DREAM

# TeamPCP: How a Security Scanner Took Down the AI Supply Chain

TLP:CLEAR

# Executive Summary

In March 2026, the financially motivated threat actor **TeamPCP** (PCPcat, ShellForce, DeadCatx3) executed a cascading supply chain attack that began with Aqua Security's **Trivy** vulnerability scanner and propagated downstream into **Checkmarx** GitHub Actions and OpenVSX extensions, **LiteLLM** on PyPI (95M+ monthly downloads), and 66+ npm packages via the self-propagating **CanisterWorm** worm. The group collaborates with **LAPSUS$** for extortion; by March 24, Mandiant reported over 1,000 SaaS environments affected.

This report provides a detailed technical analysis of TeamPCP's infrastructure, payloads, and operational tradecraft across the Checkmarx, Trivy, and LiteLLM compromises. Key findings include the identification of **AdaptixC2** as the C2 framework on the Checkmarx VPS — pre-staged nine months before the campaign; evidence that TeamPCP isolates infrastructure between the Checkmarx and Trivy workstreams (dedicated non-cloud VPS, consistent registrar selection) while the **LiteLLM compromise operated as an opportunistic extension of the Checkmarx infrastructure** — version 1.82.7 exfiltrated directly to `checkmarx[.]zone`, and even version 1.82.8's sysmon.py backdoor still beacons to `checkmarx[.]zone/raw`; and analysis of the LiteLLM backdoor's use of Python `.pth` **startup hooks** for interpreter-level persistence and **Kubernetes lateral movement** to propagate from a single compromised environment to cluster-wide access.

---

# The Cascading Compromise

## Timeline

| Date | Event |
|---|---|
| **Feb 28** | TeamPCP exploits a misconfigured GitHub Actions workflow in Aqua Security's Trivy repository, obtaining tokens with write access to 33+ repositories |
| **Mar 14–19** | Malicious Trivy v0.69.4 published to **official** channels (GitHub Releases, Docker Hub, GHCR, ECR). Credential stealer calls back to `scan.aquasecurtiy[.]org` . Exposure: ~3 hours (binaries), ~10 hours (Docker images v0.69.5–0.69.6) |
| **Mar 19–20** | Stolen npm tokens fuel **CanisterWorm** — a self-propagating npm worm across 66+ packages (141+ artifacts), using an ICP blockchain canister as C2 |
| **Mar 22** | TeamPCP re-breaches Aqua Security, publishing malicious Docker Hub images and defacing 44 internal repositories |
| **Mar 23** | Stolen credentials used to compromise **Checkmarx KICS GitHub Action** (35 force-pushed tags, ~4-hour window), **AST GitHub Action**, and two trojanized VSCode extensions on the **OpenVSX Registry** |
| **Mar 23–24** | **LiteLLM** versions 1.82.7 and 1.82.8 backdoored on PyPI via credentials harvested from a CI pipeline that pulled poisoned Trivy. Version 1.82.8 used a `.pth` file executing on every Python interpreter startup |
| **Mar 24** | ICP canister begins serving `kamikaze.sh` — a geopolitically targeted Kubernetes wiper that destroys Iranian systems and installs persistent backdoors elsewhere |

## Attack Mechanism

The Trivy compromise was an upstream supply chain attack: TeamPCP compromised the official release pipeline and published poisoned binaries through Aqua Security's legitimate distribution channels. The typosquat domain `scan.aquasecurtiy[.]org` was the C2 callback endpoint, not the delivery vector — developers pulled the malicious binary from trusted sources.

The Checkmarx compromise extended beyond the two OpenVSX extensions. TeamPCP force-pushed malicious commits to all 35 release tags of `kics-github-action`, injecting credential stealers into any CI/CD pipeline referencing Checkmarx actions by version tag rather than pinned SHA.


