

TeamPCP Wave Four: GitHub Breach via Poisoned VS Code Extension, durabletask PyPI Worm, and ~4,000 Internal Repositories Exfiltrated

By Marcus Webb

Published: 2026-05-20 · Archived: 2026-08-06 02:01:48 UTC

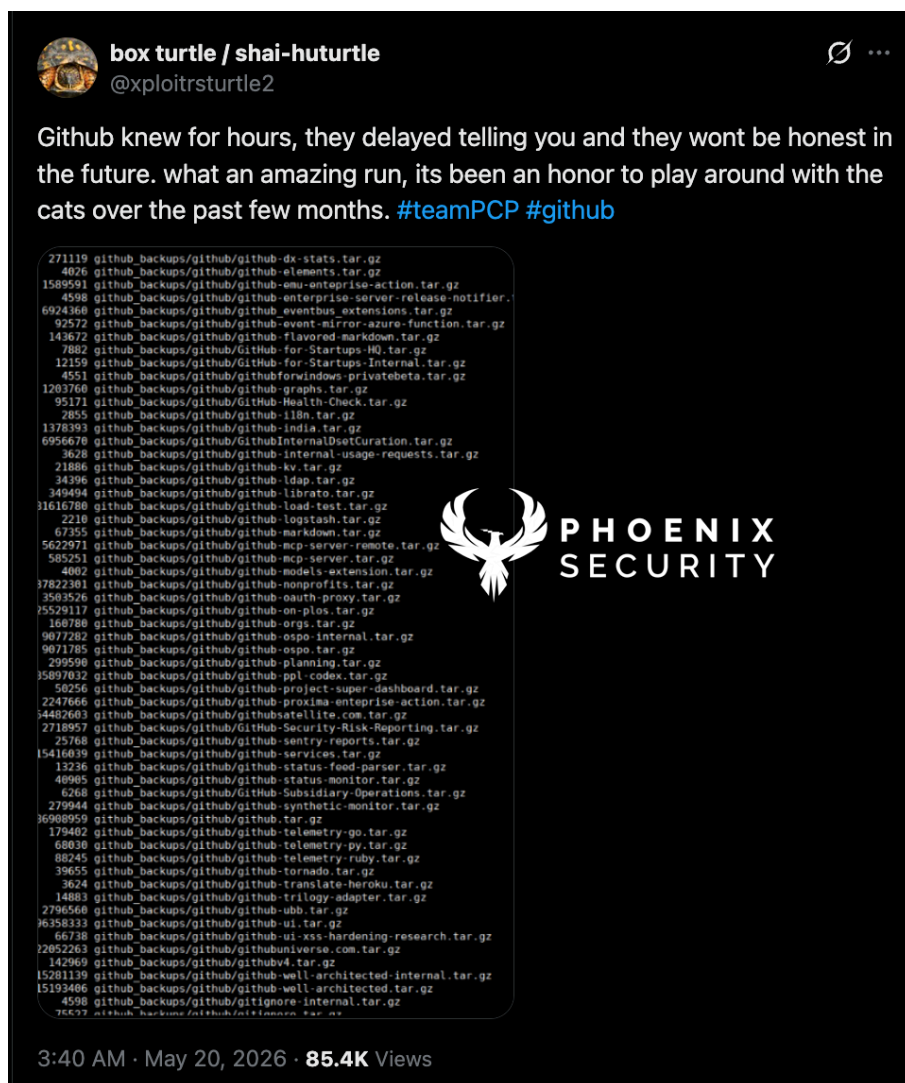
Contents

Executive Summary

On May 19–20, 2026, two events landed within hours of each other. [Wiz](#) and [Endor Labs](#) documented three malicious versions of durabletask (1.4.1, 1.4.2, and 1.4.3) pushed to PyPI within a 35-minute window on May 19, the official Microsoft Azure Python SDK for the Durable Task workflow framework. The same day, GitHub confirmed that a poisoned VS Code extension on an employee device had led to the exfiltration of approximately 3,800 internal repositories. TeamPCP listed the source code for sale at \$50,000: ‘it looks like our retirement is soon.’

