INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-13 06:37 UTC

AUR Supply Chain Attack: 400+ Packages Weaponized with eBPF Rootkit and Credential Stealer via npm Dependency

THREAT CAMPAIGN | **CRITICAL** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0449
Type	Threat Campaign
Severity	CRITICAL
CVSS Base Score	9.5
Affected Products	Arch Linux AUR (400+ packages), npm package 'atomic-lockfile', developer workstations running Arch Linux; secondary exposure: SSH keys, GitHub tokens, HashiCorp Vault secrets, Docker/Podman credentials, Slack/Discord/Microsoft Teams/Telegram session data, browser credential stores
Published	2026-06-12T13:03:55
Discovery Source	Rss

Executive Summary

Threat actors compromised more than 400 packages in the Arch Linux User Repository (AUR) by injecting malicious install scripts that deploy an eBPF rootkit and credential harvester onto developer workstations. The malware targets SSH private keys, cloud secrets, container credentials, and messaging session tokens, assets that grant downstream access to CI/CD pipelines, code repositories, and enterprise infrastructure. A single compromised developer machine can become the entry point for a broader supply chain attack affecting every organization that consumes that developer's code or infrastructure output.

Technical Analysis

Threat actors injected malicious preinstall and post-install hooks into 400+ AUR packages. The hooks fetch the npm package 'atomic-lockfile,' which drops a Linux ELF binary combining an eBPF-based rootkit (MITRE T1014) with a credential harvesting module. The eBPF component operates at kernel level, enabling process hiding and network activity concealment without requiring a traditional kernel module. The harvester targets: SSH private keys (T1552.004), GitHub personal access tokens, HashiCorp Vault secrets, Docker/Podman runtime credentials (T1552.001), browser-stored credentials (T1555.003, T1555), and messaging session tokens for Slack, Discord, Microsoft Teams, and Telegram (T1539). Delivery exploits the AUR's community-maintained, elevated-trust model via dependency confusion and supply chain injection (T1195.002,

T1195.001). Persistence is established via system service or init mechanism (T1543). Exfiltration occurs over standard HTTP/S channels (T1071.001, T1041). The compromised npm package weaponizes the software build process itself (T1554). Relevant weaknesses: CWE-732 (incorrect permission assignment), CWE-494 (download of code without integrity check), CWE-506 (embedded malicious code), CWE-312 (cleartext storage of sensitive information), CWE-829 (inclusion of functionality from untrusted control sphere). No CVE assigned. No patch is available from a vendor; remediation is procedural. Source quality score is 0.64; primary reporting from Sonatype research and BleepingComputer; treat specific package lists as subject to change as investigation matures.

