CAA-2026-0015 TeamPCP Supply Chain Attack Distributes Information Stealer via Trusted Dependencies

Summary

A threat actor known as TeamPCP has conducted multiple supply chain attacks targeting popular open-source security scanners, resulting in the distribution of malicious updates containing information stealer malware. This attack, which most recently affected the widely used `litellm` PyPi package, presents a significant risk of threat actors gaining access to development environments that use compromised products as part of the CI/CD pipelines. Organizations must assess their environment for any dependencies on the compromised products during the timeframes that malicious versions were distributed and rotate potentially compromised credentials to prevent further attacks.

Details

Since 27 February 2026, TeamPCP compromised several open-source security scanners: Aquasecurity Trivy, Checkmarx KICS, and Checkmarx AST. The group also compromised multiple npm packages, and most recently, the `litellm` PyPi package. TeamPCP used their access to the development environments of these products to publish a variety of malicious versions, Visual Studio Code (VSCode) extensions, GitHub Actions, and Docker images containing information stealer malware. The information stealer is capable of extracting secrets, including:

- SSH keys
- VPN configuration files
- Git credentials
- Cloud environment credentials, such as Azure, AWS, and GCP
- Kubernetes service account tokens
- TLS private keys, and more[1]

Execution of any infected packages will expose organizations to opportunistic threat actors that may abuse stolen credentials, exfiltrate sensitive data, and conduct data extortion.[2] SANS researchers corroborated reports of TeamPCP exfiltrating over 300GB of credentials, and partnering with the cybercriminal group LAPSUS$ to extort victims.[3][4]

TeamPCP's Supply Chain Attack Timeline

The timeline and version numbers below can help organizations assess their exposure. Several reports provide conflicting end times for when compromised versions were active on their respective repositories and have been omitted from this report.

[1] https://socket.dev/blog/trivy-under-attack-again-github-actions-compromise
[2] https://socket.dev/blog/teampcp-targeting-security-tools-across-oss-ecosystem
[3] https://www.linkedin.com/posts/mccartypaul_infosec-teampcp-lapsus-activity-7442235107521982465-2tsZ/
[4] https://www.sans.org/blog/when-security-scanner-became-weapon-inside-teampcp-supply-chain-campaign

Trivy VSCode Extension

At 2026-02-27 12:04 (UTC), a malicious Trivy VSCode extension `trivy-vulnerability-scanner` version 1.8.12 was published to the OpenVSX registry. A second malicious `trivy-vulnerability-scanner` VSCode extension version 1.8.13 was also published to the OpenVSX registry at 2026-02-28 16:28 (UTC).[5]

Trivy and Associated GitHub Actions

At approximately 2026-03-19 17:43 UTC, a malicious release of Trivy version 0.69.4, as well as 83 malicious version tags for the Trivy GitHub Actions `aquasecurity/trivy-action` and `aquasecurity/setup-trivy` were published.

The maintainers at Trivy disclosed this compromise resulted from incomplete rotation of compromised credentials from the previous incident.[6]

CanisterWorm Attack on emilgroup, teale.io and Other npm Packages

At 2026-03-20 20:13 (UTC), TeamPCP pivoted to compromising the npm ecosystem using a self-propagating worm malware dubbed CanisterWorm. The worm published 141 malicious versions of 66 different packages. Initially, most malicious npm packages were under the scope of `emilgroup` and `teale.io`, but quickly spread to further packages from `opengov`, `virtahealth`, `react`, and others.[7]

Notably, CanisterWorm leverages an Internet Computer Control (ICP) canister for command and control (C2) communication which enables the delivery of secondary payloads. ICP is a decentralized blockchain that provides infrastructure for hosting websites, applications, and more.[8] TeamPCP's ICP canister used for C2 communication was subsequently taken offline due to violating ICP policy.

CanisterWorm also establishes persistence by creating systemd services in the context of the current user.[9][10]

Trivy Docker Images

At approximately 2026-03-22 16:00 (UTC), two malicious Aquasecurity Trivy Docker images (versions 0.69.5 and 0.69.6) were published to Docker Hub.[11]

Checkmarx KICS and AST VSCode Extensions

At 2026-03-23 12:53 (UTC), two Checkmarx VSCode extensions `cx-dev-assist` version 1.7.0 and `ast-results` version 2.53.0 were published to the OpenVSX registry by a compromised account.[12] Several reports indicate the versions hosted on the official VSCode Marketplace were unaffected.[13]

Checkmarx KICS and Associated GitHub Actions for KICS and AST

At 2026-03-23 12:58 (UTC), a malicious release of KICS version 1.1, 35 GitHub Action tags for KICS (`kics-github-action`), and all GitHub Action tags for AST (`ast-github-action`) were published. Sysdig initially assessed the impact to the GitHub Action for `ast-github-action` was limited to just one tag; however, Wiz later assessed the compromise to have likely affected all tags.

