ultraviolet

THREAT ADVISORY

NuGET Malicious Packages

Services Performed By:
UltraViolet Cyber TIDE Team
tide@uvcyber.com

Published Date:
November 10, 2025
TLP:GREEN

Executive Snapshot

The discovery of delayed-activation malware within NuGet packages highlights how trusted development ecosystems can quietly become conduits for long-term compromise. Enterprises must recognize that even legitimate-looking libraries may conceal dormant threats capable of triggering years after deployment, threatening both IT and OT environments. To strengthen resilience against such attacks, organizations should adopt a structured and proactive dependency-security strategy that extends beyond traditional vulnerability scanning. UltraViolet Cyber (UVCyber) Threat Intelligence and Detection Engineering (TIDE) Team suggests the following defensive actions:

Implement strict dependency governance: Use curated and verified internal repositories, enforce code-signing for all imported packages, and avoid sourcing directly from public feeds without vetting.

Deploy automated supply-chain scanning and SBOM validation: Continuously analyze dependencies for unusual behavior, author anomalies, and delayed-execution logic during build and runtime stages.

Enhance runtime and behavioral monitoring: Instrument production environments to detect anomalous library behavior—such as intermittent write failures, process crashes, or unexpected system calls—and correlate these events to dependency origins.

Maintain long-term audit and incident-response visibility: Preserve build metadata, logs, and package provenance over multi-year periods to enable backward tracing of latent or time-delayed threats when they emerge.

Below is a list of the malicious packages recently found in NuGET:

SqlUnicornCoreTest
SqlUnicornCore
SqlUnicorn.Core
MyDbRepository
MCDbRepository
Sharp7Extend
SqlDbRepository
SqlRepository
SqlLiteRepository

TIDE Team Analysis

A recent investigation uncovered a malicious supply chain campaign within the NuGet ecosystem, where nine compromised packages were uploaded under the guise of legitimate libraries and collectively downloaded nearly 9,500 times. These packages were crafted to appear functional and benign while embedding delayed activation mechanisms—logic bombs programmed to trigger years later, with specific activation windows set for 2027 and 2028. This discovery illustrates how attackers are increasingly adopting patient, long-term tactics that weaponize trust in open-source ecosystems used throughout enterprise environments.

What makes this attack particularly deceptive is how the malicious packages blended in seamlessly with authentic libraries. They performed as expected during normal development cycles, providing full functionality and building credibility among developers. Only after the designated trigger dates would the hidden logic execute, with some payloads designed to disrupt database operations while others interfered with industrial control and automation systems. One notable example involved a package mimicking an existing PLC communication library that silently inserted code to intercept and alter control operations, demonstrating the risks of tampering within trusted industrial software layers.

Technically, the campaign revealed a high level of sophistication. By exploiting C# extension methods, the attackers ensured that the malicious logic was executed whenever certain common operations occurred—such as a database query or PLC command—without requiring any additional calls from the developer. The logic bombs used randomized timers and probabilistic behaviours to introduce instability and confusion, including intermittent write failures or process terminations. These characteristics make detection extraordinarily challenging and complicate forensic analysis, especially once the payloads activate long after initial deployment.

For enterprise environments, this presents a dangerous convergence of stealth and persistence. The dormant nature of the logic bombs means they can remain hidden for years within production systems, silently waiting to execute at a time when institutional memory has faded. By the time such malicious code activates, the original developer or system owner may have moved on, leaving minimal context for identifying the root cause. The symptoms—ranging from sporadic software crashes to apparent hardware issues—may be easily misattributed, delaying incident response and extending operational disruptions in critical sectors such as manufacturing, energy, or transportation.

From a strategic perspective, this campaign underscores the persistent vulnerability of open-source and package-based ecosystems. Many enterprises rely on third-party libraries without sufficient insight into the trustworthiness of their maintainers or the integrity of the code they import. The risk is amplified by the long-tail lifecycle of software, where components integrated today may only manifest security implications years later. Such delayed activation transforms dependency management from a short-term quality control issue into a long-term strategic risk that directly affects business continuity and cyber resilience.

