

Malware should not decide what the analyst is allowed to observe.

Problem & Motivation

A single run is quickly thwarted by the paths malware chooses to traverse and may fail against advanced timing checks or environmental fingerprinting. Malicious functionality may remain hidden in unfollowed code paths.

BEAST: Big Picture Overview

- Malware analysis framework with **complete code coverage** that overcomes anti-debugging.
- BEAST identifies and disables anti-analysis checks at runtime and implements complete code coverage to restart and revisit unfollowed code paths when malware terminates.

Anti-Analysis Neutralization

Goal: thoroughly fool malware regardless of anti-debugging, anti-emulation, or anti-VM techniques.

Technique family	Examples of checks	BEAST response
Debugger APIs and memory flags	Debugger queries, debug registers, PEB/heap flags, debug objects and self-debugging	Intercepts queries and adjusts process state to make it readily apparent it is not in a debug environment.
Exception-based control flow	INT 3, INT 2D, trap flag and guard-page faults	Runs VEH, SEH, VCH and unhandled-exception logic, following code hidden behind triggered exceptions.
Timing checks and delayed execution	Time APIs, RDTSC/RDTSCP and KUSER_SHARED_DATA	Uses one virtual clock across processes. Sleep and waits advance emulated time without real-world delays.
Anti-VM and hardware fingerprinting	CPUID, firmware, CPU count, RAM, disks, network adapters and MAC addresses	A configurable machine profile supplies plausible system information without recognizable VM artifacts.
Process, service and user checks	Process ancestry, services, windows, user and computer names, monitors and input	Simulates normal system and user activity so enumeration checks encounter a working Windows environment.
Anti-disassembly	Misleading disassembly and convoluted control flow	Logs instructions actually executed, allowing analysts to compare execution with static disassembly.

