MAY

pellera

THREAT INTEL REPORT 2026

Prepared by: Pellera Threat Intel Team
pellera.com | 800.747.8585

Driving Momentum. Accelerating Change. Empowering IT Transformation.

Pellera Technologies was born out of the combined expertise of Converge Technology Solutions and Mainline Information Systems, two industry leaders with over 35+ years of experience and a shared vision for innovation. Together, we empower businesses to achieve greater efficiency, adaptability, and growth for today and tomorrow.

Our commitment is to reshape what's possible with IT, offering advanced solutions in digital infrastructure, cloud, cybersecurity, and AI. We don't just deliver technology—we partner with you to build tailored strategies designed to simplify complexities, unlock opportunities, and drive transformational outcomes.

At Pellera, momentum builds here through collaborative, people-first technology designed to fuel progress and deliver measurable impact.

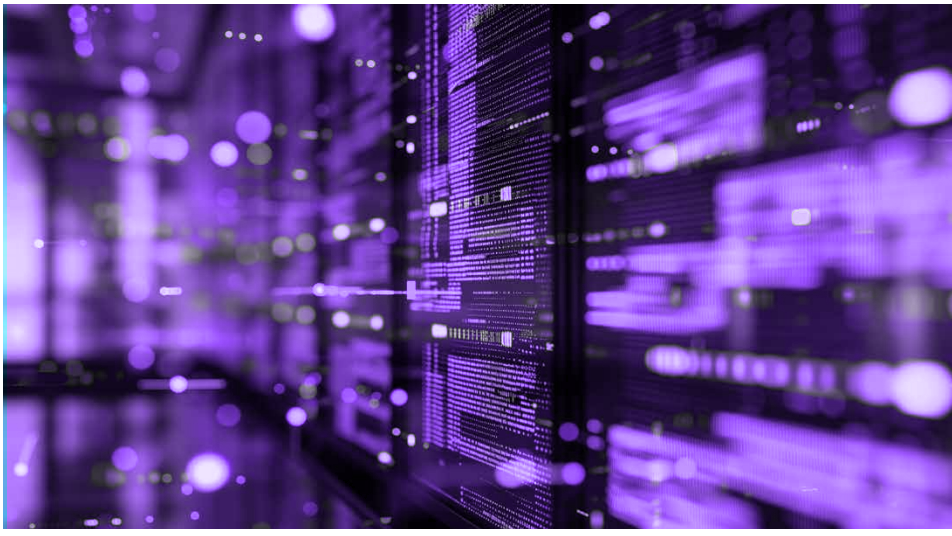
pellera

Observations for May 2026

May's attacks went after inherited trust rather than breaking through controls. **AI agent sessions**, **OAuth grants**, and **developer toolchains** were the surface across all three stories. In each case the attacker did not need a zero-day. **They found a door left unlocked by a configuration default or an unreviewed grant.**

TeamPCP ran back-to-back supply chain campaigns and used the first to fund the second. **On May 11** the group extracted OIDC tokens from GitHub Actions runner memory to compromise **TanStack**, **UiPath**, and **Mistral AI** npm packages without stealing a single credential. The **Mini Shai-Hulud worm spread those secrets into the AntV ecosystem a week later**, dropped persistent backdoors in Claude Code and VS Code configurations, and used **stolen tokens to create over 2,200 public GitHub repositories**. On May 19 GitHub confirmed that a malicious Nx Console VS Code extension had given the **group access to 3,800 of its internal repositories**. Famous Chollima ran a parallel operation, spending seven months building package trust before manipulating Claude Opus into co-authoring a commit that introduced the PromptMink malware into a cryptocurrency trading agent.

Microsoft's Azure SRE Agent carried a critical multi-tenant authentication flaw that let any outsider with a free Azure account watch another organization's live agent session in real time user prompts, executed commands, internal reasoning, and plaintext credentials with no victim-side logs generated. **Enclave disclosed CVE-2026-32173 on April 20**; the vulnerability had existed since general availability on March 10. The OAuth story ran on the same logic. A vendor employee's **Lumma Stealer infection** in February gave attackers a token for a third-party AI tool authorized in Vercel's Google Workspace. That token provided access through April through password resets, through MFA enforcement because no one revoked the grant. Malwarebytes separately documented silent OAuth redirect abuse that routes users to attacker infrastructure through legitimate Microsoft and Google authentication URLs, no token theft required.



Executive Overview

AI AGENTS IN OUR INFRASTRUCTURE

Audience

- CISO
- Cloud Security Architects
- IT Operations Managers & Teams
- Security Operations Team
- Identity & Access Management Leads
- Risk Management Professionals

RISK

GEOGRAPHIC SCOPE

BUSINESS IMPACT

On April 20, researchers at Enclave disclosed CVE-2026-32173, a critical improper authentication vulnerability in Microsoft's Azure SRE Agent rated at CVSS 8.6. The flaw allowed any outsider with a free Azure account to observe another organization's AI operations agent in real time. The cross-tenant exposure stemmed from an Entra ID app registration configured as multi-tenant, meaning any account from any Microsoft cloud tenant could obtain a valid token. The WebSocket communication channel accepted those tokens and broadcast all agent events to every connected client without filtering by identity or tenant.

