Windows Antivirus Evasion – Part 1

redfoxsec.com/blog/windows-antivirus-evasion-part-1

Joseph Simon December 12, 2023

- December 12, 2023
- Red Team
- Joseph Simon

In today's digital landscape, protecting our systems from malicious threats is of utmost importance. Antivirus software plays a significant role in defending against various forms of malware.

However, cybercriminals are constantly evolving their techniques to bypass these security measures. One such method in Windows Antivirus evasion is DLL Injection, which involves manipulating a process to load and execute an arbitrary DLL for malicious purposes. In this blog, we will delve into the world of Windows antivirus evasion, exploring the techniques used, the benefits and challenges of dynamic link libraries, and best practices for protecting your system.

Understanding Dynamic Link Libraries (DLLs)

What is a DLL?

A dynamic link library (DLL) is a collection of small programs that larger programs can load when needed to complete specific tasks. DLL files contain instructions that help the larger program handle functions that may not be core to its operation. For example, DLLs can enable communication with devices such as printers or scanners. DLLs can also contain resources like images or classes that the larger program can utilize.

DLL Injection

DLL Injection is a technique that manipulates how a process operates by running a DLL file within its address space. This allows for the execution of arbitrary commands on a target system. DLL Injection attacks are often employed for reverse engineering purposes or to gain unauthorized access to a system by increasing rights and authority. The injection process involves several steps, including placing the DLL file in the target system, allocating memory within the target process, copying the DLL file's path to the process memory, executing the DLL file within the process, and initiating its processes.

How DLL Injection Works ?

DLL Injection involves several key steps to successfully inject a DLL into a target process. The process begins with the placement of the DLL file in the target system. Once the DLL file is present, space is allocated within the target process's memory to store the path of the DLL file. The directory path is then copied to the process memory, and appropriate memory addresses are determined. Finally, the DLL file is executed within the process, allowing it to perform its intended processes.

Benefits of Dynamic Link Libraries

Dynamic Link Libraries offer several advantages in terms of system performance, memory efficiency, and modular architecture.

1. **Fewer Faults**

By avoiding repeated retrieval of DLLs from disk, DLLs can help prevent runtime errors. Additionally, multiple processes can share the same page of RAM, reducing the chances of errors in the paging process. This memory efficiency leads to faster program execution and reduced disk space usage.

2. **Memory Efficiency**

DLLs allow operating systems and programs to run faster and utilize memory more efficiently. Unlike static linking, where libraries are embedded into individual executables, DLLs are loaded only when needed. This saves memory space as DLL files are not loaded alongside the main program. Instead, they are loaded into memory just once and can be shared by multiple programs simultaneously.

3. **Modular Architecture**

DLLs contribute to a modular architecture, enabling the delivery of programs as separate components. Developers can exchange specific libraries and components without rewriting the entire application. This modular approach facilitates collaboration with other programs, simplifies software updates, and enhances system flexibility.

4. **Challenges of Dynamic Link Libraries**

While dynamic link libraries offer numerous benefits, there are also challenges associated with their use.

DLL Errors in Windows Antivirus Evasion

DLL errors occur when an application cannot find a required DLL file. This can happen due to file corruption, deletion, or incorrect installation. Resolving DLL errors often involves:

- Troubleshooting specific issues, such as restarting the computer.
- Checking the recycle bin for deleted files.
- Performing a system restore from a previous backup.

Exploits

DLL hijacking is a common exploit used by threat actors to compromise Windows applications. By replacing a legitimate DLL file with a malicious one, hackers can gain unauthorized access or deliver malware to target systems. Preventing DLL exploits requires robust security measures, including regular software updates and implementing behaviour-based detection techniques.

Speed Considerations

Dynamic linking can introduce slight performance overhead compared to static linking due to the additional runtime or load time required to link DLLs. However, modern operating systems are optimized to handle dynamic linking efficiently, resulting in minimal impact on overall system performance.

DLL Injection Techniques for Windows Antivirus

Evasion

DLL Injection techniques are employed to evade antivirus detection and execute malicious code within target processes. Here are some commonly used techniques:

Direct DLL Injection

Direct DLL Injection involves injecting a DLL directly into the target process’s memory space using functions such as LoadLibrary or CreateRemoteThread.

Process Hollowing

Process Hollowing involves:

- Creating a new process in a suspended state.
- Replacing its legitimate code with malicious code.
- Resuming its execution.

With the help of this technique allows the attacker to execute malicious code within a legitimate process, evading antivirus detection.

Reflective DLL Injection

The reflective DLL Injection technique allows the injection of a DLL without writing it to the disk. The DLL is loaded directly into memory and executed using reflective loading techniques. This technique makes it difficult for antivirus software to detect the injected DLL.

