ThreatMon
Under Cyber Wings

TeamPCP Threat Actor

REPORT

Prepared by ThreatMon Research Team | July 2026

Executive Summary

TeamPCP is a financially motivated cybercrime group that emerged in late 2025. Its initial activity focused on exposed cloud infrastructure, which it exploited to deploy ransomware and conduct unauthorized cryptocurrency mining. This phase was relatively short-lived. Within approximately three months, the group adopted a more scalable operational model centered on credential theft through compromises of the open-source software supply chain. By early 2026, TeamPCP had become associated with a significant software supply-chain campaign.

The campaign used trusted security and developer tools as malware delivery mechanisms. It began with the compromise of the Trivy vulnerability scanner in March 2026 and subsequently affected Checkmarx KICS, the LiteLLM gateway, the Telnyx SDK, and several other developer-focused tools. The activity extended across GitHub Actions, Docker Hub, npm, PyPI, and OpenVSX.

Public reporting has linked the operation to more than 1,000 affected SaaS environments, approximately 500,000 compromised credentials, and over 300 GB of exfiltrated data. A key factor increasing the campaign's impact was its chained compromise model. Credentials obtained during one intrusion were used to access additional systems and software ecosystems. The activity is therefore better assessed as a cascading compromise of interconnected trust relationships rather than a series of unrelated security incidents.

TeamPCP also appears to maintain relationships with other cybercriminal operations. We assess that the group functions partly as an access-generation operation, supplying stolen credentials and compromised environments to multiple ransomware ecosystems. Its publicly announced partnership with the Vect ransomware group, the operation of its own CipherForce ransomware brand, and reported links to other criminal actors collectively support this assessment.

This report examines TeamPCP's origins, the development of its major campaigns, its operational methods, supporting infrastructure, relationships with other cybercriminal groups, and potential future direction.

Impact at a glance

Notable victims: European Commission (AWS), Mercor AI, Sportradar AG, JobsGO (~2.3M records)

Figures compiled from public third-party reporting.

Figure 1. Campaign impact at a glance.

Background

Emergence and classification

The threat intelligence community began paying closer attention to **TeamPCP in November 2025**, although traces of the group had appeared on messaging platforms before that point. Most researchers describe it as a **financially motivated cybercrime group**, rather than a state-backed APT or a traditional hacktivist organization.

In an interview, a spokesperson using the name **T00001B** described TeamPCP as an informal group of teenagers and young adults who had struggled to find paid work. The statement cannot be independently verified, but it does align with the group's broader behavior. Its campaigns have generally been opportunistic, financially driven, and focused on targets that could be compromised with relatively little effort.

One part of the campaign, however, does not fit neatly into that pattern. At a later stage, the group introduced a **destructive capability aimed specifically at Iranian environments**. This may indicate that political or geopolitical motives influenced at least part of the operation. At present, the available evidence is not strong enough to confirm that conclusion. It is therefore more appropriate to treat it as **a developing analytical possibility rather than an established assessment.**

From cloud exploitation to the supply chain

What makes **TeamPCP** notable is not a single advanced technique, but how quickly its operations evolved. Within a few months, the group moved from opportunistically compromising exposed cloud systems to carrying out a more focused campaign against the **open-source software supply chain**.

The reasoning behind this shift is fairly clear. Security and infrastructure tools often operate in environments that contain privileged tokens, API keys, and other sensitive credentials. Compromising one widely trusted tool can therefore provide access to far more valuable data than taking over a single internet-facing server.

This became especially effective when combined with the group's credential-theft model. Credentials stolen from one victim were used to reach another, allowing the operation to spread across connected services and development environments. As a result, **a single compromised release could expose tens of thousands of secrets across multiple organizations.**

Origin of the Group

Telegram lineage and early identities

Telegram provides one of the clearest links to **TeamPCP's earlier activity**. Public reporting indicates that the group's main channel, **"Team_PCP," was previously known as "Black Witch / PCP."** An operator associated with the group also used the handle **Persy_PCP** before the TeamPCP name became established.

These changes in names and handles are useful from an analytical perspective, as they help connect activity recorded before the TeamPCP branding emerged with the actors later operating under that identity.

The group currently maintains **two active Telegram channels**. Subscriber numbers on its primary channel increased from approximately **700 in early February 2026 to more than 1,180 by late March 2026**. Much of this growth appears to have followed increased media attention surrounding the group's software supply-chain attacks.

The cloud-native exploitation phase

In its early stages, **TeamPCP operated primarily as a cloud-exploitation group**, with little resemblance to the supply-chain actor it later became. Its targets were mostly internet-facing systems, including **unprotected Docker APIs, exposed Kubernetes control planes, and misconfigured Redis instances**.

The group relied on a broad and opportunistic approach. Rather than developing custom exploits for well-defended environments, it scanned at scale for exposed services and common configuration weaknesses, then exploited whatever access was available. This pattern of **high-volume, low-complexity targeting** remained visible in its later campaigns.

After gaining access, TeamPCP used **XMRig to mine Monero** and generate direct revenue. It also deployed tunneling tools such as **FRP and GO Simple Tunnel** to maintain access and relay network traffic. Command-and-control activity was handled through **Sliver**, an open-source framework commonly used in red-team operations.

Individually, these techniques were not especially advanced. Taken together, however, they show that the group had already developed a practical level of operational capability. That foundation later carried over into its more coordinated and technically ambitious supply-chain activity.

Attribution and Aliases

Alias map

TeamPCP has been linked to **several names, accounts, and campaign labels**, each associated with a different part of its activity. Mapping these identities is important because they appear to refer to the same broader operation.

PCPcat is the earliest documented campaign name associated with the group. **ShellForce** is the identity used in connection with its data-leak activity, while **DeadCatx3** is a GitHub account that has hosted tools linked to the operation. **CipherForce** is the group's own ransomware brand, and **Persy_PCP** appears to be an earlier Telegram identity. The group has also maintained a presence on X through the account **@pcpcats**.

