

INTRODUCTION :

The primary goal of malware authors is to ensure the successful execution of their malicious code within the victim's environment. However, Windows operating systems have implemented various security measures, such as Data Execution Prevention (DEP), to prevent the execution of untrusted code. To bypass these security measures, attackers aim to place their malicious code within the executable memory region.

DLLs, which are executable files containing functionalities accessible through API calls, are always present in the executable memory region. Attackers exploit this by leveraging search order DLL hijacking, where the operating system itself loads the malicious code into these executable regions. This technique allows attackers to take advantage of the system's trust in DLLs to execute their malicious code undetected.

SEARCH ORDER DLL HIJACKING :

When an executable is run, it looks for the necessary DLLs listed in its import table. If a DLL is not already loaded in memory, the executable starts searching for it in the file system. This search is done by scanning through a predefined list of directories. If the required DLL is not found in the first directory, the search continues to the next directory on the list.

Attackers can take advantage of this DLL search order behavior by replacing a legitimate DLL with a malicious one that has the same name. By placing their malicious DLL in one of these directories, they can ensure that their code is executed instead of the legitimate DLL when processes make API calls. To avoid suspicion, attackers often retain the original DLL's API functionalities while incorporating additional malicious code.

This technique is known as Search Order DLL hijacking, and it is a popular method for achieving persistence. Since the victim is unlikely to notice the difference, the attacker's code continues to run undetected, allowing them to maintain control over the compromised system.

KnownDLLs:

KnownDLLs in Windows refer to a set of system DLLs (Dynamic Link Libraries) that are loaded into memory during the system startup process. These DLLs are commonly used by many applications and provide essential functionality to the operating system. These DLLs are listed inside the KnownDLLs registry key

'HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet \Control\Session Manager\KnownDLLs'

.

Search Order DLL Hijacking

Phase 02 A

Week 01

The actual DLL files associated with KnownDLLs can be found in the system's DLL cache, primarily located in the 'C:\Windows\System32' directory. This directory is protected and stores copies of the essential DLL files. The DLL cache ensures the availability and integrity of these system DLLs.

When an application attempts to load a DLL, Windows checks the DLL cache first to see if a copy of the DLL is available. If the DLL is not present in the KnownDLL registry path, then OS will follow a specific search order to find the DLL.

Search order :

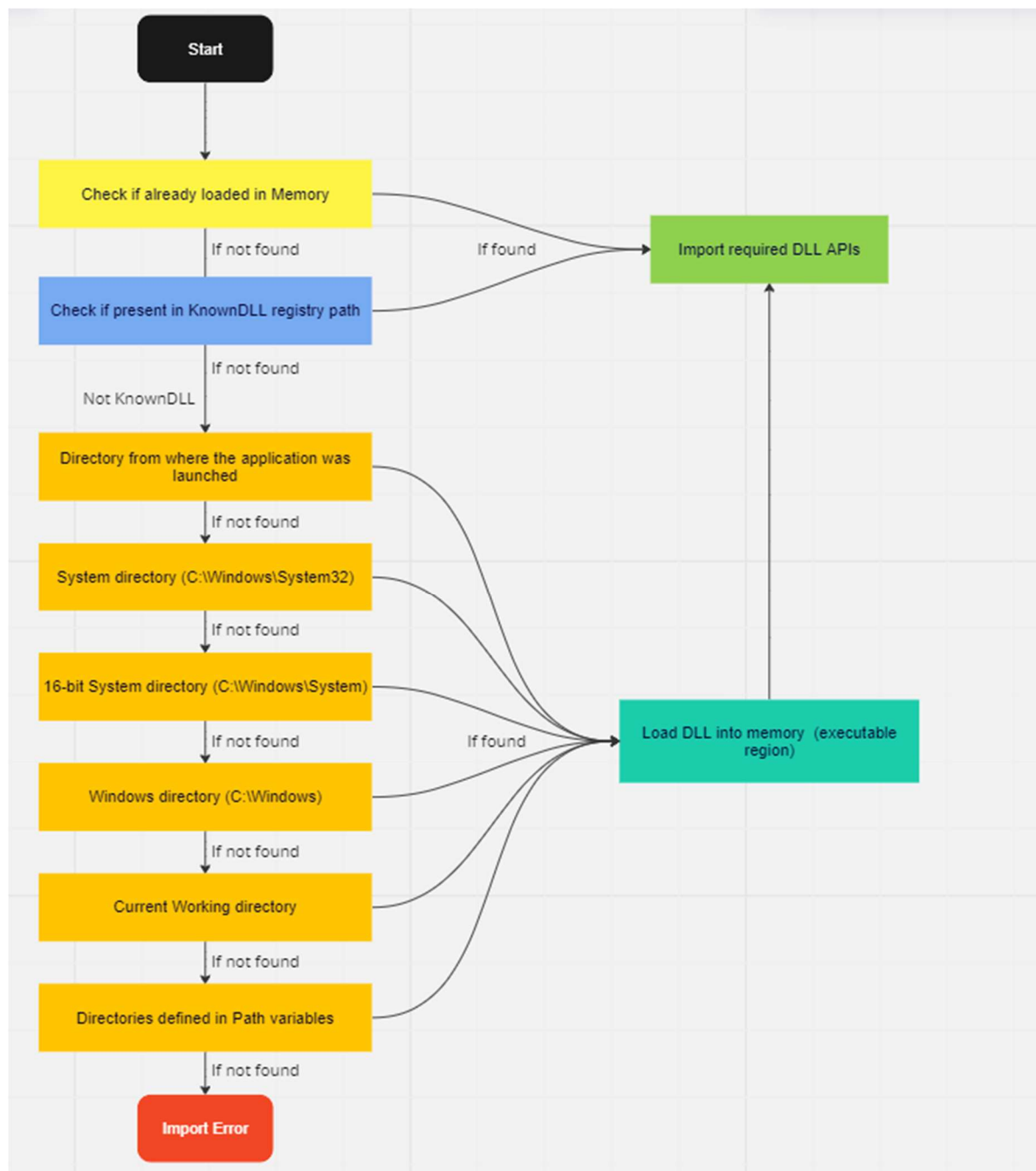
When an application needs to import a DLL, the operating system follows a specific process to locate the DLL. First, it checks if the DLL is already loaded into memory. If not, it proceeds to determine if the required DLL is a KnownDLL by referring to the KnownDLL registry path mentioned earlier.

