PLATFORM ENGINEERING

OS Independence: Inspecting DLLs Without Trusting the OS

To protect the host, the defense mechanism must observe the host from a higher plane of privilege, rather than operating as a vulnerable peer within the user-space.

Dennis Underwood, Founder & CEO, Cyber Crucible · November 2026

EXECUTIVE SUMMARY

It is insufficient to merely verify a program's signature at execution. Cyber Crucible continuously monitors the active RAM to conduct memory integrity checks on all loaded supporting libraries. By tracking parent-child relationships and utilizing Control-Flow Integrity (CFI), the engine detects the exact millisecond a trusted program's execution flow is hijacked by an untrusted, side-loaded module.

01 Executive Briefing: The Path to OS Independence

To defeat the sophisticated in-memory manipulation and DLL tradecraft exhibited by actors like MUDDYWATER, defense architectures must abandon reactive, OS-reliant models. Cyber Crucible has engineered a proprietary, patented[1] architecture that decouples its operations from the underlying Windows OS libraries.[2]

The following 10-part executive presentation details the architectural challenges presented by modern adversaries and the subsequent engineering innovations that resulted in Cyber Crucible's true OS independence.

Module 1: The Core Challenge – The Dependency Trap

The fundamental flaw in modern cybersecurity is structural. Almost every security tool relies on the exact same Windows OS DLLs it aims to protect.[3] This creates a shared fate: when the OS is blind, buggy, or under attack, the security tool is blinded and impacts system performance. Cyber Crucible recognized that to protect the host, the defense mechanism must observe the host from a higher plane of privilege, rather than operating as a vulnerable peer within the user-space.

Module 2: Weaponizing System Stability (The EDR Killer)

Client instability can result from a combination of OS bugs, kernel-level zero-days, and intentional EDR killer system disruption attacks.[3] Attackers intentionally manipulate security tool listener injections and in-memory capabilities to degrade system performance.[3] Cyber Crucible's foundational engineering mandate is to provide full, autonomous preventative capability without increasing additional load on the system during these high-stress events. The defense must remain lightweight and invisible, even when the OS is actively failing.

Module 3: The Philosophy of OS Independence

To break the Dependency Trap, Cyber Crucible embarked on a multi-year engineering initiative to sever reliance on Microsoft-provided data buses and APIs. By investing heavily in proprietary, kernel-level sensor development, Cyber Crucible achieved an engineering moat: the system is almost completely independent of the OS.[2] This ensures that even if an attacker successfully compromises a System32 DLL using the tradecraft seen in Operation Olalampo, the Cyber Crucible analytics engine remains fully operational, untampered, and highly lethal to the adversary.

Module 4: Real-Time DLL Live Inspection

To counter the precise DLL side-loading techniques utilized by MUDDYWATER (e.g., sentinelmemoryscanner.exe loading sentinelagentcore.dll), Cyber Crucible implemented continuous, real-time DLL live inspection.[2] It is insufficient to merely verify a program's signature at execution. Cyber Crucible continuously monitors the active RAM to conduct memory integrity checks on all loaded supporting libraries.[2] By tracking parent-child relationships and utilizing Control-Flow Integrity (CFI), the engine detects the exact millisecond a trusted program's execution flow is hijacked by an untrusted, side-loaded module.[2]