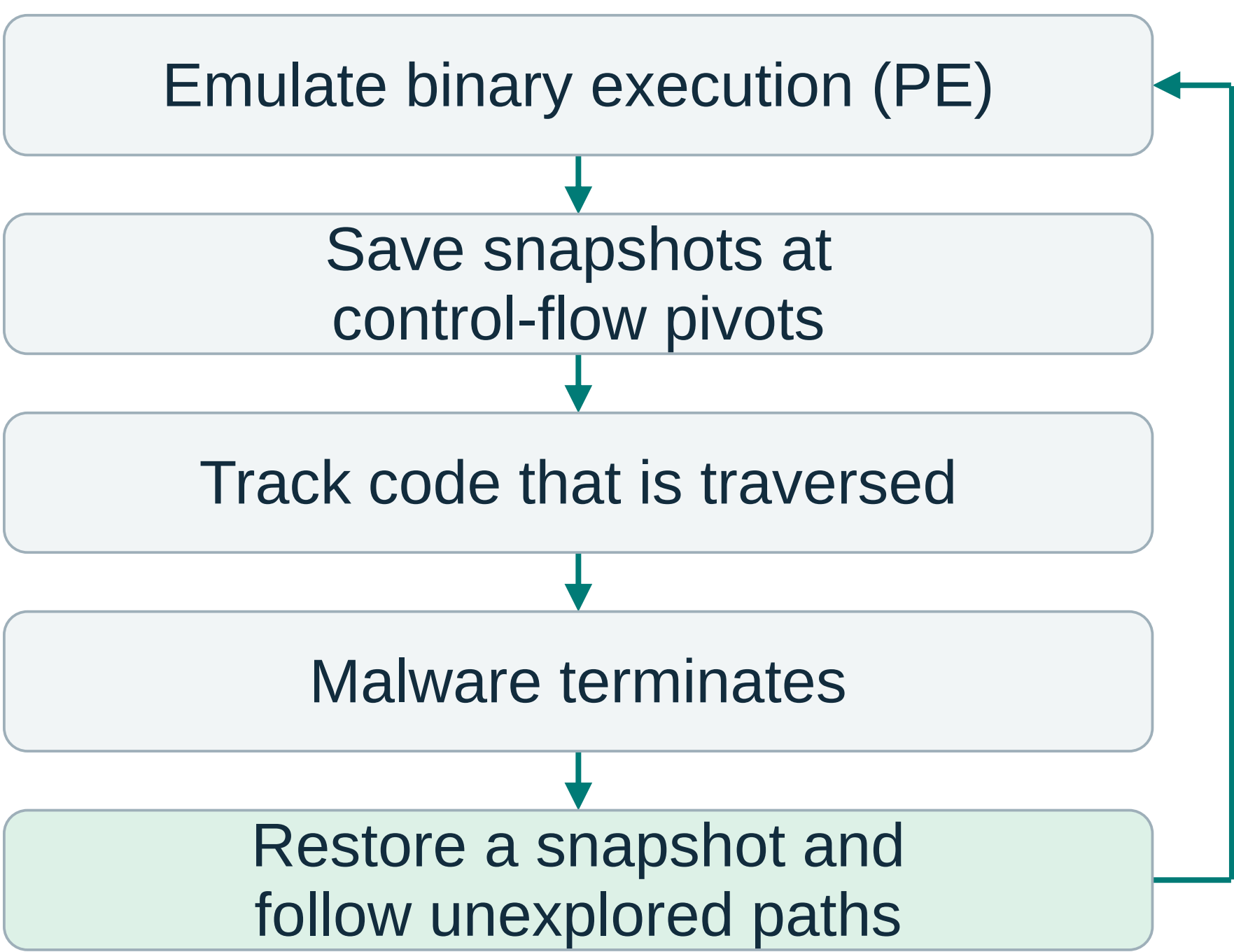
Complete Code Coverage

- Captures register states and memory at all control-flow pivots. All traversed code is tracked, logging followed code paths.
- Revisits unfollowed code paths after termination, restarting execution. The goal is to traverse all paths, or nearly all, exposing malicious functionality.



