LPE Lab 02 [**DLL Hijacking**]

Table of Contents

Setup requirements

Use the Tools below to conduct our service enumeration and identify misconfigurations that we can leverage later for the privilege escalation:

- windows 10 host (Victim)
- Kali Linux host (Attacker)
- Process Monitor

Windows programs don't do everything themselves. They load helper files called DLLs (Dynamic-Link Libraries) think of them as plug in toolboxes. When a program needs a DLL, Windows goes looking for a file with that name in a search order (a list of folders to check).

DLL hijacking is when an attacker slips a fake toolbox (malicious DLL) somewhere earlier in that search order so the program loads the attacker's file instead of the real one. Result: their code runs under the cover of a legitimate app often signed, often trusted so it can be stealthy.

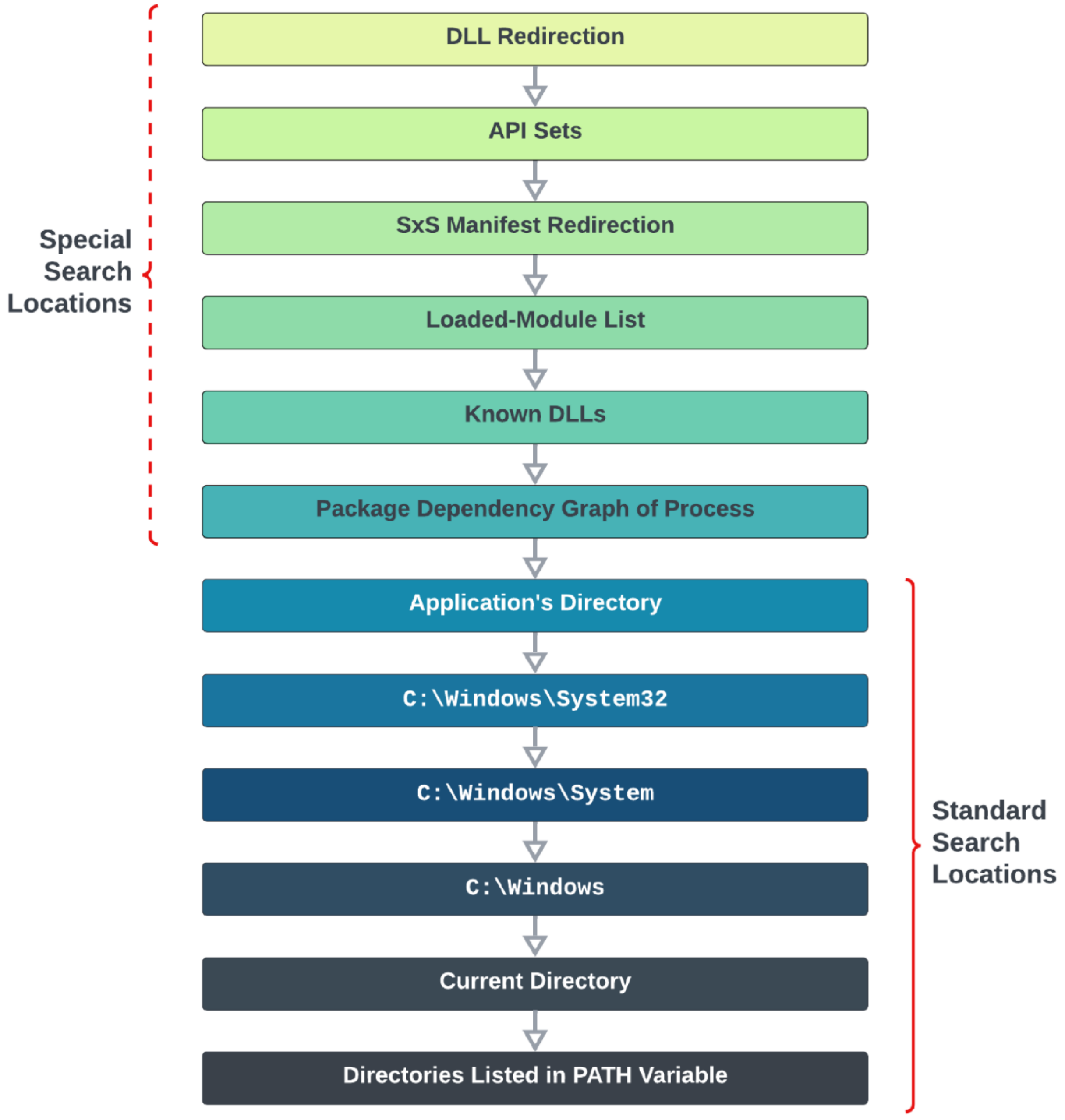
DLL Hijacking

On Windows, a **DLL** is a plug-in file that apps load to reuse code.
DLL hijacking is when a legit app accidentally loads a fake DLL (planted by an attacker) instead of the real one. That lets the attacker run code *inside* a trusted app—useful for hiding (defense evasion), getting more power (privilege escalation), or sticking around after reboot (persistence).

To see why this happens, you need the **DLL search order**: when an app asks for something.dll **by name** (not full path), Windows checks certain folders in a specific order and uses the first match it finds.

Windows DLL Search Order

DLL hijacking relies on the **DLL search order** that Windows uses when loading DLL files. This search order is a sequence of locations a program checks when loading a DLL. The sequence can be divided into two parts: special search locations and standard search locations



Special Search Locations

Special search locations are taken into account before the standard search locations, and they contain different factors that can control the locations to be searched and used to load a DLL. These locations are based on the application and the system configurations.

1. **DLL redirection** allows specifying which DLL should be loaded by the DLL loader
2. **API sets** allows dynamically routing function calls to the appropriate DLL based on the version of Windows and the availability of different features