Separately, TrendAI Research tracks the cluster as **SHADOW-WATER-058**. According to its reporting, the designation was assigned before several of the group's later incidents were publicly documented, based on recurring similarities in **infrastructure, tooling, and operational behavior.**

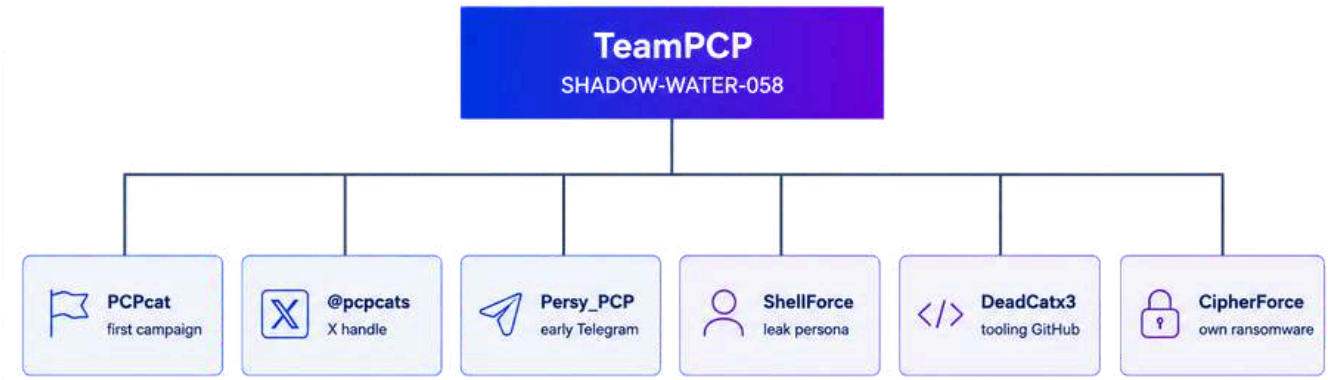

Figure 2. Alias & Identity Map

Criminal Ecosystem and Partnerships

The Vect ransomware partnership

The clearest indication that **TeamPCP operates in cooperation with other cybercriminal groups** is its reported relationship with **Vect**, a relatively new Russian-speaking ransomware-as-a-service operation. The partnership was announced through a post on BreachForums, which identified TeamPCP as the actor responsible for the Trivy and LiteLLM compromises.

According to the announcement, Vect intended to deploy ransomware within organizations affected by those intrusions. The RaaS reportedly offered affiliates **between 80% and 88% of ransom payments**, suggesting that TeamPCP's access could be monetized through Vect's extortion infrastructure.

By **15 April 2026**, Vect had reportedly begun publishing victims using data associated with credentials stolen by TeamPCP. This indicates that access obtained during the supply-chain campaign was being converted into ransomware and extortion activity **within a relatively short period**.

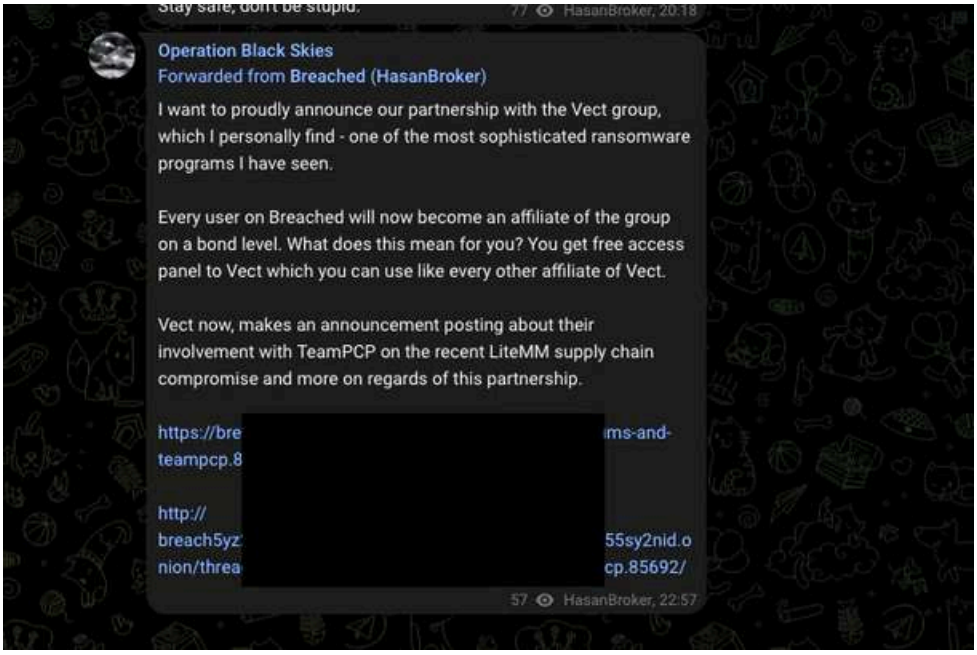

Figure 3. Operation Black Skies Telegram Screenshot

CipherForce, the in-house operation

Alongside its partnership with Vect, TeamPCP also operates **its own ransomware brand, CipherForce**. This gives the group two separate monetization channels: cooperation with an external ransomware operation and direct extortion through an internally managed project.

The relationship between these identities has also been acknowledged on Telegram. Individuals associated with the group have described **CipherForce as a newer initiative** while simultaneously identifying themselves with the **TeamPCP and ShellForce** names. These statements provide a direct link between the ransomware operation and the group's broader cybercriminal activity.

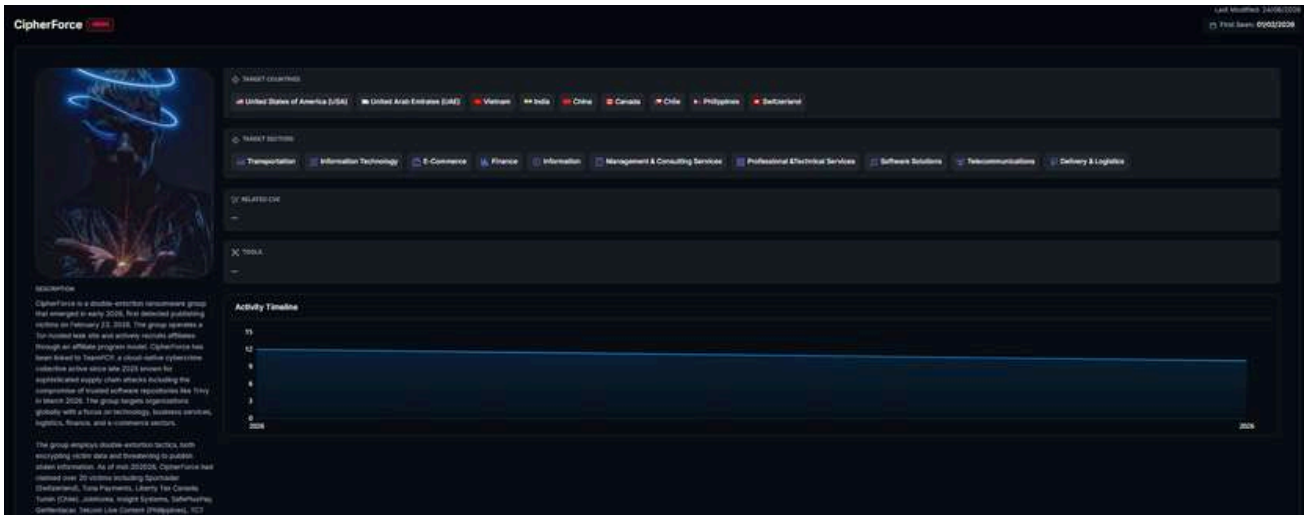

Figure 4: ThreatMon Threat Actor Monitoring Screenshot

Lapsus$, ShinyHunters, and the access engine

Public reporting has also suggested **possible links between TeamPCP and Lapsus$**. Separately, ShinyHunters was observed publishing data reportedly obtained during the European Commission breach.