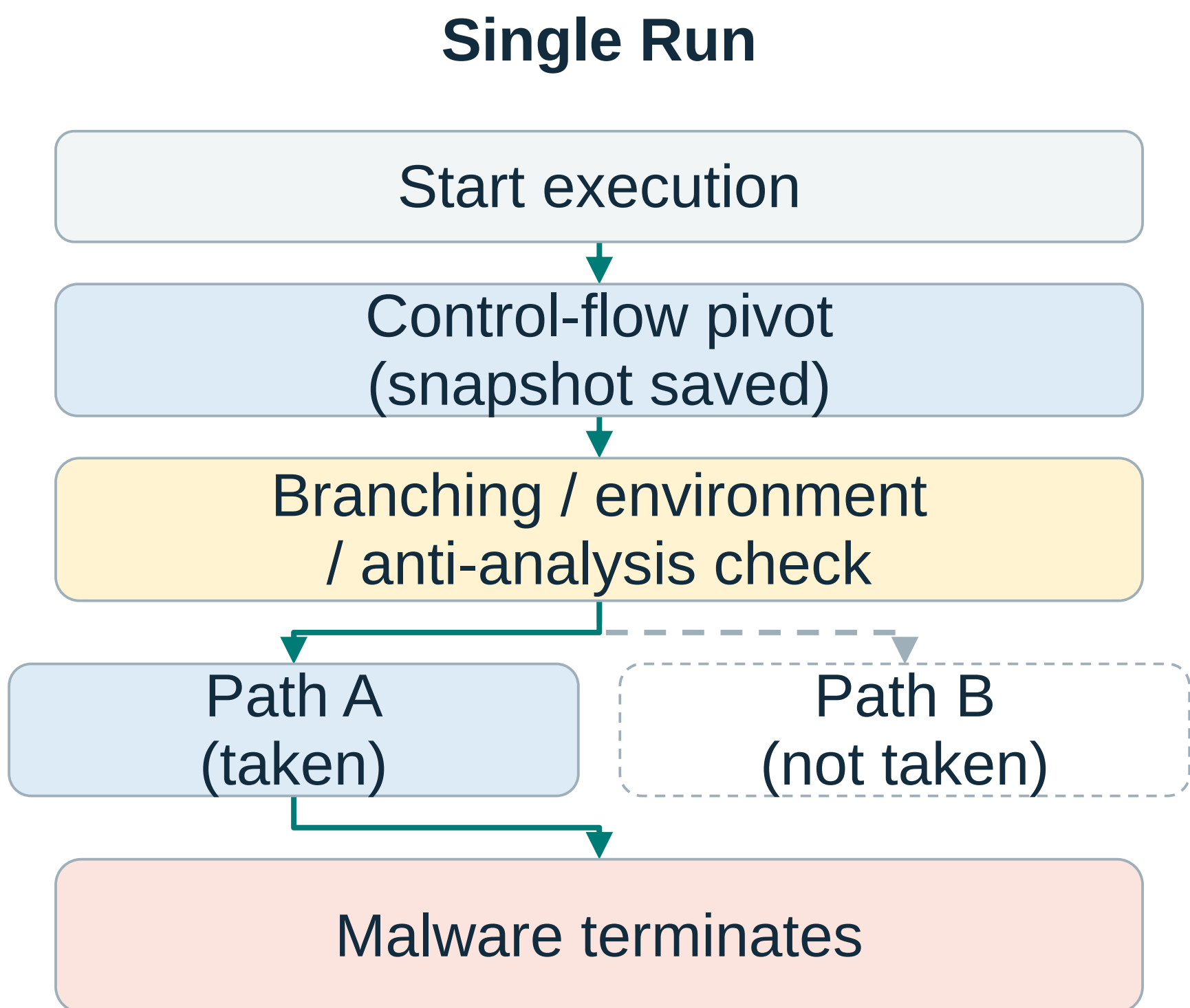
QR Code

Snapshot and Revisit Cycle

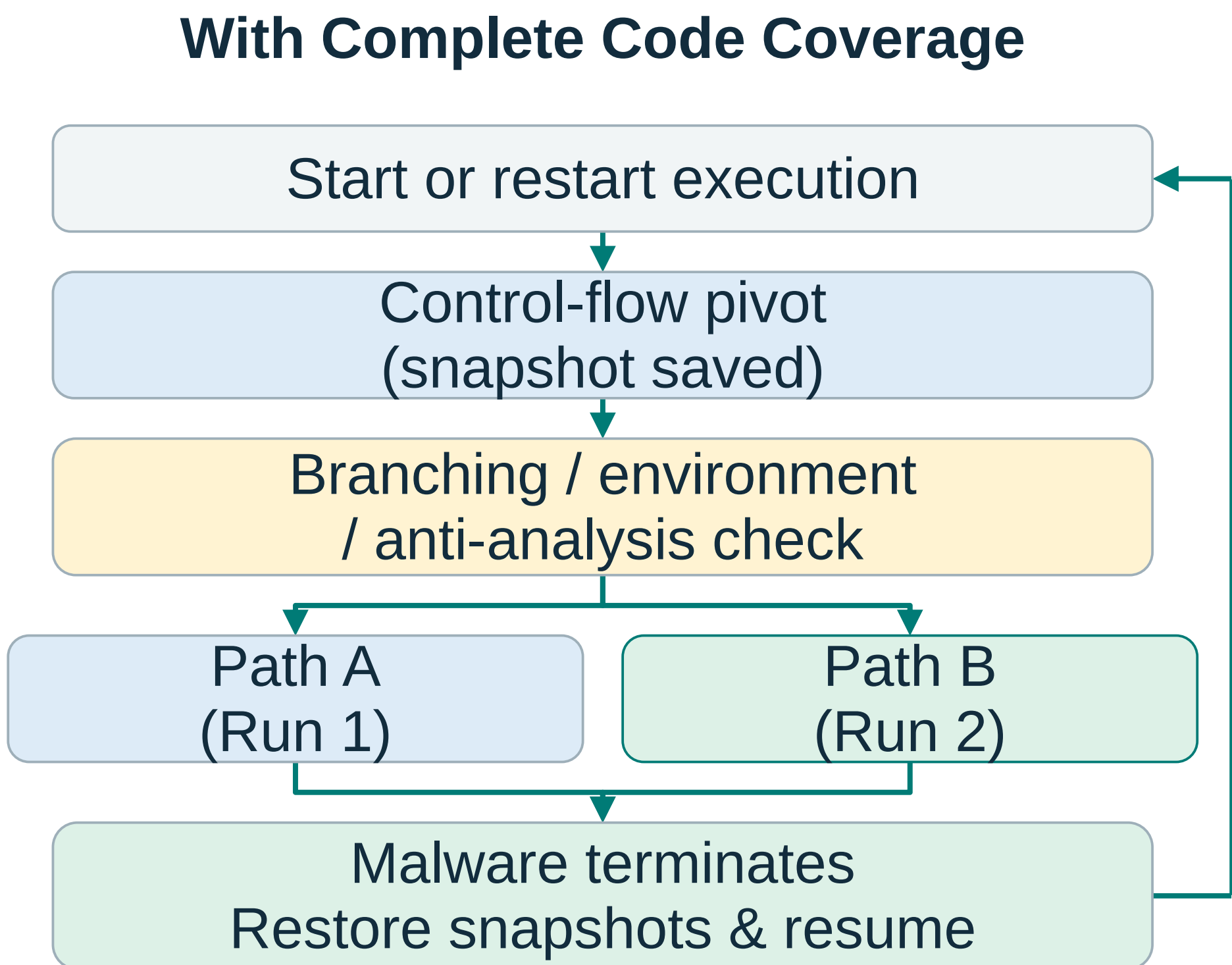


Repeat until paths are exhausted

Traversing All Paths to Find Malicious Functionality



Only the paths malware chooses are observed



Revisits unfollowed paths to expose malicious functionality

Other Noteworthy Support

Child and Multiple Processes • Separate memory, modules and thread state. • Shared virtual time and anti-analysis hooks extend analysis into child processes.	Process Injection • Remote memory, remote threads, APCs and Windows hooks. • Logs follow injected code in the target process.	Resources and DLLs • Loads and updates PE resources with numerous APIs. • Resolves all DLL dependencies and runs initialization code, including TLS callbacks.
--	--	---