# + libvlc.dll

Part of the Mijack Libs project.

Type DLL Sideloading (1 EXE)

By copying (and optionally renaming) a vulnerable application to a user-writeable folder, alongside a malicious <code>libvlc.dll</code>, arbitrary code can be executed through the legitimate application. See also MITRE ATT&CK® technique T1574.002: Hijack Execution Flow: DLL Side-Loading.

Vendor VLC

Resources news.sophos.com

www.microsoft.com

Last updated <u>over 1 year ago</u>

# **Expected Locations**

The file libvlc.dll is normally found in the following path:

> %PROGRAMFILES%\VideoLAN\VLC

### **Vulnerable Executables**

The following executable attempts to load libvlc.dll:

%PROGRAMFILES%\VideoLAN\VLC\vlc.exe

🏂 File hash av ailable

### **Detection**

Below a sample <u>Sigma</u> rule that will find processes that loaded <code>libvlc.dll</code> located in a folder that is not one of the expected locations (see above).

**Download YAML** 

Note that this rule is also included in the <u>Sigma feed</u> that comprises all DLL Hijacking entries part of this project.

# **FAQs**

### Why should I care about this?

DLL Hijacking enables the execution of malicious code through a signed and/or trusted executable. Defensive measures such as AV and EDR solutions may not pick up on this activity out of the box, and allow-list applications such as AppLocker may not block the execution of the untrusted code. There are numerous examples of threat actors that have been observed to leaverage DLL Hijacking to achieve their objectives. As such, this project wants to encourage you to monitor for unusual activity involving libvlc.dll.

#### How do I abuse this vulnerability?

As a red teamer, you will have to compile your own version of <code>libvlc.dll</code>. There are <u>various guides</u> on how this can be achieved.

#### How could the vendor have prevented this vulnerability?

Contribute to this project: https://github.com/wietze/HijackLibs

<u>DLL search order</u>. Explicitly specifying where a required DLL is located is easy and often already helps a lot. This doesn't have to hurt portability if Windows API calls are used to obtain paths, e.g. <u>GetSystemDirectory</u> to get the path of the System32 folder. Even better is to check the signature of required DLLs prior to loading them; most platforms, frameworks and/or runtimes offer means to verify DLL signatures with minimal performance impact.

#### This DLL Hijack doesn't seem to work (anymore), why is it still included?

Luckily, vendors regularly patch vulnerable applications in order to prevent DLL Hijacking from taking place. Nevertheless, older versions will remain vulnerable; for that reason, the entry won't be deleted from this project. To help others, you may want to open a pull request updating the 'precondition' tag on this entry to make the community aware of the reduced scope.

Homepage | API | Contributors