for Windows can use Native API to create a new process and to start services.[209]
S1065	Woody RAT	Woody RAT can use multiple native APIs, including `WriteProcessMemory`, `CreateProcess`, and `CreateRemoteThread` for process injection.[210]
S0161	XAgentOSX	XAgentOSX contains the execFile function to execute a specified file on the system using the NSTask:launch method.[211]
S0653	xCaon	xCaon has leveraged native OS function calls to retrieve victim's network adapter's information using GetAdapterInfo() API.[48]
S1151	ZeroCleare	ZeroCleare can call the `GetSystemDirectoryW` API to locate the system directory.[57]
S0412	ZxShell	ZxShell can leverage native API including `RegisterServiceCtrlHandler` to register a service.RegisterServiceCtrlHandler
S1013	ZxxZ	ZxxZ has used API functions such as `Process32First`, `Process32Next`, and `ShellExecuteA`.[212]

Mitigations

ID	Mitigation	Description
M1040	Behavior Prevention on Endpoint	On Windows 10, enable Attack Surface Reduction (ASR) rules to prevent Office VBA macros from calling Win32 APIs. [213]
M1038	Execution Prevention	Identify and block potentially malicious software executed that may be executed through this technique by using application control [214] tools, like Windows Defender Application Control[215], AppLocker, [216] [217] or Software Restriction Policies [218] where appropriate. [219]

Detection

ID	Data Source	Data Component	Detects
DS0011	Module	Module Load	Monitor DLL/PE file events, specifically creation of these binary files as well as the loading of DLLs into processes. Utilization of the Windows APIs may involve processes loading/accessing system DLLs associated with providing called functions (ex: ntdll.dll, kernel32.dll, advapi32.dll, user32.dll, and gdi32.dll). Monitoring for DLL loads, especially to abnormal/unusual or potentially malicious processes, may indicate abuse of the Windows API. Though noisy, this data can be combined with other indicators to identify adversary activity. Analytic 1 - Look for unusual or abnormal DLL loads, processes loading DLLs not typically associated with them `sourcetype=Sysmon EventCode=7| stats count by module_name process_name user| where module_name IN ("ntdll.dll", "kernel32.dll", "advapi32.dll", "user32.dll", "gdi32.dll")`
DS0009	Process	OS API Execution	Monitoring API calls may generate a significant amount of data and may not be useful for defense unless collected under specific circumstances, since benign use of API functions are common and may be difficult to distinguish from malicious behavior. Correlation of other events with behavior surrounding API function calls using API monitoring will provide additional context to an event that may assist in determining if it is due to malicious behavior. Correlation of activity by process lineage by process ID may be sufficient.