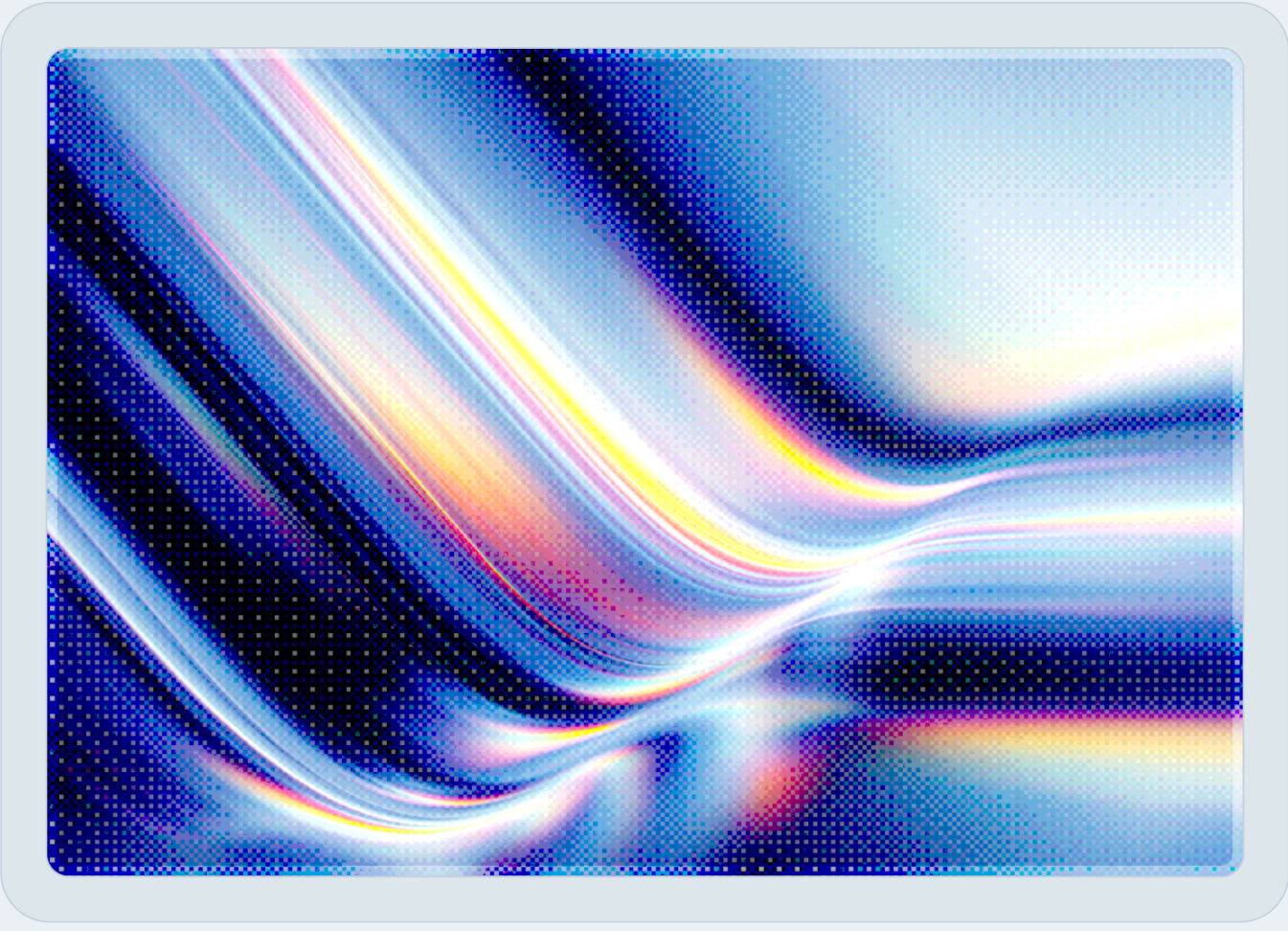
CSAI Foundation | Cloud Security Alliance

PCPJack: Cloud Worm Targeting AI Infrastructure Credentials

Credential Theft at Scale Across Exposed Container, Kubernetes, and ML Compute Services

2026-05-09

AI-assisted Rapid Research



CSAI | CSA cloud security alliance®

Key Takeaways

PCPJack is a modular credential-theft framework first identified by SentinelLABS on April 28, 2026. It worms autonomously across exposed cloud infrastructure — scanning for and exploiting vulnerable Docker daemons, Kubernetes API endpoints, Redis instances, MongoDB services, and Ray machine learning clusters — then exfiltrates harvested credentials through encrypted Telegram channels [1]. Its discovery follows a period of intense supply chain compromise activity attributed to a threat actor cluster tracked as TeamPCP. A defining behavior that gave the framework its name is its active eviction of TeamPCP tooling from infected hosts [2][12].

