CSAI Foundation

Cloud Security Alliance AI Safety Initiative

CanisterSprawl: Developer Toolchain Worm and the TeamPCP Campaign

How Self-Propagating npm Malware Is Targeting AI Developer Environments

Unofficial AI-assisted Research

2026-04-24

Key Takeaways

- **CanisterSprawl** is a self-propagating npm worm discovered on April 21, 2026, infecting packages from Namastex Labs, an agentic AI tooling company. At least 16 packages confirmed compromised or re-infected across multiple namespaces—including @automagik and @opengov—with active publication of new malicious versions continuing at time of writing [1][2].
- The worm is attributed to **TeamPCP**, a threat actor responsible for a March 2026 campaign (CanisterWorm) that compromised Aqua Security's Trivy scanner, Checkmarx KICS, LiteLLM, and more than 47 npm packages [3][4].
- **TeamPCP uses Internet Computer Protocol (ICP) blockchain canisters** as command-and-control infrastructure—representing the first publicly documented operational use of decentralized blockchain for C2 in open threat intelligence reporting—making traditional takedown approaches ineffective [3][5].
- The malware explicitly targets credentials for AI developer tools, including MCP configuration files, LLM API keys, and CI/CD secrets, extending the threat surface into agentic AI workflows [2][6].
- Within 48 hours of CanisterSprawl's initial discovery, TeamPCP's cascading campaign also compromised official Checkmarx KICS Docker Hub images and the Bitwarden CLI npm package (approximately 78,000 weekly downloads), affecting security tooling itself [6][7][8].
- Organizations using npm packages—especially those in AI/ML, agentic tooling, or CI/CD automation—should audit installed packages, rotate all developer credentials, and disable postinstall hooks in untrusted contexts immediately.

Background

The TeamPCP campaign illustrates how far software supply chain attacks have advanced from early typosquatting incidents: the March 2026 CanisterWorm alone compromised more than 47 npm packages through multi-stage token harvesting and self-replication rather than simple package name imitation [3]. The campaign documented under the TeamPCP designation targets the tools developers use to build and secure software—scanners, linters, CI/CD integrations—and weaponizes those tools to

propagate further into developer pipelines. Compromising security vendors such as Checkmarx and Aqua Security is a calculated approach: it converts supply chain integrity tooling itself into an initial access vector, reaching organizations whose security posture otherwise assumes that scanning infrastructure is trustworthy.

TeamPCP's activity was first formally characterized in a March 2026 wave that affected more than 47 npm packages across a three-week period [3]. The campaign began with the compromise of Aqua Security's Trivy vulnerability scanner: attackers with access to Aqua's GitHub organization published a malicious Trivy release (v0.69.4) containing a hardcoded connection to a typosquat domain, and simultaneously backdoored the trivy-action and setup-trivy GitHub Actions [4]. Stolen npm tokens harvested from CI/CD runners that used these actions became the seed material for a self-replicating worm—CanisterWorm—that propagated across the npm ecosystem between March 20 and March 23, 2026. Additional victims in that wave included Checkmarx's KICS static analysis tool, the LiteLLM AI gateway library, and the Telnyx Python SDK [4].

What distinguished CanisterWorm technically was its command-and-control channel. Rather than relying on a domain registrar-managed server or a traditional IP-based endpoint, TeamPCP hosted its C2 payload inside an Internet Computer Protocol (ICP) blockchain canister—a smart contract executing on the decentralized ICP network [5]. The technical architecture means that law enforcement and security operations teams cannot apply their standard playbook for disrupting infrastructure: there is no domain to sinkhole, no IP to null-route, and no hosting provider to contact. The malware polls this canister for commands and uses it as an exfiltration endpoint, and the canister itself is append-only and content-addressed, meaning removal requires compromise of the canister's controller key [5].

CSA's AI Safety Initiative documented TeamPCP's March activity in a prior research note [9], flagging it as an early indicator of a sustained campaign against the AI/ML development ecosystem. The April 2026 CanisterSprawl incidents confirm that prediction.

Security Analysis

CanisterSprawl: Technical Mechanics

On April 21, 2026, Socket and StepSecurity jointly detected malicious versions of `pgserve`—a PostgreSQL server package for Node.js development—appearing in the npm registry [1][2]. The affected packages are associated with Namastex Labs (Namastex.ai), a company that provides AI consulting and

open-source autonomous agent tooling under the Automagik brand. At least 16 packages across multiple namespaces have been confirmed compromised, including `@automagik/genie`, a CLI positioned for AI agent orchestration [1][10].