“Github knew for hours, they delayed telling you and they wont be honest in the future. what an amazing run, its been an honor to play around with the cats over the past few months.”

— @xploitrsturtle2 (box turtle / shai-huturtle, linked to TeamPCP), 3:40 AM · May 20, 2026 · 85.4K views

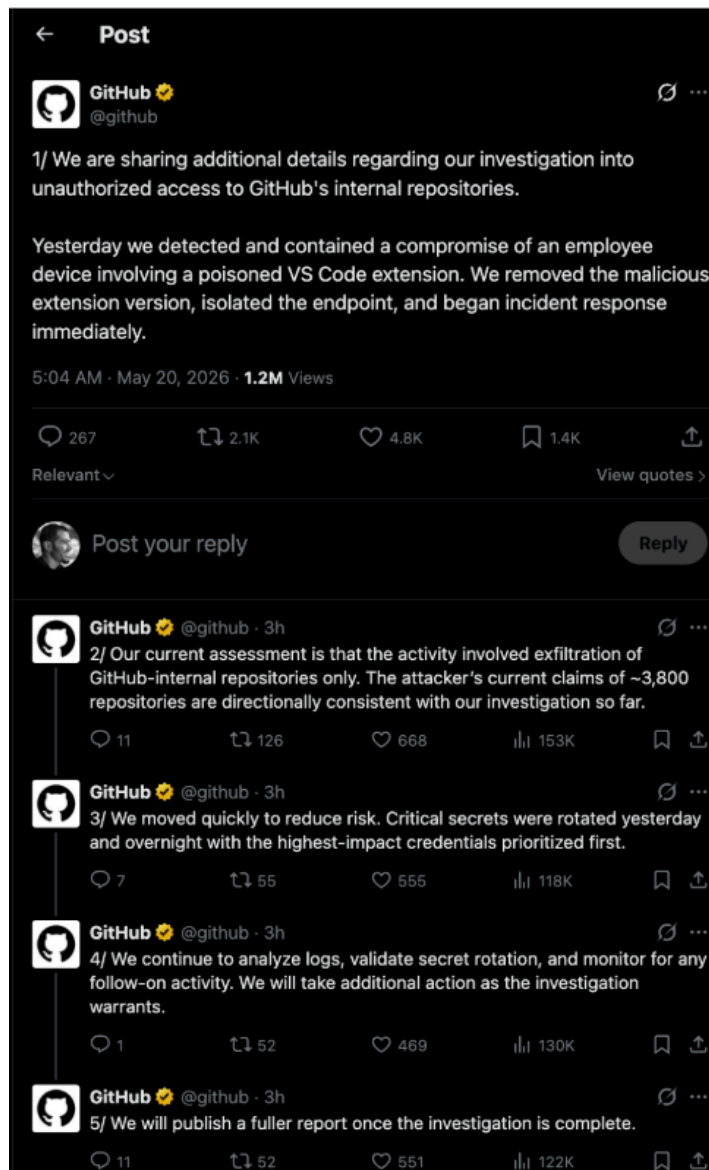


Both compromises trace to the same Mini Shai-Hulud worm infrastructure. durabletask drops rope.pyz from check.git-service[.]com, uses the FIRESALE GitHub dead-drop for C2 rotation, and creates exfiltration repositories with Russian

folklore names. All three behaviors were documented in prior TeamPCP waves going back to March 2026. Hunt.io's infrastructure analysis puts C2 IP addresses from the 83.142.209.0/24 subnet across the March Checkmarx and Telnix compromises and this wave. Same subnet, same operators.



