mpsvc.dll

Part of the Hijack Libs project.

Type	**DLL Sideloading** (1 EXE) By copying (and optionally renaming) a vulnerable application to a user-writeable folder, alongside a malicious `mpsvc.dll`, arbitrary code can be executed through the legitimate application. *See also MITRE ATT&CK® technique T1574.002: Hijack Execution Flow: DLL Side-Loading.*
Vendor	Microsoft
Resources	https://www.mcafee.com/blogs/other-blogs/mcafee-labs/revil-ransomware-uses-dll-sideloading/ https://news.sophos.com/en-us/2020/11/04/a-new-apt-uses-dll-side-loads-to-killlsomeone/ https://www.fortinet.com/blog/threat-research/dll-side-loading-technique-used-in-recent-kaseya-ransomware-attack
Last updated	Unknown

Expected Locations

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

```
%PROGRAMDATA%\Microsoft\Windows Defender\Platform\%VERSION%
```

Vulnerable Executables

The following executable attempts to load `mpsvc.dll`:

```
%PROGRAMDATA%\Microsoft\Windows Defender\Platform\%VERSION%\MsMpEng.exe
```

Detection

Below a sample Sigma rule that will find processes that loaded `mpsvc.dll` located in a folder that is not one of the expected locations (see above).

```yaml
title: Possible DLL Hijacking of mpsvc.dll
id: 9492751b-1313-48a3-6160-5b9ff8899459
status: experimental
description: Detects possible DLL hijacking of mpsvc.dll by looking for suspicious image loads, loading
this DLL from unexpected locations.
references:
    - https://hijacklibs.net/entries/microsoft/built-in/mpsvc.html
author: "Wietze Beukema"
date: 2021-12-07
tags:
    - attack.defense_evasion
    - attack.T1574.002
logsource:
    product: windows
    category: image_load
detection:
    selection:
        ImageLoaded: '*\mpsvc.dll'
    filter:
        ImageLoaded:
            - 'c:\programdata\Microsoft\Windows Defender\Platform\*\*'
```

File Image

Download YAML *Note that this rule is also included in the Sigma feed 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 `mpsvc.dll`.

How do I abuse this vulnerability?
As a red teamer, you will have to compile your own version of `mpsvc.dll`. There are various guides on how this can be achieved.

How could the vendor have prevented this vulnerability?
Most DLL Hijacking vulnerabilities are introduced by the 'lazy' loading of DLL files, which relies on Windows' default DLL search order. 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. GetSystemDirectory 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

Homepage | API | Contributors

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