TeamPCP appears to operate less as a group focused solely on conducting intrusions and directly extorting individual victims, and more as an **access-generation operation**. Its campaigns are designed to collect privileged credentials at scale and make that access available to multiple ransomware and data-extortion groups.

This model is particularly effective in a cascading supply-chain compromise. A single intrusion can produce a large number of valid and reusable credentials, allowing the group to distribute, monetize, or retain access across several victim environments rather than relying on one organization for revenue.

Leadership transition

In early April 2026, TeamPCP announced **a leadership transition on Telegram**, with the original operator, **DMT**, transferring control of the operation to an individual using the handle **T00001B**.

The new leadership stated that the group's activities would continue and that additional partnerships had already been established. This continuity, combined with TeamPCP's broader partnership model, suggests that the operation **remains active and capable of sustaining further campaigns**, rather than representing a concluded or inactive threat.

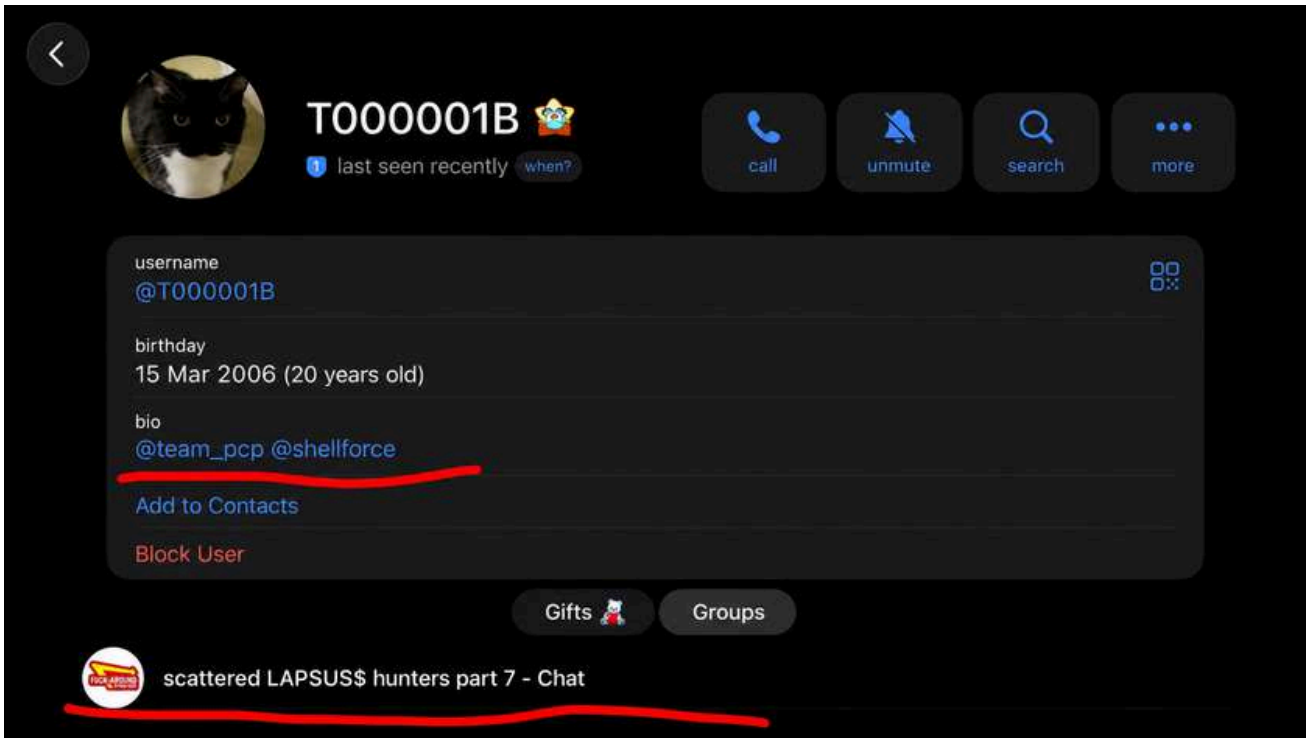

Figure 5. Threat Actor Telegram Information

The Supply Chain Pivot & Operations

The shift toward supply-chain attacks began not with a major intrusion, but with **an access token that had not been fully rotated**. From that point, the campaign developed in stages, with each compromise creating the conditions for the next. A notable feature of the operation was how rarely TeamPCP needed to rely on a new exploit. Instead, the group collected secrets from environments that executed a **compromised action, package, or software artifact**, then used those credentials to access additional repositories, registries, and developer tools. The diagram below summarizes this **cascading compromise model**, beginning with the incomplete credential rotation and showing how that initial access contributed to the later stages of the campaign.