Action Checklist

1. Step 1: Containment. Immediately audit all Arch Linux developer workstations for AUR package installations made in the past 90 days. Isolate any system that installed packages from the affected AUR set from the corporate network, CI/CD systems, and code repositories. After isolation is confirmed, generate new SSH keypairs, GitHub personal access tokens, HashiCorp Vault tokens, Docker/Podman credentials, and messaging session tokens; do not restore or reuse any credential that existed on an affected machine prior to isolation.
2. Step 2: Detection. Search developer workstations for the npm package 'atomic-lockfile' in node_modules directories and npm cache (run: find / -name 'atomic-lockfile' -type d 2>/dev/null). Look for unexpected Linux ELF binaries dropped during npm install hooks in /tmp, /var/tmp, or user home directories. Check for unusual eBPF program loads via 'bpftool prog list' or auditd records for bpf() syscalls from non-standard processes. Review ~/.ssh/ for recently accessed or modified private key files. Audit outbound network connections from developer hosts to unexpected external IPs over ports 80/443 (NIST AU-6, AU-12; CIS 8.2). Query EDR telemetry for process trees spawned by npm install hooks that invoke curl, wget, or exec on ELF binaries.
3. Step 3: Eradication. Remove all AUR packages installed from untrusted or unverified maintainers; cross-reference against the reported affected package list as published by Sonatype and updated by BleepingComputer. Purge 'atomic-lockfile' and any associated ELF binaries. Reinstall Arch Linux from a known-good image on any confirmed-compromised workstation; do not attempt to clean in place given the rootkit's kernel-level stealth capability. Rotate all credentials confirmed or suspected to have been on the affected machine: SSH keys (generate new keypairs, update authorized_keys on all target systems), GitHub tokens (revoke in GitHub Settings > Developer Settings > Personal Access Tokens), Vault tokens (revoke via vault token revoke), Docker Hub and registry credentials, and messaging platform session tokens (force re-authentication).
4. Step 4: Recovery. After workstation re-imaging and credential rotation, verify no unauthorized SSH public keys persist in authorized_keys files on downstream servers (NIST AC-2, AC-3; CIS 5.1). Audit GitHub repository access logs for commits, clones, or forks made after the estimated compromise window. Review HashiCorp Vault audit logs for secret reads. Scan CI/CD pipeline configurations for injected steps or modified scripts. Re-enable developer access only after credential rotation is confirmed complete and the workstation passes a clean build verification. Monitor for re-infection via the same vector by blocking AUR package installs that invoke npm postinstall scripts not present in the prior approved baseline (NIST SI-4; CIS 7.1).
5. Step 5: Post-Incident. Implement mandatory code review for all AUR PKGBUILD files before installation, using a dedicated review process analogous to internal code review workflows (NIST CM-7, AC-6). Enforce software supply chain integrity checks: verify package checksums and signatures before

installation (CWE-494 gap closure; CIS 7.3, 7.4). Adopt a policy prohibiting AUR package installation directly on developer machines that have network access to production CI/CD or infrastructure credentials; use isolated build environments instead. Evaluate D3FEND countermeasures D3-SFA (system file analysis) and D3-LAM (local account monitoring) for developer workstation endpoint controls. Establish a developer workstation hardening baseline referencing NIST AC-6 (least privilege) and CIS 4.6 (securely manage enterprise assets and software).

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to senior leadership, legal counsel, and potentially regulatory authorities if forensic evidence confirms that rotated credentials were used to access production systems, code repositories were modified during the compromise window, or regulated data (PII, PHI, financial records) was accessible via secrets stored on compromised developer workstations — any of these conditions may trigger breach notification obligations.
Recovery Notes	Post-reimaging, maintain enhanced monitoring on all downstream systems that accepted SSH keys or API tokens from potentially compromised developer workstations for a minimum of 90 days, specifically watching for anomalous access patterns during off-hours or from unusual source IPs. Re-validate the integrity of all CI/CD pipeline outputs (build artifacts, container images, signed releases) produced during the estimated compromise window before they are deployed to production, as the eBPF rootkit's kernel-level stealth capability means in-place forensics cannot reliably rule out pipeline tampering. Treat any code signing keys, deploy keys, or infrastructure-as-code credentials that resided on affected machines as fully compromised regardless of whether forensic evidence of their use was found.
Forensic Artifacts	/sys/fs/bpf/ — pinned eBPF objects left by the rootkit; enumerate with 'bpftool map list' and 'bpftool prog list' before host isolation to capture program IDs, names, and loaded-by process context specific to this eBPF rootkit | ~/.npm/_logs/ and ~/.npm/_cacache/ — npm debug logs and cached tarballs recording postinstall hook execution from 'atomic-lockfile', including the exact ELF binary dropper command line and any error output that reveals C2 staging infrastructure | /var/log/pacman.log — Arch Linux package manager transaction log recording the exact timestamp and package name of every AUR install in the 90-day window, enabling precise compromise window establishment for credential rotation scoping | /var/log/audit/audit.log filtered for 'syscall=bpf' and 'syscall=execve' — auditd records linking the bpf() syscall invocation to the specific npm postinstall hook process tree, establishing the rootkit load chain from PKGBUILD to kernel | ~/.ssh/id_* and ~/.vault-token and ~/.docker/config.json file access timestamps ('stat' output) — inode atime records showing when the credential stealer component accessed private key material and cloud secrets, establishing the exfiltration timeline relative to the AUR package install event

