text/plain

DLL Hijacking Just Won't Die

2015-12-182015-12-27 | ericlaw | browsers, security
The folks that build the NSIS Installer (http://sourceforge.net/projects/nsis/) have released updates to mitigate a serious security bug related to DLL loading. (v2.5 and v3.0b3 include the fixes) (http://nsis.sourceforge.net/Download).

To make a long and complicated story short, a bad guy who exploits this vulnerability places a malicious DLL into your browser's Downloads folder, then waits. When you run an installer built by an earlier version of NSIS from that folder, the elevation prompt (assuming it runs at admin) shows the legitimate installer's signature asking you for permission to run the installer. After you grant permission, the victim installer loads the malicious DLL which **runs its malicious code** with the installer's permissions. And then it's not your computer (https://technet.microsoft.com/en-us/library/hh278941.aspx) anymore.

So, how does the attacker get the malicious DLL into the browser's Downloads folder? Surely that must involve some crazy hacking or social engineering, right?

Nah. The bad guy just navigates a frame of your browser to the DLL of his choice and, if you're on Chrome or Microsoft Edge (http://justhaifei1.blogspot.com/2015/10/watch-your-downloads-risk-of-auto.html), the DLL is dropped in the Downloads folder without even asking. **Oops.**

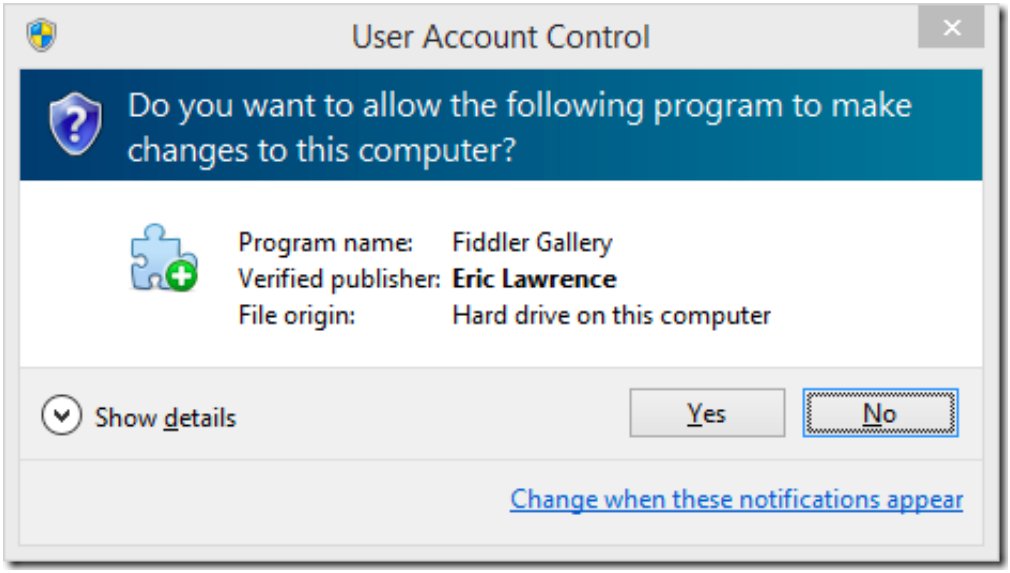
It's trivial to simulate this attack to find vulnerable installers, even as a total noob.

1. Simply click this link (https://bayden.com/dl/version.dll) which downloads a fake DLL (a simple text file containing the string ThisIsNotADLL).

2.Then download and run an installer built by an old version of NSIS:

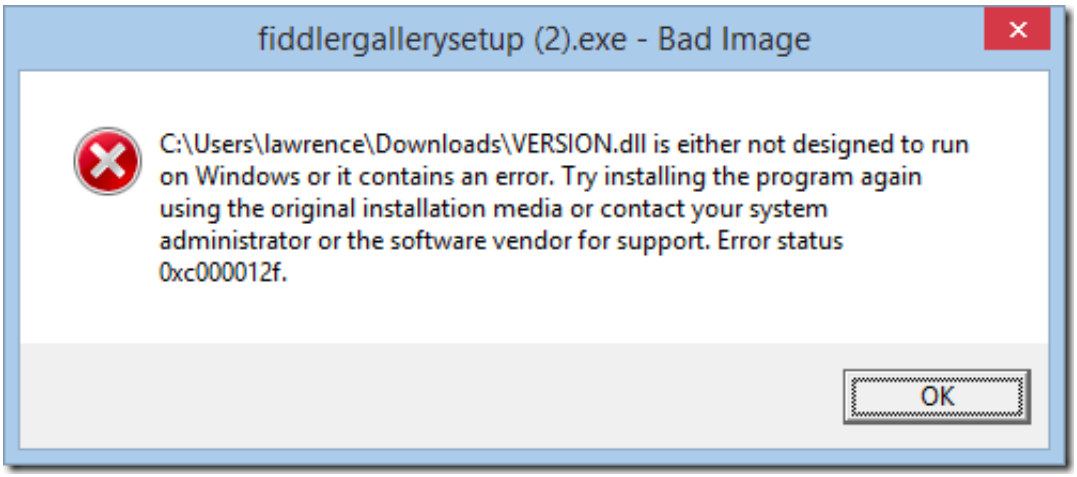
(http://textplain.files.wordpress.com/2015/12/image17.png)

3. Accept the elevation prompt:



(http://textplain.files.wordpress.com/2015/12/image18.png)

4. **Boom**… watch Windows try to load your fake DLL as code:



Of course, a real bad guy isn't going to be nice enough to use a fake DLL, he'll instead use a valid but malicious one containing code that runs during DLL load, silently rooting your computer or whatever other nefarious thing he'd like.

Around 80% of the installers in my \Downloads\ folder are vulnerable to this attack; half of those are built on NSIS and half are built using other technologies.

You can learn more about DLL Hijacking attacks here (http://blog.fortinet.com/post/a-crash-course-in-dll-hijacking).

Action Items

Here are my suggestions:

1. If you build installers with NSIS, please upgrade immediately.
2. If you build installers with other technology, test for this attack (version.dll, msi.dll,

shfolder.dll, etc). This post (http://blog.opensecurityresearch.com/2014/01/unsafe-dll-loading-vulnerabilities.html) suggests a more complete way of finding target filenames. This post notes that InnoSetup (http://seclists.org/fulldisclosure/2015/Dec/33) is vulnerable.
3. Read Microsoft's guidance (http://blogs.technet.com/b/srd/archive/2010/08/23/more-information-about-dll-preloading-remote-attack-vector.aspx) for mitigating this vulnerability.
4. Ask your browser vendor what they're doing to prevent attacks of this nature.

-Eric Lawrence

3 thoughts on "DLL Hijacking Just Won't Die"

ERICLAW SAYS: http://seclists.org/bugtraq/2015/Dec/112
2015-12-23 at 16:55 • *Reply »*
ERICLAW SAYS: "All self-extracting archives created with 7-zip are also vulnerable" https://community.rapid7.com/community/infosec/blog/2015/12/21/scannow-dll-search-order-hijacking-vulnerability-and-deprecation

2015-12-23 at 17:04 • *Reply »*
ANDERS SAYS: This is basically a Windows design flaw and anything < Vista is impossible to fix 100% by ISVs.
On Vista/7 with updates or 8 and later applications should call SetDefaultDllDirectories so dlls specified by relative paths only load from system32. That is only half the battle and only applies to things loaded by LoadLibrary at run-time. To stay secure without restricting the search to system32 only you would have to test every single Windows API function you call on every Windows version you support at all service pack levels and all new updates. Example: OleInitialize internally calls CreateWindow to create a hidden window for COM. The kernel part of CreateWindow calls back into usermode and uses LoadLibrary to load uxtheme.dll so it can hook in its Visual Styles support. To fix this particular function your application has to manually call LoadLibrary on %windir%\system32\uxtheme.dll before it calls OleInitialize. This vulnerability also exists for every PE delay-loaded function because the delay-load helper stub on these systems use a relative path to the .dll. Shell32.dll has a fair amount of delay-loaded functions in common code paths.

The other problem is implicitly linked libraries that are loaded before the application starts running. The KnownDLLs list prevents this issue somewhat but it was never designed as a security feature and the list is not the same on all Windows versions. Taking this to the extreme you could say that you can only implicitly link with kernel32.dll and all functions
1. in other dlls have to be delay-loaded after calling SetDefaultDllDirectories!

2015-12-24 at 19:15 • *Reply »*