If the DLL is not found in the KnownDLL registry path, the operating system initiates a search process based on a predetermined order. This search order is depicted in the flow chart below:

Search Order DLL Hijacking

Phase 02 A

Week 01



During the process of searching for a DLL, if the operating system locates the DLL in any of the specified directories, it stops searching and proceeds to load the DLL from secondary memory. However, this predictable search order can be exploited by attackers who are aware of this behavior.

An attacker can replace a legitimate DLL with a malicious DLL of the same name in one of the directories, preferably in the initial directories of the search order. As a result, when a process calls a DLL API from the DLL (let's say DLL A), the malicious DLL A is loaded instead of the original DLL. Consequently, the process unknowingly executes the imported DLL API

Search Order DLL Hijacking

Phase 02 A

Week 01

from the malicious DLL. By exploiting this vulnerability, attackers can execute their malicious code within the context of a legitimate process, allowing them to evade detection and persist on the system

ANALYSIS OF A MALWARE SAMPLE :

Since this was an easy and one of the prominent persistence mechanisms employed by attackers, we analyzed a malware sample that uses this mechanism.

Here, we will be working with 2 DLL samples named '**HelloDLL.dll**' (same name for both the DLLs), one non-malicious DLL which represents the original DLL that is provided by Windows or applications for their use, and the other malicious DLL which represents the malicious DLL injected by the attacker in one of the search directories.

- **Non-malicious DLL functionality:** A simple C program that exports a 'Hello()' function (as DLL API). This 'Hello()' function just prints out the "Hello World" string into cmd.
- **Malicious DLL functionality:** A C code that also exports the 'Hello()' function which opens up a TCP port and connects to a machine in the same local network listening as the server. Once connected, it sends a string "Hello World" to the server. However, in the real world, this server could be an attacker while the malicious DLL could elevate privilege or create a backdoor to the attacker or perform any other malicious activity.

We also wrote a sample C code which represents a normal user-written program ('main.c') that uses the 'Hello()' function or API present in 'HelloDLL.dll'. This code will load the 'HelloDLL.dll' DLL and get the address of the 'Hello()' function and execute it.