The GitHub breach follows directly from the worm's own mechanics. It steals GitHub tokens; those tokens get used to create exfiltration repos. A poisoned extension on a GitHub employee device is the same playbook: compromised developer tooling as the path to upstream infrastructure. Nine weeks in, TeamPCP has GitHub's internal source code. Zero CVEs exist for any package across this entire campaign. Your scanner has no record of any of it.



TL;DR for Engineering Teams

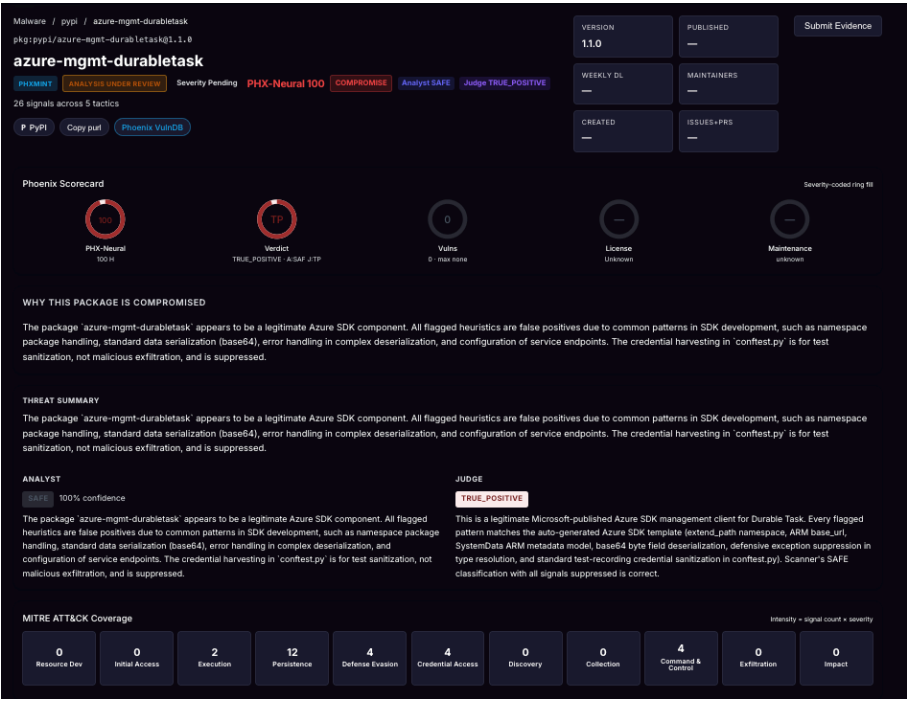
What it is: Self-propagating supply chain worm (Mini Shai-Hulud) compromising PyPI/npm packages and developer tooling. Wave Four: durabletask 1.4.1–1.4.3 backdoored; GitHub employee device compromised via poisoned VS Code extension, ~3,800 internal repos exfiltrated. Zero CVEs produced.

Where it bites: Linux CI/CD runners, cloud VMs (AWS/Azure/GCP/Kubernetes), developer machines running affected packages, GitHub Actions workflows with publish tokens.

Why it matters: 417,000 monthly downloads of durabletask. Active exploitation confirmed. Self-replicating via AWS SSM and kubectl exec. FIRESALE C2 rotation via signed GitHub commits makes domain blocking insufficient.

Patch status: durabletask 1.4.1, 1.4.2, 1.4.3 yanked from PyPI. Pin to 1.4.0. No patch required — these versions must be avoided entirely.

Immediate action: Run: `pip show durabletask` — pin to 1.4.0. Check `/tmp/managed.pyz` on every Linux host. Rotate all cloud credentials, GitHub tokens, and CI/CD secrets on any host that imported an affected version. Block `check.git-service[.]com` and `t.m-kosche[.]com` at egress. Audit VS Code extensions on developer and CI machines.