3. SxS **manifest** redirection redirects DLL loading by using application manifests
4. Loaded-module list verifies whether the DLL is already loaded into memory
5. Known DLLs checks whether the DLL name and path match the Windows list of known DLLs. This list resides in HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs
6. The **package** dependency graph of the process, in case it was executed as part of a packaged app

Standard Search Locations

The standard search locations are the ones most associated with the DLL hijacking technique, and they will usually be used by adversaries. Windows will use the following order to search for the desired DLL.

1. The application's directory (the directory containing the executable)
2. C:\Windows\System32
3. C:\Windows\System
4. C:\Windows
5. The current directory (the directory from which we execute the executable)
6. Directories listed in the PATH environment variable

Hijacking this whole DLL search order will grant an adversary the option to load their malicious DLL within the context of a legitimate application and achieve stealthy execution. They can do this by triggering a malicious DLL to load before the valid one, replacing the DLL or by altering the order (specifically the PATH environment variable).

Implementations of DLL hijacking

As the concept of DLL hijacking continues to evolve over time, threat actors have evolved as well, using different approaches to perform this kind of attack. The three most common techniques we have observed are DLL side-loading, DLL search order hijacking and phantom DLL loading. The most common technique is DLL side-loading

1) DLL side-loading (the crowd favorite)

Idea: A trusted, signed EXE looks for X.dll in its own folder first. Attacker drops a malicious X.dll **next to** that EXE. The EXE launches, happily loads the attacker's DLL.

Why it works: App uses LoadLibrary("X.dll") (name only), and the **application directory** is high in the search order.

2) DLL search-order hijacking

Idea: Place the malicious DLL in **any location** that the loader checks **before** the real one (not necessarily alongside the EXE). That might be the current working directory or a user-writable folder that's (unwisely) on PATH.

Why it works: Windows walks the search list (app dir → System32 → Windows → current dir → PATH folders) and uses the **first** match.