To verify the search order, we first compiled the user-written program ('main.c'). And execute the C executable without generating the 'HelloDLL.dll'. We see the following in the Procmon:

main.exe	8724	QueryNameInformationFile	C:\Windows\SysWOW64\KernelBase.dll	SUCCESS
main.exe	8724	QueryNameInformationFile	C:\Users\hrishi\Documents\Phase02\week1\D2\main.exe	SUCCESS
main.exe	8724	CreateFile	C:\Users\hrishi\Documents\Phase02\week1\D2\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Windows\SysWOW64\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Windows\System\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Windows\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Users\hrishi\Documents\Phase02\week1\D2\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Python39\Scripts\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Python39\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Program Files\Eclipse Adoptium\jdk-11.0.18.10-hotspot\bin\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Program Files (x86)\Common Files\Oracle\Java\javapath_target_303828\HelloDLL.dll	REPARSE
main.exe	8724	CreateFile	C:\Program Files (x86)\Common Files\Oracle\Java\javapath_target_303828\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Windows\SysWOW64\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\ProgramData\Boxstarter\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Windows\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Windows\SysWOW64\wbem\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Windows\SysWOW64\WindowsPowerShell\v1.0\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Windows\SysWOW64\OpenSSH\HelloDLL.dll	PATH NOT FOUND
main.exe	8724	CreateFile	C:\Program Files (x86)\AOEME\AOEME Backupper\7.1.2\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\ProgramData\chocolatey\bin\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Program Files\10 Editor\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Program Files\OpenJDK\jdk-19.0.2\bin\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Program Files (x86)\dotnet\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\MinGW\bin\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Program Files\OpenSSL-Win64\bin\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Program Files (x86)\Windows Kits\10\Windows Performance Toolkit\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Users\hrishi\AppData\Local\Microsoft\WindowsApps\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Tools\Cmder\HelloDLL.dll	NAME NOT FOUND
main.exe	8724	CreateFile	C:\Windows\Microsoft.NET\Framework\v4.0.30319\HelloDLL.dll	NAME NOT FOUND

Search Order DLL Hijacking

Phase 02 A

Week 01

We see the search order starts with the Directory where the 'main.exe' is present. And continues the search in the order mentioned before.

To generate DLLs in Windows, the following command must be used

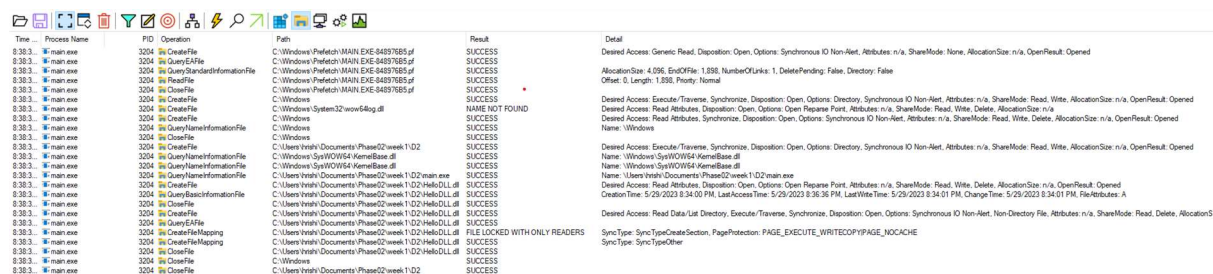
```
gcc -shared Non_malicious_dll.c -o HelloDLL.dll
```

The following steps were followed to employ the search order DLL hijacking technique:

1. Placing the non-malicious DLL in the executable's directory to verify its functionality (Pre hijacking):

For simplicity reasons, initially, we will be placing the DLL in the executable's ('main.c') directory. So the executable will check if the HelloDLL.dll is a KnownDLL or not.

Since it's not a KnownDLL, it then searches for DLL in the executable's path (the directory where the application was launched) where it finds 'HelloDLL.dll'. So it loads the DLL and calls the 'Hello()' function. The above observation can be visualized through the Procmon screenshot below:



The screenshot shows a Procmon session with 'main.exe' as the process. The 'File' filter is applied. The table below summarizes the key file operations observed:

