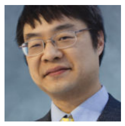
It rather involved being on the other side of this airtight hatchway: Planting DLLs into directories on the PATH for applications whose current directory is always System32

devblogs.microsoft.com/oldnewthing/20131023-00

October 23, 2013



Raymond Chen

A bunch of security vulnerability reports came in from the same person, all of the form, "Program X is vulnerable to DLL planting if you create a DLL with the name Y in a directory on the PATH," with varying values of X and Y. In all the cases, Program X runs as SYSTEM with the System32 directory as its current directory, so a current directory attack does not apply here. The only remaining directory to attack is the PATH. But if you can attack the PATH for the SYSTEM user, then you are already on the other side of the airtight hatchway. The default PATH directories are all directories which are read-only to non-administrative users. In order to plant a DLL there, you must already be an administrator, at which point you are on the other side of the airtight hatchway. If you gain administrative access, then why bother planting DLLs in sneaky places? You already pwn the machine. Just do whatever you want! In other words, planting a DLL into secured directories can be carried out only by administrators, at which point it's just the administrator attacking his own computer. Not interesting. The next angle of investigation is whether somebody can sneak an insecure directory onto the global PATH. But modifying the global PATH requires administrative privileges, so if you have the privilege to add an insecure directory to the path, you are already an administrator, so this is another case of an administrator attacking his own computer. Still not interesting. Finally, the third option is that some application, as part of its installation, added an insecure directory to the global PATH. But in that case, you don't need to do any DLL planting at all. Just put a rogue executable on the PATH. For example, you might call it `tpye.exe`, so that somebody who typos the `type` command runs your rogue executable instead. In other words, in order for DLL planting into an insecure directory on the global PATH to work, you must already have a system whose integrity has already been violated, albeit inadvertently in this case. If you have an attack along these lines, then the security vulnerability is in the application whose installer added an insecure directory to the global PATH. This is another case of *If you set up an insecure system, don't be surprised that there's a security vulnerability*.

Now, there is still a flaw in the application that gets tricked into loading the rogue DLL. It should be more specific about how it loads DLLs so it doesn’t load the wrong one by accident. But there is no elevation of privilege, since only administrators can trick the application into loading the wrong DLL.

Raymond Chen

Follow