The attack activates at install time, before any application code runs—npm's `postinstall` hook fires automatically during `npm install`, requiring no further developer action. This install-time execution model is particularly dangerous because it activates in developer laptops, build servers, and CI/CD runners alike, wherever `npm install` or equivalent commands run. The mechanism involves a 1,143-line credential harvesting script that executes without any user interaction [1][2].

The harvesting script operates in three stages. First, it sweeps environment variables for names matching known patterns for tokens, credentials, cloud provider access keys, CI/CD system secrets, container registry credentials, and LLM platform API keys [2][6]. It explicitly targets `.npmrc` files, `.git-credentials`, `.netrc`, `.env` files, SSH key directories, cloud credential stores for AWS, Google Cloud, and Azure, and MCP configuration files used by AI coding assistants [6][8]. Second, for each npm publishing token it finds, the script identifies every package the victim account can publish, bumps the patch version, injects itself into the postinstall hook, and publishes the newly poisoned version to npm [1][2]. Third, if a PyPI token is also present, the worm jumps ecosystems entirely and republishes backdoored versions to the Python Package Index, illustrating that registry boundaries offer no isolation if publishing credentials are compromised—an attacker with valid tokens can republish across any registry they have access to [2][10][14].

The cross-ecosystem propagation is a meaningful escalation from the March CanisterWorm behavior. When a developer in an AI/ML organization installs what they believe is a trusted agentic tooling package, they may inadvertently become the vector through which a dozen or more of their own published packages—potentially used by colleagues, customers, and open-source dependents—become compromised.

ICP Blockchain C2: Why Conventional Defenses Fall Short

TeamPCP's use of ICP canisters as C2 infrastructure is not merely a technical curiosity; it represents a deliberate and effective evasion of the defender's standard response toolkit [3][5]. When a malicious domain is identified, registrars can suspend it. When a malicious IP is identified, ISPs can null-route it. ICP canisters have no equivalent lever. They are deployed to a distributed network of nodes, identified only by their content-addressed canister ID, and are operationally live as long as the ICP network functions.

This architecture introduces a durable asymmetry: defenders must identify the canister ID (obtained through dynamic analysis of the malware binary), request voluntary action from the Internet Computer Association (if such a process exists), or wait for the attacker to abandon the canister. None of these are fast paths. Security vendors and government agencies that built their playbooks around domain sinkholing or IP blocking face a gap here that requires policy and technical responses at a level above the individual organization.

For defenders at the enterprise level, one practical network-layer option is blocking outbound connections to known ICP node IP ranges from developer workstations and build runners, though this approach requires maintaining a current list of ICP node endpoints and may produce false positives where legitimate ICP applications are in use.

The Checkmarx and Bitwarden Cascade

Within the same 48-hour window as CanisterSprawl's discovery, TeamPCP executed what GitGuardian characterized as a concurrent multi-platform campaign hitting npm, PyPI, and Docker Hub simultaneously [11]. On April 22, Docker's internal monitoring flagged suspicious activity in the official `checkmarx/kics` Docker Hub repository [7][15]. Analysis revealed that attackers had overwritten existing image tags—including `v2.1.20` and `alpine`—and introduced a new `v2.1.21` tag with no corresponding legitimate upstream release [7]. The poisoned KICS images contained a bundled binary modified to collect and exfiltrate CI/CD environment variables and runner process memory. The exposure window on April 22 was under 90 minutes, but sufficient for automated CI pipelines pulling `latest` or `alpine` tags to receive the malicious image during that interval [7][11].

The Bitwarden CLI compromise illustrates the cascading logic of this attack class with particular clarity. Bitwarden's CI/CD pipeline uses Checkmarx's `ast-github-action` for security scanning [8]. When that GitHub Action was compromised in the Checkmarx supply chain breach, the attacker gained the ability to inject a malicious GitHub Actions workflow into Bitwarden's build process. The resulting artifact – `@bitwarden/cli@2026.4.0` –was distributed through the official npm registry to approximately 78,000 weekly download consumers [6]. The malicious package, internally self-identified as "Shai-Hulud: The Third Coming" in a reference to previous supply chain campaigns, implemented a multi-stage credential harvester, a self-propagating npm worm, a GitHub commit-based C2 dead-drop channel using RSA-signed command delivery, and a module specifically targeting AI coding assistant configurations [6][8][13]. Bitwarden confirmed the exposure window was approximately 93 minutes on the afternoon of April 22, and stated that no end-user vault data was accessed or at risk [12].

