ADVISORY FOR ICM SECTOR

NPM supply chain attack

Original report published on: Sep 18, 2025[1]

Executive Summary

On 18 September 2025, Trend Micro reported that threat actor (TA) had launched a targeted phishing campaign compromising Node Package Manager (npm)[1] maintainer accounts by injecting malicious code into widely used JavaScript packages. The compromised packages had high global download rates of over 2.6 billion per week and affected more than 700 packages, and this reportedly included CrowdStrike packages.[2] The TA deployed payloads including cryptocurrency hijacking tools and the Shai-Hulud worm. Shai-Hulud can self-replicate and steal credentials, tokens, and repository secrets; GitHub access tokens, Continuous Integration (CI)/Continuous Delivery (CD) pipeline secrets. This impacts production infrastructure, intellectual property and operational integrity.

Background

The TA initial attack vector is by sending phishing emails that masquerade as npm security alerts containing "Update 2FA Now" links to software developers. These emails targeted package maintainers to harvest their credentials, gain privileged access, and subsequently upload malicious packages to JavaScript libraries. When these packages are installed, the malicious JavaScript executes to establish persistence, propagate across systems, and steal credentials. Shai-hulud deploys a "bundle.js" script that downloads and executes TruffleHog, a legitimate credential scanner used to collect developer and CI/CD tokens, cloud service credentials, and environment variables.

The malware used GitHub access tokens that was stolen from TruffleHog to authenticate against the GitHub API and enumerate all repositories the victim can access. It cloned the private repositories to attacker account; the newly created repositories get a suffix "-migration" to their original name.[3] Shai-Hulud then deploys malicious npm packages and writes unauthorised GitHub Actions workflow files ("shai-hulud.yaml" or "shai-hulud-workflow.yml") into the compromised repositories. As a persistence technique, the TA commits these workflow files so that the it will execute whenever CI triggers. The workflows automatically exfiltrate repository secrets and other sensitive data, creating backdoors that TA embed directly within the repository infrastructure.

The malicious workflows operate independently of the original package or infected host and continue functioning even after defenders remove packages or clean build developer systems. If TA retain unrotated tokens or CI credentials, they can trigger workflow runs or push new branches and workflow files remotely. Meanwhile, TA access exfiltrated data outside the victim's control, including data they capture in logs or transmit to TA-controlled webhooks. These capabilities enable TA to weaponise the victim's development infrastructure, creating a self-sustaining mechanism for ongoing espionage and data theft.

Detection and Mitigation

IMDA recommends organisations perform continual testing and validation of existing security controls to ensure detection and prevention identified in this advisory:

- Scan for Indicators of Compromise to detect threat activities (Annex A).

[1] NPM is a widely used JavaScript package manager that enables developers to share and manage code libraries and their dependencies.

- Refer to the MITRE ATT&CK techniques (Annex B) in this advisory:
    - Create, test and validate detection rules against the threat behaviours.
    - Validate and deny/disable processes, ports and protocols that have no business need.
- Audit and validate all npm dependencies, particularly recently updated/modified packages to ensure changes are legitimate. Remove, rebuild or roll back packages if you detect any signs of compromise.
- Revoke and regularly rotate npm account credentials or API keys as best practice.
- Audit repository inventories to identify unauthorised clones by comparing current repository counts against established baselines.
- Deploy Security Information and Event Management (SIEM) to ingest GitHub/GitLab audit logs and configure use cases to detect repositories with names ending with "-migration", or GitHub Actions workflow files ("shai-hulud.yaml" or "shai-hulud-workflow.yml") and suspicious creation or cloning events.
- Implement multi-factor authentication (MFA) for all privileged accounts across developer and CI/CD access points such as GitHub and GitLab. Enforce least privilege access controls and regularly review user permissions for new/anomalous activities.
- Deploy email security solutions to detect phishing emails.
- Conduct targeted phishing exercise on software developer for awareness.
- Track the latest advisories from the official npm registry for mitigation and new threats.

IMDA encourages organisations to conduct thorough analyses to identify potential risks and assess their potential impact prior to deploying defensive measures.

