Executive Intelligence Brief

CRITICAL SEVERITY | ACTIVE INCIDENT | MARCH 31, 2026

Maintainer Account Hijack · Cross-Platform RAT Deployment · npm Package Poisoning

100–300M	~3 Hours	3 Platforms	~3%
Weekly Downloads Exposed	Malicious Versions Were Live	Win / Mac / Linux Targeted	Confirmed Exec Rate (Wiz)

SITUATION OVERVIEW

On March 31, 2026 (00:21–03:29 UTC), a threat actor compromised the npm maintainer account of the primary developer of **axios** — the JavaScript HTTP client used in roughly 80% of cloud and code environments. Two malicious releases were published (**axios@1.14.1** and **axios@0.30.4**), each containing a hidden dependency that automatically deployed a cross-platform Remote Access Trojan on any machine running npm install during the window. No axios source code was modified, making this difficult to detect through conventional review.

CONFIRMED

- Maintainer account (jasonsaayman) compromised
- Email changed to ifstap@proton.me to lock out owner
- Both 1.x and 0.x release branches poisoned
- Malicious dependency: plain-crypto-js@4.2.1
- Self-erasing dropper via npm postinstall hook (~15 sec)
- Cross-platform RAT: macOS, Windows, Linux
- Versions removed within ~3 hours

WORKING ASSUMPTIONS

- Initial access: long-lived token or credential theft
- Attack pre-staged 18+ hours in advance
- Possible link to TeamPCP supply chain campaign
- Full RAT capability set under forensic analysis
- Additional downstream packages may remain poisoned

ATTACK TIMELINE

#	EVENT	TIMESTAMP (UTC)
1	Clean decoy plain-crypto-js@4.2.0 published to establish registry history	Mar 30 — 05:57
2	Malicious plain-crypto-js@4.2.1 published with hidden postinstall dropper	Mar 30 — 23:59
3	axios@1.14.1 published via compromised account — injects malicious dep	Mar 31 — 00:21
4	axios@0.30.4 published — legacy 0.x branch poisoned 39 min later	Mar 31 — 01:00
5	npm removes both malicious axios versions from the registry	Mar 31 — ~03:15
6	npm tombstones plain-crypto-js with security-holder stub	Mar 31 — 04:26

AFFECTED SYSTEMS & RISK TIERS

ENVIRONMENT	EXPOSURE VECTOR	RISK
Developer workstations	Manual npm install or update during the exposure window	CRITICAL
CI/CD pipelines	Automated builds with caret ranges (^1.14.x) re-resolving	CRITICAL
Docker build environments	RUN npm install in Dockerfiles without lockfile pinning	CRITICAL
Cloud / serverless runners	Fresh installs on ephemeral workers during window	HIGH
Caret-range projects	^1.14.0 or ^0.30.0 auto-resolve to malicious version	HIGH
Transitive consumers	@shadanai/openclaw and @qqbrowser/openclaw-qbot affected	MEDIUM

HOW TO DETECT A COMPROMISE

PACKAGE / LOCKFILE CHECK

Search for the malicious dependency in lockfiles:

```
grep -r "plain-crypto-js" ./package-lock.json
ls node_modules/plain-crypto-js
```

AXIOS VERSION CHECK

BAD: axios@1.14.1 or axios@0.30.4

SAFE: axios@1.14.0 or prior

RAT ARTIFACT PATHS BY OS

macOS:

```
/Library/Caches/com.apple.act.mond
```

Windows:

```
%PROGRAMDATA%\wt.exe
```

Linux:

```
/tmp/ld.py
```

Network IOC: Search firewall, DNS, and proxy logs for connections to `sfrclak.com` or `sfrclak.com:8000`. Any hit confirms RAT execution. **Note:** the dropper self-deletes — absence of the package in node_modules does NOT rule out compromise if build logs show it was installed.

IMMEDIATE RESPONSE ACTIONS

NOW (0–2 hrs)

- Block sfrclak.com at DNS/firewall; alert on all historical hits
- Audit CI/CD logs for the 00:21–03:29 UTC window on March 31
- Identify all environments that resolved axios@1.14.1 or 0.30.4
- Isolate affected systems from the network — do not clean in place

SHORT TERM (2–24 hrs)

- Pin axios to @1.14.0 in package.json; regenerate all lockfiles
- Run npm audit — check for GHSA-fw8c-xr5c-95f9
- Rebuild Docker images and containers built during the window
- Rotate ALL credentials: npm tokens, cloud keys, SSH, CI secrets, .env values

STRATEGIC (24–72 hrs)

- Enforce Granular Access Tokens and mandatory 2FA on all npm publisher accounts
- Implement trusted publishing via CI/CD — eliminate long-lived personal tokens
- Deploy runtime egress monitoring on build runners to catch C2 callbacks
- Adopt lockfile-first installs (npm ci) and SBOM generation across all pipelines

INDICATORS OF COMPROMISE (IOC REFERENCE)

TYPE	INDICATOR	NOTES
Package	`axios@1.14.1`	Malicious npm release — 1.x branch
Package	`axios@0.30.4`	Malicious npm release — 0.x branch
Package	`plain-crypto-js@4.2.1`	Malicious dropper dependency
Domain/Port	`sfrclak.com / :8000`	C2 command-and-control server
File (macOS)	`/Library/Caches/com.apple.act.mond`	Dropped RAT artifact
File (Win)	`%PROGRAMDATA%\wt.exe`	Dropped RAT artifact
File (Linux)	`/tmp/ld.py`	Dropped Python RAT
npm Account	`jasonsaayman / ifstap@proton.me`	Compromised publisher identity
Advisory	`GHSA-fw8c-xr5c-95f9 / MAL-2026-2306`	Security advisory identifiers

STRATEGIC NOTE

This incident demonstrates that even a two-to-three hour exposure window on a package with 100M+ weekly downloads produces meaningful real-world compromise. The attacker showed significant operational sophistication: pre-staging, dual-branch poisoning, triple-OS payloads, and anti-forensic self-deletion. Organizations should treat this as a forcing function to adopt lockfile-first installs, runtime egress controls on build runners, SBOM generation, and mandatory MFA for all npm publishing accounts.

Sources: StepSecurity · Socket.dev · Snyk · Wiz Security · Help Net Security · The Hacker News · Aikido.dev · GHSA-fw8c-xr5c-95f9