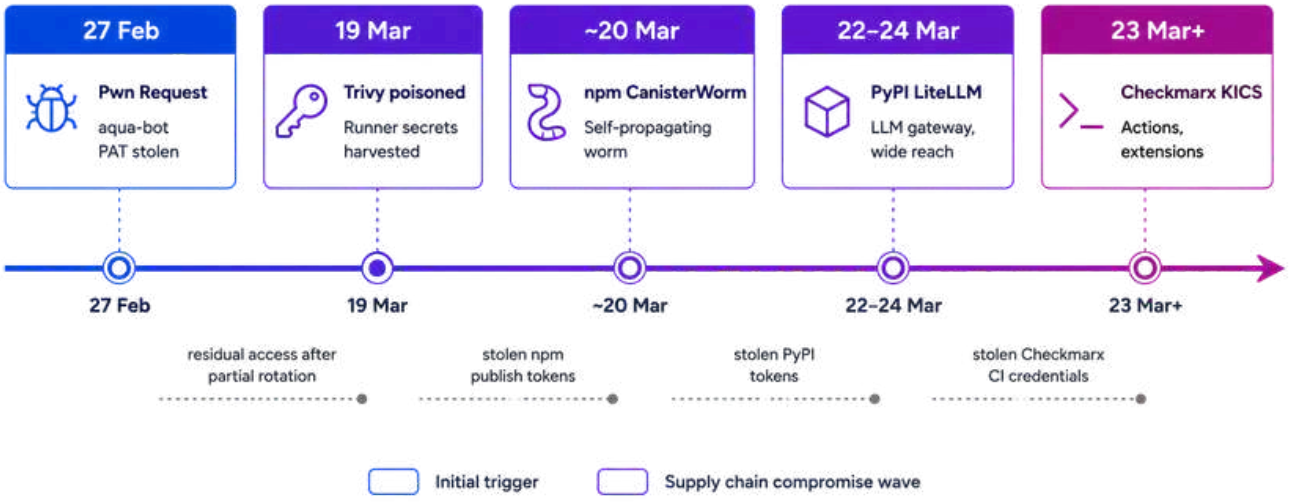

Figure 6. The credential cascade. Each stage harvested credentials that unlocked the next target. The flow is simplified for clarity; in practice secrets from the Trivy wave fed several ecosystems in parallel.

The weaknesses that enabled the campaign extend beyond the individual vendors affected. **Long-lived service account tokens** provided persistent access that remained valid even after the initial activity was detected. At the same time, CI/CD runners often contained cloud credentials and software publishing permissions while receiving less security monitoring than production systems. The use of **mutable tags rather than pinned commit hashes** created another important weakness. A force-pushed tag could redirect a trusted workflow to malicious code without an obvious change to the reference used by downstream environments. More broadly, the campaign targeted **identity and access rather than source code alone**. In modern software delivery environments, service accounts, API tokens, and publishing credentials provide access across repositories, cloud platforms, and package registries. TeamPCP's activity was structured to collect these credentials at scale and reuse them throughout the wider campaign.

Trivy supply-chain attack (19 March 2026)

The **Trivy compromise marked the first major stage of the campaign** and established the operational pattern later observed in subsequent incidents. The initial access occurred several weeks earlier. On **27 February 2026**, an actor identified as **MegaGame10418** exploited a vulnerable **pull_request_target** workflow in Trivy's CI/CD pipeline. The weakness, consistent with a **Pwn Request-style attack**, enabled the actor to exfiltrate the **aqua-bot Personal Access Token**. Aqua Security rotated the affected credentials following the incident. However, the remediation was incomplete, leaving residual access that could later be exploited.

On **19 March 2026**, TeamPCP used its remaining access to push a malicious **v0.69.4** tag to the Trivy repository. The incident, tracked as **CVE-2026-33634**, carries a **CVSS v4 score of 9.4 and a CVSS v3.1 score of 8.8**. The operators used imposter commits with a trusted contributor's identity and backdated timestamps. The malicious build contacted **scan.aquasecurtiy[.]org**, retrieved Go source files, and compiled them into the release. Because Trivy is widely used in enterprise CI/CD pipelines, the compromise turned a trusted security tool into a **malware distribution channel**.

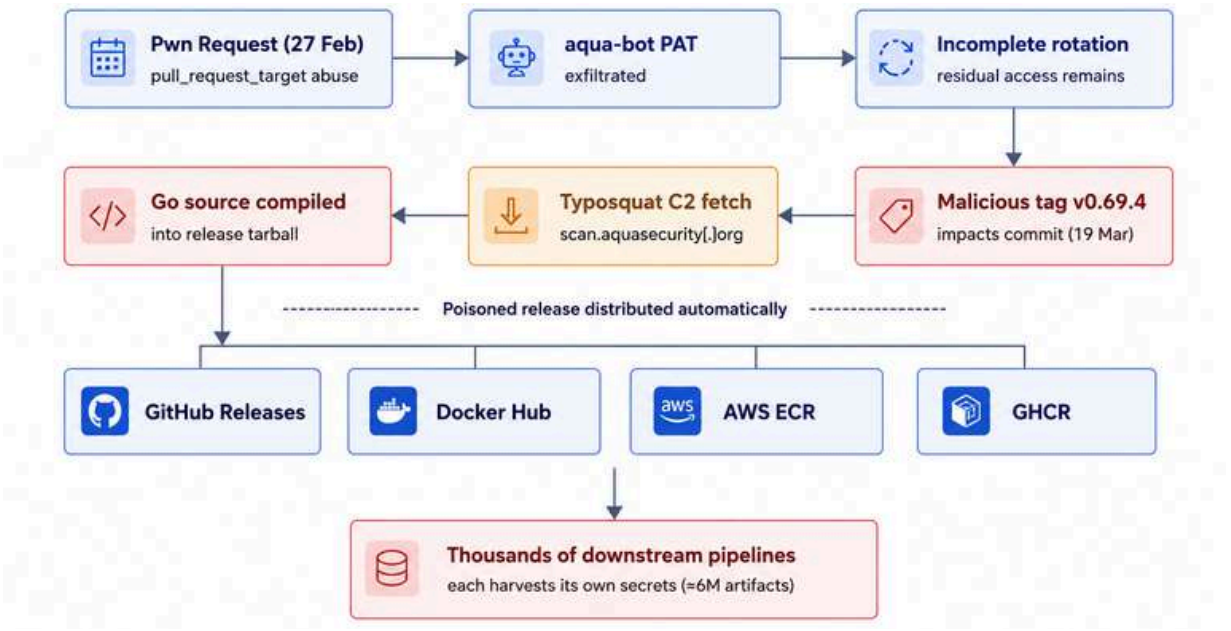

Figure 7. Trivy infection chain, from the February Pwn Request through automatic distribution across four registries.

The compromised release propagated through **GitHub Releases, Docker Hub, AWS ECR, and GitHub Container Registry**, reaching thousands of downstream pipelines within approximately four hours. Each affected pipeline collected stored credentials and transmitted them to the attackers. These stolen secrets then enabled **the subsequent stages of the campaign.**

CanisterWorm npm attack (circa 20 March 2026)

Among the credentials stolen from affected pipelines were **npm publishing tokens**, which were later used by a self-propagating worm tracked as **CanisterWorm**. This marked the point at which the campaign moved beyond a single compromised release. Using a stolen token, the worm queried the npm API to identify the account owner, enumerated every package the account could publish, increased each package's patch version, and released a malicious update. The process was fully automated and could compromise **dozens of packages in less than a minute.**