Check malware at <https://phxintel.security/malware.html>

Campaign Overview

Field	Value
Threat Actor	TeamPCP (attributed with high confidence by Wiz, Aikido, StepSecurity, Hunt.io)
Campaign Name	Mini Shai-Hulud (Wave Four)
Attack Type	Supply chain compromise — malicious package injection + developer tooling poisoning
Primary CVEs	None. Zero CVEs produced across all stages.
CWE	CWE-494 (Download of Code Without Integrity Check), CWE-506 (Embedded Malicious Code)
CVSS Score	10.0 (Critical) — supply chain RCE with lateral movement and data exfiltration
Affected Packages	durabletask 1.4.1, 1.4.2, 1.4.3 (PyPI); poisoned VS Code extension (unconfirmed name); TanStack npm (prior wave)
Active Exploitation	Confirmed — GitHub breach confirmed May 20, 2026
C2 Infrastructure	check.git-service[.]com, t.m-kosche[.]com, 83.142.209.0/24 subnet
Campaign Start	March 2026 (Trivy, Checkmarx KICS, Telnix, LiteLLM)
Wave Four Date	May 19–20, 2026

Campaign Timeline and Wave Structure

TeamPCP has run the same worm infrastructure since at least March 2026, hitting new ecosystems with each wave.

Date	Incident	Ecosystem	Key Impact
March 2026	Trivy, Checkmarx KICS, LiteLLM, Telnix compromised	PyPI, npm, Docker Hub,	CI/CD credential theft; zero CVEs

Date	Incident	Ecosystem	Key Impact
		GitHub Actions, VS Code	
March 31, 2026	Malicious Axios library via GitHub Actions workflow	npm	OpenAI macOS signing certificate exposure (attributed to UNC1069 separately)
April 2026 (early)	SAP-ecosystem npm packages (mbt, @cap-js/db-service, @cap-js/sqlite, @cap-js/postgres)	npm	SAP developer environment compromise; Mini Shai-Hulud worm deployed
May 11, 2026	TanStack compromised — 84 malicious versions across 42 @tanstack/* packages	npm	Two OpenAI employee devices compromised; code-signing certs exfiltrated
May 14–15, 2026	Mistral AI, UiPath, Guardrails AI, OpenSearch packages compromised	npm, PyPI	Hundreds of packages across high-profile AI namespaces
May 19, 2026	durabletask 1.4.1–1.4.3 published with dropper	PyPI	417,000 monthly downloads targeted; Azure Durable Functions workloads
May 20, 2026	GitHub internal breach — ~3,800 repos exfiltrated via poisoned VS Code extension	GitHub platform	GitHub source code listed for sale at \$50,000. Operator @xploitsturtle2 posts: ‘what an amazing run, its been an honor’ — suggesting campaign wind-down

Protect yourself with the latest threat intelligence, get access to PHOENIX BLUE Today

GitHub Incident Response Timeline

The following is a reconstructed timeline of public statements from GitHub and the threat actor on May 20, 2026, ordered chronologically. Times are approximate where only relative timestamps (‘8h ago’, ‘3h ago’) were available at time of capture.

Time (May 20, 2026)	Actor	Event
~12:43 AM UTC (est.)	TeamPCP [Co-Owner]	Posts GitHub source code for sale on Breached forum. Claims ~4,000 private repos. Asking price \$50,000. ‘As always this is not a ransom... if no buyer is found, we will leak it free.’
~3:40 AM UTC	@xploitsturtle2 (TeamPCP-linked)	‘Github knew for hours, they delayed telling you and they wont be honest in the future. what an amazing run, its been an honor to play around with the cats over the past few months. #teamPCP #github’
~12:43 AM – ~5:04 AM UTC	@github	Initial statement: ‘We are investigating unauthorized access to GitHub’s internal repositories.’ Confirms no evidence of customer data impact outside internal repos.
5:04 AM UTC	@github (thread 1/5)	‘Yesterday we detected and contained a compromise of an employee device involving a poisoned VS Code extension. We removed the malicious extension version, isolated the endpoint, and began incident response immediately.’ (1.2M views)
5:04 AM UTC +3h	@github (thread 2/5)	‘Our current assessment is that the activity involved exfiltration of GitHub-internal repositories only. The attacker’s current