Time	Process Name	PID	Operation	Path	Result
8:38.3	main.exe	3204	CreateFile	C:\Windows\Prefetch\MAIN.EXE-84897685.pf	SUCCESS
8:38.3	main.exe	3204	QueryFile	C:\Windows\Prefetch\MAIN.EXE-84897685.pf	SUCCESS
8:38.3	main.exe	3204	QueryStandardInformationFile	C:\Windows\Prefetch\MAIN.EXE-84897685.pf	SUCCESS
8:38.3	main.exe	3204	ReadFile	C:\Windows\Prefetch\MAIN.EXE-84897685.pf	SUCCESS
8:38.3	main.exe	3204	CreateFile	C:\Windows\Prefetch\MAIN.EXE-84897685.pf	SUCCESS
8:38.3	main.exe	3204	CreateFile	C:\Windows\System32\wow64\log.dll	NAME NOT FOUND
8:38.3	main.exe	3204	QueryNameInformationFile	C:\Windows	SUCCESS
8:38.3	main.exe	3204	CreateFile	C:\Windows	SUCCESS
8:38.3	main.exe	3204	QueryNameInformationFile	C:\Users\Insh\Documents\Phase02\week1\02	SUCCESS
8:38.3	main.exe	3204	QueryNameInformationFile	C:\Windows\System32\Wow64\KernelBase.dll	SUCCESS
8:38.3	main.exe	3204	QueryNameInformationFile	C:\Windows\System32\Wow64\KernelBase.dll	SUCCESS
8:38.3	main.exe	3204	QueryNameInformationFile	C:\Users\Insh\Documents\Phase02\week1\02\main.exe	SUCCESS
8:38.3	main.exe	3204	CreateFile	C:\Users\Insh\Documents\Phase02\week1\02\HelloDLL.dll	SUCCESS
8:38.3	main.exe	3204	QueryBasicInformationFile	C:\Users\Insh\Documents\Phase02\week1\02\HelloDLL.dll	SUCCESS
8:38.3	main.exe	3204	CreateFile	C:\Users\Insh\Documents\Phase02\week1\02\HelloDLL.dll	SUCCESS
8:38.3	main.exe	3204	QueryFile	C:\Users\Insh\Documents\Phase02\week1\02\HelloDLL.dll	SUCCESS
8:38.3	main.exe	3204	QueryFile	C:\Users\Insh\Documents\Phase02\week1\02\HelloDLL.dll	SUCCESS
8:38.3	main.exe	3204	CreateFileMapping	C:\Users\Insh\Documents\Phase02\week1\02\HelloDLL.dll	SUCCESS
8:38.3	main.exe	3204	CreateFileMapping	C:\Users\Insh\Documents\Phase02\week1\02\HelloDLL.dll	SUCCESS
8:38.3	main.exe	3204	CreateFile	C:\Users\Insh\Documents\Phase02\week1\02\HelloDLL.dll	SUCCESS
8:38.3	main.exe	3204	CreateFile	C:\Windows	SUCCESS
8:38.3	main.exe	3204	CreateFile	C:\Users\Insh\Documents\Phase02\week1\02	SUCCESS

We see that OS finds the DLL in the "main.exe's" directory. It should be noted that after finding the required DLL, OS no longer searches the next directories present in the search order.

2. Placing malicious DLL in the executable's directory to verify its functionality:

After placing, when executed, we see that the malicious 'Hello()' function from malicious DLL has been executed. Hence we can confirm that the malicious DLL has replaced the non-malicious (legitimate) DLL in the executable's directory. The same was verified through the Procmon screenshot which looks the same as the previous Procmon screenshot.

3. Placing non-malicious DLL in the 'C:\Windows' directory and malicious in the DLL executable's directory:

Many non-KnownDLLs are placed in the 'C:\Windows' directory. This is a vulnerable place since it doesn't need administrative privileges to modify this directory. So, we place the non-

Search Order DLL Hijacking

Phase 02 A

Week 01

malicious or legitimate 'HelloDLL.dll' in the 'C:\Windows'. Now the attacker has 2 main choices:

- To place malicious DLL in the directory that arrives before the Windows directory in the search order.
- To replace this legitimate DLL present in the Windows directory

Choice A:

Here placing a DLL in the same directory as that of the application (executable in this case) will reduce the chance of the victim noticing this DLL since in reality, a victim doesn't often check these directories, and also generally these application directories contain a lot of other DLLs useful for the application and executables. Hence the placed malicious DLL can camouflage itself.