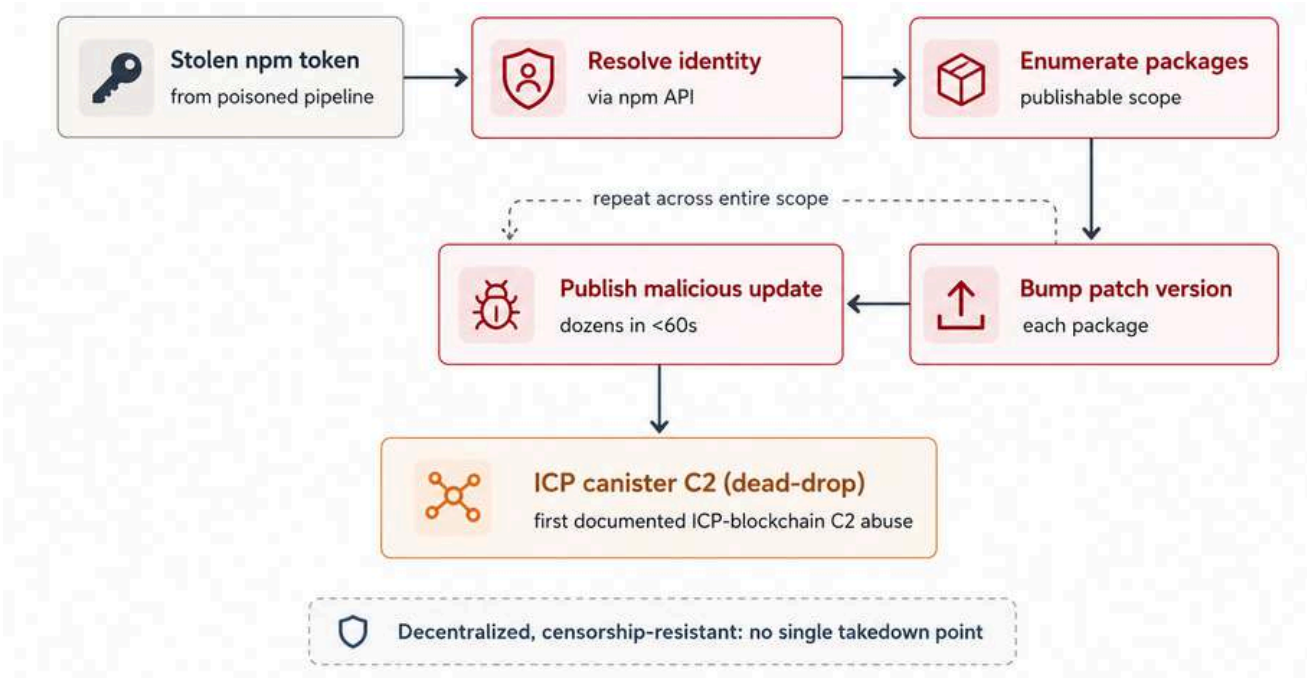

Figure 8. CanisterWorm propagation loop and its ICP-based command and control.

CanisterWorm was named after its use of an **Internet Computer Protocol canister for command-and-control and payload delivery**. Rather than relying solely on conventional domains, the malware used the canister as a decentralized dead-drop mechanism.Public reporting describes this as the **first documented use of the ICP blockchain for C2 activity**. This approach improved resilience by removing a single point of failure. Even after parts of the campaign's traditional infrastructure were seized, the **ICP-based fallback remained operational.**

LiteLLM PyPI compromise

Stolen **PyPI publishing tokens** were later used to compromise **LiteLLM**, an open-source library that routes requests across multiple LLM providers and reportedly receives more than 95 million monthly **downloads**. The malicious code was inserted into a single file within the PyPI wheel, where a short block was placed between unrelated sections of legitimate code. PyPI records show that the affected versions, **1.82.7 and 1.82.8**, were published minutes apart, approximately one day after the KICS compromise. The second version introduced an additional injection method, suggesting that the operators were modifying the payload during the campaign.

Targeting a production library also increased the potential impact. Production environments often contain **long-lived cloud IAM credentials and Kubernetes service account tokens**, which generally provide broader access than credentials exposed during temporary build processes.

Checkmarx KICS multichannel poisoning (23 March and 22 April 2026)

The Checkmarx activity shows how the campaign **moved from one vendor environment to another** while compromising several distribution channels in parallel. Stolen Checkmarx CI credentials provided access to the company's GitHub Actions environment, including **checkmarx/ast-github-action and checkmarx/kics-github-action**. On **22 April 2026**, the operators targeted three channels at the same time: the official Docker Hub repository, VS Code and OpenVSX extensions, and a GitHub Actions workflow. The malicious activity continued for approximately **83 minutes before detection**.

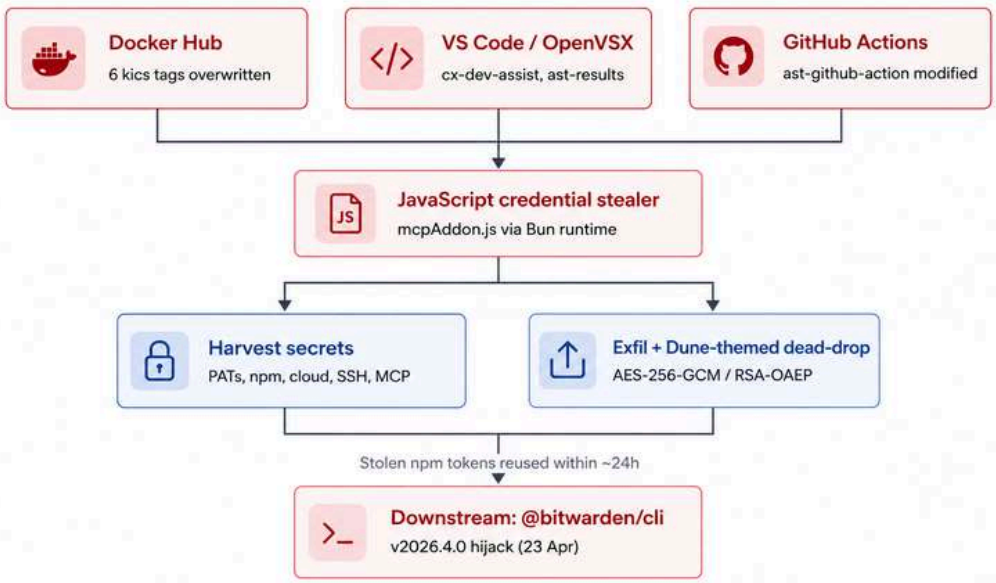

Figure 9. Checkmarx KICS multichannel poisoning and the downstream Bitwarden CLI hijack.

