Reflective Code Loading, Technique T1620 - Enterprise

Archived: 2026-04-05 13:44:37 UTC

Adversaries may reflectively load code into a process in order to conceal the execution of malicious payloads. Reflective loading involves allocating then executing payloads directly within the memory of the process, vice creating a thread or process backed by a file path on disk (e.g., Shared Modules).

Reflectively loaded payloads may be compiled binaries, anonymous files (only present in RAM), or just snubs of fileless executable code (ex: position-independent shellcode).[1][2][3][4][5] For example, the `Assembly.Load()` method executed by PowerShell may be abused to load raw code into the running process.[6]

Reflective code injection is very similar to Process Injection except that the "injection" loads code into the processes' own memory instead of that of a separate process. Reflective loading may evade process-based detections since the execution of the arbitrary code may be masked within a legitimate or otherwise benign process. Reflectively loading payloads directly into memory may also avoid creating files or other artifacts on disk, while also enabling malware to keep these payloads encrypted (or otherwise obfuscated) until execution.[3][4][7][8]

Source: https://attack.mitre.org/techniques/T1620