TeamPCP claimed credit publicly through the `@pcpcats` social media account, posting taunting messages after coverage of the Checkmarx incident broke—"Thank you OSS distribution for another very successful day at PCP inc."—suggesting significant operational confidence or a deliberate reputational signaling strategy. The actor's ultimate motivation—whether financial, ideological, or nation-state-directed—has not been publicly attributed by law enforcement or major threat intelligence vendors at time of writing [7].

AI Developer Environment as the Target Surface

A consistent pattern across all three campaign phases is the systematic targeting of AI developer tooling, including MCP configuration files, LLM API keys, and AI coding assistant contexts. The CanisterSprawl harvesting script explicitly searches for MCP configuration files, which AI coding assistants such as Claude Code use to store tool server endpoints and associated credentials [6]. LLM API keys from Anthropic, OpenAI, and other providers appear on the list of targeted environment variable patterns. The Bitwarden CLI variant specifically implements a module aimed at authenticated AI coding assistants [6][8].

This targeting pattern is consistent with the strategic value of agentic AI workflows as a persistent access vector. An AI coding agent operating in a developer's environment typically has access to file systems, shell execution, external API calls, and repository operations. If the environment credentials available to that agent—its MCP configuration, its API keys, its GitHub tokens—are stolen and replicated, an attacker gains not just credentials but a potential persistent foothold inside any future agentic workflow the developer runs. The convergence of npm supply chain attacks and AI tool targeting suggests that adversaries have recognized the developer AI environment as a high-value initial access vector.

Recommendations

Immediate Actions

Organizations should treat any npm install performed in a developer environment or CI/CD pipeline between April 19 and April 24, 2026, as a potential credential exposure event if packages from Namastex Labs, Checkmarx, or Bitwarden were involved.

- **Rotate all developer credentials** that could have been exposed through npm postinstall execution: npm tokens, GitHub tokens, AWS/Azure/GCP access keys, SSH keys, and any LLM API keys stored in `.env` files or MCP configurations.

- **Audit recent npm installs** against the list of confirmed compromised Namastex Labs packages (including `pgserve`, `@automagik/genie`, and related packages in the automagik namespace). Use a software composition analysis tool or registry audit to identify affected versions.
- **Downgrade or avoid** `@bitwarden/cli@2026.4.0` and verify the installed version against Bitwarden's signed binaries. Treat any environment that installed this version as credential-compromised and follow Bitwarden's published remediation guidance [12].
- **Remove the affected Checkmarx KICS Docker images** (`v2.1.20`, `v2.1.21`, `alpine` tags pulled on April 22 during the confirmed exposure window) from all build environments, runners, and developer workstations.

Short-Term Mitigations

The postinstall hook execution model—a known and documented attack vector in npm supply chain security—can be constrained through package manager configuration. Setting `ignore-scripts=true` in `.npmrc` globally disables postinstall hooks across all packages, eliminating the primary CanisterSprawl execution vector, though this may break packages with legitimate postinstall requirements and should be tested before broad deployment. A more targeted approach is to pin package manager permissions so that installs in CI/CD environments never run as a principal with repository publish rights; a runner that cannot call `npm publish` cannot become a propagation vector even if malware executes within it.

Organizations should implement network egress controls that block outbound connections to ICP node IP ranges from developer workstations and build runners, except where ICP is an explicitly approved dependency. Threat intelligence feeds from Socket, StepSecurity, and Sonatype are publishing indicators associated with this campaign and should be integrated into security tooling.

Review GitHub Actions configurations across all repositories for unexpected additions of `.github/workflows/format-check.yml` or equivalent files, which CanisterWorm used as a persistence and secondary exfiltration mechanism [3][4].

Strategic Considerations

The TeamPCP campaign exposes a structural trust problem in modern software development. Developers routinely grant install-time scripts the same execution context as their own code, often with implicit access to every secret on the machine. This is a default that the security community has long

criticized but that package managers have been slow to change by default. Organizations should evaluate adopting npm's `--ignore-scripts` policy in CI/CD environments now, rather than waiting for a registry-level policy change.

The TeamPCP campaign demonstrates that security scanning infrastructure carries comparable supply chain risk to application code dependencies and should be subject to equivalent scrutiny, including pinned versions, integrity verification (e.g., SHA pinning for Docker images and GitHub Actions), and tight network egress controls on the runners that execute it. The targeting of security tooling—Trivy, KICS, Checkmarx GitHub Actions—as initial access vectors is particularly significant; the trust extended to scanning infrastructure needs to be calibrated to match its actual attack surface.