PCPJack exploits five known vulnerabilities — CVE-2025-29927, CVE-2025-55182, CVE-2026-1357, CVE-2025-9501, and CVE-2025-48703 — to gain initial access across a diverse range of cloud and web targets, then moves laterally through container escape techniques, SSH key harvesting, and Kubernetes service account token abuse [1][3]. Ray ML clusters receive particular attention: PCPJack gains code execution within Ray worker environments by submitting malicious Python jobs to the unauthenticated Ray dashboard API on port 8265, a vector with a documented exploitation history dating to CVE-2023-48022 [1][9].

The malware targets credentials for AI service providers including OpenAI and Anthropic, cloud platforms such as AWS and DigitalOcean, developer tooling including GitHub, Slack, and HashiCorp Vault, and financial services including cryptocurrency exchanges. The complete absence of any cryptomining component — prevalent in prior cloud intrusion campaigns — indicates a pure credential-theft and resale monetization model [1]. Organizations with internet-exposed cloud management interfaces, ML compute clusters, or web applications built on Next.js or React Server Components without current patches face the highest immediate risk [1][2].

Background

The TeamPCP Supply Chain Campaign

To understand PCPJack's operational context, it is necessary to briefly trace the activity of TeamPCP — the threat actor whose footprint PCPJack actively displaces. Between March 19 and 25, 2026, a cluster tracked under the aliases TeamPCP, DeadCatx3, PCPcat, ShellForce, and CanisterWorm executed

cascading supply chain compromises across GitHub Actions, Docker Hub, PyPI, npm, and OpenVSX [7]. The operation injected infostealer payloads into widely trusted security tooling including Aqua Security's Trivy vulnerability scanner, Checkmarx KICS, and the BerriAI LiteLLM gateway — software deployed in the security and AI pipelines of thousands of organizations [7][8]. Researchers at Unit 42, Arctic Wolf, Wiz, and Microsoft independently documented the campaign, with data shared by vx-underground and reported by Unit 42 estimating that over 500,000 credentials were exfiltrated during the three-week window [7][8].

By late April 2026, TeamPCP's implants had established persistent footholds across cloud-connected hosts targeted during the March campaign window, though the full breadth of infected infrastructure was not publicly quantified at the time PCPJack emerged. It is into this environment that PCPJack surfaced. On April 28, SentinelLABS identified a novel shell script through a Kubernetes-focused VirusTotal hunting rule: its first execution steps were to search for and delete artifacts associated with TeamPCP tooling, leading researchers to name the framework PCPJack [1]. Whether PCPJack represents a competing criminal operation, a disgruntled former affiliate, or a deliberate disruption effort remains unconfirmed. SentinelLABS researchers note that the targeting overlap and technical familiarity with TeamPCP's tooling "suggests someone associated with the group," but they have found no definitive evidence of attribution [1].

Ray ML Clusters as a Persistent Target

The inclusion of Ray ML clusters in PCPJack's target profile reflects a recurring pattern in cloud-focused attack campaigns. Ray is an open-source distributed computing framework widely used for training and serving machine learning workloads at organizations including OpenAI, Uber, and Amazon. CVE-2023-48022, disclosed in late 2023, documented a lack of authentication on Ray's dashboard and job submission API, which operates by default on port 8265 [9]. Despite a patch becoming available, the vulnerability saw persistent exploitation in the wild — the "ShadowRay" campaign documented by Oligo Security in 2024 found thousands of publicly exposed Ray servers actively targeted in the wild [9]. PCPJack's exploitation of Ray clusters follows this established tradecraft: by submitting a Python job to the unauthenticated API, an attacker gains code execution within the Ray worker environment and can harvest cloud credentials, service account tokens, and ML API keys from environment variables [1].

