WHITE PAPER

CYTEX™
AI-Powered Resilience Command Center

Mini Shai-Hulud Worm Compromises 498 npm Packages via @antv Ecosystem Supply Chain Attack

hello@cytex.io | https://cytex.io

Mini Shai-Hulud Worm Compromises 498 npm Packages via @antv Ecosystem

Technical analysis of the active TeamPCP supply chain operation against the @antv npm namespace, with worm propagation mechanics, payload behavior, and IOCs.

What Happened

An active, fast-moving supply chain attack has compromised hundreds of packages in the **@antv npm ecosystem**, including widely used data visualization libraries @antv/g2, @antv/g6, and @antv/x6. The compromise propagated through a single maintainer account (atool) whose npm credentials were stolen and reused to publish malicious versions in rapid succession.

In the most recent wave, **639 compromised package versions across 323 unique packages** were published, including **558 versions across 279 @antv packages**. Total campaign exposure across all targets now stands at **1,048 npm versions across 498 unique packages**, plus 6 PyPI entries and 1 Composer entry.

The attack is attributed to the **Mini Shai-Hulud** worm campaign, which security researchers assess to be the work of the financially motivated threat actor **TeamPCP**. The same group has been linked to 2026 supply chain compromises affecting Trivy, Checkmarx, Bitwarden CLI, TanStack, and most recently the unauthorized access of GitHub internal repositories.

Two operational details elevate this incident beyond a typical package compromise. **First**, TeamPCP has released the Mini Shai-Hulud source code publicly as part of a supply chain attack contest. At least one unknown actor has already deployed a near-verbatim clone with independent command-and-control infrastructure. **Second**, the worm has successfully exfiltrated credentials from more than **2,500 victim environments** based on a public marker the payload leaves on compromised GitHub accounts.

Campaign Scale

Metric	Detail
Wave size (latest)	639 versions across 323 packages
@antv subset of latest wave	558 versions across 279 packages
Cumulative npm impact	1,048 versions across 498 unique packages
PyPI impact	6 packages
Composer impact	1 package
Publish burst duration	22 minutes across 317 packages (637 versions)

Metric	Detail
Compromised maintainer account	atool
Notable affected package	echarts-for-react (widely deployed React wrapper for Apache ECharts)
Victim environments evidenced via GitHub marker	2,500+

Worm Propagation Mechanics

The Mini Shai-Hulud worm uses a two-layer delivery pattern that maximizes redundancy and complicates takedown.

Layer one: preinstall hook in malicious npm packages. Each compromised version adds a preinstall hook to package.json that executes bun run index.js during installation. The hook fires automatically when a developer or CI/CD system runs npm install. No user interaction, no manual execution, and no opt-in is required for the payload to run.

Layer two: imposter commits in the legitimate antvis/G2 GitHub repository. Of the 637 malicious versions in the publish burst, **630 inject optionalDependencies entries pointing to imposter commits in the legitimate antvis/G2 repository on GitHub**. This delivers a second copy of the payload through a trusted source path. Even if the npm package is removed from the registry, the GitHub-hosted dependency path continues to resolve. The payload is delivered twice from two trust boundaries, by design.

Identical obfuscated payload. Every compromised version carries the same obfuscated payload. The repeated payload across 637 versions in 22 minutes confirms automated publishing using the stolen atool token rather than manual, per-package modification.

Payload Behavior

Once executed via the preinstall hook, the payload performs the following operations:

Credential harvesting across 20+ secret classes. The payload enumerates filesystem and environment locations for credentials and exfiltrates anything it finds. Confirmed target classes include AWS credentials, GCP service account keys, Azure credentials, GitHub tokens, npm tokens, SSH private keys, Kubernetes configuration and tokens, HashiCorp Vault credentials, Stripe API keys, and database connection strings stored in environment variables or configuration files.

Docker container escape attempts. The payload checks for and attempts to interact with the host's Docker socket (/var/run/docker.sock). Where the socket is accessible to the container running the payload, the worm uses it to attempt container escape and access the host environment. This is the same container escape primitive observed in the prior Mini Shai-Hulud variant used in the SAP package compromise earlier in 2026.

Public marker deployment. Successful exfiltration is followed by the creation of a marker on the victim's GitHub account. The payload pushes content to repositories under the compromised account with the description string "niagA oG eW ereH :duluH-iahS". Reversed, this reads "Shai-Hulud: Here We Go Again". This is both an operator signature and the indicator security researchers use to enumerate confirmed victims. The current count exceeds 2,500 repositories, each representing a distinct environment whose credentials were successfully stolen.