Annex A - Indicators of Compromise

SHA 1	Description
24a8425476dcf8106ff86e5a5dbdfe56767c3f83	Worm.JS.SHULUD.YXFIQ
0be45aa0f8f92e63b74f888fce0e8ccb3a843033	Worm.JS.SHULUD.YXFIQ
411a826870d686ba2d880efb2fd3db484d151560	Worm.JS.SHULUD.YXFIQ
8b98ab71cc71c8768de27af80a3e0d1bc6c8d809	Worm.JS.SHULUD.YXFIQ
7f64e210a3e4f0a4d4353f5b0e24cc6ed5f25f13	Worm.JS.SHULUD.YXFIP
19b5dc3aea3d2e403f6e1bfb2aadf4873a87c4b9	Trojan.JS.CRYPTOHIJACK.SMYXFII
ea8069be02451e9f78caf626547d63ce37c9e004	Trojan.JS.CRYPTOHIJACK.SMYXFII
94870190e0a2cc34cfb5800f3a7434b45273395d	Trojan.JS.CRYPTOHIJACK.SMYXFII
911ec8f2f54043c129d5a4116e0cddb04e96f71d	Trojan.JS.CRYPTOHIJACK.SMYXFII
8901bdcff8a93cc32d65f6dd5dc3a64bb702c37a	Trojan.JS.CRYPTOHIJACK.SMYXFII
60cb12384a8defcb020d996f16500cb4ae60544c	Trojan.JS.CRYPTOHIJACK.SMYXFII
f04799faa6add499aad64c9e50ecd8922656812d	Trojan.JS.CRYPTOHIJACK.SMYXFII
1fa896bff4d0aea2bdd90e2ca8ec58160d6e9130	Trojan.JS.CRYPTOHIJACK.SMYXFII
af9729d777b9837891f742db750bf35a0961c77e	Trojan.JS.CRYPTOHIJACK.SMYXFII
3cf10775ef49ed218730c2cfe9ac865d7d9782af	Trojan.JS.CRYPTOHIJACK.SMYXFII
87ecf3a97a3d760d99b1c7f91334d0e38ccc3089	Trojan.JS.CRYPTOHIJACK.SMYXFII
f8b9d1fd523282a9b620568927fb26daaeca4383	Trojan.JS.CRYPTOHIJACK.YXFIQ
e03f836f966616503dc4588bd17f80f4edf709cb	Trojan.JS.CRYPTOHIJACK.YXFIQ
7abebf9c56d06083102f0a01b10ace155c9e8855	Trojan.JS.CRYPTOHIJACK.YXFIQ
1e7dbe865a3e0408a319ee4a8b091add28452524	Trojan.JS.CRYPTOHIJACK.YXFIQ
1af9b4373657582edb9e20eab34b152be2ec70b4	Trojan.JS.CRYPTOHIJACK.YXFIQ
067648958b75806072527df55d9d3f727e4d2533	Trojan.JS.CRYPTOHIJACK.YXFIQ
47b63bc786960fda917ea9f5ff0023a3c50e2ca3	Trojan.JS.CRYPTOHIJACK.YXFIQ
ebcf69dc3d77aab6a23c733bf8d3de835a4a819a	Trojan.JS.CRYPTOHIJACK.YXFII
7c01f6ed54dc5c8dd7f3d44fb2c5e7baed2b8e84	Trojan.JS.CRYPTOHIJACK.YXFII