Security Analysis

Propagation Architecture

PCPJack's propagation model operates in two phases: host preparation and external target selection. After initial execution, the bootstrap shell script prepares the environment, downloads modular Python payloads from an attacker-controlled AWS S3 bucket, and establishes persistence before self-deleting — a technique that reduces forensic visibility of the initial dropper [1]. Target selection for external propagation combines two sources: internet-wide port scanning across documented cloud provider IP ranges maintained by the `cloud_ranges.py` module (covering AWS, Google Cloud, Azure, and Cloudflare), and targeted hostname lists extracted from Common Crawl parquet files — a technique that effectively leverages a public web index as a reconnaissance dataset without generating anomalous scanning traffic from attacker infrastructure [1][3].

The five vulnerabilities PCPJack weaponizes for initial access span diverse technology stacks, reflecting the framework's breadth. CVE-2025-29927 is a critical (CVSS 9.1) authentication bypass in Next.js middleware, exploitable by adding a crafted `x-middleware-subrequest` header to skip authorization checks entirely [5]. CVE-2025-55182 ("React2Shell") is a CVSS 10.0 deserialization flaw in React Server Components and Next.js that enables remote code execution through attacker-controlled references in the React Flight protocol — a vulnerability that saw active exploitation within hours of its December 2025 disclosure [6]. CVE-2025-48703 is a critical OS command injection flaw in CentOS Web Panel's file manager, exploitable by unauthenticated attackers on over 220,000 internet-facing instances at disclosure [4]. CVE-2025-9501 and CVE-2026-1357 target PHP injection paths in W3 Total Cache and unauthenticated file upload in the WPVivid Backup plugin respectively, extending PCPJack's reach into the vast WordPress hosting ecosystem [1][3].

Lateral Movement and Container Escape

Once inside a host, PCPJack's `lateral.py` module conducts a structured reconnaissance sweep across the local network and accessible cloud metadata endpoints. Against Kubernetes environments, it harvests service account tokens, then exploits misconfigured RBAC to access cluster resources and potentially escape container boundaries. Docker targets are reached via the Unix socket at `/var/run/docker.sock` or through unauthenticated TCP exposure on ports 2375 and 2376; bind-mount techniques are used to access host filesystems from within containers [1]. Redis instances are exploited via cron rewrite for persistence and root-level execution — a technique documented in

cloud attack toolkits for several years but still effective against unprotected deployments [1]. SSH key harvesting from `~/.ssh/` directories, combined with credential spraying against discovered SSH endpoints, rounds out the lateral movement arsenal.

Against Ray ML clusters, PCPJack submits Python jobs to the Ray job submission API that instruct the worker to exfiltrate environment variables and download the main bootstrap script — propagating the infection into the ML compute environment with a single API call to port 8265 [1].

Credential Scope and Exfiltration

The `parser.py` module categorizes harvested credentials across fourteen categories, covering AWS access keys and secrets, Kubernetes service account tokens, Docker registry credentials, GitHub personal access tokens, Slack tokens, WordPress administrative credentials, OpenAI and Anthropic API keys, HashiCorp Vault tokens, 1Password service accounts, Grafana Cloud API keys, and DigitalOcean credentials [1]. A separate financial category targets API keys and account credentials for Binance, Coinbase, Gemini, Kraken, Stripe, and cryptocurrency wallet files. Bulk messaging credentials — SendGrid, Mailgun, Mailchimp, Twilio, and Amazon SES — suggest the stolen access may be resold or used directly for spam and phishing operations [1].

Exfiltration uses a Telegram bot as the command-and-control channel. Credentials are encrypted with X25519 ECDH key exchange and ChaCha20-Poly1305 symmetric encryption, then split into 2,800-byte chunks to respect Telegram's message size limits [1]. SentinelLABS noted a tradecraft inconsistency: while most string constants in the tooling are hex-encoded, the Telegram bot token and attacker public key are left in plaintext — suggesting the operators prioritized deployment speed over obfuscation [1]. Payload hosting is conducted through a typosquatted CloudFront domain (`cdn.cloudfront-js.com:8443`) and an attacker-controlled S3 bucket, both serving as distribution infrastructure for the modular components [1].

Significance for AI Infrastructure

PCPJack's active targeting of OpenAI, Anthropic, and related AI service credentials, combined with its focus on Ray ML clusters, reflects a growing attacker focus on AI infrastructure credentials that has accelerated through 2025 and 2026. AI API keys represent high-value targets: they grant access to compute-intensive inference capabilities, and — if resold or abused — can be monetized directly through inference abuse or credential pivoting, potentially exposing training data, fine-tuned models, or proprietary system prompts. The targeting of HashiCorp Vault and 1Password credentials indicates a secondary goal of acquiring secrets-management access that can unlock cascading downstream

credentials far beyond what direct environment-variable harvesting would yield [1]. Together, these capabilities position PCPJack as a framework specifically adapted to the cloud AI infrastructure stack in a way that older credential thieves were not.

