More information about the DLL Preloading remote attack vector

By swiat

Published: 2010-08-23 · Archived: 2026-04-05 22:38:46 UTC

/ By / August 23, 2010

Today we released Security Advisory 2269637 notifying customers of a remote attack vector to a class of vulnerabilities affecting applications that load DLL's in an insecure manner. The root cause of this issue has been understood by developers for some time. However, last week researchers published a remote attack vector for these issues, whereas in the past, these issues were generally considered to be local and relatively low impact. In this blog post, we'd like to share:

- **Background about the vulnerability**
- **Information to help you make a risk assessment for your environment**
- **An optional binary update you can install to protect your systems**
- **Guidance for developers**
- **What Microsoft is doing**

The vulnerability

When an application loads a DLL without specifying a fully qualified path name, Windows will attempt to locate the DLL by searching a defined set of directories. We have discussed the DLL search path on this blog and it has also been explained well on David LeBlanc's blog. For the sake of this issue, its sufficient to say that if an attacker can cause an application to LoadLibrary() while the application's current directory is set to an attacker-controlled directory, the application will run the attacker's code.Development best practicesstate that applications should call SetDllDirectory with a blank path before calling LoadLibrary("foo.dll") to ensure that foo.dll is not loaded from the current directory. We are investigating whether any of our own applications are affected by this class of vulnerability so that we can take appropriate action to protect customers.

Making a risk assessment for your environment

The most likely exploit scenario involves an attacker convincing the victim to open a file hosted on an attacker-controlled SMB or WebDAV share. The file itself would not necessarily be malicious or malformed. The key is that the file is loaded from a location where an attacker can also place a malicious DLL with the same name as a DLL the vulnerable application loads.

If a perimeter firewall prevents a system from making outbound SMB or WebDAV connections to attacker-controlled locations, this issue poses little risk. An attack cannot be automatically launched through email or web browsing attack vectors; a user must choose to open a file. However we recognize that users will often open trusted filetypes. We continue to recommend that all outbound SMB is filtered at the perimeter firewall. In addition, thesecurity advisoryrecommends disabling the WebDAV client service on workstations to prevent outbound WebDAV connections.

A tool you can use to protect your systems

Another option for protecting your systems is to deploy a tool that can help prevent exploitation of this issue. Knowledge Base article 2264107 offers for download a tool that allows customers to selectively change the library loading behavior, either system-wide or for specific applications.

Customers can set the following two registry keys: **

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\<br> Session Manager\CWDIllegalInDllSearch<br>HKEY_LOCAL_M
```

Setting the first key will define the system-wide behavior, whereas the second key will set the behavior for one particular application. Note that this is an Image File Execution Option (IFEO), and thus will be valid for all binaries with that same name on the system.

The values for these keys have slightly different effects, depending on from where the application is started.

Scenario 1: The application is started from a local folder, such as C:\Program Files

0xffffffff	**Removes the current working directory from the default DLL search order.**
0	Uses the default DLL search path. This is the Windows default, and the least secure setting.
1	Blocks a DLL load from the current working directory if the current working directory is set to a WebDAV folder.
2	Blocks a DLL load from the current working directory if the current working directory is set to a remote folder.

Scenario 2: The application is started from a remote folder, such as \\remote\share

0xffffffff	**Removes the current working directory from the default DLL search order.**
0	Uses the default DLL search path. This is the Windows default, and the least secure setting.
1	Blocks a DLL load from the current working directory if the current working directory is set to a WebDAV folder.
2	Allows DLL load from the current working directory if the current working directory is set to a remote folder. DLL’s that are loaded from a WebDAV share are blocked if the current working

0xffffffff	**Removes the current working directory from the default DLL search order.**
	directory is set to a WebDAV share.

Scenario 3: The application is started from a WebDAV folder, such as http://remote/share

0xffffffff	**Removes the current working directory from the default DLL search order.**
0	Uses the default DLL search path. This is the Windows default, and the least secure setting.

How can developers address these issues?

Microsoft recommends that developers clearly define from where they intend to load specific libraries. This was documented in the specific LoadLibrary application programming interface documentation on MSDN. However, we recognize that this guidance may not always have been very clear. We recently published an MSDN article, "Dynamic-Link Library Security," that provides specific guidance to developers on how to load these libraries securely.

While there are several affected situations, described in detail in the above MSDN article, our general recommendations are:

- Where possible, use a fully qualified path name when loading a library;
- Remove the current directory from the search path by using SetDLLDirectory;
- Do not use SearchPath to locate a library. SearchPath was not intended to look for libraries to be loaded into the application process space, and uses an insecure search order;
- Do not attempt to load libraries purely to identify the version of Windows. Instead, use GetVersionEx or a similar function offered by the Windows API.

We've also recently drafted additional guidance to help developers understand this issue. You can find that developer guidance attached to the blog post.

What Microsoft is doing

Loading dynamic libraries is basic behavior for Windows and other operating systems, and the design of some applications require the ability to load libraries from the current working directory. Hence, this issue cannot directly be addressed in Windows without breaking expected functionality. Instead, it requires developers to ensure they code secure library loads. However, we're looking into ways to make it easier for developers to not make this mistake in the future.

Microsoft is also conducting a thorough investigation into how this new vector may affect Microsoft products. As always, if we find this issue affects any of our products, we will address them appropriately.

We hope this blog helps address any questions you may have. Thanks to Mark Debenham, Anoop KV, Hari Pulapaka, Dou Kaya, Gov Maharaj, David LeBlanc, and Michael Howard for their work on this issue.

Cheers,

Jonathan Ness, MSRC Engineering
Maarten Van Horenbeeck, MSRC Program Manager

- Mitigations
- Risk Asessment
- Workarounds
