Lessons Learned From Massive npm Supply Chain Attack Using “Shai-Hulud” Self-Replicating Malware

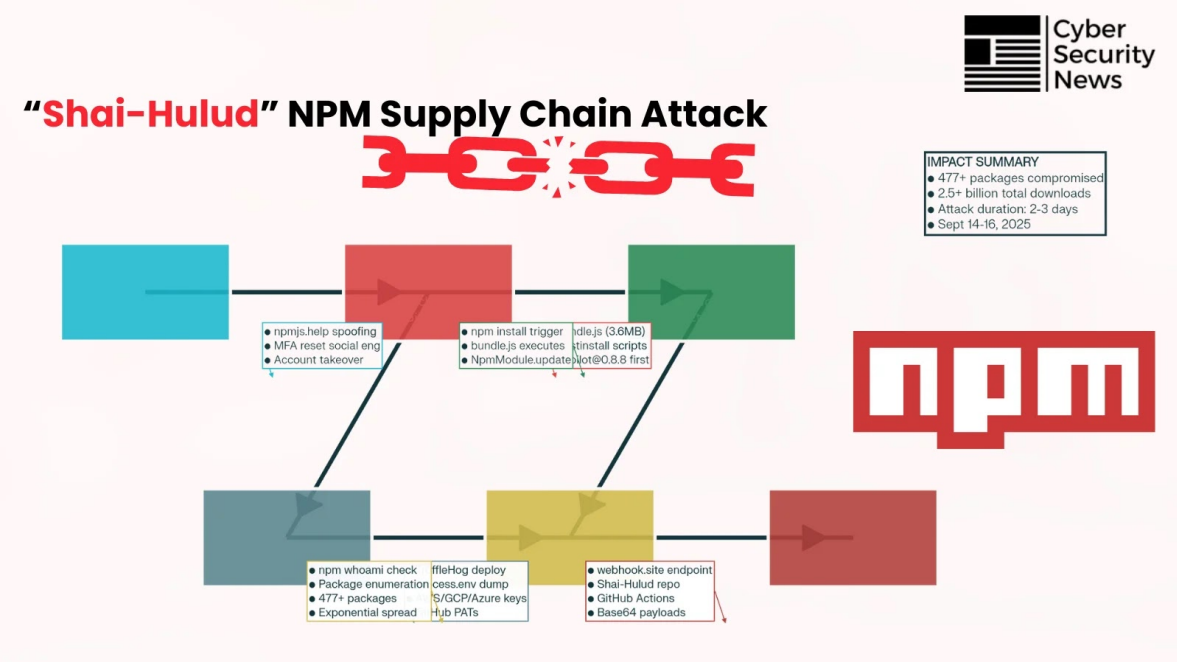
Guru Baran ⋮ ⋮ 9/18/2025

By

Guru Baran

-

September 18, 2025

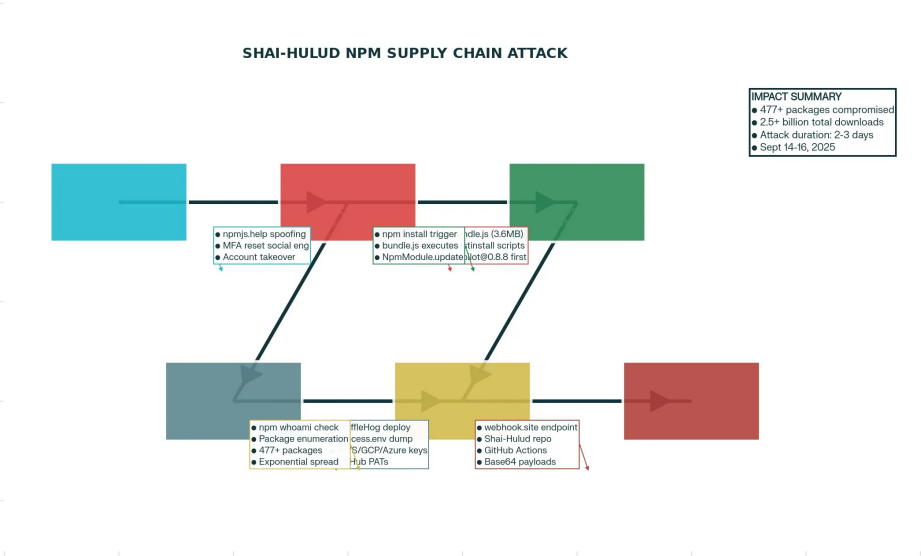


The JavaScript ecosystem experienced one of its most sophisticated and damaging supply chain attacks in September 2025, when a novel self-replicating worm dubbed “Shai-Hulud” compromised over 477 npm packages, marking the first successful automated propagation campaign in the npm registry’s history.

