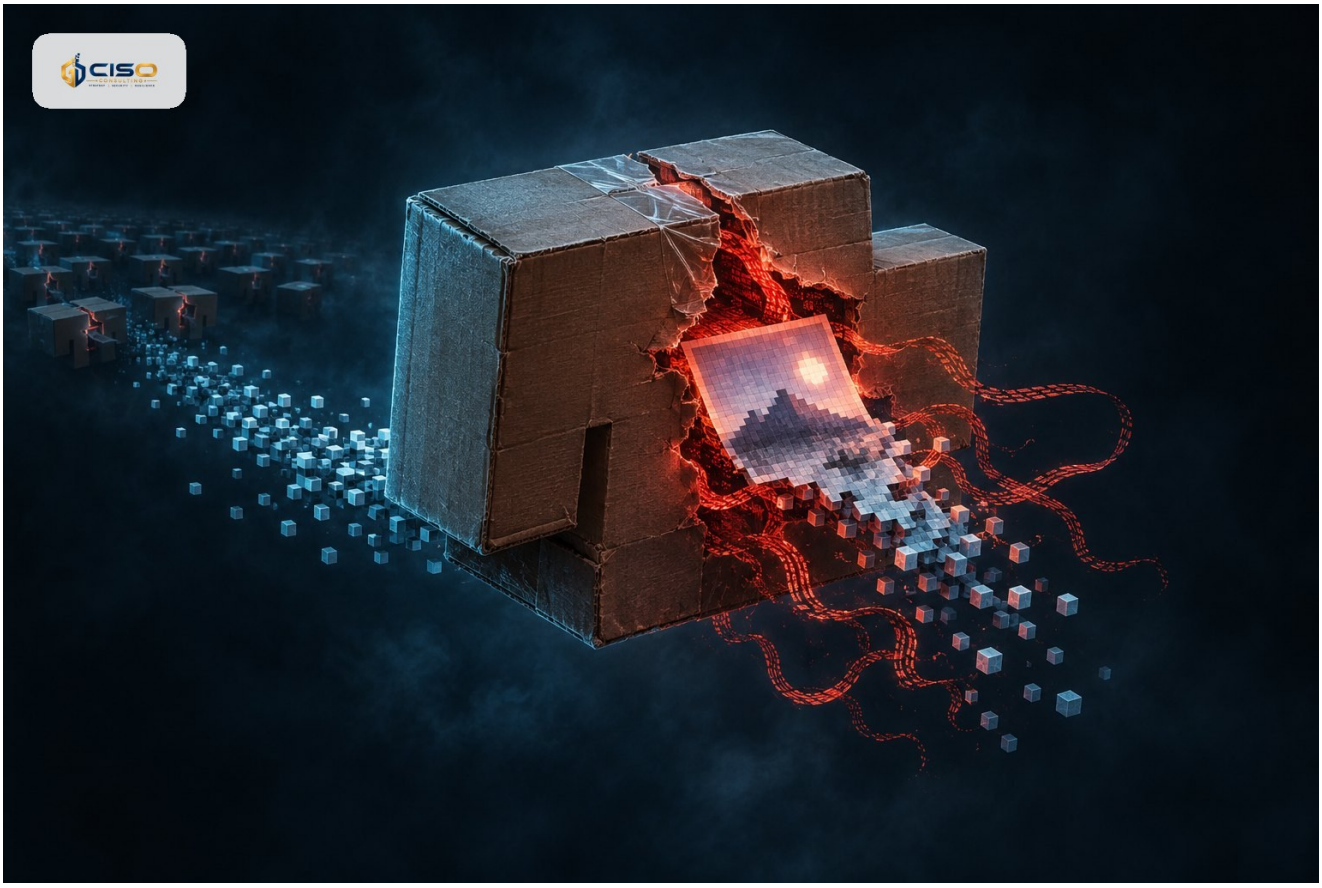
Malicious npm package 'imgdecode' linked to MALFEX supply chain campaign



Researchers found 'imgdecode', a malicious npm package tied to the MALFEX campaign. Uploaded on 30 September 2026, it had 979 downloads and was still listed at the time of reporting.

- The package 'imgdecode' is part of a software supply chain campaign named MALFEX.
- It was uploaded to npm on 30 September 2026 by a user named 'malfexteste8'.
- It had been downloaded 979 times and remained available at the time of reporting.
- The source headline says it hides a shellcode loader inside a Windows native addon.

Cybersecurity researchers have identified a malicious npm package named "imgdecode" as part of MALFEX, a software supply chain attack campaign disclosed recently.

What is known

The package was published to the npm registry on 30 September 2026 by an account called "malfexteste8". At the time of reporting it had been downloaded 979 times and was still available for download. According to the source headline, it conceals a shellcode loader inside a Windows native addon.

The excerpt available to us does not describe the loader's behaviour, its payload, or the wider scope of the MALFEX campaign, so we do not speculate on these points.

Why it matters

Organisations that build or maintain JavaScript and Node.js applications pull in third-party packages routinely. A package that looks like a utility but ships a native addon can run code on developer workstations or build servers, so dependency intake is a risk worth checking even where no incident is suspected.

Recommended approach

Teams should search their repositories, lock files and build logs for the package name and treat any match as a potential compromise of the host that installed it. Further technical detail should be taken from the original researchers' report before finalising any response.

What to do now

1. Search repositories, lock files and CI/CD logs for 'imgdecode' and for installs from the user 'malfexteste8'.
2. If found, isolate the affected workstation or build host and start incident handling, including credential rotation.
3. Review the original researchers' report for indicators of compromise before closing the investigation.
4. Restrict npm installs to an approved internal registry or allow-list of vetted packages.
5. Add dependency review for native addons to the secure development process.