The scope of exposure was substantial. Every message sent to the agent, every response, the agent's internal reasoning traces before each action, every command executed including full command details, and every command output including plaintext credentials were visible to any connected outsider. In the researchers' test environment, routine agent tasks returned deployment credentials for live web applications in plain text. The victim organization generated no logs of the connection, so neither detection nor forensic investigation was possible at the time. Microsoft's Azure App Service team uses Azure SRE Agent to cut incident resolution time from 40 hours to three minutes. That performance profile indicates the agent operates with deep access to production infrastructure.

READ MORE >> AI AGENTS IN OUR INFRASTRUCTURE

SUPPLY CHAIN ATTACKS ESCALATE

Audience

- CISO
- Application Security Leads
- DevOps & Platform Engineering
- Security Operations Teams
- Incident Response Teams
- Software Engineering Leadership

RISK

GEOGRAPHIC SCOPE

BUSINESS IMPACT

On May 11, TeamPCP, also tracked as UNC6780, launched a coordinated supply chain attack against the npm and PyPI ecosystems, compromising packages across the TanStack, UiPath, Mistral AI, and Guardrails AI namespaces. The TanStack compromise targeted @tanstack/react-router, one of the most widely used routing libraries in the React ecosystem at approximately 12 million weekly downloads. The attack used a novel GitHub Actions cache-poisoning technique: the attacker created a fork, opened a pull request that triggered a pull_request_target workflow, and poisoned the GitHub Actions pnpm cache. When legitimate maintainer PRs were later merged, the release workflow restored the

poisoned cache and the attacker extracted OIDC tokens directly from runner process memory using /proc/<pid>/mem, publishing malicious packages without ever stealing npm credentials.

The Mini Shai-Hulud worm extended the campaign into the AntV ecosystem the following week, harvesting credentials from 130-plus file paths and exfiltrating through a GitHub API dead-drop in the legitimate antvis/G2 repository. Stolen tokens were used to create over 2,200 public GitHub repositories. On May 19, GitHub confirmed a separate breach: an employee had installed a malicious Nx Console VS Code extension that harvested developer secrets, allowing TeamPCP to clone approximately 3,800 internal repositories and list them for sale at upwards of $50,000 USD.

READ MORE >> SUPPLY CHAIN ATTACKS ESCALATE

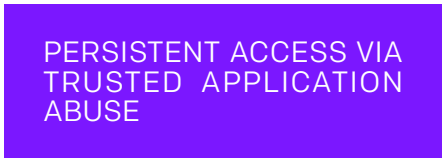
Audience

- **CISO**
- **IT Security Managers**
- **Identity & Access Management Leads**
- **Security Operations Team**
- **Cloud Security Architects**
- **Incident Response Teams**

OAUTH ATTACKS ON THE RISE

RISK

GEOGRAPHIC SCOPE



BUSINESS IMPACT

In February 2026, a vendor employee's corporate device picked up Lumma Stealer. By April, that infection had given attackers two months of access to Vercel's internal systems via an OAuth token nobody revoked. The token belonged to a third-party AI productivity tool authorized inside Vercel's Google Workspace. Password resets and MFA enforcement throughout the dwell period did nothing — the access ran through an authorized application grant, not a credential.

Malwarebytes documented a parallel attack pattern that does not require token theft at all. Attackers are abusing OAuth's built-in redirect mechanism by crafting URLs that start with legitimate Microsoft or Google authentication domains and include intentionally invalid scopes, triggering a silent authentication flow that redirects the user to attacker-controlled infrastructure without completing sign-in or requiring token capture. The link appears to originate from login.microsoftonline.com or accounts.google.com, passes URL reputation filters, and delivers the victim to a phishing or malware page through the provider's own redirect infrastructure.

READ MORE >> OAUTH ATTACKS ON THE RISE

Tactical Guidance

AI AGENTS IN OUR INFRASTRUCTURE

Overview & Impact

Microsoft's own Azure App Service team uses Azure SRE Agent to cut mean incident resolution from 40 hours to three minutes. That is what tells you what the agent is doing. It watches for alerts, diagnoses outages, restarts services, rolls back deployments, and runs commands across production infrastructure around the clock. It has access to source code, logs, system metrics, PagerDuty, ServiceNow, and anything else the team uses to run the environment. Watching that agent's live session means watching all of it.

