OVERVIEW ABOUT LOADER

Bui Viet Dung

June 26, 2025

1 Intensive definition of Malware Loader

In modern context, a Malware Loader is defined as a specialized malicious code with the main purpose of downloading and executing one or more malicious parts into the memory of the target system.The core function of Loader is as a means of stealth transportation, creating a safe bridge between the initial penetration stage and the stage of executing the main toxic behavior, such as deploying ransomware, trojan stealing information (infostealer), or a remote control backdoor (SV).

The basic difference of modern loader compared to previous generations lies in the fact that they are designed to operate mainly or completely in memory.This trend is a direct response to the development of anti -virus solutions (AV) and Endpoint detection and response (EDR) traditional, which focuses strongly on scanning and analyzing files on disks.By avoiding writing the final Payload on the disk , the loaders can perform ”filess attack ” attacks, significantly reducing traces and the possibility of being detected.

The operation of a typical loader consists of a series of carefully staged actions: a new memory, decoding or decompression of the hidden Payload, dynamic resolution of the necessary API addresses to avoid static detection, and finally, transfer the execution right for that Payload.

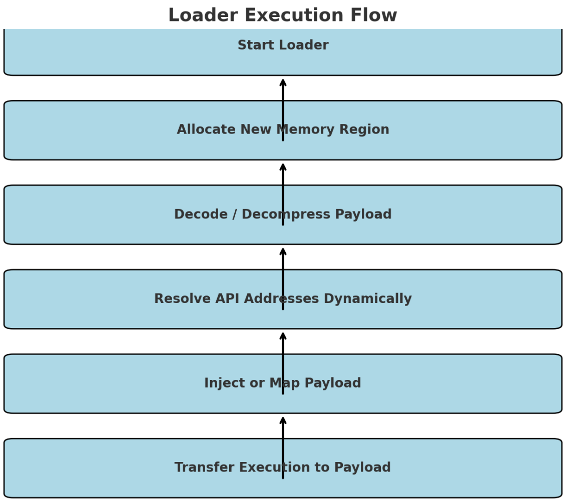

Loader Execution Flow
Start Loader
Allocate New Memory Region
Decode / Decompress Payload
Resolve API Addresses Dynamically
Inject or Map Payload
Transfer Execution to Payload

Figure 1: Normal excution flow of loader

2 The Role of Loaders in the Malware Execution Chain

Loaders play a pivotal role in the malware lifecycle by acting as a staging mechanism to deliver, unpack, and execute the main payload while minimizing detection. Typically positioned after initial access but before the main malware activation, the loader orchestrates multiple tasks essential for successful infection and persistence. This section outlines the core functionalities of a loader in a modern malware execution chain.

1. Payload Decryption and Unpacking

Malware payloads are often stored in encrypted or obfuscated formats to evade static detection by antivirus (AV) and endpoint detection and response (EDR) systems. Loaders are responsible for decoding these payloads using symmetric ciphers (e.g., AES, RC4), XOR encoding, or custom obfuscation routines before deployment.

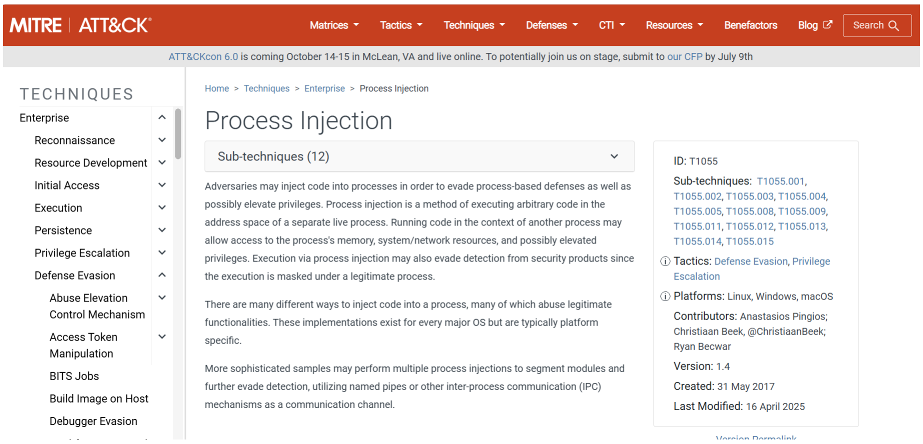
2. Remote Retrieval and Staging

Loaders may retrieve the final payload from external command-and-control (C2) infrastructure. This minimizes the size and footprint of the initial dropper. Staging mechanisms include HTTP/HTTPS, DNS tunneling, cloud services (e.g., Dropbox, GitHub), and peer-to-peer (P2P) delivery.