LiteLLM PyPi Package

At 2026-03-24 10:39 (UTC), malicious versions `litellm` 1.82.7 and 1.82.8 were published to the PyPi marketplace. Analysts at FutureSearch note malicious versions of `litellm` had been downloaded over 46,000 times, demonstrating the significant blast radius of this attack.[14]

---

[5] https://socket.dev/blog/unauthorized-ai-agent-execution-code-published-to-openvsx-in-aqua-trivy-vs-code-extension
[6] https://github.com/aquasecurity/trivy/security/advisories/GHSA-69fq-xp46-6x23
[7] https://socket.dev/supply-chain-attacks/canisterworm
[8] https://learn.internetcomputer.org/hc/en-us/articles/33152818663444-What-is-ICP
[9] https://www.aikido.dev/blog/teampcp-deploys-worm-npm-trivy-compromise
[10] https://www.endorlabs.com/learn/canisterworm
[11] https://www.wiz.io/blog/trivy-compromised-teampcp-supply-chain-attack
[12] https://checkmarx.com/blog/checkmarx-security-update/
[13] https://www.wiz.io/blog/teampcp-attack-kics-github-action#compromised-artifacts-43
[14] https://futuresearch.ai/blog/litellm-hack-were-you-one-of-the-47000/

LiteLLM determined their internal development environment was compromised because of using Trivy for security scanning in their CI/CD pipeline. This allowed attackers to obtain credentials for the LiteLLM publishing pipeline.[15]

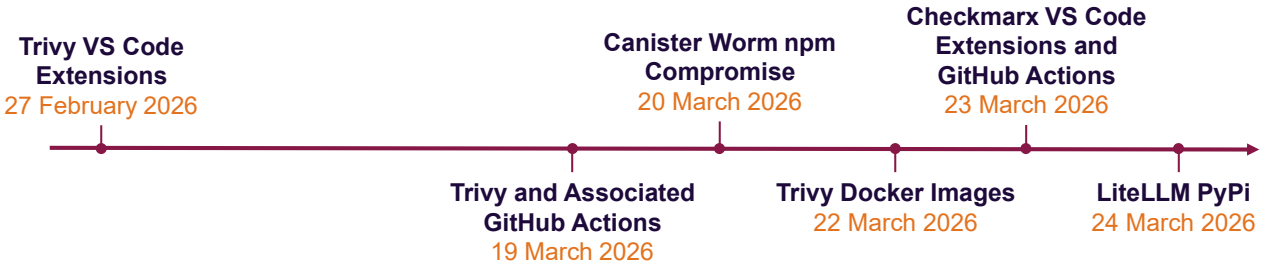

Figure 1. TeamPCP's Supply Chain Attack Timeline

Assessment

CyberAlberta Threat Intelligence assesses that TeamPCP will likely continue attempting to compromise development environments of popular open-source projects, attracting opportunistic threat actors seeking to exploit this access.[16] Similar to past Shai-Hulud campaigns, the use of self-propagating malware in supply chain attacks highlights the importance for up-to-date asset inventories so organizations can quickly determine their level of exposure.[17] Moreover, as organizations and users adopt agentic workflows, code dependencies will likely grow without the explicit awareness of security teams.

Recommendations

To assess the level of exposure to TeamPCP's supply chain attacks, organizations should:

- Confirm the presence of any of the following compromised products:
  - Aquasecurity Trivy VSCode extension `trivy-vulnerability-scanner` versions 1.8.12 and 1.8.13
  - Aquasecurity Trivy version 0.69.4
  - Aquasecurity Trivy GitHub Action Tags `aquasecurity/trivy-action` and `aquasecurity/setup-trivy`
  - The npm packages reported by Socket[18]
  - Aquasecurity Trivy Docker images versions 0.69.5 and 0.69.6
  - Checkmarx KICS OpenVSX extensions `ast-results` version 2.53.0 and `cx-dev-assist` version 1.7.0
  - Checkmarx KICS version 1.1
  - Checkmarx KICS GitHub Actions `kics-github-action` and `ast-github-action`
  - PyPi package `litellm` 1.82.7 and 1.82.8
- Determine if any malicious updates were installed during their respective timeframe of compromise.
- Exposed organizations must reset SSH keys and credentials for AWS, GCP, Azure, Kubernetes service account tokens, and cryptocurrency wallets.
- Update affected products to the latest, remediated version where appropriate.
- Developers should pin GitHub Actions by commit SHA instead of tags to remediate similar attacks relying on GitHub Actions tags.