All three delivery channels deployed the same **JavaScript credential stealer**, named mcpAddon.js. The payload was executed through a downloaded Bun runtime during a process that appeared to be a legitimate security scan. The stealer targeted **GitHub PATs, npm tokens, cloud credentials, SSH keys, and AI tooling configuration files**. Collected data was encrypted using **AES-256-GCM and RSA** before being sent to an endpoint impersonating Checkmarx. As a secondary exfiltration method, the malware created **public GitHub repositories with Dune-themed names** and used them as dead-drop locations.

Downstream Bitwarden CLI hijack

Within approximately one day, npm tokens stolen during the KICS compromise were used to publish a malicious version of **@bitwarden/cli.** This exposed developers and CI/CD pipelines that downloaded the Bitwarden CLI during the affected period. The KICS and Bitwarden payloads used the **same command-and-control domain, encryption method, and runtime**, indicating that they were part of the same operation.

The Bitwarden variant also included a fallback mechanism that queried the **GitHub commit search API** for alternative exfiltration domains. This allowed the operators to rotate infrastructure without releasing an updated payload.

Elementary-data script injection (24 April 2026)

The elementary-data incident used a simpler attack path than the KICS compromise but required **no stolen maintainer credentials**. An attacker-controlled account posted a malicious comment on an open pull request. Because the workflow inserted **github.event.comment.body** directly into a shell command without sanitization, GitHub Actions executed the injected command on a runner with a **GITHUB_TOKEN granting repository write access**.

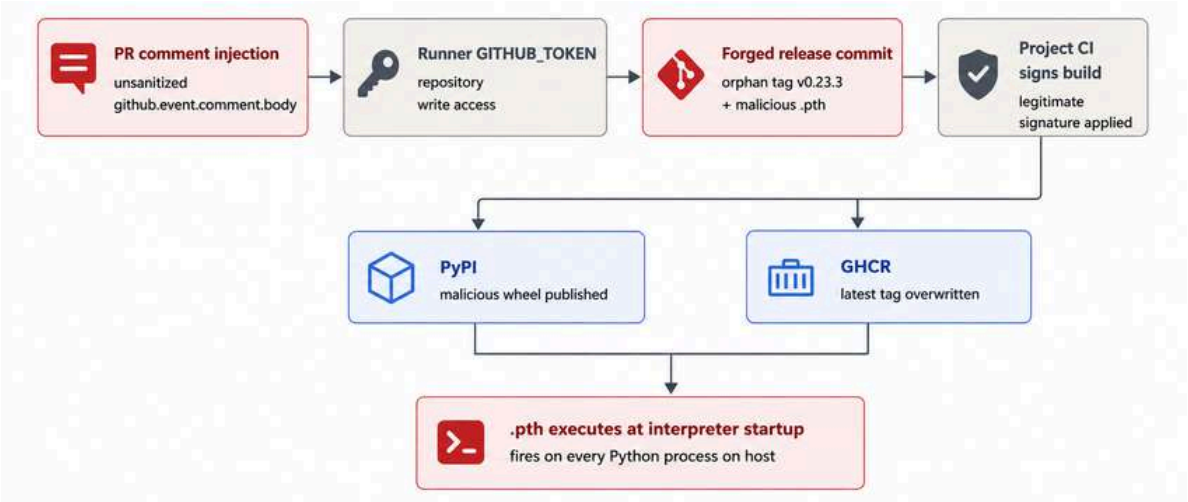

Figure 10. elementary-data script injection chain, from pull request comment to a CI-signed malicious release.

Using the exposed token, the attacker created an orphan-tagged commit containing a malicious **.pth file** and then triggered the project's legitimate release workflow. Because the package was built and signed through the project's own infrastructure, it received a **valid cryptographic signature** and passed standard signature verification. The malicious wheel was subsequently published to **PyPI and GHCR**. Python automatically processes **.pth** files during interpreter startup. As a result, the payload executed on affected systems **before the application code and without requiring the package to be explicitly imported**.

Telnyx and Xinference

TeamPCP continued targeting PyPI packages using a similar approach. On **27 March 2026**, the group compromised the **Telnyx Python SDK**, which reportedly receives around **742,000 monthly downloads**. The payload used the same steganographic delivery method observed in earlier incidents, but communicated with a newly introduced C2 server. **Xinference** was compromised during the **22 April 2026** wave alongside KICS. Both incidents contained the same operator branding and **Session identifier**, linking the Python-focused compromises to the same broader operation rather than separate threat actors.

Campaign Timeline

Figure 11. Campaign timeline, March to April 2026.

Date	Event	Significance
19 Mar 2026	Trivy GitHub Actions compromised	First confirmed TeamPCP supply chain wave
23 Mar 2026	KICS Action, LiteLLM PyPI, Checkmarx OpenVSX	Expansion into a second vendor and package registries
27 Mar 2026	Telnyx Python SDK compromised	Continued expansion using the same delivery pattern
2-3 Apr	European Commission AWS breach attributed	First nation-state-tier institutional victim
15 Apr	Vect ransomware begins publishing victims	Active monetization of stolen credentials
22 Apr	Xinference and KICS multichannel wave	Simultaneous Docker Hub, OpenVSX, and Actions poisoning
23 Apr	Bitwarden CLI downstream hijack	Stolen npm tokens reused within ~24 hours
24 Apr	elementary-data script injection	No stolen credentials needed; CI-signed malicious release

Tools/Components used by TeamPCP

Tool or component	Category	Observed use
XMRig	Cryptocurrency miner	Unauthorized Monero mining
Sliver	Command-and-control framework	Remote access and post-exploitation
FRP	Tunneling utility	Traffic forwarding and relay
GO Simple Tunnel	Tunneling utility	Encrypted network tunneling
CanisterWorm	npm worm	Automated poisoning of accessible packages
mcpAddon.js	JavaScript credential stealer	Collection of tokens, keys, and cloud credentials
Bun	Legitimate JavaScript runtime	Execution of the KICS stealer
Telnyx Windows loader	Multi-stage malware loader	Decoding and loading the Windows implant
Telnyx inner DLL	Remote-access implant	Command execution and file collection
dllhost.exe	Windows system component	Hosting the decoded payload in memory
Malicious .pth file	Python startup mechanism	Automatic execution when Python starts
ICP canister	Decentralized infrastructure	Payload dead-drop and resilient delivery

Indicators of Compromise

Trivy

Indicator	Type	Purpose
scan[.]aquasecurtiy[.]org	Domain	Primary credential exfiltration C2
45[.]148[.]10[.]212	IPv4	IP address resolving from the typosquatted C2
tdtqy-oyaaa-aaaae-af2dq-cai[.]raw[.]icp0[.]io	Domain	ICP-hosted payload dead-drop
plug-tab-protective-relay[.]trycloudflare[.]com	Domain	GitHub Actions credential exfiltration