*the attack flow: from trivy, to creds harvesting, to litellm*

# Attribution

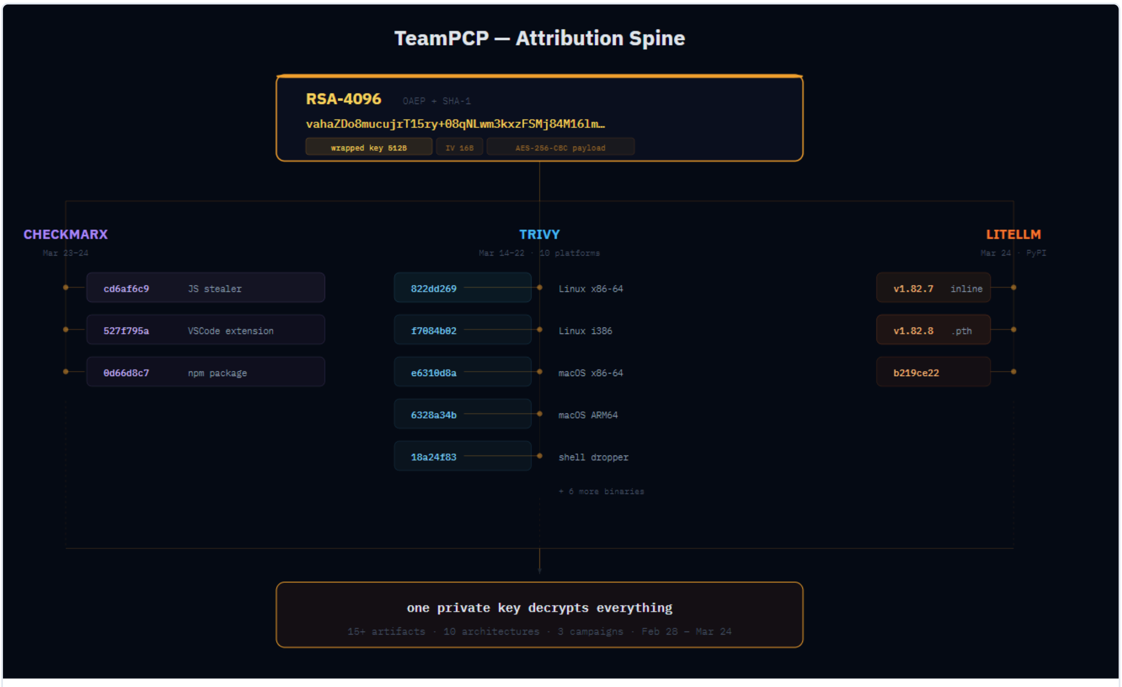
## Public Record

TeamPCP is tracked by Mandiant, Wiz, Microsoft, and multiple supply chain security vendors as a financially motivated actor collaborating with **LAPSUS$** for extortion. The group targets cloud infrastructure — primarily Azure and AWS — through exposed Docker APIs, Kubernetes clusters, and supply chain compromise.

## Cryptographic Attribution Link

A single **RSA-4096 public key** — identifiable by modulus fragment `vahaZDo8mucujrT15ry+` — is hardcoded in every TeamPCP payload across the Checkmarx and Trivy campaigns: the JavaScript credential stealer, all 10 trojanized Trivy binary variants, the shell dropper, and the sysmon.py Python backdoor. The same encryption scheme appears in the LiteLLM compromise, which reused the Checkmarx campaign's codebase and infrastructure before partially migrating to `models.litellm[.]cloud`.

The key wraps per-session AES-256-CBC keys before exfiltrating stolen credentials, producing a bundle format of `[RSA-encrypted key (512B)] + [IV (16B)] + [AES-encrypted payload]`. Only the holder of TeamPCP's private key can decrypt the material.




*Shared RSA across the campaign*

## Cross-Campaign TTP Fingerprint

The `~/.config/sysmon/` **persistence namespace** appears identically in both campaigns: `sysmon.js` for the Checkmarx campaign and `sysmon.py` for the Trivy campaign, each installed as a systemd user service named "System Telemetry." The sysmon.py backdoor is functionally identical across the Trivy and LiteLLM compromises — and the LiteLLM variant still contains a hardcoded reference to `checkmarx[.]zone/raw`, a residual from the Checkmarx campaign codebase. This persistence component is a reliable cross-campaign detection anchor.

CanisterWorm uses an analogous pattern: a `pgmon` service masquerading as PostgreSQL tooling, with payload downloads to `/tmp/pglog` and state tracking in `/tmp/.pg_state`.

