Process Injection: Process Doppelgänging, Sub-technique T1055.013 - Enterprise

Archived: 2026-04-05 17:37:12 UTC

Adversaries may inject malicious code into process via process doppelgänging in order to evade process-based defenses as well as possibly elevate privileges. Process doppelgänging is a method of executing arbitrary code in the address space of a separate live process.

Windows Transactional NTFS (TxF) was introduced in Vista as a method to perform safe file operations. [1] To ensure data integrity, TxF enables only one transacted handle to write to a file at a given time. Until the write handle transaction is terminated, all other handles are isolated from the writer and may only read the committed version of the file that existed at the time the handle was opened. [2] To avoid corruption, TxF performs an automatic rollback if the system or application fails during a write transaction. [3]

Although deprecated, the TxF application programming interface (API) is still enabled as of Windows 10. [4]

Adversaries may abuse TxF to a perform a file-less variation of Process Injection. Similar to Process Hollowing, process doppelgänging involves replacing the memory of a legitimate process, enabling the veiled execution of malicious code that may evade defenses and detection. Process doppelgänging's use of TxF also avoids the use of highly-monitored API functions such as `NtUnmapViewOfSection`, `VirtualProtectEx`, and `SetThreadContext`. [4]

Process Doppelgänging is implemented in 4 steps [4]:

- Transact – Create a TxF transaction using a legitimate executable then overwrite the file with malicious code. These changes will be isolated and only visible within the context of the transaction.
- Load – Create a shared section of memory and load the malicious executable.
- Rollback – Undo changes to original executable, effectively removing malicious code from the file system.
- Animate – Create a process from the tainted section of memory and initiate execution.

This behavior will likely not result in elevated privileges since the injected process was spawned from (and thus inherits the security context) of the injecting process. However, execution via process doppelgänging may evade detection from security products since the execution is masked under a legitimate process.
