Dynamic-Link Library Security - Win32 apps

By stevewhims

Archived: 2026-04-05 15:25:44 UTC

When an application dynamically loads a dynamic-link library without specifying a fully qualified path name, Windows attempts to locate the DLL by searching a well-defined set of directories in a particular order, as described in Dynamic-Link Library Search Order. If an attacker gains control of one of the directories on the DLL search path, it can place a malicious copy of the DLL in that directory. This is sometimes called a *DLL preloading attack* or a *binary planting attack*. If the system does not find a legitimate copy of the DLL before it searches the compromised directory, it loads the malicious DLL. If the application is running with administrator privileges, the attacker may succeed in local privilege elevation.

For example, suppose an application is designed to load a DLL from the user's current directory and fail gracefully if the DLL is not found. The application calls **LoadLibrary** with just the name of the DLL, which causes the system to search for the DLL. Assuming safe DLL search mode is enabled and the application is not using an alternate search order, the system searches directories in the following order:

1. The directory from which the application loaded.
2. The system directory.
3. The 16-bit system directory.
4. The Windows directory.
5. The current directory.
6. The directories that are listed in the PATH environment variable.

Continuing the example, an attacker with knowledge of the application gains control of the current directory and places a malicious copy of the DLL in that directory. When the application issues the **LoadLibrary** call, the system searches for the DLL, finds the malicious copy of the DLL in the current directory, and loads it. The malicious copy of the DLL then runs within the application and gains the privileges of the user.

Developers can help safeguard their applications against DLL preloading attacks by following these guidelines:

- Wherever possible, specify a fully qualified path when using the **LoadLibrary**, **LoadLibraryEx**, **CreateProcess**, or **ShellExecute** functions.
- Use the LOAD_LIBRARY_SEARCH flags with the **LoadLibraryEx** function, or use these flags with the **SetDefaultDllDirectories** function to establish a DLL search order for a process and then use the **AddDllDirectory** or **SetDllDirectory** functions to modify the list. For more information, see Dynamic-Link Library Search Order.

  Windows 7, Windows Server 2008 R2, Windows Vista and Windows Server 2008: These flags and functions are available on systems with KB2533623 installed.

- On systems with KB2533623 installed, use the LOAD_LIBRARY_SEARCH flags with the **LoadLibraryEx** function, or use these flags with the **SetDefaultDllDirectories** function to establish a DLL search order for a process and then use the **AddDllDirectory** or **SetDllDirectory** functions to modify the list. For more information, see Dynamic-Link Library Search Order.

- Consider using DLL redirection or a manifest to ensure that your application uses the correct DLL.

- When using the standard search order, make sure that safe DLL search mode is enabled. This places the user's current directory later in the search order, increasing the chances that Windows will find a legitimate copy of the DLL before a malicious copy. Safe DLL search mode is enabled by default starting with Windows XP with Service Pack 2 (SP2) and is controlled by the **HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\SafeDllSearchMode** registry value. For more information, see Dynamic-Link Library Search Order.

- Consider removing the current directory from the standard search path by calling **SetDllDirectory** with an empty string (""). This should be done once early in process initialization, not before and after calls to **LoadLibrary**. Be aware that **SetDllDirectory** affects the entire process and that multiple threads calling **SetDllDirectory** with different values can cause undefined behavior. If your application loads third-party DLLs, test carefully to identify any incompatibilities.

- Do not use the **SearchPath** function to retrieve a path to a DLL for a subsequent **LoadLibrary** call unless safe process search mode is enabled. When safe process search mode is not enabled, the **SearchPath** function uses a different search order than **LoadLibrary** and is likely to first search the user's current directory for the specified DLL. To enable safe process search mode for the **SearchPath** function, use the **SetSearchPathMode** function with BASE_SEARCH_PATH_ENABLE_SAFE_SEARCHMODE. This moves the current directory to the end of the **SearchPath** search list for the life of the process. Note that the current directory is not removed from the search path, so if the system does not find a legitimate copy of the DLL before it reaches the current directory, the application is still vulnerable. As with **SetDllDirectory**, calling **SetSearchPathMode** should be done early in process initialization and it affects the entire process. If your application loads third-party DLLs, test carefully to identify any incompatibilities.

- Do not make assumptions about the operating system version based on a **LoadLibrary** call that searches for a DLL. If the application is running in an environment where the DLL is legitimately not present but a malicious copy of the DLL is in the search path, the malicious copy of the DLL may be loaded. Instead, use the recommended techniques described in Getting the System Version.

The Process Monitor tool can be used to help identify DLL load operations that might be vulnerable. The Process Monitor tool can be downloaded from https://technet.microsoft.com/sysinternals/bb896645.aspx.

The following procedure describes how to use Process Monitor to examine DLL load operations in your application.

To use Process Monitor to examine DLL load operations in your application

1. Start Process Monitor.
2. In Process Monitor, include the following filters:
    - Operation is CreateFile
    - Operation is LoadImage
    - Path contains .cpl
    - Path contains .dll
    - Path contains .drv
    - Path contains .exe
    - Path contains .ocx
    - Path contains .scr
    - Path contains .sys
3. Exclude the following filters:
    - Process Name is procmon.exe
    - Process Name is Procmon64.exe
    - Process Name is System
    - Operation begins with IRP_MJ_
    - Operation begins with FASTIO_
    - Result is SUCCESS
    - Path ends with pagefile.sys
4. Attempt to start your application with the current directory set to a specific directory. For example, double-click a file with an extension whose file handler is assigned to your application.
5. Check Process Monitor output for paths that look suspicious, such as a call to the current directory to load a DLL. This kind of call might indicate a vulnerability in your application.