41b328df338a31e5afb05e4e37b3e89b29394523	Trojan.JS.CRYPTOHIJACK.YXFIJ
d4117240a8122c9f5c463a4d5b8a4d34cd243147	Trojan.JS.CRYPTOHIJACK.YXFIJ
bd398b4c641cc656510398bd70e181d572a9bc7b	Trojan.JS.CRYPTOHIJACK.YXFIJ
b43a8985746997b08842d55d32e8050dd943349a	Trojan.JS.CRYPTOHIJACK.YXFIJ
b28a3ab62a108d8094d2d2a8fa9f60a6af9189e6	Trojan.JS.CRYPTOHIJACK.YXFIJ
b0c9ed032985beda979cb0becba7b4a47b1de30c	Trojan.JS.CRYPTOHIJACK.YXFIJ
5518bc3a1df75f8e480efb32fa78de15e775155d	Trojan.JS.CRYPTOHIJACK.YXFIJ
9d893b6e0b50221889fbd2136d77112208746483	Trojan.JS.CRYPTOHIJACK.YXFIJ
9c14e3b712695d02c886df3503ce9ceadf67b99e	Trojan.JS.CRYPTOHIJACK.YXFIJ
3bc38b1fb607e2e393f0586ad137bec99e8a22dc	Trojan.JS.CRYPTOHIJACK.YXFIJ
8ad058047c5f2875f53cc12236cd715ab40918bb	Trojan.JS.CRYPTOHIJACK.YXFIJ
7e091778fdc88f043f3a5ad02647ca0ecb106311	Trojan.JS.CRYPTOHIJACK.YXFIJ
4b2d21961eb5ae538ae00c85655b28156c5135e3	Trojan.JS.CRYPTOHIJACK.YXFIJ
7a1ca7d142305e2886b988c2f0b524f89f003940	Trojan.JS.CRYPTOHIJACK.YXFIJ
78b18ee8f16e3d06997189ebac933c1048c74687	Trojan.JS.CRYPTOHIJACK.YXFIJ
014228830250bf081fce9db0826b10305bf4a075	Trojan.JS.CRYPTOHIJACK.YXFIJ
81f533be5a9ec9bb167634e509ed907896d6ea16	Trojan.JS.CRYPTOHIJACK.YXFIJ
2252418758a34f8b2708d13d641b8eea3a76a91c	Trojan.JS.CRYPTOHIJACK.YXFII
f416d1e4c19a8293306968d35fe27aa2be0a5d80	Trojan.JS.CRYPTOHIJACK.YXFII
c6490428b140893c27274b9e3bb33d2ab48a478d	Trojan.JS.CRYPTOHIJACK.YXFII
e97440fa7b29d5e4986bc88d7b2d8cec6f251267	Trojan.JS.CRYPTOHIJACK.YXFII
c577059020b7ae370c67cf0a3170eff4d7f2b038	Trojan.JS.CRYPTOHIJACK.YXFII
499756844aa7249d94c3ca3fd3f5346b3bdcabfe	Trojan.JS.CRYPTOHIJACK.YXFII
6323eac15e6029f92d7f53f786909dec04acc22a	Trojan.JS.CRYPTOHIJACK.YXFII
ef25127522cd65bf943000f78f9dd9bcdd8217f0	Trojan.JS.CRYPTOHIJACK.YXFII
70957568e6802538949197cf17709f8f29757c86	Trojan.JS.CRYPTOHIJACK.YXFII

URL	**Description**
webhook[.]site/bb8ca5f6-4175-45d2-b042-fc9ebb8170b7	C&C Server
npmjs[.]help	Malicious URL

Annex B - MITRE ATT&CK Tactics and Techniques

Tactic	**Technique ID**	**Remark**
Initial Access	T1566 – Phishing	Attack began with a phishing email masquerading as a npm security alert to trick a developer into revealing credentials.
Credential Access	T1528 – Steal Application Access Token	Harvested developer and CI/CD tokens, cloud service credentials, and environment variables from compromised hosts and developer environments and reused them to authenticate to APIs.
Execution	T1059.007 – Command & Scripting Interpreter: JavaScript	Deployed a "bundle.js" script that downloads and executes TruffleHog to collect credentials.

Command & Control / Ingress	T1105 – Ingress Tool Transfer	Downloaded and ran TruffleHog on infected hosts to scan for secrets and credentials.
Collection	T1005 / T1530 (credential & code collection)	Collected secrets from environment variables, config files, and cloud metadata; staged secrets and token material.
Discovery	T1592 – Gather Victim Network	Used GitHub access tokens stolen from TruffleHog to authenticate to GitHub API and enumerate repositories accessible to the victim.
Persistence	T1574 – Hijack Execution Flow	Deployed malicious npm packages and write unauthorised GitHub Actions workflow files ("shai-hulud.yaml" or "shai-hulud-workflow.yml") into compromised repositories. Execute workflows automatically when CI triggers after committing files.
Impact / Supply-Chain	T1195 – Supply Chain Compromise	Used stolen GitHub tokens to clone private repositories with "-migration" suffixes and inject malicious npm packages and unauthorised GitHub Actions workflows to compromise the software supply chain.
Exfiltration	T1567.001 – Exfiltration to Code Repository	Executed curl command to exfiltrated harvested secrets and token dumps to a webhook site.

References

1. What We Know About the NPM Supply Chain Attack

2. Massive Malicious NPM Package Attack Threatens Software Supply Chains

3. Self-Replicating Worm Hits 180+ npm Packages to Steal Credentials in Latest Supply Chain Attack