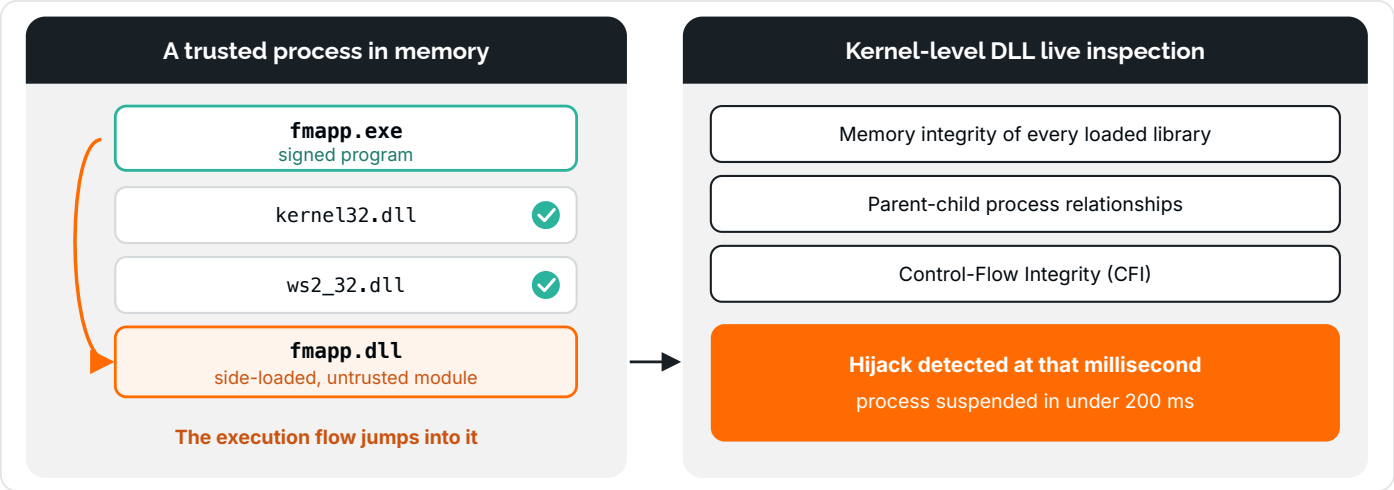

Figure 1. Real-time DLL live inspection: checking what a signed program loads, not only its signature

Module 5: Engineering the Unfair Advantage

Released in November 2025, Cyber Crucible Core version 4.5.1.0 marked a significant milestone in OS decoupling. Recognizing that Windows SSL and HTTP libraries are frequent targets for exploitation and manipulation, Cyber Crucible severed its reliance on them. The platform transitioned to utilizing OpenSSL for all SSL encryption/decryption processes, and implemented libcurl to entirely bypass the native Windows HTTP libraries. This ensures that critical telemetry and C2 communications remain secure and functional even if the host's networking stack is compromised.

Module 6: Precision Kernel Cryptography

By January 2026 (Version 4.5.1.2), Cyber Crucible had successfully deployed custom, kernel-level cryptography libraries developed directly from published Microsoft RFCs on Authenticode. Initially, native Microsoft libraries were utilized simultaneously to double-check mathematical calculations and transmit deviations to Cyber Crucible servers for error checking. After 10 months of zero observed calculation errors, it was determined that the native Microsoft libraries were actually causing unacceptable system instability under heavy load. Consequently, the reliance on Microsoft backup-calculations was entirely dropped, relying solely on Cyber Crucible's optimized kernel cryptography.

Module 7: Severing the Metadata Umbilical

In February 2026, subsequent updates dramatically advanced the OS independence initiative. Version 4.5.2.1 successfully removed dependency on multiple data bus and metadata production kernel APIs. Shortly thereafter, Version 4.5.2.2 removed additional dependencies on tertiary Windows APIs. At this stage, the Cyber Crucible system achieved near-total independence from the underlying operating system, retaining only a few highly specific OS dependencies necessary for basic hardware interaction.

Module 8: The Cyber Crucible Core Evolution

April 2026 saw the release of the Cyber Crucible Core (Version 4.5.3.0). This update introduced configurable, asynchronous DLL analytics designed specifically for machines with minimal hardware resources, ensuring robust defense without performance degradation. Furthermore, process creation and process injection analytics and telemetry were upgraded to utilize disk persistence. Crucially, this version entirely removed dependencies on Windows MiniFilters for these analytics, eliminating a major vector for kernel-level interference.

Module 9: Kernel Buffer Operation Enhancements

The Cyber Crucible kernel driver leverages three primary buffers to temporarily store telemetry before it is transmitted to the customer dashboard (distinct from local behavioral model operations). If a kernel buffer overloads due to a massive OS bug or a concentrated attack on the system, CPU load can spike. To mitigate this, Version 4.5.3.2 introduced critical enhancements: all three primary buffers' usage percentages are now continuously transmitted to Cyber Crucible infrastructure for metrics gathering. Administrators can remotely adjust the size of these buffers, and an automated customer notification system for very high buffer usage will be fully available by June 1, 2026. This guarantees Cyber Crucible can manage system load dynamically during EDR killer attacks.

