DLL Hijacking: A Comprehensive Analysis

July 2025

Abstract

DLL hijacking is a persistent and sophisticated cybersecurity threat that exploits the Windows Dynamic Link Library (DLL) search order to execute malicious code. This paper provides an in-depth examination of DLL hijacking, including its definition, mechanisms, historical context, real-world examples (including recent vulnerabilities like CVE-2025-24076 and CVE-2025-24994), impact, and effective mitigation strategies. By understanding this attack vector and implementing recommended defenses, organizations can significantly reduce their vulnerability to such attacks.

1 Introduction

DLL hijacking is a cyberattack technique where attackers exploit the way Windows applications load Dynamic Link Libraries (DLLs) to execute malicious code. By placing a malicious DLL in a location that is searched before the legitimate DLL, attackers can trick applications into loading their malicious code instead of the intended library. This method is particularly dangerous because it can lead to privilege escalation, data theft, and the installation of persistent malware, often without the user's knowledge.

DLL hijacking has been a known threat for decades, with numerous high-profile incidents highlighting its effectiveness and persistence. This paper focuses exclusively on DLL hijacking, providing a comprehensive overview of its technical details, historical significance, and practical defenses.

2 Background

2.1 What are DLLs?

Dynamic Link Libraries (DLLs) are shared libraries in Windows that contain code and data used by multiple programs. They allow for modular programming, enabling applications to reuse code and reduce redundancy. DLLs are loaded into memory when an application requires their functionality, such as handling graphics, networking, or other system tasks.

2.2 Windows DLL Search Order

When an application needs to load a DLL, Windows follows a specific search order to locate it:

1. The directory from which the application loaded.
2. The system directory (e.g., C:\Windows\System32).
3. The 16-bit system directory (e.g., C:\Windows\System).
4. The Windows directory (e.g., C:\Windows).
5. The current working directory.
6. Directories listed in the PATH environment variable.

This search order is critical to understanding DLL hijacking, as attackers can exploit it by placing malicious DLLs in directories that are searched before the legitimate ones [11].

3 How DLL Hijacking Works

DLL hijacking exploits the DLL search order by placing a malicious DLL with the same name as a required DLL in a directory that is searched first. When the application loads, it unknowingly loads the malicious DLL instead of the legitimate one, executing the attacker's code.

3.1 Exploiting Search Order

Attackers place a malicious DLL in the application's directory or another directory in the search order. For example, if an application is launched from a user-writable directory, such as the Downloads folder, a malicious DLL placed there will be loaded before the legitimate one in the system directory [6].

3.2 Execution of Malicious Code

The malicious DLL can contain code that performs various malicious actions, such as stealing data, installing malware, or escalating privileges. This code runs within the context of the application, inheriting its permissions, which can amplify the attack's impact [2].

4 Real-World Examples

DLL hijacking has been used in numerous high-profile attacks, demonstrating its effectiveness and persistence. The following table summarizes key incidents:

4.1 Historical Incidents

- 2010 Windows Applications Vulnerability: Over 200 Windows applications, including Microsoft Office, VLC Media Player, and uTorrent, were vulnerable to DLL hijacking. Attackers could place malicious DLLs in directories like network shares or removable drives, leading to remote code execution when users opened files from these locations. Microsoft issued Security Advisory 2269637, recommending blocking network share attacks, and vendors patched their software to use absolute paths or safer loading methods [5].

Table 1: Notable DLL Hijacking Incidents

Year	Incident	Description
2010	Windows Applications Vulnerability	Over 200 applications, including Microsoft Office and VLC Media Player, were vulnerable, allowing remote code execution via malicious DLLs in untrusted directories [1].
2010	Stuxnet Worm	Exploited Siemens' WinCC/SCADA software to manipulate industrial systems [14].
2014	WinRAR Vulnerability	Malicious DLLs in compressed file directories enabled code execution [13].
2020	Zoom Client Vulnerability	Enabled privilege escalation via malicious DLLs in Zoom's directory [12].
2023	ToneShell Backdoor	Used DLL sideloading for persistence and data exfiltration [4].
2025	CVE-2025-24076 & CVE-2025-24994	DLL hijacking vulnerabilities in Windows 11's "Mobile devices" feature, allowing privilege escalation from low-privileged users to SYSTEM level within 300 milliseconds [15].