This attack represents a significant evolution in supply chain threats, leveraging both social engineering and technical automation to achieve unprecedented scale and persistence across the open-source software ecosystem.

The Shai-Hulud campaign began with a sophisticated phishing operation targeting npm package maintainers through fake domains spoofing the official npm registry.

Attackers created convincing emails from the fraudulent domain npmjs[.]help, closely mimicking the legitimate npmjs[.]com, and urged maintainers to “update” their multi-factor authentication credentials under threat of account lockout.



Shai-Hulud NPM Supply Chain Attack

This social engineering approach proved devastatingly effective, as it exploited the trust relationship between developers and the npm platform while creating a sense of urgency that bypassed normal security caution.

The attack's sophistication was further evidenced by Unit 42's assessment that the threat actors likely leveraged Large Language Models (LLMs) to assist in writing the malicious bash scripts, based on the inclusion of comments and emojis in the code.

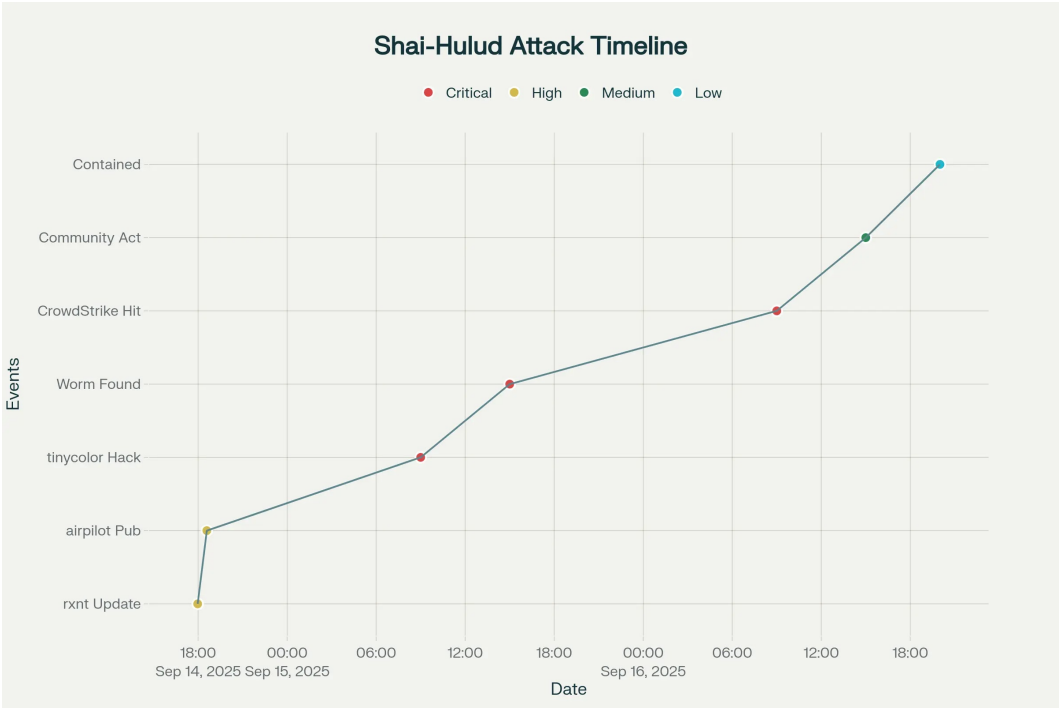
This represents a concerning trend in cybercriminal operations, where AI tools are increasingly being weaponized to enhance the quality and effectiveness of malicious code development.