[15] https://docs.litellm.ai/blog/security-update-march-2026
[16] https://x.com/intelrat/status/2036924437908373618
[17] https://cyberalberta.ca/the-scattering-the-shai-hulud-worm-malware
[18] https://socket.dev/supply-chain-attacks/canisterworm

Detection Opportunities

The following KQL queries can help detect usage of the compromised products. These queries are dependent on the products being stored in their default folder paths.

```
let Compromised_VS_Code_Extensions = dynamic(['trivy-vulnerability-scanner-1.8.12', 'trivy-
vulnerability-scanner-1.8.13', 'checkmarx.ast-results-2.53.0', 'checkmarx.cx-dev-assist-
1.7.0']);
union isfuzzy=true DeviceEvents, DeviceFileEvents, DeviceNetworkEvents, DeviceProcessEvents
| where Timestamp >= datetime("2026-02-27")
| where
    InitiatingProcessCommandLine matches regex @"[\\/]\.VS Code[\\/]extensions[\\/]"
    or
    ProcessCommandLine matches regex @"[\\/]\.VS Code[\\/]extensions[\\/]"
    or
    FolderPath matches regex @"[\\/]\.VS Code[\\/]extensions[\\/]"
    or
    InitiatingProcessFolderPath matches regex @"[\\/]\.VS Code[\\/]extensions[\\/]"
| extend VS Code_Extension =
    coalesce
    (
        extract(@"\.VS Code[\\/]+extensions[\\/]+([^\\/]+)", 1,
InitiatingProcessCommandLine),
        extract(@"\.VS Code[\\/]+extensions[\\/]+([^\\/]+)", 1, ProcessCommandLine),
        extract(@"\.VS Code[\\/]+extensions[\\/]+([^\\/]+)", 1, FolderPath),
        extract(@"\.VS Code[\\/]+extensions[\\/]+([^\\/]+)", 1,
InitiatingProcessFolderPath)
    )
| where VS Code_Extension matches regex @"^[a-z0-9][a-z0-9._-]*$" and VS Code_Extension
has_any (Compromised_VS_Code_Extensions)
| summarize FirstSeen = min(Timestamp), LastSeen = max(Timestamp) by VS Code_Extension,
DeviceName, InitiatingProcessAccountName
| sort by LastSeen desc
```

Figure 2. KQL Query for Identifying VSCode Extensions

```
union isfuzzy=true DeviceEvents, DeviceFileEvents, DeviceNetworkEvents, DeviceProcessEvents
| where Timestamp >= datetime("2026-03-20")
| where
    InitiatingProcessCommandLine matches regex @"[\\/]+node_modules[\\/]+"
    or
    ProcessCommandLine matches regex @"[\\/]+node_modules[\\/]+"
    or
    FolderPath matches regex @"[\\/]+node_modules[\\/]+"
    or
    InitiatingProcessFolderPath matches regex @"[\\/]+node_modules[\\/]+"
| extend npm_Package =
    coalesce
    (
        extract(@"node_modules[\\/]+@?([^\\/]+)", 1, InitiatingProcessCommandLine),
        extract(@"node_modules[\\/]+@?([^\\/]+)", 1, ProcessCommandLine),
        extract(@"node_modules[\\/]+@?([^\\/]+)", 1, FolderPath),
        extract(@"node_modules[\\/]+@?([^\\/]+)", 1, InitiatingProcessFolderPath)
    )
| where npm_Package matches regex @"^[a-z0-9][a-z0-9._-]*$"
```

```
| summarize FirstSeen = min(Timestamp), LastSeen = max(Timestamp) by npm_Package,
DeviceName, InitiatingProcessAccountName
| sort by LastSeen desc
```