- Stuxnet 2010: The Stuxnet worm used DLL hijacking to target Siemens' WinCC/SCADA software. By replacing the legitimate s7otbxdx.dll with a malicious version, Stuxnet intercepted communications between WinCC and Siemens PLC devices, enabling it to modify code on PLC devices unnoticed and cause physical damage to centrifuges. This was addressed by Siemens in a 2011 patch [14].

- WinRAR 2014: WinRAR versions 5.30 and earlier were vulnerable to DLL hijacking, where attackers could place malicious DLLs (e.g., UXTheme.dll, RichEd32.dll) in the same directory as a compressed file, such as the user's Downloads folder. When WinRAR extracted the file, it loaded the malicious DLL, enabling code execution. This was patched in later versions [13].

- Zoom 2020: A DLL hijacking vulnerability (CVE-2020-9767) in Zoom Meeting versions prior to 5.1.4 allowed attackers to place malicious DLLs (e.g., SHCore.dll) in the Zoom application directory. If Zoom ran with elevated privileges, this could lead to privilege escalation. The vulnerability was fixed in version 5.1.4 [12].

- ToneShell Backdoor 2023: Used by the threat actor Stately Taurus (aka Mustang Panda), ToneShell employed DLL sideloading to install a backdoor. It consisted of three DLL components (persistence, networking, and functionality) loaded into

legitimate processes, enabling data exfiltration and command-and-control communication. This campaign targeted Southeast Asian governments [4].

4.2 Recent Vulnerabilities

In April 2025, two critical vulnerabilities related to DLL hijacking were disclosed in Windows 11's "Mobile devices" feature, which allows users to link their mobile phones to their Windows computer:

- CVE-2025-24076: Allows attackers to gain local system privileges from a low-privileged user by leveraging DLL hijacking. The vulnerability involves a system process loading a user-modifiable DLL (CrossDevice.Streaming.Source.dll) within a 300-millisecond window, enabling rapid privilege escalation. It was discovered through automated scans using tools like PrivescCheck and fixed in March 2025 [15].
- CVE-2025-24994: A related vulnerability where the user process fails to verify the loaded DLL, potentially enabling user-to-user attacks. This was also fixed in March 2025 [15].

These vulnerabilities, reported to Microsoft through their responsible disclosure program, highlight the ongoing relevance of DLL hijacking and the need for robust defenses [15].

5 Impact of DLL Hijacking

The impact of DLL hijacking can be severe:

- Privilege Escalation: If the application runs with elevated privileges, the malicious DLL inherits those privileges, allowing attackers to gain administrative access [8].
- Data Theft: Malicious DLLs can log sensitive information, such as passwords, session tokens, or proprietary data [1].
- Persistence: DLL hijacking can be used to install persistent malware or backdoors, ensuring long-term access to the compromised system [4].
- Lateral Movement: Attackers can use DLL hijacking to move laterally within a network, compromising additional systems [7].

6 Mitigation Strategies

To prevent DLL hijacking, a defense-in-depth approach is essential, combining developer practices, system configurations, and user precautions:

- Use Absolute Paths: Developers should specify the full path to DLLs (e.g., C:\Windows\System32
to bypass the default search order, ensuring only the intended DLL is loaded [11].
- Safe DLL Loading Functions: Use functions like LoadLibraryEx with flags such as LOAD_LIBRARY_SEARCH_SYSTEM32 to restrict loading to trusted directories, reducing the risk of hijacking [11].

- Code Signing: Digitally sign DLLs and verify signatures before loading to ensure authenticity, preventing the execution of unauthorized DLLs [6].
- Directory Permissions: Restrict write access to application directories (e.g., C:\Program Files) to prevent attackers from placing malicious DLLs, a critical lesson from the 2010 Windows applications vulnerability [1].
- Enable Safe DLL Search Mode: Set the registry key HKEY_LOCAL_MACHINE\System\Curren Manager\SafeDllSearchMode to 1 to exclude the current working directory from the search order [11].
- Application Whitelisting: Use tools like AppLocker to restrict which DLLs can be loaded, enhancing security by limiting unauthorized code execution [6].
- Monitoring and Auditing: Employ tools like Process Monitor to monitor real-time DLL loading and detect suspicious behavior, such as "NAME NOT FOUND" errors indicating potential hijacking attempts. Windows Event Logs can also be used for auditing [9].
- Detection Tools: Use specialized tools like DLLSpy, which employs static, dynamic, and recursive methods to identify vulnerable DLL loading, and PowerSploit functions (e.g., Find-ProcessDLLHijack, Find-PathDLLHijack) to automate vulnerability discovery [9, 8].
- Regular Updates: Keep software and operating systems updated to patch known vulnerabilities, as seen in the fixes for WinRAR, Zoom, and Windows 11's "Mobile devices" feature [10].