# Technical Analysis

## AdaptixC2 on the Checkmarx VPS

The Checkmarx C2 VPS ( `83[.]142[.]209[.]11` , DEMENIN, Netherlands) runs **AdaptixC2**, an open-source C2 framework, on port 2222. Three corroborating artifacts confirm the identification: the stock AdaptixC2 404 HTML page ( `af98111ef` , first seen June 2025), the `/endpoint/login` authentication route, and the full AdaptixC2 source archive (AdaptixC2-main.zip, 3.3MB).

This VPS was staged with AdaptixC2 approximately **nine months** before the Checkmarx campaign became active. Twenty additional AdaptixC2 operator deployments share the same stock 404 HTML artifact across distinct IPs, confirming framework adoption across multiple unrelated threat actors — none sharing infrastructure overlap with TeamPCP.

## Checkmarx Payload: JavaScript Credential Stealer

The core credential stealer ( `cd6af6c9` ) is a **Linux-only Node.js script** (6,986 bytes) that exits on Windows detection. Using only Node.js built-in modules, it targets:

- **Environment variables**: `AWS_` , `KUBE` , `K8S` , `GOOGLE` , `GCLOUD` , `AZURE` , `DATABASE` , `VAULT` , Solana wallet strings
- **Process environments**: `/proc/<pid>/environ` enumeration
- **Filesystem**: 50+ paths including SSH keys ( `~/.ssh/` ), cloud credentials ( `~/.aws/` , `~/.config/gcloud/` , `~/.azure/` ), Kubernetes tokens ( `~/.kube/config` , service account tokens), Docker configs, cryptocurrency wallets (Bitcoin, Ethereum, Solana), WireGuard VPN configs, npm tokens ( `~/.npmrc` ), and shell history

Stolen data is encrypted with a **hybrid RSA-4096 OAEP / AES-256-CBC scheme**: a randomly generated 32-byte session key encrypts the payload, then the session key is wrapped with the actor's RSA-4096 public key. The exfiltration bundle is sent via HTTPS POST to `checkmarx[.]zone/vsx` with header `X-Filename: tpcp.tar.gz` .

**Persistence**: A Base64-encoded Python backdoor is deployed as `~/.config/sysmon/sysmon.js` and registered as a systemd user service ("System Telemetry"). The backdoor beacons to `checkmarx[.]zone/raw` every **50 minutes** — a deliberately long interval that avoids standard sandbox analysis windows. A `GITHUB_ACTIONS` environment check skips persistence installation in CI/CD pipelines. The `"youtube"` keyword in the C2 response functions as an operator-controlled killswitch — the ICP canister initially served a RickRoll URL, gating payload delivery until the operator was ready.

## Trojanized VSCode Extensions

The trojanized VSIX packages were built from specific legitimate Checkmarx releases — AST Results v2.52.0 and CX Dev Assist v1.6.0 — with a single malicious JavaScript module ( `environmentAuthChecker.js` ) substituted into each. Both trojanized extensions were submitted to the OpenVSX Registry within 56 seconds of each other. The legitimate base versions contain no instance of this module.

## Trojanized Trivy Binaries

The poisoned Trivy binaries execute the legitimate scanner and malicious code **in parallel**, maintaining functional appearance. TeamPCP built variants across **10 platforms**: Linux (x86-64, i386, ARM64, ARM, PPC64LE, s390x), macOS (x86-64, ARM64), FreeBSD (64-bit), and Windows (64-bit).

The binaries inherit the legitimate Trivy import table, causing them to cluster with genuine Trivy releases under standard binary similarity analysis — a deliberate evasion strategy. Internal build versioning ( `from-amd-1-7-6` , `from-arm-1-7-5` ) runs parallel to upstream Trivy version numbers, confirming active tracking of Trivy releases and a multi-architecture build pipeline.

When the primary C2 returns a non-2xx status, the malware falls back to creating a **`tpcp-docs`** repository in the victim's GitHub account and uploading the encrypted credential bundle as a release asset. A **Cloudflare Tunnel** ( `plug-tab-protective-relay.trycloudflare.com` ) provided an additional exfiltration channel from the GitHub Actions compromises.