The complete absence of cryptomining code is also analytically significant. Cryptomining has historically represented a prevalent monetization strategy in cloud intrusion campaigns targeting compute-capable infrastructure. PCPJack's operators appear to have concluded that the resale or direct exploitation value of stolen AI infrastructure credentials now exceeds the income from commandeering compute for mining — a judgment consistent with the monetization calculus SentinelLABS observed in assessing the campaign [1].

Recommendations

Immediate Actions

Organizations with cloud infrastructure matching PCPJack's target profile should treat the following as urgent operational tasks. All five CVEs exploited by the framework — CVE-2025-29927, CVE-2025-55182, CVE-2025-48703, CVE-2025-9501, and CVE-2026-1357 — should be remediated immediately; each has patches available from the respective vendors. The Next.js (CVE-2025-29927) and React Server Components (CVE-2025-55182) patches in particular have been available for months and represent well-understood, high-priority fixes [5][6]. Any Ray ML dashboard or job submission API (port 8265) that is reachable from untrusted networks should be firewalled immediately; if token authentication was not enabled as part of Ray's post-ShadowRay hardening work, it must be enabled now [9]. Docker management interfaces on TCP ports 2375 and 2376 should be disabled; all Docker management should occur through the Unix socket with appropriate access controls [1]. CentOS Web Panel (CWP) should be upgraded to version 0.9.8.1205 or later [4].

Alongside these patching steps, organizations should conduct a sweep for PCPJack indicators of compromise: check for the systemd service `spm-monitor.service`, files in `/var/lib/.spm/`, cron entries invoking Python from unusual temporary paths, and outbound Telegram API connections [1]. Any credentials that may have been exposed through prior TeamPCP compromise should be rotated — the overlap between TeamPCP-infected environments and PCPJack's targeting means that organizations affected by the March 2026 supply chain campaigns face elevated risk from both frameworks [7][8]. Kubernetes service account RBAC bindings warrant review as well; any service account capable of creating jobs or executing commands cluster-wide should be scoped to the minimum required [1].

Short-Term Mitigations

Beyond immediate patching, organizations should enforce AWS Instance Metadata Service v2 (IMDSv2) on all EC2 instances. IMDSv2 requires a session-oriented token for metadata access, preventing the class of environment-variable credential harvest that PCPJack's `worm.py` module performs against IMDS endpoints [1]. Plaintext credential storage in environment variables, configuration files, and shell history should be replaced with secrets management tooling; where vault solutions are already deployed, the vault access credentials themselves require the same rotation and monitoring as any other high-value secret. Outbound Telegram API traffic from cloud infrastructure hosts is uncommon in enterprise environments and, where no legitimate use exists, blocking it at the egress layer would disrupt PCPJack's C2 channel even on already-infected hosts [1][3].

Container image vulnerability scanning should be integrated into the CI/CD pipeline with a policy that blocks deployment of images containing critical or high-severity CVEs for which patches are available. Restricting S3 downloads in execution environments to an approved allowlist of buckets would prevent the payload delivery mechanism PCPJack uses without impairing legitimate use [1].

Strategic Considerations

The PCPJack campaign is consistent with a recurring pattern in cloud AI infrastructure: the speed at which machine learning frameworks, developer tooling, and cloud-native services are adopted frequently outpaces the security hardening of those environments. Ray ML clusters, Docker sockets, and Kubernetes API servers are legitimate and powerful infrastructure components, but their default configurations often assume a trusted network perimeter that does not exist in contemporary cloud deployments. Organizations building AI infrastructure should establish explicit network policies for every ML compute component before deploying to internet-adjacent environments.

The competitive dynamic between PCPJack and TeamPCP is also worth noting from a threat modeling perspective. The active displacement of a competing threat actor's tooling suggests that compromised cloud hosts are valuable enough to contest. Organizations that discover PCPJack artifacts should not conclude their incident is limited to the PCPJack framework — the same exposure that invited PCPJack may have previously hosted TeamPCP tooling, and both frameworks may have harvested credentials before the displacement occurred. Incident response investigations should account for the full potential window of compromise, beginning with the TeamPCP campaign period in March 2026.