The flaw was in /agentHub, the WebSocket endpoint the agent streams activity through. Connecting required a token. The Entra ID app registration behind it was configured as multi-tenant, which meant any account from any Microsoft cloud tenant could get a valid token from Microsoft's own authentication infrastructure. The hub checked whether the token was legitimate and whether the audience claim matched. It never asked whether the caller belonged to the target's tenant or held any role on the target resource. Once connected, every event broadcast to all clients with no filtering: user prompts, agent responses, internal reasoning before each action, every command executed with full detail, and every line of output — including plaintext deployment credentials returned during routine tasks.

One free Azure account. The target agent's address, which follows a predictable format. Fifteen lines of code. Every deployed instance was reachable this way. The victim organization's logs showed nothing. The only record of the connection was on the attacker's machine.

The exposure path, step by step:

- Attacker creates a free Azure account in any Microsoft cloud tenant
- Attacker obtains a valid token from Microsoft's authentication infrastructure using the multi-tenant app registration
- Attacker connects to the target organization's /agentHub WebSocket endpoint using the valid token
- The hub confirms token validity and audience but does not verify tenant ownership
- All agent events stream to the attacker in real time: user prompts, agent responses, reasoning traces, executed commands, and command output including credentials
- No logs of the connection are generated on the victim organization's side
- **CVE-2026-32173 rated critical, CVSS 8.6**; any free Azure account could observe any organization's Azure SRE Agent sessions.
- Exposed data included user prompts, agent reasoning, executed commands, and plaintext credentials returned during routine tasks.
- No victim-side logs generated — real-time detection and post-incident investigation were both impossible.
- Attack required one free Azure account, a predictable target URL, and 15 lines of code.
- Microsoft's Azure App Service team uses the agent to cut incident resolution from 40 hours to 3 minutes, reflecting the depth of production access it holds.

Observations

- Multi-tenant Entra ID app registration is not an edge-case misconfiguration. It was the default on Azure SRE Agent at general availability. Any AI agent product built on the same pattern carries this flaw until someone explicitly checks.
- Everything the attacker observed through that WebSocket — commands, reasoning traces, credentials in command output — reflects access you would need a privileged admin account to get through traditional means. The agent had that access. The streaming channel did not protect it.
- The victim side had zero logs. Not incomplete logging, zero. You cannot detect, investigate, or scope a breach you cannot see. That is not a defect in one product. It is what happens when streaming channels are not designed with forensic visibility as a requirement.
- Token validity and audience claim are not the same as tenant ownership. The hub passed both checks and still let in anyone with a free Azure account. Tenant isolation needs to be an explicit validation step, not an assumed property of token issuance.

Guidance

Strategic Intelligence

Trend

- Azure SRE Agent hit general availability on March 10. CVE-2026-32173 was disclosed April 20. The vulnerability was there at launch. That gap is not unique to this product. Security review that treats AI agents as a distinct privileged-access category needs to happen before GA, not after the researcher publishes.
- An AI agent with production access is a privileged actor. It reads credentials, executes commands against live infrastructure, and integrates with incident management and source control. The identity controls, session logging, and least-privilege constraints that govern a human admin need to govern it too. Most deployments skip this.
- SOC alerting assumes dwell time of days. An autonomous agent can touch and return a full credential set in one task cycle. The window is not hours, it is seconds. Detection logic and playbooks built for human operators do not fit the speed at which agents move.

Operational Intelligence

Threat Vectors

- Multi-tenant Entra ID app registrations on AI agent platforms that accept tokens from any tenant
- WebSocket or streaming communication channels that broadcast events without per-connection identity filtering
- AI agents with direct access to secrets, credentials, and deployment infrastructure
- Predictable or discoverable agent endpoint URLs that reduce reconnaissance requirements
- Context injection and prompt manipulation targeting AI systems that write or review code

Monitoring & Detection Gaps

- No audit log of inbound connections to AI agent endpoints — an attacker can observe a live session without generating any record on the victim's side
- AI agent platforms not evaluated for multi-tenant authentication defaults before production deployment

- No per-session record of what an agent accessed, executed, or returned — forensic investigation after a compromise is effectively impossible
- No internal policy covering which AI agent platforms are approved or what security controls are required before connecting one to production
- AI agent platforms not included in privileged-access security review at the pre-deployment stage

Response Actions

- Audit all AI agent platforms in your environment for multi-tenant authentication configurations — Azure SRE Agent is unlikely to be the only product with this default
- Pull session logs for Azure SRE Agent and look for inbound WebSocket connections from outside your tenant prior to April 20
- Rotate any credentials that appeared in agent task output during the exposure window
- Confirm Microsoft's server-side fix for CVE-2026-32173 is applied to your Azure SRE Agent instances
- Ask AI platform vendors directly whether their connection endpoints validate tenant ownership, not just token validity — get it in writing