## LiteLLM PyPI Backdoor

The LiteLLM compromise is the clearest example of the cascading trust chain at work: LiteLLM's CI/CD pipeline ran the poisoned Trivy action as its vulnerability scanner, which harvested the project's PyPI publishing token. TeamPCP then used that token to publish two backdoored versions — 1.82.7 and 1.82.8 — directly to PyPI on March 24, bypassing GitHub entirely. The packages were available for approximately three hours before PyPI quarantined the project. With ~3.4 million daily downloads, the exposure window was short but the blast radius was wide.

**Two injection vectors.** The two versions use different delivery mechanisms. Version 1.82.7 embeds the payload as 12 lines of double-base64-encoded code in `litellm/proxy/proxy_server.py` , executing at module import time — any process that imports LiteLLM's proxy server triggers the stealer. Version 1.82.8 escalates: a 34KB file named `litellm_init.pth` is placed in `site-packages/` . Python's `site.py` processes `.pth` files at interpreter startup — not at import, but every time Python runs. Scripts, notebooks, CI jobs, IDE language servers, even `pip` itself: if 1.82.8 is installed, the stealer fires regardless of whether the code ever touches LiteLLM. The `.pth` launcher uses `subprocess.Popen` to execute the payload in a detached background process, producing no visible output or startup delay. This is functionally equivalent to hooking the interpreter itself.

**Expanded credential scope.** The LiteLLM payload shares the same core structure as the Checkmarx JavaScript stealer — environment variables, SSH keys, cloud credentials, Kubernetes tokens, shell history — but extends it in two ways. It implements a full **AWS SigV4 signing routine** to authenticate directly to Secrets Manager and SSM Parameter Store, pulling secrets that exist only in managed services rather than on disk. And because LiteLLM is an LLM API gateway that handles keys for OpenAI, Anthropic, Google, Azure, and dozens of other providers by design, a compromised LiteLLM installation exposes not just infrastructure credentials but the API keys of every LLM provider the organization routes through it.

**Kubernetes lateral movement.** When the payload detects a Kubernetes service account token at `/var/run/secrets/kubernetes.io/serviceaccount/token` , it enumerates all cluster nodes and deploys a privileged pod to each one in `kube-system` . Each pod runs `alpine:latest` with `privileged: true` , `hostPID` , `hostNetwork` , mounts the host root filesystem at `/host` , and tolerates all taints — scheduling on every node including control plane. The pod chroots into the host and installs the sysmon.py persistence backdoor as a systemd user service, turning a single compromised Python environment into cluster-wide persistent access.

**Payload development evidence.** An operational security failure left two earlier payload iterations as commented-out base64 blobs on lines 131–132 of `proxy_server.py` . Decoding all three reveals TeamPCP's development progression: the first iteration used `exec()` with RC4 obfuscation; the second was transitional, carrying both old and new versions; the third (active) payload dropped `exec()` and RC4 in favor of `subprocess` piping and `tempfile`. The C2 endpoints, encryption scheme, persistence mechanism, and credential scope remained identical across all three — only the delivery and evasion wrappers evolved.

**Exfiltration.** The same hybrid RSA-4096/AES-256-CBC scheme used across all TeamPCP payloads. Version 1.82.7 exfiltrates directly to `checkmarx[.]zone` — the Checkmarx campaign's C2 server — confirming LiteLLM was not deployed on independent infrastructure but operated as an extension of the Checkmarx cluster. Version 1.82.8 partially migrated exfiltration to `models.litellm[.]cloud`, a domain registered March 23 via Spaceship to mimic LiteLLM's legitimate `litellm.ai`. However, the sysmon.py backdoor dropped by both versions still beacons to `checkmarx[.]zone/raw` for C2 tasking, leaving the Checkmarx infrastructure dependency intact. The `X-Filename: tpcp.tar.gz` header is present in both.