Supply Chain Attack Using "Shai-Hulud" Self-Replicating Malware

The malware's core innovation lies in its self-replicating mechanism, implemented through the `NpmModule.updatePackage` function. Unlike traditional supply chain attacks that require manual intervention for each compromised package, Shai-Hulud operates as a true worm, automatically identifying and infecting additional packages maintained by compromised developers.

The propagation process follows a systematic approach: downloading existing package tarballs, modifying `package.json` files to inject malicious `postinstall` scripts, embedding the ~3.6MB minified `bundle.js` payload, repackaging the archives, and republishing them to the npm registry.

This automated approach enabled exponential growth in affected packages, with the malware spreading from an initial handful of compromised packages to over 477 infected packages within approximately 72 hours.



Shai-Hulud NPM Supply Chain Attack Timeline

The worm's design ensures persistence across the ecosystem by leveraging legitimate maintainer credentials and publishing rights, effectively turning trusted developers into unwitting vectors for malware distribution.

The malware execution begins when users install compromised packages via `npm install`, triggering the `postinstall` script that launches the `bundle.js` payload.

This Webpack-bundled script performs comprehensive system reconnaissance, beginning with environment variable extraction (`process.env`) to capture sensitive credentials immediately available in the execution context.

The payload then deploys TruffleHog, a legitimate open-source secret scanning tool, using the command `trufflehog filesystem . --json --results=verified` to systematically scan the local filesystem for over 800 different types of credentials.

The malware demonstrates sophisticated credential validation capabilities, using `npm whoami` commands to verify the authenticity of discovered npm tokens and access cloud service APIs to confirm the validity of AWS, Google Cloud Platform, and Microsoft Azure credentials.

This validation step ensures that only working credentials are exfiltrated, maximizing the value of stolen data for subsequent malicious activities.

Comprehensive Package Analysis

The attack timeline reveals a rapid escalation that caught the security community off-guard. The earliest confirmed malicious package, `airpilot@0.8.8`, was published on September 14, 2025, at 18:35:07.600Z UTC.

The campaign gained significant momentum with the compromise of `@ctrl/tinycolor@4.1.1`, a package with over 2.2 million weekly downloads, which was first reported by security researcher Daniel Pereira on September 15, 2025.

The attack's scope expanded dramatically on September 16, when security researchers identified compromised packages belonging to enterprise vendors, including multiple CrowdStrike npm packages.

This expansion demonstrated the worm's ability to breach high-value targets and potentially access enterprise development environments, raising the stakes significantly for affected organizations.

Affected Package Inventory

Package_Name	Compromised_Version	Status
	ctrl/tinycolor	Removed
rxnt-authentication	0.0.6	Removed
airpilot	0.8.8 (earliest identified)	Removed
angulartics2	14.1.2	Removed
	ctrl/deluge	Removed
	ctrl/golang-template	Removed
	ctrl/magnet-link	Removed
	ctrl/ngx-codemirror	Removed
	ctrl/ngx-csv	Removed
	ctrl/ngx-emoji-mart	Removed
	ctrl/ngx-rightclick	Removed
	ctrl/qbittorrent	Removed
	ctrl/react-adsense	Removed
	ctrl/shared-torrent	Removed
	ctrl/torrent-file	Removed
	ctrl/transmission	Removed
	ctrl/ts-base32	Removed
encounter-playground	0.0.5	Removed
json-rules-engine-simplified	0.2.4, 0.2.1	Removed
koa2-swagger-ui	5.11.2, 5.11.1	Removed
	nativescript-community/gesturehandler	Removed
	nativescript-community/sentry	Removed
	nativescript-community/text	Removed
	nativescript-community/ui-collectionview	Removed
	nativescript-community/ui-drawer	Removed
	nativescript-community/ui-image	Removed
	nativescript-community/ui-material-bottomsheet	Removed
	nativescript-community/ui-material-core	Removed
	nativescript-community/ui-material-core-tabs	Removed
ngx-color	10.0.2	Removed
ngx-toastr	19.0.2	Removed
ngx-trend	8.0.1	Removed
react-complaint-image	0.0.35	Removed
react-jsonschema-form-conditionals	0.3.21	Removed
react-jsonschema-form-extras	1.0.4	Removed
rxnt-healthchecks-nestjs	1.0.5	Removed
rxnt-kue	1.0.7	Removed
swc-plugin-component-annotate	1.9.2	Removed
ts-gaussian	3.0.6	Removed