Tactical Intelligence

Mitigation Strategies

- Require AI agent platforms to validate tenant ownership on every connection, not just token validity and audience claim
- Require victim-side session logging for AI agent communication channels — the same bar you set for any privileged data access path
- Scope AI agent service identities to the minimum permissions the task requires; do not run agents under subscription owner or equivalent roles
- Implement just-in-time access for AI agent sessions that need elevated permissions; auto-terminate on task completion

Preventive Measures

- **Treat AI agent platforms as privileged-access products** — security review before production deployment, not after the researcher publishes
- Write an internal policy covering approved AI agent platforms and minimum required controls before any team connects one to production infrastructure
- **Add AI agent vendor advisories to your security feed rotation** — CVE databases are behind on this product category
- **Run a tabletop where the AI agent is the attacker's pivot:** what does your SOC do when the agent itself is compromised?
- **Inventory AI agent identities, permissions, and runtime access** — AISPM tooling exists for this and most environments currently have zero visibility

Threat Hunting Hypotheses

UNAUTHORIZED EXTERNAL CONNECTION TO AZURE SRE AGENT

Hypothesis: An external party connected to our Azure SRE Agent WebSocket endpoint before the server-side fix for CVE-2026-32173 was applied and may have observed live agent sessions.

Investigation Steps

- Query Azure network and application logs for inbound WebSocket connections to any /agentHub endpoint from the period prior to April 20, 2026.
- Identify connection source IP ranges and cross-reference against your organization's known-good IP inventory.

Tactical Guidance

- Check for connections authenticated with tokens issued to accounts outside your own Entra ID tenant.
- Review the agent's task history for any sessions that returned credentials, secrets, or sensitive configuration in command output.
- If unauthorized connections are identified, scope which tasks ran during each session and identify all credentials that may have been exposed.
- Rotate any credentials identified in task output for the affected period and notify downstream systems that relied on those credentials.

AI AGENT CREDENTIAL EXPOSURE IN TASK OUTPUT

Hypothesis: An AI agent running in our environment has returned credentials or secrets in plain text as part of task execution, which may have been captured by an unauthorized observer or retained in unprotected logs.

Investigation Steps

- Pull AI agent task logs for the past 90 days and search for patterns consistent with credential material: tokens, passwords, API keys, connection strings, and private key headers.
- Identify which tasks produced output containing credential material and which systems those credentials grant access to.
- Determine whether task output logs are stored in encrypted, access-controlled storage or are accessible to broader platform roles.
- Cross-reference with CVE-2026-32173 exposure window to assess whether any credential-containing task output was observable externally.
- If credential exposure is confirmed, rotate affected credentials immediately and audit downstream system access for the period following exposure.

Sources

- **Mallory Intelligence: Critical Azure SRE Agent Token Flaw Exposed Cross-Tenant AI Sessions (April 20, 2026)**
- **BankInfoSecurity: A Token Flaw Turned Azure's AI Agent Into a Spy (April 20, 2026)**
- **CSO Online: Azure SRE Agent Flaw Lets Outsiders Silently Eavesdrop on Enterprise Cloud Operations (April 21, 2026)**
- **Wiz: Securing Agentic AI – What Cloud Teams Need to Know**

SUPPLY CHAIN ATTACKS ESCALATE

Overview & Impact

TeamPCP has been running the same playbook since March: compromise a developer tool or maintainer account, push malicious versions of something widely used, harvest the secrets, and feed them into the next campaign. The axios and Trivy operations in March set the pattern. May ran two campaigns back to back and used the first to fund the second.

The TanStack compromise did not need a stolen credential. The attacker forked TanStack/router under the alias zblgg/configuration to stay off fork-list searches, opened a pull request that fired a pull_request_target workflow, and let that workflow check out and run the attacker's fork code. That poisoned the GitHub Actions pnpm cache. When a legitimate maintainer later merged a PR to main, the release workflow pulled the poisoned cache back in. Attacker binaries inside the runner then read OIDC tokens from process memory via /proc/<pid>/mem and published malicious package versions. No npm credentials stolen.

The payload those packages carried, Mini Shai-Hulud, reads GitHub Actions runner memory for masked CI/CD secrets in plaintext, sweeps 130-plus file paths for AWS, GCP, Azure, Kubernetes, Vault, and cryptocurrency credentials, and exfiltrates through a dead-drop inside the legitimate antvis/G2 repository so the traffic looks like normal GitHub API calls. It also writes backdoors into Claude Code and VS Code configuration directories that survive IDE restarts. By May 19 the stolen tokens had been used to spin up over 2,200 public GitHub repositories named with Dune-universe terminology. The worm was advertising its own blast radius.

Famous Chollima (APT37) was running a separate operation against the same target pool. Under the PromptMink campaign the group spent seven months building a supply chain of 60-plus packages across 300-plus versions, presenting them as Web3 utilities while wiring malware through secondary dependencies. The distinguishing move came in February: the group manipulated Claude Opus through context injection into co-authoring a commit that added @validate-sdk/v2 to a cryptocurrency trading agent. First documented case of an AI model being fed context to introduce a malicious dependency into production code.

