CODEHUNTER®
ZERO TRUST FOR CODE

Behavioral Risk Brief

Notepad ++

July 24, 2026

The Claim

Governance frameworks that treat a legitimate, signed application as inherently safe create systemic risk when that application can be paired with malicious components it will execute automatically. Zero Trust for Code addresses this by requiring a pre-execution trust decision on what an application actually loads and runs, rather than extending trust to everything bundled alongside a recognized binary.

The Incident

CERT-UA disclosed a campaign attributed to the threat cluster UAC-0099 that delivers a fake Notepad++ plugin to compromise Windows systems. A phishing email leads victims through a shortened link to a ZIP archive containing a VBScript disguised as a PDF, which displays a decoy document while silently retrieving a second archive. That archive contains a complete and legitimate copy of Notepad++, a malicious DLL named NppExport.dll, and supporting tools. The script launches the genuine Notepad++ binary, which automatically loads the malicious DLL as a plugin. That DLL unpacks additional components and establishes persistence through a scheduled task running every three minutes, ultimately delivering a loader called “BURNYBEAR” and a modified payload tracked as “MATCHBOIL.V2”.

The Governance Failure

The governance failure is not that Notepad++ was impersonated, but that the actual application delivered to the victim is the real legitimate Notepad++ binary that is unmodified and fully functional. Any control built around verifying the identity or signature of the executable being launched would find nothing to object to. The compromise lives entirely in what that legitimate binary is permitted to load and execute once it runs and a layer signature verification is never reached.

This is compounded by how plugin architectures function. A trusted application loading a DLL from its own plugins directory is completely ordinary behavior. It is indistinguishable at launch from any legitimate extension a user might install. Trust granted to the parent executable extends implicitly to whatever it loads next, without a separate decision evaluating what that plugin does once it holds the parent process's privileges.

The underlying breakdown is the absence of a trust decision at the moment a loaded component begins executing, as distinct from when the parent application launches. This pattern is not unique to Notepad++. Any application with a plugin or module-loading architecture creates the same exposure, and as long as trust is assigned only to the parent process, this technique remains available regardless of which application it is built around.

The Regulatory and Business Exposure

- Persistent access established through a scheduled task disguised as routine application behavior.
- Resource exhaustion triggered as a fallback if the malicious loader runs outside expected conditions.
- Reliance on signature checks that verify the parent application while loaded plugins go unexamined.

What Your Auditor Will Ask

- How do you validate the behavior of plugins or modules loaded by applications your organization already trusts?
- How do you distinguish a legitimate binary from the components it is permitted to load once running?
- What controls detect a scheduled task established through an otherwise trusted application?
- How do you verify that application allowlisting accounts for what a binary executes after launch?
- What evidence do you maintain that a loaded plugin's behavior was evaluated independently of the parent application's trust status?

A consistent signal is the disconnect between the executable that was verified and the code that actually ran. A legitimate binary passing every identity check says nothing about what it will load once permitted to execute.

CODEHUNTER

Zero Trust for Code Value

Zero Trust for Code introduces a trust decision at the point a loaded component is about to execute, evaluating what that plugin or module will do independently of whether the parent application is legitimate or already allowlisted. A plugin is assessed on its own behavior, not granted the trust status of the executable that loads it.

This directly addresses the governance weakness exposed by UAC-0099: the assumption that verifying a parent application is sufficient because everything it loads inherits that same trust. By requiring a pre-execution decision for loaded components specifically, organizations can stop a malicious plugin even when the application launching it is entirely legitimate.

The result is a governance model where trust is evaluated at every layer, the parent binary and everything it loads, rather than assumed to flow automatically from one to the other.

Governance Action Brief

- Establish governance controls that evaluate plugin behavior independently of the parent application's trust status.
- Require a pre-execution trust decision for any component loaded dynamically by an approved application.
- Enforce execution policies covering plugin directories for widely used developer and productivity applications.
- Monitor for scheduled tasks or persistence mechanisms established through processes descending from trusted applications.
- Maintain a record of loaded components evaluated separately from the parent binary's approval.

Sources

Analysis based on reporting from The Hacker News (July 24, 2026) on CERT-UA's disclosure of the UAC-0099 campaign, CodeHunter Labs evaluation of governance gaps in plugin execution trust, and alignment with NIST 800-53 and NIST SSDF integrity control objectives.

CODEHUNTER