Per-Action IR Details

Step 1: Containment — Immediately audit all Arch Linux developer workstations for AUR package installations made in the past 90 days. Isolate any system that installed packages from the affected AUR set from the corporate network, CI/CD systems, and code repositories. Revoke all SSH keys, GitHub personal access tokens, HashiCorp Vault tokens, Docker/Podman credentials, and messaging session tokens that were accessible from potentially compromised systems. Do not reuse any credential that existed on an affected machine.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy

Controls: NIST AC-2 (Account Management), NIST AC-12 (Session Termination), CIS 6.2 (Establish an Access Revoking Process)

Compensating: For teams without NAC or EDR: run 'pacman -Qm' on each Arch workstation to list all foreign (AUR) packages with install dates, then cross-reference output against the Sonatype-published affected package list using a bash diff loop. To audit SSH key exposure without EDR, parse 'last -F' and ~/.bash_history for ssh-agent invocations and SSH key file access timestamps ('stat ~/.ssh/id_*'). Use 'ausearch -sc bpf --start today' on any host with auditd enabled to surface bpf() syscall activity before network isolation cuts audit log forwarding.

Evidence: Before isolating the host or revoking any credential, capture: (1) full RAM image using LiME kernel module ('sudo insmod lime.ko path=/external/ram.lime format=lime') to preserve eBPF program maps and in-memory credential buffers the rootkit may be holding; (2) 'bpftool prog list' and 'bpftool map list' output to enumerate live eBPF programs loaded by the rootkit; (3) 'ss -tnp' or 'netstat -ano' to record active outbound C2 connections before network cut; (4) 'pacman -Qm --info' for all AUR packages to timestamp installs; (5) copies of ~/.ssh/known_hosts, ~/.ssh/authorized_keys, and shell history files (~/.bash_history, ~/.zsh_history) which may record credential exfiltration commands injected by the PKGBUILD postinstall hook.

Step 2: Detection — Search developer workstations for the npm package 'atomic-lockfile' in node_modules directories and npm cache (run: find / -name 'atomic-lockfile' -type d 2>/dev/null). Look for unexpected Linux ELF binaries dropped during npm install hooks in /tmp, /var/tmp, or user home directories. Check for unusual eBPF program loads via 'bpftool prog list' or auditd records for bpf() syscalls from non-standard processes. Review ~/.ssh/ for recently accessed or modified private key files. Audit outbound network connections from developer hosts to unexpected external IPs over ports 80/443 (NIST AU-6, AU-12; CIS 8.2). Query EDR telemetry for process trees spawned by npm install hooks that invoke curl, wget, or exec on ELF binaries.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), NIST AU-12 (Audit Record Generation), CIS 8.2 (Collect Audit Logs)

Compensating: Without EDR: deploy Sysmon for Linux (sysmon-for-linux) with a config capturing ProcessCreate, FileCreate, and NetworkConnect events, then grep for 'npm' parent processes spawning 'curl', 'wget', 'sh', or 'bash' child processes. Use 'find /tmp /var/tmp $HOME -name "*.elf" -o -name ".* " -newer /usr/bin/npm -type f 2>/dev/null' to locate dropped ELF binaries. For eBPF detection without EDR, run 'cat /proc/*/maps | grep -i bpf' and 'ls -la /sys/fs/bpf/' to enumerate pinned eBPF objects. Use Wireshark or 'tcpdump -i any -w /external/cap.pcap port 80 or port 443' before isolation to record C2 beacon traffic patterns from the credential stealer.

Evidence: This step is detection and does not alter live state, but capture before any subsequent action: (1) full output of 'bpftool prog dump xlated id ' for each non-standard eBPF program to preserve rootkit bytecode; (2) 'find ~/.npm/_cacache -name "*.tgz" | xargs -I{} tar -tzf {} | grep -i postinstall' to extract postinstall script payloads from npm cache before cache purge; (3) auditd log entries for bpf() syscalls — default path /var/log/audit/audit.log, filter with 'ausearch -sc bpf'; (4) file access timestamps on ~/.ssh/id_rsa, ~/.config/gh/hosts.yml (GitHub CLI token), ~/.vault-token, and ~/.docker/config.json to establish the credential access window; (5) process accounting logs (/var/log/pacct if enabled) to reconstruct PKGBUILD execution chains.