The TanStack GitHub Actions attack chain

- Attacker creates fork of TanStack/routerrepository under alias to evade fork-list detection
- Pull request opened against upstream, triggering a pull_request_target workflow that checks out and executes attacker fork code
- GitHub Actions pnpm cache poisoned with attacker-controlled binaries
- When legitimate maintainer PRs merge to main, release workflow restores the poisoned cache
- Attacker binaries extract OIDC tokens from GitHub Actions runner process memory via /proc/<pid>/mem
- Tokens used to publish malicious package versions to npm without any credential theft
- Published packages contain the Mini Shai-Hulud worm payload which self-propagates through stolen CI/CD secrets

- **@tanstack/react-router:** approximately 12 million weekly downloads; @uipath, @mistralai/mistralai, guardrails-ai, and AntV namespace packages also affected.
- Mini Shai-Hulud worm harvests credentials from 130-plus file paths including AWS, GCP, Azure, Kubernetes, Vault, and cryptocurrency wallets.
- Persistent backdoors written to Claude Code and VS Code configuration directories survive across sessions; over 2,200 public GitHub repositories created with stolen tokens.
- **GitHub internal breach:** approximately 3,800 repositories cloned via malicious Nx Console extension and listed for sale at $50,000 USD-plus.
- **PromptMink (APT37/Famous Chollima):** 60-plus packages, 300-plus versions tracked over seven months; first documented case of an AI model co-authoring a malicious dependency commit.

Observations

- pull_request_target with an unprotected cache key is all it took. The attacker forked the repo under an alias, triggered a workflow that ran their code, poisoned the pnpm cache, and waited for a legitimate maintainer merge to detonate it. No credentials stolen.
- GitHub Actions masks tokens in logs with asterisks. Those asterisks do not protect the token in runner process memory. /proc/<pid>/mem returns plaintext. Log masking is not a security control for in-process token data.

- Reinstalling the affected npm package does not clean the backdoor. Mini Shai-Hulud writes to Claude Code and VS Code configuration directories and those files survive across sessions. The package removal is not the remediation.
- The GitHub breach came directly out of the TanStack/AntV campaign. One set of stolen CI/CD secrets fed 3,800 internal repos listed for sale within a week. These are not separate incidents — they are one operation.

Guidance

Strategic Intelligence

Trend

- TeamPCP is running a compounding operation. March secrets fed May. May's TanStack tokens got GitHub's internal repos listed for sale in 48 hours. Each campaign is not contained when the package is pulled. It is contained when every downstream secret that touched the poisoned environment is rotated.
- pull_request_target with unprotected cache keys is a documented, common risk that most maintainers have not fixed. The TanStack attack needed no npm credentials. It read OIDC tokens from runner memory and published clean. Log masking did nothing. This is an immediate hardening item, not a roadmap one.
- Famous Chollima spent seven months building package trust for a technique that produced one documented case so far. As AI-assisted development becomes the default workflow, injecting context that makes malicious commits look clean is going to get easier. This is the first documented case, not the last.

Operational Intelligence

Threat Vectors

- GitHub Actions pull_request_target workflow exploitation to poison CI/CD caches without credential theft
- OIDC token extraction from GitHub Actions runner process memory bypassing log masking
- Compromised npm maintainer accounts used to publish malicious package versions at scale
- Self-propagating worm payload that harvests credentials from 130-plus file paths and creates GitHub API dead-drops for exfiltration
- Malicious VS Code extensions that harvest developer secrets and access tokens from IDE local environments
- Context injection and prompt manipulation targeting AI coding assistants to introduce malicious dependencies

Monitoring & Detection Gaps

- No enforcement of short-lived OIDC tokens in GitHub Actions workflows; long-lived tokens survive cache poisoning
- GitHub Actions cache not scoped per-branch or per-PR, allowing poisoned cache to be restored by legitimate workflows
- No detection for process memory reads targeting GitHub Actions runner processes
- Claude Code and VS Code configuration directories not monitored for unauthorized modifications or backdoor installation
- Third-party VS Code extensions installed on developer workstations without security review or integrity verification
- AI-assisted commits not reviewed for dependency additions or package changes that do not appear in the PR diff

Response Actions

- Audit all GitHub Actions workflows for use of pull_request_target and evaluate whether cache keys are scoped to prevent cross-PR poisoning
- Rotate all CI/CD secrets, cloud credentials, and developer tokens on any system that ran a TanStack, AntV, or related affected package version between May 11 and May 20
- Check Claude Code and VS Code configuration directories on developer workstations and CI runners for unauthorized files or modifications
- Remove the Nx Console VS Code extension from all developer machines and verify it against the known-good version hash; audit any tokens accessible from devices where the compromised version was installed
- Search package dependency trees for @validate-sdk/v2 and any packages flagged under the PromptMink campaign; review recent AI-assisted commits for unusual dependency additions
- **Block C2 infrastructure associated with Mini Shai-Hulud:** t.m-kosche.com and any GitHub repositories matching the Dune-universe naming pattern used for staging