Time (May 20, 2026)	Actor	Event
		claims of ~3,800 repositories are directionally consistent with our investigation so far.'
5:04 AM UTC +3h	@github (thread 3/5)	'We moved quickly to reduce risk. Critical secrets were rotated yesterday and overnight with the highest-impact credentials prioritized first.'
5:04 AM UTC +3h	@github (thread 4/5)	'We continue to analyze logs, validate secret rotation, and monitor for any follow-on activity. We will take additional action as the investigation warrants.'
5:04 AM UTC +3h	@github (thread 5/5)	'We will publish a fuller report once the investigation is complete.'

GitHub Official Statements — Primary Source

GitHub's initial statement (posted approximately 8 hours before the 8:43 AM screenshot capture):

"We are investigating unauthorized access to GitHub's internal repositories. While we currently have no evidence of impact to customer information stored outside of GitHub's internal repositories (such as our customers' enterprises, organizations, and repositories), we are closely monitoring our infrastructure for follow-on activity."

— @github, initial statement · 17K likes · 6.2K retweets · 7M views

"If any impact is discovered, we will notify customers via established incident response and notification channels."

— @github, follow-up · 1.6K likes · 436K views

GitHub's detailed thread, published 5:04 AM May 20, 2026 (1.2M views):

1/ "Yesterday we detected and contained a compromise of an employee device involving a poisoned VS Code extension. We removed the malicious extension version, isolated the endpoint, and began incident response immediately."

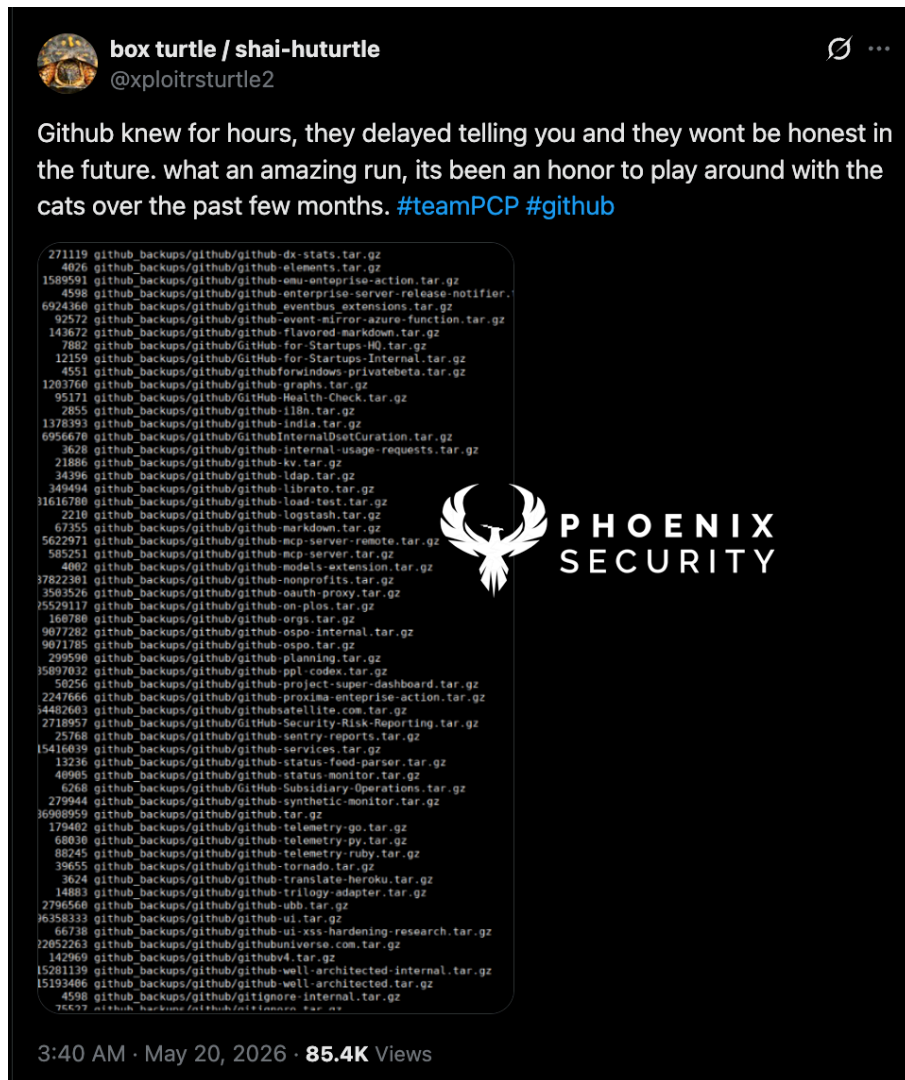
2/ "Our current assessment is that the activity involved exfiltration of GitHub-internal repositories only. The attacker's current claims of ~3,800 repositories are directionally consistent with our investigation so far."