Module 10: The Final Frontier – Network Stack Independence Roadmap

The final, remaining vulnerability involves the ws2_32.dll (Winsock) dependency. MUDDYWATER and similar actors have been observed launching targeted attacks designed to block telemetry from being transmitted by Network Interface Cards (NICs), effectively shutting down cloud-dependent MDR visibility. The short-term mitigation provided by Cyber Crucible is a "Partially offline" notification alert. However, the long-term, definitive mitigation is the total removal of the ws2_32.dll dependency, scheduled for release by August 2026. This will render Cyber Crucible completely immune to network stack disruption attacks.

02 Autonomous Kernel-Level Prevention in Action

The culmination of OS independence, real-time DLL live inspection, and advanced memory analytics is a defense platform that does not react, but prevents. Cyber Crucible

replaces the latency of cloud-based analysis with a patented,[1] edge-computing analytics engine powered by Genetic AI.[2]

Unlike Large Language Models (LLMs) or probabilistic AI that analyze text patterns to guess an outcome, Cyber Crucible's Genetic AI uses massive, deeply technical variable sets to produce definitive, deterministic outcomes.[4] Because these models are translated into raw machine language and execute locally within the kernel of the endpoint, they operate with absolute autonomy and zero network latency.[4]

When a malicious behavior is initiated—such as a side-loaded DLL attempting to utilize Direct Syscalls to bypass App-Bound Encryption and steal Chrome session tokens—the Genetic AI evaluates the intent of the data access in a fraction of a second.[2] If malicious intent is detected, the engine autonomously intercepts and suspends the offending process in under 200 milliseconds.[2]

This immediate, localized suspension freezes the attacker and the compromised process in memory, preventing data exfiltration or encryption while seamlessly preserving the memory state for subsequent forensic root-cause analysis.[4] Operations continue without downtime, disruption, or the need for a system reboot.[4]

Capability	Legacy EDR/XDR	Cyber Crucible
Operating architecture	User-space telemetry, highly OS-dependent	**Kernel-level sensors, OS-independent**
Decision location	Cloud-based analytics (high latency)	**Edge-native, on-endpoint processing**
Response time	Minutes to hours (network + human delay)	**Under 200 milliseconds (fully autonomous)**
Detection methodology	Signatures, basic heuristics, probabilistic AI	**Deterministic Genetic AI, behavioral intent**
DLL / memory defense	Blind to advanced in-memory library manipulation	**Continuous DLL live inspection, memory integrity validation**
System load management	Crashes under EDR-killer attacks	**Dynamic kernel buffer scaling, OS decoupled**

Figure 2. Legacy EDR/XDR and Cyber Crucible, capability by capability[2]

References

1. Cyber Crucible Inc., "Systems and methods for ransomware detection and mitigation," US 11,720,678 B2; EP 3 999 984 B1; CA 3,148,437 C; AU 2020315905 B2; JP 7393517 B2.
2. Cyber Crucible, collected white papers, 24 February 2026. Published individually on the Cyber Crucible Intelligence Hub. https://www.cybercrucible.com/intelligence-hub
3. Cyber Crucible, "OS Independence" presentation.
4. Cyber Crucible, "Cyber Crucible Defense" presentation, version 9.

Related papers

- **Part 1 of 5:** DLL Side-Loading in Real Attacks: Executive Summary www.cybercrucible.com/intelligence-hub/dll-side-loading-in-real-attacks
- **Part 2 of 5:** DLL Side-Loading: How Attackers Hide Inside Trusted Software www.cybercrucible.com/intelligence-hub/dll-side-loading-how-attackers-hide-in-trusted-software
- **Part 3 of 5:** Stealing Browser-Protected Credentials: Defeating App-Bound Encryption www.cybercrucible.com/intelligence-hub/stealing-browser-credentials-app-bound-encryption
- **Part 4 of 5:** Why Allow-Listing, Cloud Collectors and EDR Killers Miss It www.cybercrucible.com/intelligence-hub/allow-listing-cloud-collectors-edr-killers-dll-side-loading
- **Ransomware Defense:** The Technical and Business Failures of Ransomware Key Capture Defense www.cybercrucible.com/intelligence-hub/ransomware-key-capture-defense-failures

Read online (six languages): www.cybercrucible.com/intelligence-hub/os-independence-inspecting-dlls-without-trusting-the-os
Book a live technical demo: www.cybercrucible.com/demo