Tactical Intelligence

Mitigation Strategies

- Restrict pull_request_target workflows to not execute code from untrusted forks; use pull_request instead for workflows that do not require write permissions
- Scope GitHub Actions cache keys to include a branch or PR identifier to prevent poisoned caches from being restored by unrelated workflows
- Enforce a minimum release age of 48 to 72 hours on critical npm packages so newly published versions sit in quarantine before any pipeline picks them up
- Require hardware security keys and OIDC trusted publisher flows for all maintainer accounts; rotate any long-lived npm tokens to short-lived alternatives
- Run EDR on developer workstations with visibility into IDE extension activity and process memory access patterns

Preventive Measures

- Build an SBOM for every production system and tie it to an automated scanner that flags new versions of dependencies against a known-bad list
- Monitor GitHub Actions runner logs for unusual process memory access calls and unexpected outbound connections during CI runs
- Audit VS Code extension inventory across engineering teams; restrict extension installation to approved extensions managed through a central policy
- Review AI coding assistant usage policies and establish code review requirements for any AI-assisted commit that adds or modifies dependencies
- Add supply chain worm scenarios to tabletop exercises — focus specifically on what happens when stolen CI/CD secrets reach the next environment in the chain

Threat Hunting Hypotheses

MINI SHAI-HULUD WORM INFECTION IN CI/CD ENVIRONMENT

Hypothesis: A CI/CD pipeline installed a compromised TanStack, AntV, or related npm package version and the Mini Shai-Hulud worm has harvested secrets and installed persistent backdoors.

Tactical Guidance

Investigation Steps

- Search package installation logs for affected @tanstack, @antv, timeago.js, echarts-for-react, jest-canvas-mock, and related package versions installed between May 11 and May 20, 2026.
- Check for DNS queries or outbound connections to t.m-kosche.com from CI runners or developer workstations.
- Search for unusual write activity to the antvis/G2 GitHub repository from organizational tokens.
- Inspect Claude Code and VS Code configuration directories on affected systems for unauthorized files or configuration changes.
- Review GitHub Actions logs for process memory access patterns, unusual subprocess spawning, or outbound connections during install steps.
- If infection is confirmed, isolate the affected runner or workstation, rotate all secrets the system had access to, audit GitHub repositories for any newly created repositories using exfiltrated tokens, and rebuild from clean state.

MALICIOUS VS CODE EXTENSION HARVESTING DEVELOPER CREDENTIALS

Hypothesis: A developer installed a malicious or compromised VS Code extension that has harvested access tokens, CI/CD secrets, or cloud credentials from their local environment.

Investigation Steps

- Audit installed VS Code extensions on developer workstations and compare against approved extension inventory and known-good version hashes for Nx Console and other high-privilege extensions.
- Review IDE local storage, extension output logs, and network activity for connections to unexpected domains during extension load or IDE startup.
- Check for anomalous GitHub API activity from developer tokens immediately following VS Code startup, including unexpected repository cloning, secret access, or repository creation.
- **Cross-reference developer access token usage against expected patterns:** tokens used from unusual IP ranges, at unusual hours, or against repositories the developer does not normally access.
- If a malicious extension is found, quarantine the workstation, revoke all tokens the developer had access to, audit downstream systems for unauthorized access, and perform a full credential rotation.

Sources

- BleepingComputer: GitHub Links Repo Breach to TanStack npm Supply-Chain Attack (May 21, 2026)
- The Hacker News: GitHub Internal Repositories Breached via Malicious Nx Console VS Code Extension
- Sophos: GitHub Internal Repositories Breached (May 2026)
- Microsoft Security Blog: Mini Shai-Hulud: Compromised @antv npm Packages Enable CI/CD Credential Theft (May 20, 2026)
- StepSecurity: Shai-Hulud: Here We Go Again – Mass npm Supply Chain Attack Hits the AntV Ecosystem (May 19, 2026)
- The Hacker News: Mini Shai-Hulud Pushes Malicious AntV npm Packages via Compromised Maintainer Account
- Wiz: Mini Shai-Hulud Strikes Again: TanStack and More npm Packages Compromised (May 2026)
- ReversingLabs: Claude Adds PromptMink Malicious Dependency to Crypto Agent
- Infosecurity Magazine: Malicious npm Dependency Linked to AI-Assisted Commit Targets Crypto Wallets (April 29, 2026)
- Group-IB: Six Supply Chain Attack Groups to Watch Out for in 2026 (March 13, 2026)
- Zscaler ThreatLabz: Supply Chain Attacks Surge in March 2026 (April 3, 2026)

OAUTH ATTACKS ON THE RISE

Overview & Impact

Most organizations have thousands of active OAuth grants across their SaaS environment, many of them issued months or years ago and never reviewed. Each one is a standing authorization: whoever holds the token can access the granted resources from any location, on any device, without re-authenticating, until someone explicitly revokes the grant. Password resets, MFA changes, and session termination do not touch it. The Vercel dwell time of two months is a direct consequence. Nobody revoked the grant.