AI developer tooling now constitutes a distinct attack surface that most organizations have not yet mapped in their asset inventories. MCP configuration files, AI coding assistant contexts, and LLM API keys are functionally equivalent to other privileged credentials and should be managed accordingly—stored in secrets managers rather than plaintext environment variables or `.env` files, rotated regularly, and monitored for anomalous use.

CSA Resource Alignment

This incident directly engages several active CSA research areas and frameworks.

The **MAESTRO framework** for agentic AI threat modeling addresses the risk of compromised tool execution contexts (Layer 3: Agent Frameworks, and Layer 6: Infrastructure). CanisterSprawl's explicit targeting of MCP configurations and LLM API keys is a practical example of the MAESTRO-identified threat of supply chain compromise propagating into agentic execution environments. Organizations building or deploying AI coding agents should consult MAESTRO's guidance on securing agent tool invocation chains and validating the integrity of tool server configurations.

The **CSA AI Controls Matrix (AICM)**—a superset of the Cloud Controls Matrix (CCM) that incorporates AI-specific control objectives—provides relevant control mappings for software supply chain integrity (supply chain management controls), credential management (identity and access management controls), and logging and monitoring (audit and accountability controls). The AICM's supply chain controls specifically address integrity verification for dependencies, a gap this campaign exploited.

The **Zero Trust guidance** published by CSA is applicable to the CI/CD pipeline architecture exposed here. A Zero Trust approach to build infrastructure would limit the blast radius of a compromised runner by ensuring that build credentials are scoped to the minimum necessary privileges, time-limited, and subject to continuous authentication verification—preventing the token-to-worm propagation chain that CanisterSprawl depends on.

CSA's prior **research note on TeamPCP's March 2026 supply chain campaign** [9] provides foundational context on the actor's TTPs and should be reviewed alongside this note for a complete picture of the evolving campaign.

The **CSA STAR program** registry offers a mechanism for software vendors—including those whose packages appear in npm—to publish their security practices. Many open-source developer tools and utilities currently lack STAR attestations, leaving procurement and security review teams without a standardized security posture baseline for these dependencies. Organizations with dependencies on unattested packages should apply proportionally more scrutiny to those packages in their supply chain reviews.

References

[1] Socket. "Namastex.ai npm Packages Hit with TeamPCP-Style CanisterWorm." Socket Security Blog, April 2026.

[2] StepSecurity. "CanisterWorm: How a Self-Propagating npm Worm Is Spreading Backdoors Across the Ecosystem." StepSecurity Blog, April 2026.

[3] Unit 42. "Weaponizing the Protectors: TeamPCP's Multi-Stage Supply Chain Attack on Security Infrastructure." Palo Alto Networks, 2026.

[4] SANS Institute. "When the Security Scanner Became the Weapon: Inside the TeamPCP Supply Chain Campaign." SANS Blog, 2026.

[5] The Hacker News. "Self-Propagating Supply Chain Worm Hijacks npm Packages to Steal Developer Tokens." The Hacker News, April 22, 2026.

[6] Aikido Security. "Is Shai-Hulud Back? Compromised Bitwarden CLI Contains a Self-Propagating npm Worm." Aikido Security Blog, April 2026.

[7] Socket. "Malicious Checkmarx Artifacts Found in Official KICS Docker Repository." Socket Security Blog, April 2026.

[8] Socket. "Bitwarden CLI Compromised in Ongoing Checkmarx Supply Chain Campaign." Socket Security Blog, April 2026.

[9] Cloud Security Alliance AI Safety Initiative. "TeamPCP: Cascading Supply Chain Attack on AI/ML Tooling." CSA Labs, March 2026.

[10] InfoWorld. "Malicious pgserve, automagik Developer Tools Found in npm Registry." InfoWorld, April 2026.

[11] GitGuardian. "No Off Season: Three Supply Chain Campaigns Hit npm, PyPI, and Docker Hub in 48 Hours." GitGuardian Blog, April 2026.

[12] Bitwarden. "Bitwarden Statement on Checkmarx Supply Chain Incident." Bitwarden Community Forums, April 2026.

[13] Endor Labs. "The Bitwarden CLI Supply Chain Attack: What Happened and What to Do." Endor Labs Blog, April 2026.

[14] Sonatype. "Self-Propagating npm Malware Turns Trusted Packages Into Attack Paths." Sonatype Blog, April 2026.

[15] Bleeping Computer. "New Checkmarx Supply-Chain Breach Affects KICS Analysis Tool." Bleeping Computer, April 2026.