Longer-term, the targeting of AI API keys as a primary credential category argues for treating those keys with the same rigor as privileged cloud credentials: short-lived tokens where platform support exists, dedicated secrets management rather than environment-variable storage, per-workload key scoping,

and usage monitoring with anomaly alerting. AI service providers and enterprise customers alike should adopt usage-based anomaly detection for API keys — a sudden spike in inference requests from an unfamiliar IP range may indicate a stolen key in active use rather than a legitimate workload.

CSA Resource Alignment

Several existing CSA frameworks and publications provide directly applicable guidance for the vulnerabilities and attack patterns PCPJack exploits.

CSA's *Agentic AI Red Teaming Guide* addresses the threat modeling dimensions most relevant to PCPJack's targeting of AI compute infrastructure [10]. Its taxonomy of threat categories — including Agent Authorization and Control Hijacking and multi-agent orchestration exploitation — maps onto the scenario where PCPJack's job-submission technique against Ray ML gives an attacker code execution within an agentic compute pipeline. The guide's recommendations for testing authentication boundaries on AI orchestration components and validating RBAC controls on compute services apply directly to hardening Ray, Kubernetes-based ML inference clusters, and LLM API gateway deployments.

CSA's 2019 *Best Practices for Implementing a Secure Application Container Architecture* addresses the Docker and Kubernetes attack surfaces that PCPJack exploits through container escape and service account token abuse [11]. The publication's guidance on container runtime security, privilege restriction, and network segmentation for containerized workloads provides a foundational control framework applicable to the lateral movement techniques PCPJack demonstrates; organizations should supplement this foundation with more recent container security guidance that reflects developments in Kubernetes RBAC, runtime security tooling, and supply chain controls since its publication.

The Zero Trust principles articulated in CSA's Zero Trust guidance publications — particularly the principle that no network path should be implicitly trusted — provide the architectural lens through which PCPJack's propagation model should be understood. The framework's success in moving from an initial web application exploit to lateral movement across Kubernetes, Docker, and Ray services depends on implicit trust between adjacent services within a cloud environment. A mature Zero Trust posture implementing mutual authentication between services, restricted east-west traffic flows, and workload identity requirements for all inter-service calls would meaningfully constrain PCPJack's lateral movement, though implementation completeness matters — partial Zero Trust deployments may remain vulnerable to the Ray and Kubernetes API abuse techniques PCPJack employs.

The AI Controls Matrix (AICM) provides control identifiers relevant to AI-specific credential management and API access governance. Specifically, AICM controls addressing AI system access management, key lifecycle governance, and cloud infrastructure configuration apply to the OpenAI, Anthropic, and AI

infrastructure credential categories PCPJack targets. Organizations using AICM as a compliance framework should map PCPJack's credential theft capabilities against their AICM control coverage as a gap-analysis exercise.

References

[1] SentinelLABS. "PCPJack | Cloud Worm Evicts TeamPCP and Steals Credentials at Scale." SentinelOne, May 2026.

[2] BleepingComputer. "New PCPJack worm steals credentials, cleans TeamPCP infections." BleepingComputer, May 2026.

[3] The Hacker News. "PCPJack Credential Stealer Exploits 5 CVEs to Spread Worm-Like Across Cloud Systems." The Hacker News, May 2026.

[4] Help Net Security. "Critical Control Web Panel vulnerability is actively exploited (CVE-2025-48703)." Help Net Security, November 2025.

[5] NIST National Vulnerability Database. "CVE-2025-29927 Detail." NVD, 2025.

[6] Akamai Security Research. "CVE-2025-55182: React and Next.js Server Functions Deserialization RCE." Akamai, 2025.

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

[8] Microsoft Security Blog. "Guidance for detecting, investigating, and defending against the Trivy supply chain compromise." Microsoft, March 2026.

[9] Oligo Security. "ShadowRay: First Known Attack Campaign Targeting AI Workloads Actively Exploited in the Wild." Oligo Security, 2024.

[10] Cloud Security Alliance. "Agentic AI Red Teaming Guide." CSA, 2025.

[11] Cloud Security Alliance. "Best Practices for Implementing a Secure Application Container Architecture." CSA, 2019.

[12] SecurityWeek. "'PCPJack' Worm Removes TeamPCP Infections, Steals Credentials." SecurityWeek, May 2026.