The attack chain started outside Vercel. A vendor employee's device picked up Lumma Stealer, which harvested credentials that gave attackers access to a third-party AI productivity tool's environment. That tool had been authorized inside Vercel's Google Workspace. The attacker used its OAuth tokens to operate as the application — reading environment variables, pulling deployment credentials, working entirely within the access the grant defined. Initial compromise was February. Breach disclosed in April. Two months of access through one token no one looked at.

Silent redirect abuse takes a different path to the same outcome. The attacker builds a URL that opens at login.microsoftonline.com or accounts.google.com with an intentionally invalid scope in the parameters. The silent authentication flow fires, nothing authenticates, and the provider's own error-handling redirect drops the browser at attacker infrastructure. The user sees a link starting at a trusted domain. URL reputation scanners see the same thing. The delivery mechanism is Microsoft's or Google's own infrastructure and the redirect takes milliseconds.

THE VERCEL OAUTH SUPPLY CHAIN ATTACK CHAIN

- Vendor employee downloads software containing Lumma Stealer malware on a corporate device
- Infostealer harvests credentials enabling access to the third-party AI productivity tool's environment
- Attacker accesses the tool's stored OAuth tokens for the Vercel Google Workspace integration
- OAuth token grants access to Vercel's internal systems as a legitimate authorized application
- Attacker operates within the environment reading environment variables and internal credentials for approximately two months
- Access persists through password resets and MFA enforcement because the OAuth grant is never revoked
- Breach discovered and disclosed in April 2026; initial compromise traced to February 2026
- **Vercel breach:** approximately two months of dwell time via OAuth token persistence, surviving password resets and MFA enforcement.
- Environment variables and internal deployment credentials exposed across affected teams' projects.
- Silent OAuth redirect attacks use legitimate Microsoft and Google authentication domains as the delivery mechanism, bypassing URL reputation filtering.
- OAuth tokens are independent of SSO and MFA — once issued, valid until explicitly revoked regardless of authentication control changes.
- **Obsidian Security:** most enterprises cannot audit which third-party apps hold risky OAuth scopes — the same visibility gap that made the Vercel breach possible.

Observations

- The Vercel attacker did not log in. They held a token from a grant made months earlier. Password reset, MFA re-enrollment, conditional access policy update — none of that touches an OAuth grant. The token is valid until someone explicitly revokes it.
- Every organization that authorized that third-party AI tool became a second-order target the moment the vendor was compromised. The blast radius was not just Vercel — it was every enterprise that connected the same application.
- Silent redirect abuse needs no token. The attacker stuffs an invalid scope into a Microsoft or Google authentication URL and the provider's own error-handling delivers the user to attacker infrastructure. The entry URL passes reputation filters clean.
- Infostealers pull OAuth tokens from browser storage alongside passwords. Rotating a compromised password means nothing if the tokens granted before the rotation are still active.

Guidance

Strategic Intelligence

Trend

- The token is the persistence. Vercel's attacker did not need malware, a C2 channel, or a persistent backdoor. They needed one OAuth grant that nobody revoked. Breach response that stops at password resets and session termination without auditing and revoking OAuth grants is not finished.
- A seed-stage startup with one employee who authorized a corporate AI tool is now a material risk to your entire environment. That is the Vercel breach in one sentence. Every application in your OAuth grant list is a vendor relationship, most of which have never been assessed.
- RFC 9700 came out in January 2025 because the previous OAuth security guidance had not been updated since 2020 and real attacks proved it was insufficient. The patterns documented this month are exactly what that RFC was written in response to. If your OAuth grant hygiene has not been reviewed against it, it has not been reviewed.

Operational Intelligence

Threat Vectors

- Third-party OAuth application authorization in enterprise SaaS environments, particularly applications with access to cloud resources, email, and files
- Infostealer malware on vendor or employee devices harvesting OAuth tokens from browser storage and application local data
- Silent OAuth redirect flows using intentionally invalid scopes to redirect users through legitimate authentication domain infrastructure
- Consent phishing campaigns presenting fake OAuth authorization requests to Entra ID and Google Workspace users
- Long-lived OAuth tokens stored by platform-as-a-service environments that expose environment variables with internal access