LiteLLM

Indicator	Type	Purpose
models[.]litellm[.]cloud	Domain	Encrypted credential archive exfiltration
checkmarx[.]zone	Domain	Persistence and payload infrastructure
checkmarx[.]zone/raw	URL Path	Returns the next-stage binary URL

Telnyx

Indicator	Type	Purpose
83[.]142[.]209[.]203	IPv4	Telnyx payload and exfiltration server
83[.]142[.]209[.]203:8080	IPv4/port	HTTP C2 infrastructure
hxxp://83[.]142[.]209[.]203:8080/hangup[.]wav	URL	Windows payload carrier
hxxp://83[.]142[.]209[.]203:8080/ringtone[.]wav	URL	Linux/non-Windows payload carrier
hxxp://83[.]142[.]209[.]203:8080/	URL	Credential archive exfiltration endpoint
modesl[.]litellm[.]cloud	Domain	Reported TeamPCP infrastructure

Malicious File Hashes

Indicator	Type
887e1f5b5b50162a60bd03b66269e0ae545d0aef0583c1c5b00972152ad7e073	hash
f7084b0229dce605ccc5506b14acd4d954a496da4b6134a294844ca8d601970d	hash
bef7e2c5a92c4fa4af17791efc1e46311c0f304796f1172fce192f5efc40f5d7	hash
e64e152afe2c722d750f10259626f357cdea40420c5eedae37969fbf13abbecf	hash
ecce7ae5ffc9f57bb70efd3ea136a2923f701334a8cd47d4fbf01a97fd22859c	hash
d5edd791021b966fb6af0ace09319ace7b97d6642363ef27b3d5056ca654a94c	hash
e6310d8a003d7ac101a6b1cd39ff6c6a88ee454b767c1bdce143e04bc1113243	hash
6328a34b26a63423b555a61f89a6a0525a534e9c88584c815d937910f1ddd538	hash
0880819ef821cff918960a39c1c1aada55a5593c61c608ea9215da858a86e349	hash

CanisterWorm hashes

Indicator	Type	Purpose
e9b1e069efc778c1e77fb3f5fcc3bd3580bbc810604cbf4347897ddb4b8c163b	index.js	Wave 1 dry run
61ff00a81b19624adaad425b9129ba2f312f4ab76fb5ddc2c628a5037d31a4ba	index.js	Wave 2, armed ICP backdoor
0c0d206d5e68c0cf64d57ffa8bc5b1dad54f2dda52f24e96e02e237498cb9c3a	index.js	Wave 3, self-propagating test
c37c0ae9641d2e5329fcdee847a756bf1140fdb7f0b7c78a40fdc39055e7d926	index.js	Wave 4, final form
f398f06eefcd3558c38820a397e3193856e4e6e7c67f81ecc8e533275284b152	deploy.js	Initial verbose variant
7df6cef7ab9aae2ea08f2f872f6456b5d51d896ddda907a238cd6668ccdc4bb7	deploy.js	Added --tag latest
5e2ba7c4c53fa6e0cef58011acdd50682cf83fb7b989712d2fcf1b5173bad956	deploy.js	Minified silent variant

LiteLLM hashes

Indicator	Purpose
8395c3268d5c5dbae1c7c6d4bb3c318c752ba4608cfcd90eb97ffb94a910eac2	LiteLLM 1.82.7 wheel
d2a0d5f564628773b6af7b9c11f6b86531a875bd2d186d7081ab62748a800ebb	LiteLLM 1.82.8 wheel
a0d229be8efcb2f9135e2ad55ba275b76ddcfeb55fa4370e0a522a5bdee0120b	Compromised proxy_server.py
71e35aef03099cd1f2d6446734273025a163597de93912df321ef118bf135238	litellm_init.pth

Checkmarx OpenVSX hashes

Indicator	Purpose
65bd72fcddaf938cefdf55b3323ad29f649a65d4ddd6aea09afa974dfc7f105d	ast-results-2.53.0.vsix
744c9d61b66bcd2bb5474d9afeee6c00bb7e0cd32535781da188b80eb59383e0	cx-dev-assist-1.7.0.vsix

Telnyx Windows malware hashes

Indicator	Purpose
7290353a3bc2b18e9ea574d3294b09e28edaa6b038285bb101cf09760f187dcd	msbuild.exe
7e270255567866d37ad56e3f06977b695e39530eede74a10a0848ba71560cb45	Embedded malicious PNG
dafc1cc5d39bc303562d8587b698b6351e843b77c01764efa8b423a36b88fa6d	Inner DLL payload

Malicious Packages and Versions

Ecosystem	Package or artifact	Malicious/affected version
Trivy	aquasecurity/trivy	v0.69.4
Docker Hub	Trivy images	0.69.5, 0.69.6
PyPI	litellm	1.82.7, 1.82.8
PyPI	telnyx	4.87.1, 4.87.2
OpenVSX	ast-results	2,530
OpenVSX	cx-dev-assist	170
GitHub Actions	checkmarx/kics-github-action	Actions executed during affected window

Filesystem and Persistence Indicators

LiteLLM and Trivy variants

Indicator	Description
~/.config/sysmon/sysmon.py	Python persistence backdoor
~/.config/systemd/user/sysmon.service	User-level systemd service
/tmp/pglog	Downloaded next-stage binary
/tmp/.pg_state	Previously downloaded URL/state tracking
litellm_init.pth	Executes payload during Python startup
tpcp.tar.gz	Encrypted exfiltration archive
X-Filename: tpcp.tar.gz	HTTP exfiltration header
node-setup-*	Malicious Kubernetes pod naming pattern
kube-system/node-setup-*	Expected attacker pod location
tpcp-docs	Fallback GitHub exfiltration repository

CanisterWorm

Indicator	Description
~/.local/share/pgmon/service.py	CanisterWorm Python backdoor
~/.config/systemd/user/pgmon.service	Persistence unit
/tmp/pglog	Downloaded binary
/tmp/.pg_state	C2 payload tracking file

Telnyx

Indicator	Description
%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\msbuild.exe	Persistent executable
%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\msbuild.exe.lock	Twelve-hour execution lock
~/.config/audiomon/audiomon.py	Python persistence payload
~/.config/systemd/user/audiomon.service	User-level systemd service
hangup.wav	Encoded executable carrier
ringtone.wav	Encoded Python payload carrier

Checkmarx

Indicator	Description
checkmarx.ast-results-2.53.0.vsix	Malicious OpenVSX extension
checkmarx.cx-dev-assist-1.7.0.vsix	Malicious OpenVSX extension
setup.sh	Script injected into affected CI/CD runners
tpcp.tar.gz	Credential archive
tpcp-docs	Unexpected GitHub repository
docs-tpcp	Alternate suspicious repository name