3) Phantom DLL loading (aka "missing DLL planting")

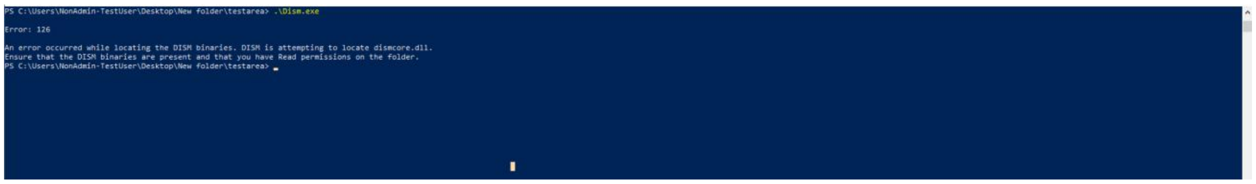
Idea: The app requests a DLL that normally **doesn't exist** on disk (import table or runtime call). Windows searches, fails repeatedly, and if an attacker later **plants** a DLL with that missing name in an early search location, the app will load it.

Why it works: The loader doesn't find the expected file, so **any** correctly named drop-in wins.

Lab Setup

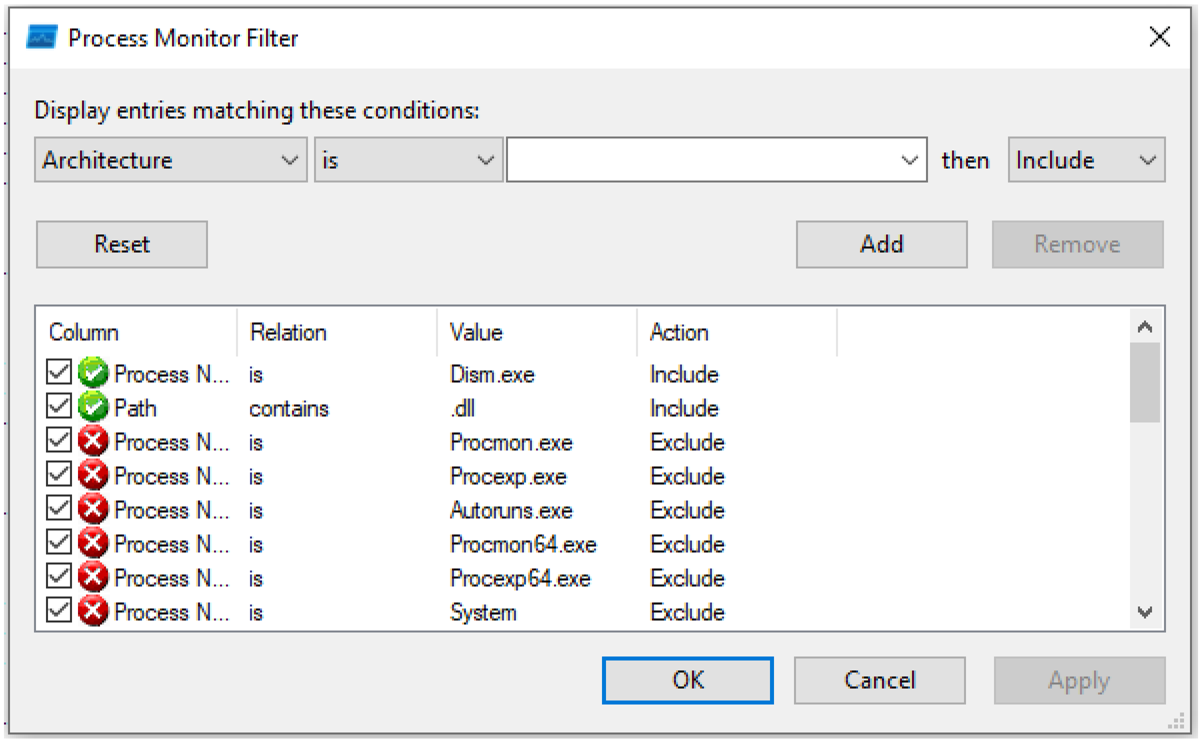
Windows loads DLLs by checking locations in a fixed **search order** and uses the **first** match it finds. In *phantom loading*, an app asks for a DLL that doesn't exist on disk. If a file with that missing name later appears in an early search location (like the app's folder), the loader will pick it up.