*LiteLLM .pth Interpreter Hook + K8s Lateral Movement*

## ICP Blockchain C2

The ICP canister ( `tdtqy-oyaaa-aaaae-af2dq-cai.raw.icp0[.]io` ) serves multiple functions:

- **C2 dead-drop**: The sysmon.py backdoor and CanisterWorm poll the canister for follow-on payloads. The `/on313` endpoint serves a 148-byte JSON configuration directing the backdoor
- **Credential harvesting**: A 46KB HTML phishing kit with lure pages for login, financial, and commercial flows uses **JotForm** — a legitimate form service — as its harvesting backend, eliminating the need for attacker-controlled receiving infrastructure
- **Destructive payload staging**: `kamikaze.sh` deploys a privileged Kubernetes DaemonSet — on Iranian systems it executes `rm -rf / --no-preserve-root` ; on non-Iranian systems it installs a persistent Python backdoor

ICP canisters are stored on the Dfinity blockchain and governed by smart contract logic; they cannot be taken down via registrar or hosting provider action. The canister was eventually made unavailable via an ICP network policy violation.

## Infrastructure

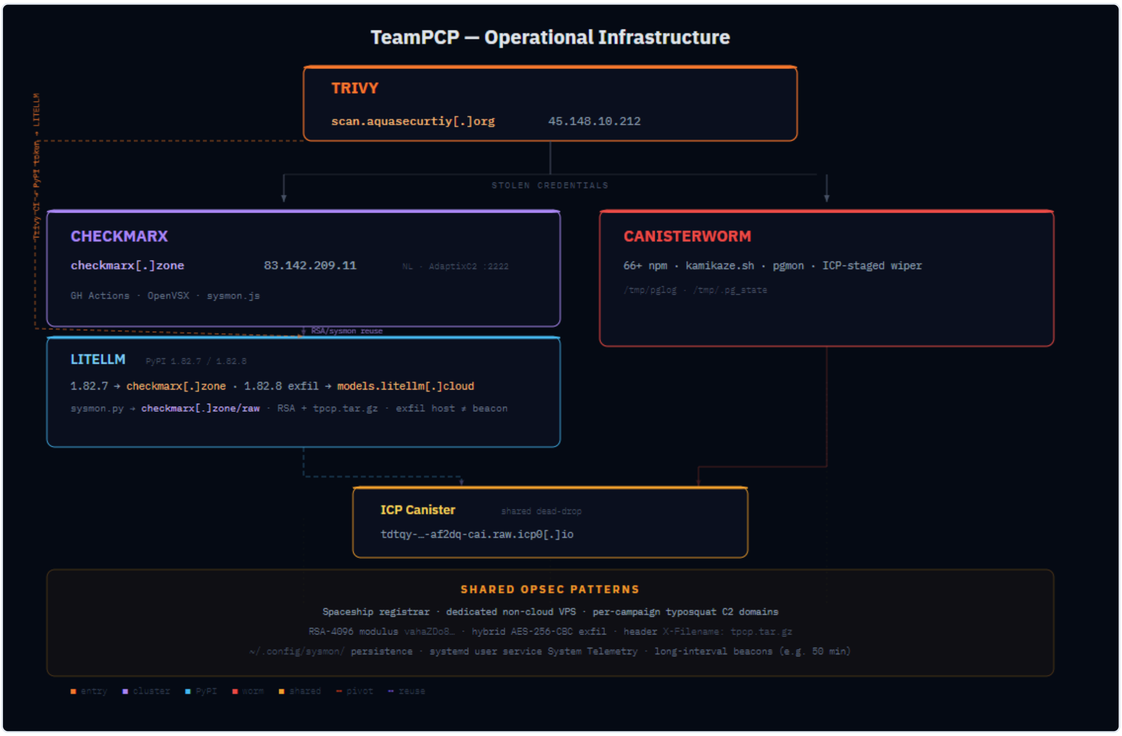
TeamPCP's infrastructure separates into **two isolated clusters** — Checkmarx and Trivy — with the LiteLLM compromise operating as an **extension of the Checkmarx cluster**, not a third independent deployment.