Step 3: Eradication — Remove all AUR packages installed from untrusted or unverified maintainers; cross-reference against the reported affected package list as published by Sonatype and updated by BleepingComputer. Purge 'atomic-lockfile' and any associated ELF binaries. Reinstall Arch Linux from a known-good image on any confirmed-compromised workstation — do not attempt to clean in place given the rootkit's kernel-level stealth capability. Rotate all credentials confirmed or suspected to have been on the affected machine: SSH keys (generate new keypairs, update authorized_keys on all target systems), GitHub tokens (revoke in GitHub Settings > Developer Settings > Personal Access Tokens), Vault tokens (revoke via vault token revoke), Docker Hub and registry credentials, and messaging platform session tokens (force re-authentication).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST AC-2 (Account Management), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 2.3 (Address Unauthorized Software)

Compensating: For teams without automated reimaging infrastructure: use a verified Arch Linux ISO (validate SHA256 against https://archlinux.org/download/ before use) to perform a bare-metal reinstall, preserving only user data directories after scanning them with ClamAV ('clamscan -r --remove /mnt/userdata'). For credential rotation without a secrets manager: use a scripted approach — 'ssh-keygen -t ed25519 -C "rotated-$(date +%Y%m%d)"' for new SSH keys and automate authorized_keys replacement on downstream servers via a temporary jump host that was not in the compromised AUR install scope. Document each rotation action with timestamps for the post-incident record.

Evidence: Before reimaging or rotating any credential, ensure all volatile captures from Steps 1 and 2 are complete and stored to external read-only media. Additionally capture: (1) a full disk image of the compromised workstation using 'dd if=/dev/sda bs=4M | gzip > /external/compromised-disk.img.gz' or similar forensic imaging tool before wipe, preserving eBPF object files potentially pinned under /sys/fs/bpf/ and any rootkit artifacts in kernel module load paths (/lib/modules/$(uname -r)/); (2) copy of /var/log/pacman.log showing full AUR install history with timestamps; (3) copy of npm debug logs (~/.npm/_logs/) which record postinstall hook execution and any error output from the malicious ELF dropper.

Step 4: Recovery — After workstation re-imaging and credential rotation, verify no unauthorized SSH public keys persist in authorized_keys files on downstream servers (NIST AC-2, AC-3; CIS 5.1). Audit GitHub repository access logs for commits, clones, or forks made after the estimated compromise window. Review HashiCorp Vault audit logs for secret reads. Scan CI/CD pipeline configurations for injected steps or modified scripts. Re-enable developer access only after credential rotation is confirmed complete and the workstation passes a clean build verification. Monitor for re-infection via the same vector by blocking AUR package installs that invoke npm postinstall scripts not present in the prior approved baseline (NIST SI-4; CIS 7.1).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AC-2 (Account Management), NIST AC-3 (Access Enforcement), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Without a SIEM for GitHub audit log analysis: use the GitHub Audit Log API ('curl -H "Authorization: token " https://api.github.com/orgs//audit-log?phrase=action:git.push&include=git') to export push/clone/fork events during the compromise window into a CSV for manual review. For Vault audit log review without a log aggregator, parse /var/log/vault/vault-audit.log directly filtering for 'operation":"read"' entries against secrets paths accessible to the compromised developer. For CI/CD pipeline integrity, use 'git diff HEAD -- .github/workflows/' to surface any injected steps in GitHub Actions YAML files.