3/ "We moved quickly to reduce risk. Critical secrets were rotated yesterday and overnight with the highest-impact credentials prioritized first."

4/ "We continue to analyze logs, validate secret rotation, and monitor for any follow-on activity. We will take additional action as the investigation warrants."

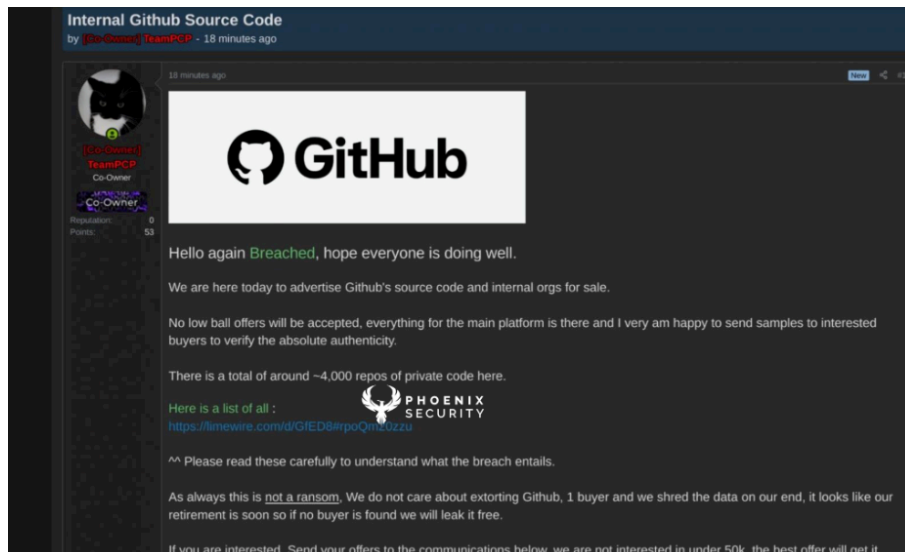
5/ "We will publish a fuller report once the investigation is complete."

— @github thread, 5:04 AM · May 20, 2026 · 1.2M views



Threat Actor Statement — TeamPCP Operator

The operator post on X from @xploitsturtle2 (account associated with 'box turtle / shai-huturtle') was published at 3:40 AM May 20, 2026, approximately 80 minutes before GitHub's detailed thread. The phrasing 'GitHub knew for hours' and the present tense suggest the actor had visibility into GitHub's internal response before public disclosure. The statement 'its been an honor to play around with the cats over the past few months' is consistent with the retirement framing in the Breached forum post. Attribution of this account to TeamPCP is based on consistent campaign branding (#teamPCP, shai-huturtle naming referencing Mini Shai-Hulud) and operational timing; this assessment carries medium-to-high confidence.