**Checkmarx vs. Trivy: Fully Isolated.** No IP or hosting overlap exists between these two workstreams.

| Attribute | Checkmarx Cluster | Trivy Cluster |
|---|---|---|
| C2 Domain | checkmarx[.]zone | scan.aquasecurtiy[.]org |
| Registrar | Spaceship | Spaceship |
| C2 VPS | 83[.]142[.]209[.]11 (DEMENIN, NL) | 45[.]148[.]10[.]212 (DMZHOST) |
| VPS Type | Dedicated, non-cloud | Dedicated, non-cloud |
| Pre-staging | ~9 months | ~3 years |
| Exfil Endpoint | /vsx | /upload |
| Beacon Path | /raw | /dido.mp3 |

Both VPS hosts are dedicated non-cloud providers that do not appear in cloud IP blocklists. Both C2 domains are freshly registered typosquats of the impersonated brand. The Trivy shell dropper was additionally delivered via **Socket.dev's CDN** ( `socketusercontent[.]com` ) — a developer security platform whose infrastructure may be inherently trusted by security controls.

**LiteLLM: Infrastructure-Dependent on Checkmarx.** The LiteLLM compromise reused Checkmarx infrastructure rather than deploying its own C2 — consistent with an opportunistic extension executed within hours of obtaining the stolen PyPI token. Version 1.82.7 exfiltrated stolen credentials directly to `checkmarx[.]zone` — the same C2 server, same VPS, same endpoint structure. Version 1.82.8 introduced `models.litellm[.]cloud` as a new exfiltration domain (registered March 23 via Spaceship, same registrar), but the migration was incomplete: the sysmon.py persistence backdoor deployed by v1.82.8 still beacons to `checkmarx[.]zone/raw` for C2 tasking. This residual is both an OPSEC failure and a hard attribution link — any LiteLLM-compromised host phones home to Checkmarx C2 infrastructure regardless of which backdoored version was installed.




*TeamPCP campaign Infra - shared opsec patterns*

## Detection & Mitigation

**Block campaign infrastructure.** Block outbound connections to all domains, IPs, and URLs listed in the IOC section. Monitor DNS for queries to `checkmarx[.]zone`, `scan.aquasecurtiy[.]org` (note the typosquat), `models.litellm[.]cloud`, and the ICP canister address.

**Hunt for AdaptixC2 artifacts.** TeamPCP's Checkmarx VPS runs AdaptixC2 on port 2222 with the stock 404 HTML page and `/endpoint/login` route. Organizations can proactively scan for AdaptixC2 deployments on non-standard ports as an infrastructure hunting technique — the framework's default artifacts are consistent across deployments.

**Detect the sysmon persistence backdoor.** The sysmon.py/sysmon.js backdoor is TeamPCP's most reliable fingerprint — it is reused identically across the Trivy, Checkmarx, and LiteLLM campaigns. Alert on creation of `~/.config/sysmon/` directories, `sysmon.service` or `sysmon.js` / `sysmon.py` files, and systemd user services named "System Telemetry." The LiteLLM variant retains a hardcoded `checkmarx[.]zone/raw` beacon URL even when exfiltrating to a different domain, making this string a durable detection signature.

**Detect CanisterWorm persistence.** Alert on `pgmon` systemd services masquerading as PostgreSQL tooling. Monitor for `/tmp/pglog` (payload download path) and `/tmp/.pg_state` (state tracking file).

**Hunt for tpcp-docs repositories.** The presence of a `tpcp-docs` repository in any GitHub organization indicates successful exfiltration via TeamPCP's fallback mechanism.

**Pin GitHub Actions to SHA hashes.** Version tags can be force-pushed to malicious commits. Pin all third-party GitHub Actions to full commit SHAs.

**Audit affected package versions.** Verify no compromised Trivy, Checkmarx, or LiteLLM versions were pulled during exposure windows. Rotate all credentials on affected systems — SSH keys, cloud tokens, Kubernetes service account tokens, Docker registry credentials, npm publish tokens, and PyPI credentials.

**Scan for `.pth` file persistence.** Search all Python environments for unexpected `.pth` files in `site-packages/` — any `.pth` file executes on every interpreter startup. Specifically hunt for `litellm_init.pth` (34KB). This technique abuses a CPython feature acknowledged as a security risk (cpython#113659) but unpatched.