3. Memory Injection Techniques

To avoid leaving artifacts on disk, loaders use in-memory execution via process injection techniques such as:

- *Process Hollowing*
- *DLL Injection* and *Reflective DLL Injection*
- *Thread Hijacking* and *QueueUserAPC*
- *Process Doppelgänging* and *Transacted Hollowing*
- *Early Bird Injection*



ATT&CKcon 6.0 is coming October 14-15 in McLean, VA and live online. To potentially join us on stage, submit to our CFP by July 9th

Process Injection

Adversaries may inject code into processes in order to evade process-based defenses as well as possibly elevate privileges. Process injection is a method of executing arbitrary code in the address space of a separate live process. Running code in the context of another process may allow access to the process's memory, system/network resources, and possibly elevated privileges. Execution via process injection may also evade detection from security products since the execution is masked under a legitimate process.

There are many different ways to inject code into a process, many of which abuse legitimate functionalities. These implementations exist for every major OS but are typically platform specific.

More sophisticated samples may perform multiple process injections to segment modules and further evade detection, utilizing named pipes or other inter-process communication (IPC) mechanisms as a communication channel.

ID: T1055
Sub-techniques: T1055.001, T1055.002, T1055.003, T1055.004, T1055.005, T1055.008, T1055.009, T1055.011, T1055.012, T1055.013, T1055.014, T1055.015
Tactics: Defense Evasion, Privilege Escalation
Platforms: Linux, Windows, macOS
Contributors: Anastasios Pingios; Christiaan Beek, @ChristiaanBeek; Ryan Becwar
Version: 1.4
Created: 31 May 2017
Last Modified: 16 April 2025

Figure 2: https://attack.mitre.org/techniques/T1055/

These allow the payload to be executed in the address space of a legitimate process.

4. Execution of the Final Payload

Once injected or mapped into memory, the loader initiates the execution of the payload. This may be achieved via low-level APIs such as `NtCreateThreadEx`, `CreateRemoteThread`, or by triggering callbacks. Some loaders use direct or indirect syscalls to bypass user-mode hooks.

5. Privilege Escalation (Optional)

Some advanced loaders attempt to elevate privileges by exploiting system misconfigurations or local vulnerabilities. Techniques include token stealing, UAC bypass (e.g., `fodhelper.exe`, `eventvwr.exe`), or even kernel exploits if required by the payload.

6. Establishing Persistence

To ensure continued access after system reboot, loaders may configure persistence using:

- Windows Registry keys (`Run`, `RunOnce`)
- Scheduled tasks and services

- Startup folder shortcuts
- WMI event subscriptions
- DLL hijacking or COM hijacking

7. Anti-Forensics and Cleaning

Post-deployment, loaders often delete themselves from disk and memory to hinder forensic analysis. Techniques include:

- `MoveFileEx` with `MOVEFILE_DELAY_UNTIL_REBOOT`
- Overwriting PE headers in memory
- Clearing event logs or prefetch files

8. Anti-Analysis and Stealth

To avoid dynamic analysis, loaders may implement:

- **Anti-Debugging**: Using APIs like `IsDebuggerPresent`, `NtQueryInformationProcess`
- **Anti-VM Detection**: Checking CPUID, MAC address, registry keys
- **Obfuscation**: API hashing, control flow flattening
- **Indirect Syscalls**: To bypass userland hooks

9. Self-Update or Re-Drop Capability

Advanced loaders can download and replace themselves with updated variants or redrop new payloads to adapt to evolving detection mechanisms.

3 Deep dive into techniques

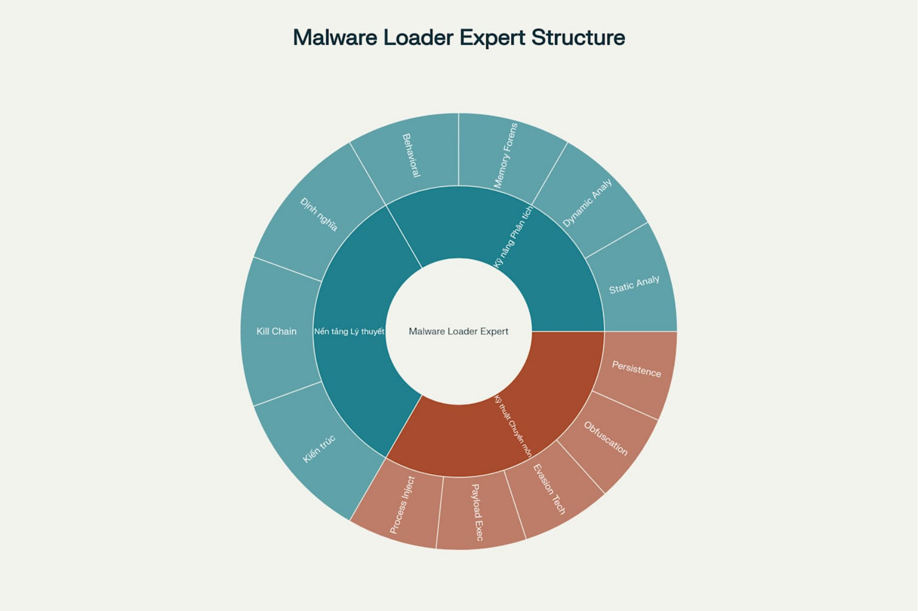


Figure 3: Enter Caption

https://www.elastic.co/security-labs/qbot-malware-analysis