DISM.exe (Deployment Image Servicing and Management) is a Microsoft tool for servicing Windows images (.wim, .ffu, .vhd/.vhdx) and for certain OS maintenance tasks. We'll use it as the **legitimate host** that triggers a DLL lookup.



```
PS C:\Users\NonAdmin-TestUser\Desktop\New folder\testarea> .\Dism.exe

Error: 126

An error occurred while locating the DISM binaries. DISM is attempting to locate dismcore.dll.
Ensure that the DISM binaries are present and that you have Read permissions on the folder.
PS C:\Users\NonAdmin-TestUser\Desktop\New folder\testarea>
```

now set up your visibility. Launch ProcMon with administrative rights. Open the filter dialog and include process name is "dism.exe" and the path ends with ".dll . Start capturing. With ProcMon recording, open an elevated Command Prompt in your lab folder and run a harmless DISM invocation such as dism.exe /? (or dism.exe /online /?) to trigger module lookups without making any changes to the OS.


Process Monitor Filter

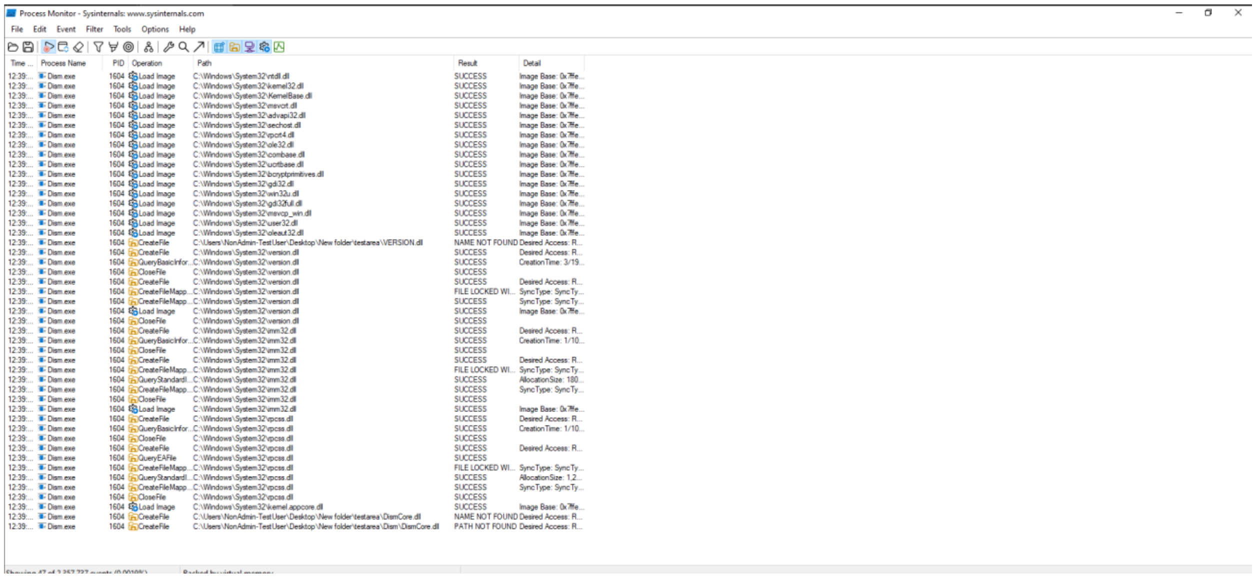
Display entries matching these conditions:

Architecture | is | | then | Include

Column	Relation	Value	Action
☑ ✓ Process N...	is	Dism.exe	Include
☑ ✓ Path	contains	.dll	Include
☑ ✗ Process N...	is	Procmon.exe	Exclude
☑ ✗ Process N...	is	Procexp.exe	Exclude
☑ ✗ Process N...	is	Autoruns.exe	Exclude
☑ ✗ Process N...	is	Procmon64.exe	Exclude
☑ ✗ Process N...	is	Procexp64.exe	Exclude
☑ ✗ Process N...	is	System	Exclude

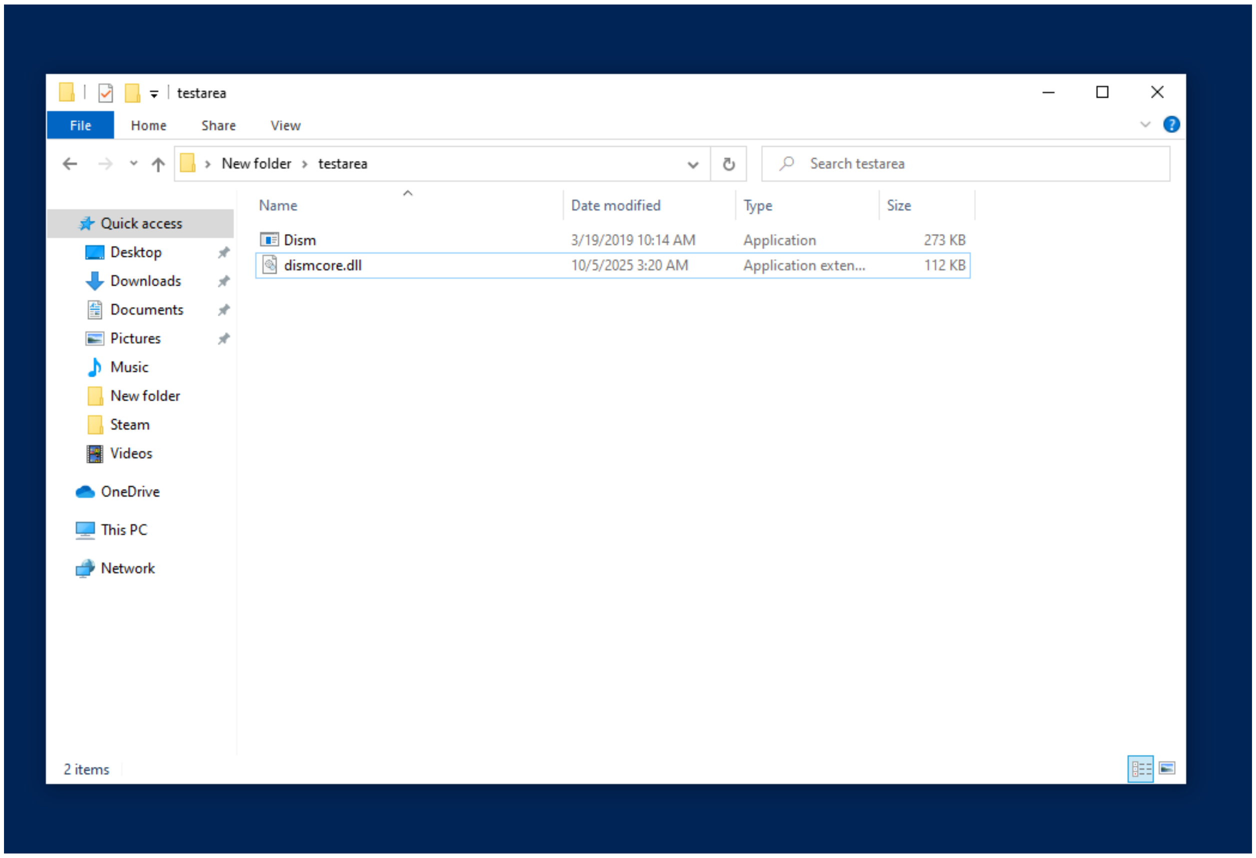
As the help text prints, flip back to ProcMon and observe the sequence of DLL resolution attempts. In phantom DLL loading, a signed application asks for a DLL that isn't present on disk. You'll see a series of "NAME NOT FOUND" results across various folders in the search order, followed by a successful "Load Image" from your lab directory when the loader finds a file with the expected name there. That successful load from a non-system path is the behavior you're demonstrating: the legitimate process has accepted the first matching DLL the search order presented. If you want to corroborate what you saw, open Process Explorer, locate dism.exe, open its properties, and check the DLLs tab; you should see your benign demonstration DLL listed with its full on-disk path and, if you verify signatures, note that dism.exe is Microsoft-signed while your stub is not.