**Hunt for TeamPCP Kubernetes pods.** In clusters where LiteLLM or compromised Trivy ran, check `kube-system` for pods named `node-setup-*` running `alpine:latest` with privileged security context and host filesystem mounts. These are the lateral movement pods deployed by the LiteLLM and kamikaze.sh payloads.

**Verify VSCode extension sources.** The trojanized Checkmarx extensions were distributed via the **OpenVSX Registry**, not the official VS Code Marketplace.

**Monitor for the RSA key modulus in binaries.** The string `vahaZDo8mucujrT15ry+` appears in every TeamPCP payload and can be used as a YARA rule anchor for retrospective scanning.

**Watch for infrastructure pre-staging patterns.** TeamPCP acquires dedicated non-cloud VPS months to years before campaigns. Monitor for freshly registered typosquat domains on **Spaceship** registrar resolving to dedicated VPS providers outside major cloud platforms.

# Indicators of Compromise

## Network Indicators

| Type | Indicator | Description |
|---|---|---|
| **Domain** | checkmarx[.]zone | Checkmarx/LiteLLM cluster C2 (typosquat) |
| **Domain** | scan.aquasecurtiy[.]org | Trivy campaign C2 (typosquat of aquasecurity.org) |
| **Domain** | models.litellm[.]cloud | LiteLLM v1.82.8 exfiltration (v1.82.7 used checkmarx[.]zone) |
| **Domain** | tdtqy-oyaaa-aaaae-af2dq-cai.raw.icp0[.]io | ICP blockchain C2 dead-drop |
| **Domain** | plug-tab-protective-relay.trycloudflare.com | Cloudflare Tunnel C2 |
| **IP** | 83[.]142[.]209[.]11 | Checkmarx/LiteLLM C2 VPS (DEMENIN, NL) — AdaptixC2 on port 2222 |
| **IP** | 45[.]148[.]10[.]212 | Trivy C2 VPS (DMZHOST) |

## URLs

| Indicator | Description |
|---|---|
| hxxps://checkmarx[.]zone/vsx | Credential exfiltration (Checkmarx + LiteLLM v1.82.7) |
| hxxps://checkmarx[.]zone/raw | C2 beacon endpoint (50-min backdoor) |
| hxxp://checkmarx[.]zone/raw/tmp/pglog | C2 staging path |
| hxxps://checkmarx[.]zone/static/checkmarx-util-1.0.4.tgz | Malicious npm package delivery |
| hxxps://scan.aquasecurtiy[.]org/upload | Credential exfiltration (Trivy) |
| hxxps://scan.aquasecurtiy[.]org/main.go | Trivy binary delivery |
| hxxps://scan.aquasecurtiy[.]org/dido.mp3 | Trivy C2 beacon path |
| hxxps://socketusercontent[.]com/blob/QGKJPg-gHR5Q43Kt6GATFGgDa_B1SZpimbgZA0eXdZxo | Shell dropper delivery (Socket.dev CDN) |
| hxxps://models.litellm[.]cloud/ | Credential exfiltration (LiteLLM v1.82.8) |
| hxxps://tdtqy-oyaaa-aaaae-af2dq-cai.raw.icp0[.]io/on313 | ICP canister C2 endpoint |

## File Hashes (SHA-256)

**Checkmarx Campaign:**

| Hash | Description |
|---|---|
| `cd6af6c9ba149673ff89a1f1ccc8ec40a265a3b54ad455fbef28dc2967a98e45` | JavaScript credential stealer (index.js) |
| `527f795a201a6bc114394c4cfd1c74dce97381989f51a4661aafbc93a4439e90` | VSCode extension payload (environmentAuthChecker.js) |
| `0d66d8c7e02574ff0d3443de0585af19c903d12466d88573ed82ec788655975c` | Malicious npm package (checkmarx-util-1.0.4.tgz) |
| `65bd72fcddaf938cefdf55b3323ad29f649a65d4ddd6aea09afa974dfc7f105d` | Trojanized VSIX (ast-results-2.53.0) |
| `744c9d61b66bcd2bb5474d9afeee6c00bb7e0cd32535781da188b80eb59383e0` | Trojanized VSIX (cx-dev-assist-1.7.0) |