DLL Side-Loading

DLL Side-Loading exploits the way some applications load DLL files. By placing a malicious DLL file in a location where a legitimate DLL is expected to be loaded, the attacker can trick the application into loading the malicious DLL instead. This technique takes advantage of insecure DLL loading behaviour in certain applications.
Evading Antivirus Detection

To successfully evade antivirus detection, attackers employ various techniques to obfuscate their malicious code. Here are some commonly used methods:

1. **Polymorphic Code**

Polymorphic code is a technique where the attacker constantly modifies the code's structure, making it difficult to detect using signature-based antivirus solutions. Each instance of the malware appears unique, evading traditional detection methods.

Code Obfuscation

In software development, making code confusing and hard to understand is called "obfuscation." Just like in language, it can use overly complicated ways to say things. Programmers might do this on purpose for a few reasons:

- **To hide what the code is for:**Like a secret code, it's harder for others to figure out what the program does.
- **To protect the logic:**Keeping the inner workings hidden makes it harder to steal or copy ideas.
- **To make it a puzzle:**For some, it's a fun challenge to write and read obfuscated code.

2. **Anti-Emulation Techniques**

In Windows, Antivirus software often relies on emulating code to detect malware. To evade this detection, attackers employ anti-emulation techniques that detect when their code is being run in an emulated environment. They may introduce delays, employ anti-debugging techniques, or use conditional branching to confuse emulators.

3. **Encryption and Packaging**

Attackers may encrypt their malicious code and package it within seemingly harmless files or documents. This technique ensures that the malicious code remains undetected until it is decrypted and executed on the target system.

4. **Undocumented Windows**

Undocumented Windows APIs are functions within the Windows operating system not officially documented by Microsoft. Discovered through reverse engineering, they pose risks such as compatibility issues, security vulnerabilities, and legal concerns. Relying on them can lead to unstable applications and hinder long-term support. Developers are advised to use only officially documented APIs for stability and security.

Best Practices

Safeguarding your system from Windows antivirus evasion demands a proactive strategy and commitment to implementing best practices.

1. **Regularly Update Security Software**

Keeping your antivirus software updated is crucial for detecting and preventing the latest malware threats. Consistent updates guarantee that your security software possesses the latest virus definitions and detection capabilities.

2. Implement Behaviour-Based Detection

Behaviour-based detection techniques focus on analysing the behaviour of programs rather than relying solely on signature-based detection. By monitoring for suspicious activities and behavioural deviations, antivirus software can detect previously unknown threats.

3. **Utilize Sandboxing Techniques**

Sandboxing involves running potentially malicious programs in isolated environments to prevent them from affecting the underlying system. Sandbox solutions provide an additional layer of defence by containing and analysing potentially dangerous executables in a controlled environment.

Future Trends

As technology advances, new trends and techniques in Windows antivirus evasion continue to emerge. Here are some areas to watch in the future:

1. **Machine Learning and AI-Based Detection**

Antivirus software is incorporating machine learning and artificial intelligence (AI) at an escalating rate for the detection and prevention of emerging threats. These advanced technologies enable the analysis of extensive data sets, pattern recognition, and the ability to adapt to evolving attack vectors.

2. **Zero-Day Exploits and Advanced Persistent Threats**

Zero-day exploits refer to vulnerabilities that are unknown to software vendors. Attackers leverage these vulnerabilities to carry out targeted attacks. Advanced persistent threats (APTs) are sophisticated attacks that persistently target specific organizations or individuals. Detecting and mitigating such threats requires a combination of advanced detection techniques and proactive security measures.

3. **Implanting Malicious DLLs in Firmware**

As attackers explore new avenues for evasion, there is a growing concern regarding the implantation of malicious DLLs in firmware. Firmware-level attacks can have far-reaching consequences and pose significant challenges for detection and mitigation.

TL;DR

Windows antivirus evasion through DLL injection poses a significant threat to system security. Understanding the tactics employed by attackers and integrating best practices for defence is vital in ensuring the security of your systems.

By staying vigilant, regularly updating security software, and employing behaviour-based detection techniques, you can significantly reduce the risk of falling victim to DLL injection attacks.

Redfox Security is a diverse network of expert security consultants with a global mindset and a collaborative culture. If you are looking to improve your organization's security posture, **contact us** today to discuss your security testing needs. Our team of security professionals can help you **identify vulnerabilities and weaknesses in your systems and provide recommendations to remediate them.**

"Join us on our journey of growth and development by signing up for our comprehensive **courses.**"

PreviousUnderstanding the Pen Test Program Life Cycle
NextUnderstanding XML External Entity Injection (XXE) Attacks

Recent Blog

September 09, 2025
Is APK Decompilation Legal? What You Need To Know
September 06, 2025
When Hackers Hit the Road: The Jaguar Land Rover Cyberattack
September 05, 2025
This Is the Hacker's Swiss Army Knife. Have You Heard About It?