TeamPCP Campaign Context and the Public Source Release

This incident sits inside a broader TeamPCP operational pattern that has run continuously across 2026. The pattern is consistent: compromise a high-trust component upstream, harvest credentials downstream, use harvested credentials to fund the next iteration of compromise, monetize the result.

Prior 2026 TeamPCP operations include the **Trivy GitHub Action** compromise with downstream credential harvesting, the **Checkmarx** supply chain incident, the **Bitwarden CLI** package compromise, the **TanStack ecosystem** incident, the **SAP package** compromise that used the same Mini Shai-Hulud payload now observed in @antv, and the **GitHub internal repositories** unauthorized access incident that exposed approximately 3,800 internal repos via a malicious Visual Studio Code extension.

The new development with this @antv wave is the **public release of the Mini Shai-Hulud source code**. TeamPCP announced a supply chain attack contest in coordination with BreachForums and released the worm's source code to participants. Within days, at least one unknown threat actor uploaded four malicious npm packages containing a near-verbatim copy of the Shai-Hulud worm pointed at independent command-and-control infrastructure.

The defensive implication is that the Mini Shai-Hulud worm is now a **commodity tool**. Attribution to TeamPCP for individual future compromises will become harder. Multiple actors operating independent variants of the worm against different package ecosystems should be expected.

Indicators of Compromise

Compromised maintainer account:

- npm username: atool

Affected package families (non-exhaustive):

- @antv/g2 and its sub-packages
- @antv/g6 and its sub-packages
- @antv/x6 and its sub-packages
- echarts-for-react

- Additional data visualization, graphing, mapping, and charting packages across the @antv namespace and unrelated maintainers

Package-level indicators:

- Presence of a preinstall hook running bun run index.js
- optionalDependencies entries pointing to commits in the antvis/G2 GitHub repository that are not part of the project's legitimate release history
- Package versions published during the 22-minute burst window
- Versions published by the atool account during the campaign window

Host and environment indicators:

- Unexpected network connections from build hosts, CI runners, or developer workstations to exfiltration endpoints during or shortly after npm install execution
- Access attempts against /var/run/docker.sock from processes spawned by npm install or bun
- Reads against credential locations including ~/.aws/credentials, ~/.config/gcloud/, ~/.azure/, ~/.ssh/id_*, ~/.kube/config, ~/.npmrc, ~/.gitconfig, and environment-variable enumeration from node, npm, or bun processes

Post-exfiltration markers on GitHub:

- Repositories on victim accounts containing the description string "niagA oG eW ereH :duluH-iahS" (reverse: Shai-Hulud: Here We Go Again)
- Newly created public repositories on developer accounts immediately following installation of affected packages
- Unexpected commits to existing repositories under the victim account

Immediate Actions

Rotate all credentials in any environment that installed an affected version. Treat any AWS, GCP, Azure, GitHub, npm, SSH, Kubernetes, Vault, Stripe, or database credential present on a build host, CI runner, or developer workstation that ran the malicious install as compromised. Rotate the credential and audit the access logs for the exposure window.

Audit GitHub accounts and organization repositories for the Shai-Hulud marker string. Search across personal and organizational repositories for the description string "niagA oG eW ereH :duluH-iahS". Any match is a confirmed exfiltration event.

Pin to known-good versions and lock dependencies. Switch affected packages to versions published before the campaign window. Use lockfile integrity verification (package-lock.json, yarn.lock, pnpm-lock.yaml) and audit lockfile changes for unexpected updates.

Inspect package.json and lockfiles across recent dependency updates. Look for the presence of preinstall hooks invoking bun run index.js and optionalDependencies entries pointing to GitHub commit references rather than published registry versions.

Enable two-factor authentication on all package registry accounts. The initial compromise of the atool account is the precondition that allowed the entire publish burst. Mandatory 2FA on npm publisher accounts reduces the population of accounts that can be compromised the same way.

Restrict Docker socket access from build containers. Where the build process does not legitimately require host Docker access, the socket should not be mounted into the container. This removes the escape primitive the payload depends on.

Enable egress monitoring on CI/CD infrastructure. Outbound connections from build environments to non-standard destinations during install phases are a strong behavioral indicator. Allowlisting outbound destinations on build hosts limits exfiltration channels.

Plan for further variants. With the source code public and at least one independent clone already deployed, multiple parallel campaigns using the same worm logic should be expected. Build inventory, lockfile discipline, and credential rotation procedures should be staged as standing capabilities rather than incident response.

Sources

- GitHub repository enumeration of the "niagA oG eW ereH :duluH-iahS" marker string
- Underground forum monitoring of the BreachForums supply chain attack contest and Mini Shai-Hulud source code release