If the console or a calling program reports **Error 126**, that typically indicates the application failed to load a DLL. To confirm the cause rather than guess, use Process Monitor exactly as described: apply the filters, start the capture, and then run dism.exe while ProcMon is recording. Review the "Load Image" events to see which DLL names returned "NAME NOT FOUND," which locations were tried, and whether a later successful load came from an unexpected directory. This evidence-first approach turns a vague error code into a precise explanation: which DLL was sought, where it was found (or not), and why.


Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help

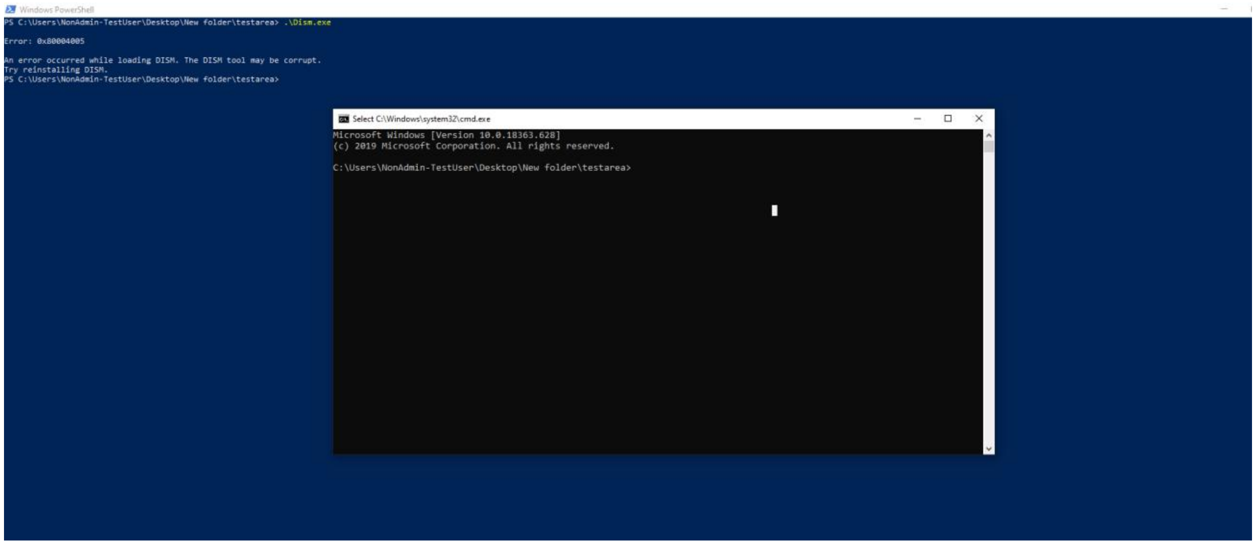
```
Time ...  Process Name  PID   Operation           Path                                                                 Result           Detail
12:39:... Dism.exe      1604  Load Image          C:\Windows\System32\ntdll.dll                                        SUCCESS          Image Base: 0x7ffe...
12:39:... Dism.exe      1604  Load Image          C:\Windows\System32\kernel32.dll                                     SUCCESS          Image Base: 0x7ffe...
12:39:... Dism.exe      1604  Load Image          C:\Windows\System32\KernelBase.dll                                   SUCCESS          Image Base: 0x7ffe...
12:39:... Dism.exe      1604  Load Image          C:\Windows\System32\msvcrt.dll                                       SUCCESS          Image Base: 0x7ffe...
12:39:... Dism.exe      1604  Load Image          C:\Windows\System32\advapi32.dll                                     SUCCESS          Image Base: 0x7ffe...
12:39:... Dism.exe      1604  Load Image          C:\Windows\System32\sechost.dll                                      SUCCESS          Image Base: 0x7ffe...
12:39:... Dism.exe      1604  Load Image          C:\Windows\System32\rpcrt4.dll                                       SUCCESS          Image Base: 0x7ffe...
12:39:... Dism.exe      1604  Load Image          C:\Windows\System32\ole32.dll                                        SUCCESS          Image Base: 0x7ffe...
12:39:... Dism.exe      1604  Load Image          C:\Windows\System32\combase.dll                                      SUCCESS          Image Base: 0x7ffe...
12:39:... Dism.exe      1604  Load Image          C:\Windows\System32\ucrtbase.dll                                     SUCCESS          Image Base: 0x7ffe...
12:39:... Dism.exe      1604  Load Image          C:\Windows\System32\bcryptprimitives.dll                             SUCCESS          Image Base: 0x7ffe...
12:39:... Dism.exe      1604  Load Image          C:\Windows\System32\gdi32.dll                                        SUCCESS          Image Base: 0x7ffe...
12:39:... Dism.exe      1604  Load Image          C:\Windows\System32\win32u.dll                                       SUCCESS          Image Base: 0x7ffe...
