DLL hooks in the Microsoft Windows Operating System

Roger Jakobsen
Roger.Jakobsen.5225@student.uu.se
Håkan Larsson
hakan.larsson@streamingemotions.se

2005-10-19

Abstract

DLL files presents a windows programmer with a pool of functions that are necessary to truly interact with the environment. DLL hooking is a way to use DLL files to communicate with different components of the Microsoft Windows Operating System. DLL hooking is often used to monitor system events. This concept can also be used to generate a securityrisk. Password sniffing is one example of an application that is easy to implement using DLL hooking.

Contents

1 DLL-files

1.1 What is a DLL-file?

A DLL-file, or dynamic link library file, is a set of functions. These functions may be called from any processes running on the system. The purpose of most DLL-files is to offer an interface between different components of a computer system. The DLL-files that offers interfaces between software and hardware on a system is usually referred to as device drivers. The set of DLL-files on a system can be compared to the library routines that is provided with most larger programming languages, for example C and C++. DLL files is linked into programs at runtime, not compiled with the main program.

1.2 Main advantages

The main advantage of DLL files is that by calling them from a program the process requires less space in the internal memory. DLL files that is used in programs is not loaded into the internal memory when the program is loaded. Instead the DLL-files is loaded upon direct demand. If a user runs an instance of a text editor program the printer routine is not always needed, because we never print a document every time you edit it. The idea is that the code needed to print a document should only be loaded when the code is actually needed. The DLL-files represent a function pool that is available to programs running on a system.

2 DLL-hooks

2.1 What are DLL-hooks?

A hook is a function that may be created either as part of a DLL-file or an application. You can use hooks to monitor various events in the system environment. The basic idea is to create a function that is called every time a certain event is generated. For example when a user presses a key on the keyboard, moves the mouse or when an application 'looses focus'.(for example when a user switches to another window) Hooks can be used to many things. The original reason for providing hooks in the windows operating system were that it eased debugging. Hooks are, however, used in many different ways

to solve different tasks. For example keyboard stroke recording or mouse movement detection, more about this later.There are basically 2 types of hooks in windows. A local hook is a hook that monitors things happening only for a specific thread(read program). A global hook can monitor the entire system (all threads). Both types of hooks have a similar structure, the main difference between local hooks and global hooks is that for a local hook, the function to be called can be within the program it is monitoring, but with a global hook the function must be stored and loaded from a separate DLL. Global hooks must be in a DLL-file because the DLL-files are loaded into a shared memory segment. This way the hook is available to all running threads on the system.

2.2 When can a DLL-hook be used?

A hook can be used to intercept most of the messages sent in windows, low level keyboard strokes, minimizing or maximizing windows and much more. One hook that might be of high value by an attacker is the WH_KEYBOARD Hook that allows an attacker to intercept keyboard strokes. He could then use the hook for 'sniffing'password and other sensitive information typed in by the user. Another more friendly use of a hook can be for a curious employer to monitor the workers efficiency during the day. By installing a hook that records and logs every time a program gets and looses focus. Later using this information he can make a summary of the workers day. Ex, 50 percent of time spent in Visual studio, 30 percent spent in explorer and 20 spent in windows media player. By some clever techniques the worker will never find the process doing the statistic collection.

2.3 Protection against hooks

There are a few ways to protect one against hooks. The spy safe edit control is one way to ask the user for sensitive information without the risk of exposing the information. What the control does is that it actually generates a random sequence of simulated keystrokes for every user keystroke. The hook will intercept all the keystroke and assume they are the sensitive information. So if the user enters the string 'hello' the hook will in the end get 'lkjvhwefesdlsdvwevlvdsvov'. It is not fail safe but it makes it all much harder. Another solution is a bit more drastic and might create other problems on the way. The WH_DEBUG-hook is a hook that is called before calling hook procedures associated with any other hook in the system, this hook can be used

in order to allow the system to call hook procedures associated with other types of hooks. So setting this hook and not allowing any other hook solves the initial problem, it also denies any other legitimate hooks that needs to be made. And what happens if someone hooks our DEBUG-hook and so on.

2.4 Destructive use of hooks

Say that we have managed to install a evil keyboard hook in to a system that is firewalled and the security is considered good. Now we are sitting on all these password and we want to send them somewhere where they later can be used. One solution would be to upload them to an remote ftp server or similar. But most firewall's will detect and block unauthorized programs from connecting to the Internet. A solution to this problem would be to install a global DLL-hook and then let it check if the running program is for example explorer or firefox, load some additional code for uploading the information. (The hook would call the LoadLibrary function or similar for loading some larger piece of code for the upload process)

3 Alternate Data Streams(ADS)

3.1 What is ADS?

In NTFS, there is a feature that enables to fort a file data into existing files without affecting their functionality, size, or display to file browsing utilities like Windows Explorer. A file consists of different data streams. One stream holds the security information (access rights and such things), another one holds the 'real data' you expect to be in a file. There may also be alternate data streams holding data the same way the standard stream does. The only downside of these are that they are completely hidden from the user. You might have a file that is 1byte in size but has many MB in a alternate data stream. The actual file is bigger than 1 byte but the standard file managers will report it as 1 byte. Making this feature more dangerous you dont need any special privileges to add alternate data streams to a file or directory, as long you are allowed to write you can do it.

3.2 A small example of how to use ADS

type atextfile - visible.txt:hidden2.txt
This will create another hidden stream hidden2.txt in the file visible.txt.
more - visible.txt:hidden2.txt - newfile.txt
This will create a file newfile.txt from the hidden stream hidden2.txt in the file visible.txt. Using these streams you can hide your code in a legitimate file that looks harmless when viewed. Not only can you hide code inside the file but when you execute this alternate stream it looks like the legitimate is executed. So now we have a way of executing our code without drawing to much attention to us. The ADS feature is a feature that cannot be turned off.

References

[1] http://delphi.about.com/library/bluc/text/uc063001a.htm :dll hooking, 2005.

[2] http://msdn.microsoft.com/library :dll hooks, 2005.

[3] http://searchwin2000.techtarget.com :dll-file, 2005.

[4] http://www.codeproject.com/dll/hooks.asp :dll hooks, 2005.

[5] http://www.codeproject.com/system/keylogger.asp :dll hooks, 2005.

[6] http://www.heysoft.de/nt/ntfs-ads.htm :ads, 2005.