Monitoring & Detection Gaps

- No inventory of which third-party applications have active OAuth grants or what permissions they hold
- OAuth grant revocation not in the incident response playbook — most teams reset passwords and terminate sessions and stop there
- No alerting on OAuth token usage from IP ranges or geographies inconsistent with the vendor's known infrastructure
- Email security tools check the visible domain; silent redirect abuse starts at a trusted domain and never triggers reputation filters
- No recurring review of OAuth grants — applications authorized years ago and no longer in use still hold active access

Response Actions

- Audit all OAuth grants in Google Workspace, Microsoft 365, and connected SaaS platforms now
- Revoke grants for any third-party app you cannot confirm is active, necessary, and secure — if you cannot answer all three, revoke it
- Rotate environment variables and deployment credentials for any system where a compromised third-party OAuth app held internal access
- Pull OAuth token usage logs for the past 90 days and flag access from IP ranges inconsistent with the vendor's known infrastructure
- Stop allowing users to self-authorize third-party OAuth apps without a review step; gate new authorizations behind admin approval

Tactical Intelligence

Mitigation Strategies

- Deploy OAuth application governance tooling — you need continuous visibility into what is authorized and immediate alerts when new grants appear
- Enforce minimum-scope OAuth grants; re-authorize third-party apps on a defined schedule and revoke anything you cannot justify keeping
- Enable Continuous Access Evaluation for Microsoft 365 and Google Workspace to shorten the valid window on stolen tokens
- Gate high-risk OAuth scopes (mail read, file access, user impersonation) behind admin approval and log every consent grant event
- Preventive Measures
- Audit all active OAuth grants now and revoke anything dormant, unknown, or no longer in use — this is probably the highest-return 30 minutes your team can spend on SaaS security today
- Strip or defang prompt=none and empty-scope parameters in authentication URLs at the email gateway to block silent redirect delivery
- Any third-party app requesting OAuth access to corporate SaaS gets the same vendor security review as a direct data vendor — the Vercel breach started with a startup no one had assessed
- Add OAuth grant revocation to the incident response runbook alongside credential rotation — it is currently missing from most playbooks
- **Train employees on consent phishing with one clear rule:** any authorization dialog from a trusted provider can grant permanent access to an untrusted third party — require approval before clicking Allow

Threat Hunting Hypotheses

PERSISTENT OAUTH ACCESS FOLLOWING THIRD-PARTY APPLICATION COMPROMISE

Hypothesis: A third-party application authorized in our SaaS environment has been compromised and an attacker is using its OAuth tokens to access our internal resources.

Investigation Steps

- Pull OAuth application grant logs from Google Workspace and Microsoft 365 admin portals for all applications authorized with access to mail, files, cloud resources, or user data.
- Identify applications that have been used in the past 30 days from IP ranges, ASNs, or geographies inconsistent with the vendor's known infrastructure or the authorizing user's normal patterns.
- Check for bulk data access, mass email reads, or large file downloads associated with OAuth application tokens in the audit logs.
- Cross-reference third-party application providers against known breach disclosures or threat intelligence feeds for vendor-side compromises.
- For any application showing anomalous access, revoke the grant immediately, rotate all resources the application had access to, and review the full access history for the period of suspected compromise.
- Notify the vendor of the suspicious access pattern and request their security incident log for the relevant period.

SILENT OAUTH REDIRECT PHISHING DELIVERY IN EMAIL

Hypothesis: Users have received phishing emails containing silent OAuth redirect URLs that appear to originate from legitimate Microsoft or Google authentication infrastructure.

Investigation Steps

- Search email security logs for inbound messages containing URLs with the pattern login.microsoftonline.com or accounts.google.com in the visible domain with parameters including prompt=none, empty scope values, or unusual state parameters.
- Review click telemetry from web proxy or SWG for any users who followed such links in the past 60 days.
- Check for immediate redirects from those authentication URLs to domains not owned by Microsoft or Google, particularly newly registered domains.
- Correlate with any subsequent anomalous authentication activity, OAuth consent grants, or download events for users who clicked.
- If delivery is confirmed, block the source sending infrastructure, notify affected users, audit for any grants or credentials exposed, and submit the URLs to the email security vendor for signature update.

Sources

- **Proofpoint: When Trusted Apps Become Trusted Backdoors – What the Vercel Incident Shows About OAuth Risk (May 11, 2026)**
- **Malwarebytes: Attackers Abuse OAuth's Built-in Redirects to Launch Phishing and Malware Attacks (March 4, 2026)**
- **The Hacker News: The Back Door Attackers Know About — and Most Security Teams Still Haven't Closed**
- **Trend Micro: The Vercel Breach – OAuth Supply Chain Attack Exposes the Hidden Risk in Platform Environment Variables (April 20, 2026)**
- **Obsidian Security: OAuth Vulnerabilities Every Security Team Should Know (February 4, 2026)**
- **Grip Security: The Rise of OAuth Attacks to Access Sensitive Systems (March 23, 2025)**

pellera

Contact the Pellera Threat Intel Group at getsecure@pellera.com

pellera.com