Tensorlake npm package compromised in Shai-Hulud supply chain attack



Version 0.5.144 of the tensorlake npm package was compromised and carries obfuscated malware that steals credentials and secrets. Teams using the package should check their dependencies.

- The npm package tensorlake, a TypeScript SDK for Tensorlake applications, sandboxes and cloud services, was compromised.
- The malicious release is version 0.5.144.
- The incident is tied to a ChainDrop / Shai-Hulud supply chain attack.
- Socket reports the malware harvests credentials, exfiltrates secrets, sets up persistence and runs remotely supplied code.

What happened

The npm package "tensorlake", a TypeScript software development kit for Tensorlake applications, sandboxes and cloud services, was compromised as part of a ChainDrop / Shai-Hulud supply chain attack. The affected release is version 0.5.144.

According to Socket, the malicious version contains obfuscated malware. It harvests credentials, exfiltrates secrets, establishes persistence and executes code supplied remotely by the attacker.

Why it matters

A package like this is typically installed in developer workstations and build pipelines, where tokens, keys and cloud credentials are often available. Malware that steals secrets and keeps a foothold can therefore extend well beyond the machine that installed it. The source does not say how many installations were affected or how the package was compromised, so institutions should not assume they are unaffected without checking.

What to do next

Start by establishing whether version 0.5.144 was ever installed in your environments. If it was, treat the host and any secrets reachable from it as exposed, and rotate them. The actions below set out a practical order of work.

What to do now

1. Search lockfiles, repositories and build logs for tensorlake version 0.5.144.
2. If found, isolate the affected host or runner and treat credentials, tokens and keys accessible from it as compromised.
3. Rotate exposed secrets, including cloud and CI/CD credentials, and review them for unauthorised use.
4. Remove the malicious version, pin to a verified safe release and check for persistence on affected systems.
5. Review controls on third-party package use, such as version pinning and delayed adoption of new releases.