**Trojanized Trivy Binaries:**

| Hash | Description |
|---|---|
| `822dd269ec10459572dfaaefe163dae693c344249a0161953f0d5cdd110bd2a0` | Linux x86-64 (v0.69.4) |
| `f7084b0229dce605ccc5506b14acd4d954a496da4b6134a294844ca8d601970d` | Linux i386 (v0.69.4) |
| `e64e152afe2c722d750f10259626f357cdea40420c5eedae37969fbf13abbecf` | Linux ARM64 (v0.69.4) |
| `25214414b8284b8ab0f954d3eac92105008a068fcf635bb659cb508b9422324d` | Linux ARM64 (v0.69.6) |
| `7b5cc85e82249b0c452c66563edca498ce9d0c70badef04ab2c52acef4d629ca` | Linux x86-64 (internal v1-7-6) |
| `bef7e2c5a92c4fa4af17791efc1e46311c0f304796f1172fce192f5efc40f5d7` | Linux ARM |
| `887e1f5b5b50162a60bd03b66269e0ae545d0aef0583c1c5b00972152ad7e073` | FreeBSD 64-bit |
| `ecce7ae5ffc9f57bb70efd3ea136a2923f701334a8cd47d4fbf01a97fd22859c` | Linux PPC64LE |
| `d5edd791021b966fb6af0ace09319ace7b97d6642363ef27b3d5056ca654a94c` | Linux s390x |
| `e6310d8a003d7ac101a6b1cd39ff6c6a88ee454b767c1bdce143e04bc1113243` | macOS x86-64 (v0.69.4) |
| `6328a34b26a63423b555a61f89a6a0525a534e9c88584c815d937910f1ddd538` | macOS ARM64 (v0.69.4) |
| `0880819ef821cff918960a39c1c1aada55a5593c61c608ea9215da858a86e349` | Windows 64-bit |

**Persistence and Delivery:**

| Hash | Description |
|---|---|
| `b219ce2263c913655269946b884c38ee3dc577a7f1221d4524f2b2bceb1f55ad` | sysmon.py Python backdoor |
| `18a24f83e807479438dcab7a1804c51a00dafc1d526698a66e0640d1e5dd671a` | Shell dropper (Socket.dev CDN) |
| `c39450ba91a2567d198ac3d81da8970485c1ae312f8cc3def57365d3e2d24b11` | Docker image (trivyhack.tar) |
| `ca84ead8f68a43d64b0205e69931e37fc45edccc1ba2e52664733a30dbc02c06` | ICP phishing kit HTML |
| `dd36fba6c1c1183d14bafb614616a22a54af73ffaaf5f7fcc5bd76cd132cca29` | ICP canister JSON config |
| `af98111ef3078ab381b276c64d8c1bcd9e6a9bd1534ea10c57b716763fe86941` | AdaptixC2 404 HTML (Checkmarx VPS) |

## Host-Based Indicators

| Type | Indicator | Description |
|---|---|---|
| **File Path** | `~/.config/sysmon/sysmon.js` | Checkmarx campaign backdoor |
| **File Path** | `~/.config/sysmon/sysmon.py` | Trivy/LiteLLM campaign backdoor |
| **File Path** | `litellm_init.pth` (in `site-packages/` ) | LiteLLM v1.82.8 interpreter-level persistence |
| **File Path** | `/tmp/pglog` | CanisterWorm payload path |
| **File Path** | `/tmp/.pg_state` | CanisterWorm state file |
| **Service** | `sysmon.service` ("System Telemetry") | Persistence — Checkmarx/Trivy/LiteLLM |
| **Service** | `pgmon` (PostgreSQL masquerade) | Persistence — CanisterWorm |
| **K8s** | Pods `node-setup-*` in `kube-system` | LiteLLM K8s lateral movement (privileged, host mount) |
| **GitHub** | Repository `tpcp-docs` | Fallback exfiltration channel |
| **Crypto** | RSA-4096 modulus: `vahaZDo8mucujrT15ry+` | Actor attribution key — all payloads |
| **HTTP Header** | `X-Filename: tpcp.tar.gz` | Exfiltration POST header |