7 Future Trends

As Windows evolves, DLL hijacking is likely to remain a threat due to its low-noise nature and ability to bypass security controls by mimicking legitimate behavior. Emerging features, such as those in Windows 12, may introduce new vulnerabilities if not properly secured. Advances in detection tools, such as machine learning-based behavioral analytics, and stricter DLL loading policies could mitigate future risks. Organizations should stay informed about new attack vectors and adapt their defenses accordingly [16].

8 Conclusion

DLL hijacking remains a significant threat in the Windows ecosystem due to its ability to exploit the DLL search order for malicious purposes. Historical incidents like Stuxnet and recent vulnerabilities like CVE-2025-24076 demonstrate its impact. However, it can be effectively mitigated through careful development practices, system hardening, and user awareness. By understanding how DLL hijacking works and implementing the recommended strategies, organizations can protect themselves against this insidious attack vector. As cyber threats evolve, ongoing vigilance and adaptation of security practices will be crucial to staying ahead of attackers.

References

[1] UpGuard. (2025). What is DLL Hijacking? The Dangerous Windows Exploit. https://www.upguard.com/blog/dll-hijacking

[2] MITRE ATT&CK. (2020). Hijack Execution Flow: DLL, Sub-technique T1574.001. https://attack.mitre.org/techniques/T1574/001/

[3] PentestLab. (2017). DLL Hijacking. https://pentestlab.blog/2017/03/27/dll-hijacking/

[4] Unit 42. (2023). Cyberespionage Attacks Against Southeast Asian Government Linked to Stately Taurus. https://unit42.paloaltonetworks.com/stately-taurus-attacks-se-asian-government/

[5] Reuters. (2010). Windows DLL load hijacking exploits go wild. https://www.reuters.com/article/urnidgns852573c400693880000257789006d820c/windows-dll-load-hijacking-exploits-go-wild-idUS2168761020100825/

[6] Okta. (2024). DLL Hijacking Definition Tutorial & Prevention. https://www.okta.com/identity-101/dll-hijacking/

[7] CrowdStrike. (2022). 4 Ways Adversaries Hijack DLLs. https://www.crowdstrike.com/en-us/blog/4-ways-adversaries-hijack-dlls/

[8] HackTricks. (2024). Dll Hijacking. https://book.hacktricks.xyz/windows-hardening/windows-local-privilege-escalation/dll-hijacking

[9] CyberArk. (2019). DLLSpy – Tighten Your Defense by Discovering DLL Hijacking Easily. https://www.cyberark.com/resources/threat-research-blog/dllspy-tighten-your-defense-by-discovering-dll-hijacking-easily

[10] SecurityScorecard. (2025). What Is DLL Hijacking? Understanding and Preventing the Threat. https://securityscorecard.com/blog/what-is-dll-hijacking-understanding-and-preventing-the-threat/

[11] Microsoft. (2023). Dynamic-link library redirection. https://docs.microsoft.com/en-us/windows/win32/dlls/dynamic-link-library-redirection

[12] GitHub. (2020). Zoom-dll-hijacking: A dll hijacking vulnerability in zoom meeting $< 5.1.4$. CVE-2020-9767. https://github.com/shubham0d/Zoom-dll-hijacking

[13] Packet Storm. (2014). WinRAR 5.30 DLL Hijacking. https://packetstormsecurity.com/files/135665/WinRAR-5.30-DLL-Hijacking.html

[14] CISA. (2013). Siemens SIMATIC STEP 7 DLL Vulnerability. https://www.cisa.gov/news-events/ics-advisories/icsa-12-205-02

[15] Compass Security Blog. (2025). 300 Milliseconds to Admin: Mastering DLL Hijacking and Hooking to Win the Race (CVE-2025-24076 and CVE-2025-24994). https://blog.compass-security.com/2025/04/3-milliseconds-to-admin-mastering-dll-hijacking-and-hooking-to-win-the-race-cve-2025-24076-and-

[16] Unit 42. (2024). Intruders in the Library: Exploring DLL Hijacking. https://unit42.paloaltonetworks.com/dll-hijacking-techniques/