12:39:... Dism.exe      1604  Load Image          C:\Windows\System32\gdi32full.dll                                    SUCCESS          Image Base: 0x7ffe...
12:39:... Dism.exe      1604  Load Image          C:\Windows\System32\msvcp_win.dll                                    SUCCESS          Image Base: 0x7ffe...
12:39:... Dism.exe      1604  Load Image          C:\Windows\System32\user32.dll                                       SUCCESS          Image Base: 0x7ffe...
12:39:... Dism.exe      1604  Load Image          C:\Windows\System32\oleaut32.dll                                     SUCCESS          Image Base: 0x7ffe...
12:39:... Dism.exe      1604  CreateFile          C:\Users\NonAdmin-TestUser\Desktop\New folder\testarea\VERSION.dll   NAME NOT FOUND   Desired Access: R...
12:39:... Dism.exe      1604  CreateFile          C:\Windows\System32\version.dll                                      SUCCESS          Desired Access: R...
12:39:... Dism.exe      1604  QueryBasicInfor...  C:\Windows\System32\version.dll                                      SUCCESS          CreationTime: 3/19...
12:39:... Dism.exe      1604  CloseFile           C:\Windows\System32\version.dll                                      SUCCESS
12:39:... Dism.exe      1604  CreateFile          C:\Windows\System32\version.dll                                      SUCCESS          Desired Access: R...
12:39:... Dism.exe      1604  CreateFileMapp...   C:\Windows\System32\version.dll                                      FILE LOCKED WI...  SyncType: SyncTy...
12:39:... Dism.exe      1604  CreateFileMapp...   C:\Windows\System32\version.dll                                      SUCCESS          SyncType: SyncTy...
12:39:... Dism.exe      1604  Load Image          C:\Windows\System32\version.dll                                      SUCCESS          Image Base: 0x7ffe...
12:39:... Dism.exe      1604  CloseFile           C:\Windows\System32\version.dll                                      SUCCESS
12:39:... Dism.exe      1604  CreateFile          C:\Windows\System32\imm32.dll                                        SUCCESS          Desired Access: R...
12:39:... Dism.exe      1604  QueryBasicInfor...  C:\Windows\System32\imm32.dll                                        SUCCESS          CreationTime: 1/10...
12:39:... Dism.exe      1604  CloseFile           C:\Windows\System32\imm32.dll                                        SUCCESS
12:39:... Dism.exe      1604  CreateFile          C:\Windows\System32\imm32.dll                                        SUCCESS          Desired Access: R...
12:39:... Dism.exe      1604  CreateFileMapp...   C:\Windows\System32\imm32.dll                                        FILE LOCKED WI...  SyncType: SyncTy...
12:39:... Dism.exe      1604  QueryStandardI...   C:\Windows\System32\imm32.dll                                        SUCCESS          AllocationSize: 180...
12:39:... Dism.exe      1604  CreateFileMapp...   C:\Windows\System32\imm32.dll                                        SUCCESS          SyncType: SyncTy...