Evidence: Recovery verification requires confirming no attacker persistence artifacts remain; collect before re-enabling developer access: (1) diff of all authorized_keys files on downstream servers against the pre-incident baseline ('diff <(sort authorized_keys.backup) <(sort ~/.ssh/authorized_keys)') to detect keys added by the eBPF credential stealer; (2) GitHub repository event logs for the compromise window, specifically filtering for push events from IP addresses not associated with the developer's known locations; (3) HashiCorp Vault audit log entries showing secret reads on paths containing CI/CD credentials, cloud provider keys, or signing certificates during the compromise window; (4) CI/CD pipeline run history and configuration file hashes to detect injected build steps that could establish persistence in the supply chain downstream of the compromised workstation.

Step 5: Post-Incident — Implement mandatory code review for all AUR PKGBUILD files before installation, using a dedicated review process analogous to internal code review workflows (NIST CM-7, AC-6). Enforce software supply chain integrity checks: verify package checksums and signatures before installation (CWE-494 gap closure; CIS 7.3, 7.4). Adopt a policy prohibiting AUR package installation directly on developer machines that have network access to production CI/CD or infrastructure credentials — use isolated build environments. Evaluate D3FEND countermeasures D3-SFA (system file analysis) and D3-LAM (local account

monitoring) for developer workstation endpoint controls. Establish a developer workstation hardening baseline referencing NIST AC-6 (least privilege) and CIS 4.6 (securely manage enterprise assets and software).

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity

Controls: NIST AC-6 (Least Privilege), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), CIS 4.6 (Securely Manage Enterprise Assets and Software), MITRE D3FEND D3-SFA (System File Analysis), MITRE D3FEND D3-LAM (Local Account Monitoring)

Compensating: For PKGBUILD review without a commercial supply chain tool: implement a mandatory peer-review Git workflow where PKGBUILDs are committed to an internal repository and reviewed via pull request before 'makepkg' is invoked — specifically inspect the 'prepare()', 'build()', and 'package()' functions and all 'source=' entries for remote downloads that lack a corresponding sha256sums entry. For npm postinstall hook auditing without commercial tooling: use 'npm pack ' followed by 'tar -tzf .tgz | xargs -I{} sh -c "tar -xOf .tgz {} 2>/dev/null"' to inspect package contents, and create a YARA rule matching ELF magic bytes (0x7f454c46) dropped to /tmp or /var/tmp by node processes to flag this specific dropper pattern.

Evidence: Post-incident evidence collection to drive control improvements: (1) complete timeline reconstruction of PKGBUILD postinstall hook execution across all affected workstations, correlating pacman.log timestamps with auditd bpf() syscall records to establish the eBPF rootkit load sequence; (2) list of all unique external IPs and domains contacted by the credential stealer, extracted from pre-isolation network captures, for IOC sharing and firewall block-list creation; (3) inventory of all credentials confirmed or suspected accessed, mapped to downstream systems they granted access to, to scope the full blast radius for breach notification decisions; (4) documentation of which CI/CD pipelines and infrastructure components were reachable via credentials on compromised workstations, to assess whether downstream systems require their own forensic review.

Detection Guidance

Primary detection targets: the npm package 'atomic-lockfile' on developer hosts, eBPF rootkit activity, and credential access patterns. Run 'find / -path '*/node_modules/atomic-lockfile' 2>/dev/null' and 'npm ls -g atomic-lockfile 2>/dev/null' on all Arch Linux developer machines. Use 'bpftool prog list' to enumerate loaded eBPF programs; flag any program loaded by a process outside known security or observability tooling. Enable auditd rules for the bpf() syscall (syscall=321 on x86_64) and alert on calls from npm, node, sh, bash, or any installer process. Look for ELF binary execution spawned as child processes of npm or package managers in /tmp or home directories. Monitor outbound HTTPS from developer workstations to domains or IPs not in an approved baseline, particularly short-lived connections immediately following package install events (MITRE T1071.001, T1041). Review ~/.ssh/id_* file access timestamps for reads outside normal developer working hours (T1552.004). Check git credential stores (~/.git-credentials, ~/.netrc) and browser profile directories (Chrome: ~/.config/google-chrome/Default/Login Data; Firefox: ~/.mozilla/firefox/*/logins.json) for recent access by non-browser processes (T1555.003). For CI/CD exposure: review GitHub Actions workflow run logs for unexpected jobs, secret references, or external network calls during the compromise window. No confirmed file hashes or C2 IP addresses are available from verified sources at this time; treat IOCs as preliminary pending Sonatype's full disclosure.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`atomic-lockfile (npm package name)`	Malicious npm package delivering the eBPF rootkit and credential harvester; pulled via AUR package postinstall hooks	HIGH
URL	`https://www.npmjs.com/package/atomic-lockfile`	npm registry entry for the malicious package — search-retrieved, requires human validation before use as a block rule	MEDIUM