Scope of Exposed Repositories

The file listing shared by TeamPCP as proof of breach contains repository archives across GitHub's internal infrastructure. Selected repositories visible in the leaked listing include:

Repository Name	Inferred Function / Sensitivity
github-mcp-server	GitHub's Model Context Protocol server — high sensitivity given MCP is developer tooling infrastructure
github-mcp-server-remote	Remote MCP server component
github-models-extension	GitHub Models product extension code
github-Security-Risk-Reporting	Internal security risk reporting tooling
github-ui-xss-hardening-research	Internal XSS hardening research — knowledge of unfixed attack surface
github-oauth-proxy	OAuth proxy infrastructure — credential flow code
github-InternalDSFCuration	Internal developer success tooling
github-internal-usage-requests	Internal usage request tooling
github-ospo-internal	Open Source Programs Office internal tooling
github-sentry-reports	Error monitoring and telemetry reports
github-synthetic-monitor	Synthetic monitoring infrastructure
github-telemetry-go / -py / -ruby	Multi-language telemetry pipeline code
github-load-balancers / github-load-test	Load balancing infrastructure and test tooling
github-satellite.com	GitHub Satellite conference infrastructure
github-enterprise-server-release-notifier	Enterprise Server release automation
github-well-architected / -internal	Internal architecture standards and tooling
github-gitignore-internal / github-gitignore-Eternal	Internal gitignore tooling
github-for-Startups-HQ / -Internal	GitHub for Startups program infrastructure

Repository Name	Inferred Function / Sensitivity
github-githubfundingprivatebeta	GitHub Sponsors / funding private beta code
github-Subsidiary-Operations	Subsidiary operations tooling
github-ppl-codex	People and codex internal tooling
github-project-super-dashboard	Internal project management dashboard
github-sponsors	GitHub Sponsors platform code
githubuniverse.com	GitHub Universe conference site code
github-v4	Likely GitHub GraphQL API v4 related code

The repository list includes *github-mcp-server* and *github-mcp-server-remote*. Given MCP is now a core integration layer between AI coding agents and developer infrastructure, exposure of this code is particularly relevant for teams building on GitHub's MCP surface.

The full claimed list runs to approximately 4,000 entries. GitHub's own statement ('~3,800 repositories are directionally consistent with our investigation') confirms the order of magnitude. The repositories above represent a selection of those with the highest inferred sensitivity based on naming conventions visible in the leaked file listing. This is not an exhaustive list, and the actual contents of each archive are unknown.

Technical Analysis

Root Cause: Credential Theft Enabling Package Publishing

The durabletask compromise follows the same access pattern documented by Wiz: credentials stolen in a prior attack compromised a GitHub account, repository secrets were dumped, and a recovered PyPI publishing token was used to push malicious versions directly. Three versions in 35 minutes: 1.4.1 at 16:19 UTC, 1.4.2 at 16:49, 1.4.3 at 16:54. Each release added more injection points to cover more import paths.

The GitHub breach came through a different vector. GitHub confirmed a poisoned VS Code extension on an employee device. The Nx Console project had a concurrent compromise pushing a multi-stage credential stealer; GitHub hasn't publicly confirmed the extension name, but timing and mechanism align. From there the malware harvested credentials with access to GitHub's internal repositories.