The complete inventory of affected packages spans multiple maintainer namespaces and includes both popular libraries and specialized tools. Key compromised packages include:

High-Impact Packages:

- `@ctrl/tinycolor@4.1.1, 4.1.2` – 2.2 million weekly downloads
- `angulartics2@14.1.2` – Popular Angular analytics library
- `ngx-toastr@19.0.2` – Widely-used notification component
- Multiple `@nativescript-community` packages affecting mobile development workflows

Enterprise and Security-Related Packages:

- Multiple CrowdStrike npm packages (specific package names were rapidly removed by npm administrators)
- `rxnt-authentication@0.0.6` – Authentication-related functionality

- Various `@ctrl` namespace packages spanning file management, networking, and media processing

The malware's selection of targets appears strategic, focusing on packages with high download counts and broad dependency graphs to maximize infection potential.

The inclusion of enterprise vendor packages suggests either sophisticated targeting or opportunistic exploitation of compromised maintainer accounts with access to commercial package repositories.

Indicators of Compromise (IOCs) and Detection Methods

Category	Indicator	Value
file_hashes	bundle.js	46faab8ab153fae6e80e7cca38eab363075bb524edd79e42269217a083628
network_indicators	webhook_url	https://webhook.site/bb8ca5f6-4175-45d2-b042-fc9ebb8170b7
network_indicators	trufflehog_download	Downloaded and executed from filesystem
file_system_indicators	malicious_workflow	.github/workflows/shai-hulud-workflow.yml
file_system_indicators	github_branch	shai-hulud
file_system_indicators	bundle_file	bundle.js (varies in size, ~3.6MB minified)
file_system_indicators	public_repo	Shai-Hulud repository created in victim accounts
process_indicators	npm_commands	npm whoami, npm publish commands
process_indicators	trufflehog_command	trufflehog filesystem . –json –results=verified
process_indicators	postinstall_script	node bundle.js

Security teams can identify potential compromises through several file system artifacts. The primary indicator is the presence of malicious `bundle.js` files with the SHA-256 hash `46faab8ab153fae6e80e7cca38eab363075bb524edd79e42269217a083628f09`.

However, researchers note that this hash may vary across different campaign iterations, requiring behavioral detection rather than relying solely on static signatures.

Critical file system indicators include:

- `.github/workflows/shai-hulud-workflow.yml` – Malicious GitHub Actions workflow
- `shai-hulud` branch creation in Git repositories
- Public repositories named "Shai-Hulud" containing credential dumps
- Unexpected `postinstall` script additions to `package.json` files

The malware communicates with a specific command-and-control infrastructure for data exfiltration. The primary exfiltration endpoint is `https://webhook.site/bb8ca5f6-4175-45d2-b042-fc9ebb8170b7`, which received stolen credentials and system information in JSON format. Network monitoring teams should watch for:

- Outbound connections to `webhook.site` domains
- Base64-encoded HTTP POST requests containing credential data
- GitHub API abuse for repository creation and workflow injection
- TruffleHog binary downloads and filesystem scanning activity

The malware exhibits distinctive behavioral patterns that can aid in detection and incident response. Key process indicators include:

- Execution of `npm whoami` commands for credential validation
- Automated `npm publish` operations from compromised accounts
- TruffleHog process execution with filesystem scanning parameters
- GitHub API calls for repository enumeration and modification

Credential Harvesting and Data Exfiltration

Shai-Hulud implements a comprehensive credential harvesting strategy targeting multiple credential types and storage locations.

The malware prioritizes high-value credentials, including npm publishing tokens, GitHub Personal Access Tokens (PATs), and cloud service credentials for AWS, Google Cloud Platform, and Microsoft Azure.

The systematic approach includes scanning `.npmrc` files for npm registry tokens, extracting SSH private keys (`id_rsa`) from default locations, and parsing Git configuration files (`.git/config`) for embedded credentials.