Figure 3. KQL Query for Identifying npm Package Installs since TeamPCP's CanisterWorm Attack

```
union isfuzzy=true DeviceEvents, DeviceFileEvents, DeviceNetworkEvents, DeviceProcessEvents
| where Timestamp >= datetime("2026-03-24")
| where
    InitiatingProcessCommandLine matches regex @"[\\/]+site-packages[\\/]+"
    or
    ProcessCommandLine matches regex @"[\\/]+site-packages[\\/]+"
    or
    FolderPath matches regex @"[\\/]+site-packages[\\/]+"
    or
    InitiatingProcessFolderPath matches regex @"[\\/]+site-packages[\\/]+"
| extend PyPi_Package =
    coalesce
    (
        extract(@"site-packages[\\/]+([^\\/]+)", 1, InitiatingProcessCommandLine),
        extract(@"site-packages[\\/]+([^\\/]+)", 1, ProcessCommandLine),
        extract(@"site-packages[\\/]+([^\\/]+)", 1, FolderPath),
        extract(@"site-packages[\\/]+([^\\/]+)", 1, InitiatingProcessFolderPath)
    )
| where PyPi_Package matches regex @"^[a-z0-9][a-z0-9_-]*$"
| where PyPi_Package contains "litellm"
| summarize FirstSeen = min(Timestamp), LastSeen = max(Timestamp) by PyPi_Package,
DeviceName, InitiatingProcessAccountName
| sort by LastSeen desc
```

Figure 4. KQL Query for Identifying Installs of LiteLLM PyPi Packages Since the The Start of The Compromise

Indicators of Compromise (IOCs)

The following IOCs characterize TeamPCP's activity described in this report.

Description	Indicator
Informaiton stealer, C2 domains	scan.aquasecurtiy[.]org
	plug-tab-protective-relay.trycloudflare[.]com
	models.litellm[.]cloud
	checkmarx[.]zone
Information stealer, C2 IP	45.148.10[.]212
Informaiton stealer, SHA256 hash	18a24f83e807479438dcab7a1804c51a00dafc1d526698a66e0640d1e5dd671a
	887e1f5b5b50162a60bd03b66269e0ae545d0aef0583c1c5b00972152ad7e073
	f7084b0229dce605ccc5506b14acd4d954a496da4b6134a294844ca8d601970d
	822dd269ec10459572dfaaefe163dae693c344249a0161953f0d5cdd110bd2a0
	bef7e2c5a92c4fa4af17791efc1e46311c0f304796f1172fce192f5efc40f5d7

	e64e152afe2c722d750f10259626f357cdea40420c5eedae37969fbf13abbecf
	ecce7ae5ffc9f57bb70efd3ea136a2923f701334a8cd47d4fbf01a97fd22859c
	d5edd791021b966fb6af0ace09319ace7b97d6642363ef27b3d5056ca654a94c
	e6310d8a003d7ac101a6b1cd39ff6c6a88ee454b767c1bdce143e04bc1113243
	6328a34b26a63423b555a61f89a6a0525a534e9c88584c815d937910f1ddd538
	0880819ef821cff918960a39c1c1aada55a5593c61c608ea9215da858a86e349
CanisterWorm, ICP canister C2	tdtqy-oyaaa-aaaae-af2dq-cai.raw.icp0[.]io
CanisterWorm test payload, SHA256 hash	e9b1e069efc778c1e77fb3f5fcc3bd3580bbc810604cbf4347897ddb4b8c163b
	0c0d206d5e68c0cf64d57ffa8bc5b1dad54f2dda52f24e96e02e237498cb9c3a
CanisterWorm ICP backdoor payload, SHA256 hash	61ff00a81b19624adaad425b9129ba2f312f4ab76fb5ddc2c628a5037d31a4ba
CanisterWorm self-propagating backdoor payload, SHA256 hash	c37c0ae9641d2e5329fcdee847a756bf1140fdb7f0b7c78a40fdc39055e7d926
CanisterWorm self-propagating worm, SHA256 hash	f398f06eefcd3558c38820a397e3193856e4e6e7c67f81ecc8e533275284b152
	7df6cef7ab9aae2ea08f2f872f6456b5d51d896ddda907a238cd6668ccdc4bb7
	5e2ba7c4c53fa6e0cef58011acdd50682cf83fb7b989712d2fcf1b5173bad956
OpenVSX Extension ast-results-2.53.0.vsix, SHA256 hash	65bd72fcddaf938cefdf55b3323ad29f649a65d4ddd6aea09afa974dfc7f105d
OpenVSX Extension cx-dev-assist-1.7.0.vsix, SHA256 hash	744c9d61b66bcd2bb5474d9afeee6c00bb7e0cd32535781da188b80eb59383e0
OpenVSX Extension checkmarx-util-1.0.4.tgz, SHA256 hash	0d66d8c7e02574ff0d3443de0585af19c903d12466d88573ed82ec788655975c
OpenVSX Extension environmentAuthChecker.js, SHA256 hash	527f795a201a6bc114394c4cfd1c74dce97381989f51a4661aafbc93a4439e90

Table 1. TeamPCP Indicators of Compromise