GitHub Actions Commit Indicators

Indicator
8afa9b9f9183b4e00c46e2b82d34047e3c177bd0
386c0f18ac3d7f2ed33e2d884761119f4024ff8a
384add36b52014a0f99c0ab3a3d58bd47e53d00f
7a4b6f31edb8db48cc22a1d41e298b38c4a6417e
6d8d730153d6151e03549f276faca0275ed9c7b2
99b93c070aac11b52dfc3e41a55cbb24a331ae75
f4436225d8a5fd1715d3c2290d8a50643e726031

MITRE ATT&CK Mapping

Tactic	ID	Technique	Observation
Initial Access	T1195.002	Supply Chain: Compromise Software Supply Chain	Poisoned Trivy, KICS, Docker Hub images, VS Code extensions, and PyPI packages
Initial Access	T1078	Valid Accounts	Stolen publisher credentials reused to republish across channels
Execution	T1059.007	Command and Scripting: JavaScript	KICS mcpAddon.js executed via Bun runtime
Execution	T1059.006	Command and Scripting: Python	.pth payloads executed at interpreter startup
Persistence	T1543.002	Create/Modify System Process	pgmon.service with Restart=always
Persistence	T1546.018	Boot or Logon Autostart Execution	.pth fires on every Python interpreter startup
Defense Evasion	T1027	Obfuscated Files or Information	WAV steganography; hybrid and XOR crypto by operation
Defense Evasion	T1070.004	Indicator Removal: File Deletion	Temp-directory context manager deletes archives on exit
Credential Access	T1552.001	Unsecured Credentials in Files	SSH keys, cloud files, developer tokens harvested from disk
Credential Access	T1552.005	Cloud Instance Metadata API	EC2 IMDS and ECS task credential endpoints queried

Discovery	T1526	Cloud Service Discovery	Live Secrets Manager and SSM enumeration
Collection	T1560	Archive Collected Data	Hybrid-encrypted archives (tpcp.tar.gz, trin.tar.gz)
Lateral Movement	T1021.004	Remote Services: SSH	SSH key and auth log harvesting for subnet spread
Command and Control	T1102.001	Dead Drop Resolver	GitHub commit-search API and ICP canister fallback C2
Exfiltration	T1567.001	Exfiltration to Code Repository	Dead-drop to auto-created victim GitHub repositories
Impact	T1485	Data Destruction	Iran-targeted wiper on Kubernetes nodes and hosts
Impact	T1657	Financial Theft	Cryptocurrency wallet harvesting

Defensive Recommendations

The whole campaign traces back to a credential rotation that was not done atomically. So the advice below centers on treating identity and supply chain security as one problem, and on the structural controls that blunt this class of attack no matter which project gets hit next.

- Pin CI/CD dependencies and GitHub Actions to immutable commit SHAs rather than mutable tags. Tag-based trust becomes a silent credential collection point the moment a tag is force-pushed.
- Perform credential rotation atomically. Rotating one token is incomplete unless every linked account and residual access path is closed at the same time. Rotate from a clean host, not from within a potentially compromised CI environment.
- Harden non-human identities by narrowing token scopes, enforcing least privilege, and segmenting CI infrastructure from production systems at the network level.
- Apply network egress controls and container image digest pinning on CI runners. These two structural controls would have blocked exfiltration in several of the confirmed waves.
- Audit GitHub Actions workflows for user-controlled expressions interpolated directly into shell commands, the structural weakness that enabled the elementary-data attack.
- Expand monitoring to the software delivery pipeline. Alert on Python interpreter processes initiating outbound HTTPS at startup, Bun runtime execution in CI, unexpected privileged DaemonSets, and CloudTrail Secrets Manager enumeration from pipeline roles.
- Hunt for campaign markers: archives named tpcp.tar.gz or trin.tar.gz, repositories following the Dune-themed naming pattern, the actor-branded exfiltration headers, and pinned versions of the compromised packages in lock files.
- Enforce artifact signing and provenance validation, and increase SBOM visibility across the release process.

Conclusion

TeamPCP's development from opportunistic cloud exploitation into a coordinated software supply-chain operation illustrates how quickly a financially motivated group can scale when it begins targeting trust rather than individual systems.

The campaign's effectiveness did not depend on a single advanced exploit. It came from combining exposed infrastructure, unsafe CI/CD workflows, incomplete credential rotation, overprivileged automation accounts, mutable release references, and the implicit trust placed in security tools and open-source packages. Once the group gained access to one trusted environment, the credentials collected there opened the next repository, registry, package, or cloud account. That cascading model turned several apparently separate incidents into one connected operation.

The most important lesson is that modern supply-chain security is primarily an identity and workflow problem. Signed artifacts, trusted vendor names, official repositories, and familiar version numbers cannot provide assurance when the systems producing and publishing them have been compromised. Defenders must be able to verify how software was built, which identity authorized its release, what permissions were available during execution, and whether the resulting behavior matches its stated purpose.

TeamPCP also appears to have built an operational model that extends beyond initial compromise. By combining its own CipherForce operation with external ransomware and extortion partnerships, the group can monetize stolen access through several channels while separating the supply-chain intrusion from the final impact. This makes the threat more durable and complicates attribution, containment, and victim notification.

Organizations should therefore assume that credentials exposed through development tooling may be reused quickly and outside the environment in which they were stolen. Effective defense requires short-lived credentials, isolated runners, tightly scoped publishing rights, immutable dependency references, verifiable build provenance, continuous release monitoring, and an incident-response process capable of revoking trust across multiple ecosystems at once.

TeamPCP may change its infrastructure, package names, identities, and partners, but the weaknesses it exploits are structural and widely shared. Closing those gaps will not only reduce exposure to this group; it will improve resilience against the broader class of supply-chain actors that increasingly target the identities and automation underpinning modern software delivery.

ThreatMon
Under Cyber Wings

More Information About ThreatMon

One Platform for all intelligence needs.

ThreatMon End-to-end intelligence is a cutting-edge, cloud-based SaaS platform that continuously monitors the dark and surface web, providing early warnings and actionable insights into emerging threats.

We are a SaaS platform designed to help businesses proactively detect and address threats before a cyber attack occurs. Unlike traditional cyber threat intelligence, we provide comprehensive and holistic cyber intelligence.

- *Attack Surface Intelligence*
- *Fraud Intelligence*
- *Dark and Surface Web Intelligence*
- *Threat Intelligence*

Contact Us:

Email Address
team@threatmonit.io

https://x.com/MonThreat

https://www.linkedin.com/company/threatmon