Upon executing, we see that the malicious DLL has been loaded onto the memory. Indicating, that OS found malicious DLL before non-malicious DLL. The below Procmon screenshot verifies the same:

Time...	Process Name	PID	Operation	Path	Result	Detail
9:16.5	main.exe	2148	CreateFile	C:\Windows	SUCCESS	Desired Access: Execute/Traverse. Synchronize. Disposition: Open. Options: Directory, Synchronous IO Non-Alert. Attributes: n/a. ShareMode: Read. Write. AllocationSize: n/a. OpenResult: Opened
9:16.5	main.exe	2148	CreateFile	C:\Windows\System32\wow64api.dll	NAME NOT FOUND	Desired Access: Read Attributes. Disposition: Open. Options: Open Reparse Point. Attributes: n/a. ShareMode: Read. Write. Delete. AllocationSize: n/a
9:16.5	main.exe	2148	CreateFile	C:\Windows	SUCCESS	Desired Access: Read Attributes. Synchronize. Disposition: Open. Options: Synchronous IO Non-Alert. Attributes: n/a. ShareMode: Read. Write. Delete. AllocationSize: n/a. OpenResult: Opened
9:16.5	main.exe	2148	QueryNameInformationFile	C:\Windows	SUCCESS	Name: \Windows
9:16.5	main.exe	2148	CreateFile	C:\Users\Irish\Documents\Phase02\week1\02	SUCCESS	Desired Access: Execute/Traverse. Synchronize. Disposition: Open. Options: Directory, Synchronous IO Non-Alert. Attributes: n/a. ShareMode: Read. Write. AllocationSize: n/a. OpenResult: Opened
9:16.5	main.exe	2148	QueryNameInformationFile	C:\Windows\System32\wow64api.dll	SUCCESS	Name: \Windows\System32\wow64api.dll
9:16.5	main.exe	2148	CreateFile	C:\Windows\System32\wow64api.dll	SUCCESS	Name: \Windows\System32\wow64api.dll
9:16.5	main.exe	2148	QueryNameInformationFile	C:\Users\Irish\Documents\Phase02\week1\02\main.exe	SUCCESS	Name: \Users\Irish\Documents\Phase02\week1\02\main.exe
9:16.5	main.exe	2148	CreateFile	C:\Users\Irish\Documents\Phase02\week1\02\HelloDLL.dll	SUCCESS	Desired Access: Read Attributes. Disposition: Open. Options: Open Reparse Point. Attributes: n/a. ShareMode: Read. Write. Delete. AllocationSize: n/a. OpenResult: Opened
9:16.5	main.exe	2148	QueryBasicInformationFile	C:\Users\Irish\Documents\Phase02\week1\02\HelloDLL.dll	SUCCESS	CreationTime: 5/29/2023 9:08:32 PM. LastAccessTime: 5/29/2023 9:16:18 PM. LastWriteTime: 5/29/2023 9:16:18 PM. ChangeTime: 5/29/2023 9:16:18 PM. FileAttributes: A
9:16.5	main.exe	2148	CreateFile	C:\Users\Irish\Documents\Phase02\week1\02\HelloDLL.dll	SUCCESS	Desired Access: Read Data/List Directory. Execute/Traverse. Synchronize. Disposition: Open. Options: Synchronous IO Non-Alert. Non-Directory File. Attributes: n/a. ShareMode: Read. Delete. AllocationSize: n/a. OpenResult: Opened
9:16.5	main.exe	2148	CreateFileMapping	C:\Users\Irish\Documents\Phase02\week1\02\HelloDLL.dll	FILE LOCKED WITH ONLY READERS	SyncType: SyncTypeCreateSection. PageProtection: PAGE_EXECUTE_WRITECOPY/PAGE_NOCACHE
9:16.5	main.exe	2148	CreateFileMapping	C:\Users\Irish\Documents\Phase02\week1\02\HelloDLL.dll	SUCCESS	SyncType: SyncTypeOther
9:16.5	main.exe	2148	CreateFile	C:\Windows\System32\wow64api.dll	SUCCESS	Name: \Windows\System32\wow64api.dll
9:16.5	main.exe	2148	QueryNameInformationFile	C:\Windows\System32\wow64api.dll	SUCCESS	Desired Access: Read Attributes. Disposition: Open. Options: Open Reparse Point. Attributes: n/a. ShareMode: Read. Write. Delete. AllocationSize: n/a. OpenResult: Opened
9:16.5	main.exe	2148	CreateFile	C:\Windows\System32\wow64api.dll	SUCCESS	CreationTime: 9/7/2022 8:09:36 PM. LastAccessTime: 5/29/2023 9:09:59 PM. LastWriteTime: 9/7/2022 8:09:36 PM. ChangeTime: 5/28/2023 6:30:31 AM. FileAttributes: A
9:16.5	main.exe	2148	CreateFile	C:\Windows\System32\wow64api.dll	SUCCESS	Desired Access: Read Data/List Directory. Execute/Traverse. Synchronize. Disposition: Open. Options: Synchronous IO Non-Alert. Non-Directory File. Attributes: n/a. ShareMode: Read. Delete. AllocationSize: n/a. OpenResult: Opened
9:16.5	main.exe	2148	CreateFileMapping	C:\Windows\System32\wow64api.dll	FILE LOCKED WITH ONLY READERS	SyncType: SyncTypeCreateSection. PageProtection: PAGE_EXECUTE_WRITECOPY/PAGE_NOCACHE
9:16.5	main.exe	2148	CreateFileMapping	C:\Windows\System32\wow64api.dll	SUCCESS	SyncType: SyncTypeOther
9:16.5	main.exe	2148	CreateFile	C:\Windows\System32\wow64api.dll	SUCCESS	Offset: 168,960. Length: 32,768. I/O Flags: Non-cached. Paging I/O. Synchronous Paging I/O. Priority: Normal
9:16.5	main.exe	2148	ReadFile	C:\Windows\System32\wow64api.dll	SUCCESS	
9:16.5	main.exe	2148	CloseFile	C:\Users\Irish\Documents\Phase02\week1\02	SUCCESS	