Framework Mappings

MITRE-ATTACK

- **T1014** — Rootkit
- **T1195.002** — Compromise Software Supply Chain
- **T1071.001** — Web Protocols
- **T1552.001** — Credentials In Files
- **T1554** — Compromise Host Software Binary
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1059.004** — Unix Shell
- **T1543** — Create or Modify System Process
- **T1041** — Exfiltration Over C2 Channel
- **T1078** — Valid Accounts
- **T1555.003** — Credentials from Web Browsers
- **T1555** — Credentials from Password Stores
- **T1539** — Steal Web Session Cookie
- **T1552.004** — Private Keys

NIST-800-53R5

- **CM-7** — Least Functionality
- **SA-9** — External System Services
- **SR-3** — Supply Chain Controls and Processes
- **SI-7** — Software, Firmware, and Information Integrity
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)

- **IA-5** — Authenticator Management
- **AC-3** — Access Enforcement
- **CM-3** — Configuration Change Control
- **SR-2** — Supply Chain Risk Management Plan

OWASP-TOP10-2021

- **A01:2021** — Broken Access Control
- **A08:2021** — Software and Data Integrity Failures

CIS-V8

- **3.3** — Configure Data Access Control Lists
- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **6.3** — Require MFA for Externally-Exposed Applications
- **15.1** — Establish and Maintain an Inventory of Service Providers

HIPAA-SECURITY

- **164.312(d)** — Person or Entity Authentication

SOC2-TSC

- **CC6.1** — Logical access security software, infrastructure, and architectures
- **CC9.2** — Manages risks associated with vendors and business partners

NIST-CSF-2

- **GV.SC-01** — Cybersecurity supply chain risk management program

ISO-27001-2022

- **A.5.21** — Managing information security in the ICT supply chain

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1014	Rootkit	Defense-Evasion
T1195.002	Compromise Software Supply Chain	Initial-Access
T1071.001	Web Protocols	Command-And-Control
T1552.001	Credentials In Files	Credential-Access
T1554	Compromise Host Software Binary	Persistence
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1059.004	Unix Shell	Execution

Technique ID	Technique Name	Tactic
T1543	Create or Modify System Process	Persistence
T1041	Exfiltration Over C2 Channel	Exfiltration
T1078	Valid Accounts	Defense-Evasion
T1555.003	Credentials from Web Browsers	Credential-Access
T1555	Credentials from Password Stores	Credential-Access
T1539	Steal Web Session Cookie	Credential-Access
T1552.004	Private Keys	Credential-Access

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/over-400-arch-linux-...	T3
Over 400 Arch Linux packages compromised to push rootkit ... - Reddit	https://www.reddit.com/r/cybersecurity/comments/1u41u02/over_400_ar...	T3
Atomic Arch npm Campaign Adds Malicious Dependency - Sonatype	https://www.sonatype.com/blog/atomic-arch-npm-campaign-adds-malicio...	T3
Arch Linux AUR Hit By NEW Malware Attack Over 1300 ... - YouTube	https://www.youtube.com/watch?v=ZhWxVH8hd3Q	T3
Malicious Commits in Arch Linux AUR Packages Pulled npm-Based ...	https://www.mallory.ai/stories/019eb95c-48fc-7268-992c-72efefd1b205	T3

Generated 2026-06-13 06:37 UTC by TJS Security Command Center