The malware also targets environment-specific credential storage, including `.env` files commonly used in development environments and CI/CD pipeline configurations. This comprehensive approach ensures maximum credential exposure across different development workflows and deployment scenarios.

Critical supply chain attack on npm package @ctrl/tinycolor infecting 40+ packages with self-propagating malware and a critical severity level

The attack employs a dual-channel exfiltration strategy to ensure data persistence and accessibility. Primary exfiltration occurs through webhook endpoints at `webhook.site`, providing immediate access to stolen credentials via HTTP POST requests containing JSON-encoded credential data.

The secondary exfiltration method involves creating public GitHub repositories named “Shai-Hulud” within compromised accounts, where complete credential dumps are stored as base64-encoded files.

The malware also establishes persistence through GitHub Actions workflows, injecting `.github/workflows/shai-hulud-workflow.yml` files that execute on code pushes and automatically exfiltrate repository secrets using the `toJSON(secrets)` function.

This persistence mechanism ensures continued data collection even after the initial infection is removed from development machines.

The compromise of CrowdStrike npm packages represents a significant escalation in the attack’s potential impact on enterprise environments.

While specific package names were rapidly removed by npm administrators and CrowdStrike’s incident response team, the compromise demonstrates the malware’s ability to infiltrate packages belonging to major cybersecurity vendors.

This development raises concerns about supply chain security in enterprise software development and the potential for insider threat scenarios resulting from compromised vendor packages.

CrowdStrike confirmed that they acted quickly to remove the compromised packages upon discovery, but the incident highlights the challenges faced by enterprise software vendors in maintaining supply chain integrity.

The compromise also underscores the importance of comprehensive dependency scanning and package integrity verification in enterprise development workflows.

Security researchers have identified significant operational and technical overlaps between Shai-Hulud and previous npm supply chain attacks, particularly the S1ngularity/Nx compromise that occurred in late August 2025.

Both campaigns share similar credential harvesting techniques, GitHub repository manipulation methods, and a preference for creating public repositories to store stolen data. The technical similarities suggest either the same threat actor group or shared tooling and methodologies between related groups.

The progression from the S1ngularity attack to Shai-Hulud demonstrates a clear evolution in attacker capabilities, with the addition of self-propagating worm functionality representing a significant advancement in automated supply chain exploitation.

This evolution suggests that threat actors are continuously refining their techniques and investing in more sophisticated attack infrastructure.

Lessons Learned and Future Implications

The Shai-Hulud attack represents a watershed moment in supply chain security, demonstrating how traditional security measures are inadequate against self-propagating threats that operate at CI/CD speed.

The attack's success highlights the need for fundamental changes in how organizations approach dependency management and package validation.

Traditional approaches that focus on static vulnerability scanning and known-bad package identification are insufficient against dynamic, self-modifying threats that leverage legitimate credentials and publishing infrastructure.

The attack also underscores the critical importance of maintainer account security, as compromise of a single high-privilege account can cascade across entire package ecosystems.

The Shai-Hulud npm supply chain attack represents a paradigm shift in supply chain threats, combining sophisticated social engineering with automated propagation mechanisms to achieve unprecedented scale and impact.

The attack's success in compromising over 477 packages within a three-day period demonstrates the vulnerability of trust-based ecosystems to well-executed adversarial operations.

The incident's lessons extend beyond immediate technical remediations to fundamental questions about ecosystem security architecture and the balance between accessibility and security in open-source software distribution.

As the JavaScript ecosystem continues to grow and enterprises increase their reliance on npm packages, the security implications of Shai-Hulud will influence supply chain security practices for years to come.

The attack has proven that traditional security approaches are inadequate against adaptive, self-propagating threats, necessitating new approaches that combine automated detection, community collaboration, and enhanced maintainer security practices.

Future supply chain security must evolve to address not just known threats, but the innovative attack methodologies that sophisticated adversaries continue to develop.

The npm ecosystem's recovery from Shai-Hulud has demonstrated both its resilience and its vulnerabilities, providing a critical learning opportunity for improving supply chain security across all software distribution platforms.

The lessons learned from this incident must inform not only technical security improvements but also policy changes, community practices, and organizational security strategies to better defend against the next generation of supply chain attacks.