To reduce exposure, organizations must adopt stronger controls around software dependencies, treating package integrity as an extension of their broader supply chain security. This includes curated whitelists of trusted libraries, stricter code-signing and validation processes, and the use of Software Bills of Materials to maintain visibility into every external component. Runtime monitoring should also evolve to detect anomalies that might indicate malicious library behaviour, including subtle deviations in database activity or control system responses. Equally important is maintaining detailed audit trails and long-term logging, enabling investigators to trace back anomalous events to specific dependencies introduced years earlier.

Ultimately, this campaign serves as a critical reminder that trust in public package ecosystems cannot be assumed. Attackers have demonstrated the patience and technical capability to weaponize dependencies that appear harmless for years, only to activate at moments designed for maximum disruption. For enterprises, the imperative is to move beyond reactive patching and toward proactive dependency governance, runtime validation, and long-term monitoring. Without these safeguards, a single malicious NuGet package could compromise not only code integrity but the operational reliability and reputation of entire organizations long after initial deployment.

Why It Matters

The NuGet ecosystem has long been a cornerstone of enterprise .NET development, providing rapid access to reusable code and reducing software delivery timelines. However, this convenience has repeatedly been exploited by threat actors who recognize the trust developers place in package repositories. Over the past decade, several campaigns have targeted NuGet with malicious or Typosquatted packages—ranging from credential-stealing modules to payload droppers designed to exfiltrate environment variables, cloud tokens, and API keys from build systems. In 2023 and 2024, security researchers identified multiple NuGet campaigns where attackers uploaded packages that masqueraded as legitimate Microsoft or utility libraries, some even embedding PowerShell loaders and obfuscated scripts. These incidents demonstrate that malicious NuGet packages are not theoretical risks but active and recurring tactics in the software supply chain.

The latest wave of delayed-activation malware within NuGet represents a dangerous evolution of this pattern. Whereas earlier attacks often relied on immediate execution to deliver payloads or steal credentials, this new generation of threats is patient, subtle, and engineered for longevity. By embedding logic bombs designed to activate years after installation, attackers ensure their code can persist undetected across software lifecycles, silently propagating through enterprise CI/CD pipelines, developer endpoints, and production environments. Such attacks are particularly dangerous in operational technology contexts, where software libraries are rarely updated and systems may run for years without scrutiny. The combination of trusted origins, delayed activation, and broad ecosystem reach means that a single compromised package can embed risk across dozens of dependent systems before any signs of compromise appear.

Protecting NuGet and broader developer environments requires a shift in mindset from reactive patching to active ecosystem defense. Organizations must treat developer workstations, build servers, and package feeds as part of their critical infrastructure, not merely as convenience tools. Comprehensive measures such as code-signing enforcement, private mirror repositories, SBOM tracking, and multi-layered runtime validation can reduce exposure to malicious dependencies. Equally important is embedding security awareness within developer culture—training teams to verify package maintainers, monitor for Typosquatting, and scrutinize unexpected dependency updates. As attackers increasingly exploit the implicit trust of package ecosystems, defending NuGet and other developer platforms becomes central to maintaining the integrity, reliability, and safety of modern enterprise software.

How to Respond

- Strictly adhere to CyberSecurity Fundamentals and ensure all personnel undergo annual phishing and social engineering training. Speak with your UltraViolet Cyber TAM Representative to schedule a live phishing engagement.
- Ensure software and services in your environment which are maintained by Third Party Vendors include a current and updated Software Bill of Materials (SBOM). Ensure that all SBOMs are shared with your UVCyber TAM Representative so they can be analyzed for out-of-date, vulnerable, or malicious packages.
- Perform annual tech refresh reviews to gain a holistic understanding of your infrastructure. Speak with your UltraViolet Cyber TAM Representative to schedule a RedTeam or PurpleTeam engagement to gain insight into the vulnerabilities in your environment.

What UltraViolet Cyber is Doing

- Performing hypothesis driven Threat Hunts in Customers Environments to survey and understand the risks that NuGET could pose, while also applying bespoke detection rules where applicable.
- Parsing available victim dump data for any social, financial, business, or technical relations to UVCyber Clients and partner organizations.
- Aggregating threat intelligence from myriad sources and applying the most up-to-date knowledge to proactive threat hunting and response.