12:39:... Dism.exe      1604  CloseFile           C:\Windows\System32\imm32.dll                                        SUCCESS
12:39:... Dism.exe      1604  Load Image          C:\Windows\System32\imm32.dll                                        SUCCESS          Image Base: 0x7ffe...
12:39:... Dism.exe      1604  CreateFile          C:\Windows\System32\rpcss.dll                                        SUCCESS          Desired Access: R...
12:39:... Dism.exe      1604  QueryBasicInfor...  C:\Windows\System32\rpcss.dll                                        SUCCESS          CreationTime: 1/10...
12:39:... Dism.exe      1604  CloseFile           C:\Windows\System32\rpcss.dll                                        SUCCESS
12:39:... Dism.exe      1604  CreateFile          C:\Windows\System32\rpcss.dll                                        SUCCESS          Desired Access: R...
12:39:... Dism.exe      1604  QueryEAFile         C:\Windows\System32\rpcss.dll                                        SUCCESS
12:39:... Dism.exe      1604  CreateFileMapp...   C:\Windows\System32\rpcss.dll                                        FILE LOCKED WI...  SyncType: SyncTy...
12:39:... Dism.exe      1604  QueryStandardI...   C:\Windows\System32\rpcss.dll                                        SUCCESS          AllocationSize: 1,2...
12:39:... Dism.exe      1604  CreateFileMapp...   C:\Windows\System32\rpcss.dll                                        SUCCESS          SyncType: SyncTy...
12:39:... Dism.exe      1604  CloseFile           C:\Windows\System32\rpcss.dll                                        SUCCESS
12:39:... Dism.exe      1604  Load Image          C:\Windows\System32\kernel.appcore.dll                               SUCCESS          Image Base: 0x7ffe...
12:39:... Dism.exe      1604  CreateFile          C:\Users\NonAdmin-TestUser\Desktop\New folder\testarea\DismCore.dll  NAME NOT FOUND   Desired Access: R...
12:39:... Dism.exe      1604  CreateFile          C:\Users\NonAdmin-TestUser\Desktop\New folder\testarea\Dism\DismCore.dll  PATH NOT FOUND  Desired Access: R...

Showing 47 of 2,357,737 events (0.0019%)    Backed by virtual memory
```

Now we identified that missing DismCore.dll is missing and application is search it in the current working directory so in this case lets add the missing DismCore.dll but with a malisious code.



Name	Date modified	Type	Size
Dism	3/19/2019 10:14 AM	Application	273 KB
dismcore.dll	10/5/2025 3:20 AM	Application exten...	112 KB

And now if we were to run the dism.exe the dll we created will executed with it and give us the access



```
PS C:\Users\NonAdmin-TestUser\Desktop\New folder\testarea> .\Dism.exe

Error: 0x80004005

An error occurred while loading DISM. The DISM tool may be corrupt.
Try reinstalling DISM.
PS C:\Users\NonAdmin-TestUser\Desktop\New folder\testarea>
```

```
Microsoft Windows [Version 10.0.18363.628]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\NonAdmin-TestUser\Desktop\New folder\testarea>
```