We see the first directory searched is the one in which the 'main.exe' executable is present. Once found, OS terminates its search and loads the found DLL. So, from the attacker's perspective, 'HelloDLL.dll' has been hijacked.

Choice B:

This time, the attacker could just place the malicious DLL in the 'C:\Windows' directory, hence replacing the legitimate HelloDLL.dll in the Windows directory.

This approach also has its advantage. If the sizes of original and malicious DLLs are similar, it wouldn't raise suspicion among the victims. However, this method would not work if the legitimate DLL is already present in the application's directory.

Search Order DLL Hijacking

Phase 02 A

Week 01

When executed, the malicious DLL ('HelloDLL.dll') will be loaded from the Windows directory. Since initially, before replacing non-malicious DLL, 'HelloDLL.dll' was loaded from the Windows directory, this time as well it would not raise suspicion.

.. main_delete.exe	9036	Load Image	C:\Windows\SysWOW64\ws2_32.dll	SUCCESS	Image Base: 0x76760000, Image Size: 0x63000
.. main_delete.exe	9036	Load Image	C:\Windows\SysWOW64\port4.dll	SUCCESS	Image Base: 0x75fd0000, Image Size: 0xbf000
.. main_delete.exe	9036	Load Image	C:\Windows\HelloDLL.dll	SUCCESS	Image Base: 0x8f500000, Image Size: 0x10000
.. main_delete.exe	9036	Load Image	C:\Windows\SysWOW64\ws2_32.dll	SUCCESS	Image Base: 0x76760000, Image Size: 0x63000
.. main_delete.exe	9036	Load Image	C:\Windows\SysWOW64\port4.dll	SUCCESS	Image Base: 0x75fd0000, Image Size: 0xbf000
.. main_delete.exe	9036	Load Image	C:\Windows\SysWOW64\mwssock.dll	SUCCESS	Image Base: 0x72c20000, Image Size: 0x52000

By looking at the above screenshot, we do not see any suspicious thread activity under Procmon. However, the loaded 'HelloDLL.dll' is the malicious one. This just indicates the effectiveness of the attack.

This concludes the work on Search order DLL hijacking.

Conclusion:

After learning its severity, its also important to look at some mitigation strategies employed:

- **Secure DLL Loading:** Ensure that applications follow secure DLL loading practices. This includes using absolute paths when loading DLLs instead of relying on the system's default search order. By specifying the exact location of the DLL, the risk of hijacking is reduced.
- **Manifest File Implementation:** Implement manifest files for applications. Manifest files provide explicit instructions to the operating system regarding DLL loading behavior. By specifying the DLL dependencies and their locations, the risk of search order hijacking can be mitigated.
- **DLL Side-by-Side (SxS) Assemblies:** Utilize SxS assemblies to isolate the dependencies of applications. SxS allows applications to have their own private copies of DLLs, reducing the reliance on system-wide DLLs and minimizing the risk of hijacking.

In conclusion, search order hijacking poses a significant security risk by allowing attackers to load malicious DLLs. To mitigate this threat, it is important to enforce secure DLL loading mechanisms, such as using strong cryptographic signatures, and implementing code integrity checks. Regularly updating software and maintaining a robust patch management process can also help prevent the exploitation of known vulnerabilities.

By adopting these measures, organizations can enhance their defenses against search order hijacking attacks.

Search Order DLL Hijacking

Phase 02 A

Week 01