AlSch092 @ Github **For: MITRE ATT&CK** **Dec. 28 2023**

Technique Name: Modify Exported Function Names within Running Processes

Tactic: TA0002 (Execution)

Platform: Windows

Required Permissions: User

Techniques: This is a sub-technique of T1129

Data Sources: Windows API, Export Directory Table

Description:

The names of exported functions can be modified by a malicious actor at runtime to avoid detection mechanisms and alter the program's execution flow. This can be done by using the Windows API `ImageDirectoryEntryToData` with the flag `IMAGE_DIRECTORY_ENTRY_EXPORT` to get an image export directory, and then using `ImageRvaToVa` to get the address of the export function's string value. Typically the name of each exported function is located at some offset of its module. Any process can perform this technique on itself or other running processes as long as the addresses where export names reside at is writable (calling `VirtualProtect/Ex` might be needed to ensure memory is writable). As a result of applying this technique, calls to the Windows API `GetProcAddress` will return abnormal results.

Program flow can be changed if the name of a different exported function is written over the original. For example: if 'USER32.dll' exports `MessageBoxA`, we can replace this string value with `MessageBoxW` - then if a program uses `GetProcAddress` to look up the address of `MessageBoxA`, they will instead be given 0 as the address. If they looked up `MessageBoxW`, they would be given the address of `MessageBoxA`. An attacker can also cause an availability issue for these exported functions by changing their names to random strings, since `GetProcAddress` will then return 0.

This concept also works with exported user-defined functions. If a programmer makes 'exampleProgram.exe' which contains the exported functions `ExportFuncA` and `ExportFuncB`, an attacker could swap the two string values to make sure any calls to `GetProcAddress` return incorrect (swapped) results.

Detection:

DS0011 (Module, Module Load): Hashes of a module's exported function names can be gathered at program startup and later compared to.

Mitigation:

M1045 (Code Signing): Ensure that memory is not writable in the module, either through proper permissions or by re-mapping the module's sections with undocumented flags (SEC_NO_CHANGE).

M1044 (Restrict Library Loading): Ensure an attacker cannot unload and then re-load a module with modified export names

Adversary Use:

No examples could be found as this is a newly discovered technique. Further data must be collected to determine if any past malware samples have used this technique.

Additional References:

Here is a reference from the researcher who discovered this technique, which includes a description and code examples: (https://github.com/alsch092/ModifyExports)