Attack Chain: durabletask Wave

The durabletask attack runs in three stages:

- **Stage 1 — Import-time dropper:** A 13-line block injected at the top level of `__init__.py` (and progressively more module files across versions 1.4.1–1.4.3) runs on any import durabletask call. The dropper is Linux-only, detaches via `start_new_session=True`, swallows errors via `except: pass`, and downloads `rope.pyz` from `check.git-service[.].com` to `/tmp/managed.pyz` before launching it as a background process. No function call needed; cold start in Azure Functions triggers it before any user code runs.
- **Stage 2 — Second-stage payload (rope.pyz):** A Python zipapp with 19 modules. Pre-flight checks exit on non-Linux, Russian locale (`$LANG` starting with `ru`), or CPU count ≤ 2 (sandbox evasion). The FIRESCALE dead-drop searches GitHub commit API for 'FIRESCALE' patterns containing base64-encoded C2 URLs with RSA-SHA256 signatures, providing the attacker a censorship-resistant C2 rotation mechanism requiring only a public Git commit.
- **Stage 3 — Full credential harvest and worm propagation:** Eight concurrent credential modules via `ThreadPoolExecutor` target: AWS IAM/SSM/Secrets Manager (all 19 regions), Azure service principals/Key Vaults, GCP service account keys/Secret Manager, Kubernetes secrets across all contexts, HashiCorp Vault KV v1+v2, 1Password, Bitwarden, pass/gopass. Additionally targets AI developer tooling configs: `claude_desktop_config.json`, Cursor MCP config, VS Code MCP config, Codeium, Zed, Continue. Propagates to up to five AWS EC2 instances via SSM `SendCommand` and up to five Kubernetes pods via `kubectl exec`.

Version Expansion Pattern

Version	Infected Files	Import Paths Covered
1.4.1	durabletask/__init__.py	import durabletask
1.4.2	__init__.py + task.py	+ from durabletask.task import ...
1.4.3	5 files: __init__.py, task.py, entities/__init__.py, extensions/__init__.py, payload/__init__.py	All major import paths covered

When the primary C2 (`check.git-service[.]com`) is unreachable, `rope.pyz` queries the GitHub commit search API:

Each result is inspected for the pattern:

The base64 URL is only trusted if the RSA-SHA256 signature verifies against a hardcoded 4096-bit public key baked into the binary. The upshot: the attacker rotates C2 by pushing a single signed commit anywhere on GitHub. Taking down `check.git-service[.].com` doesn't stop the worm. Only rotating the FIRESALE signing key does.

Hunt.io tied the GitHub breach to the same TeamPCP infrastructure through overlapping C2 IP ranges. Three addresses in the 83.142.209.0/24 subnet: 83.142.209.194 on the TanStack wave, 83.142.209.11 and .203 on the March 2026 Checkmarx and Telnix attacks. Both .11 and .203 had SSH active from November 15–21, 2025, roughly four months before TanStack went public. Pre-staged infrastructure, not coincidence. The same subnet shows up across every TeamPCP wave Hunt.io tracked through May 2026.

Exfiltration with GitHub Token Abuse

When both C2 channels fail, rope.pyz uses stolen GitHub tokens to create public repositories named after Russian folklore (BABA-YAGA, KOSCHEI, FIREBIRD, PTITSA, RUSALKA, MOROZKO, LESHY, DOMOVOI, VODYANOY) and uploads the encrypted credential bundle as results.json. The victim's own GitHub account becomes the exfil channel. From the outside it looks like ordinary repository activity.

Destructive Payload

On hosts fingerprinted as Israeli or Iranian (timezone, /etc/timezone, /etc/localtime, \$LANG, \$LC_ALL, \$LC_MESSAGES checks for he_IL or fa_IR), roulette.py rolls a 1-in-6 die. On a hit: downloads audio.mp3 from check.git-service[.]com, sets system volume to 100% via pactl, plays it through mpv, then runs `rm -rf /*`. The payload is operator-activated. The attacker triggers it by returning HTTP 200 from /v1/models during the check-in call.

The screenshot shows the Phoenix Security dashboard for the package 'azure-mgmt-durabtask'. The dashboard includes a header with the package name and version (1.1.0), a 'Submit Evidence' button, and a 'Severities Pending' section. Below this, there are tabs for 'PHISHMIT', 'ANALYSIS UNDER REVIEW', and 'Severities Pending'. The 'Severities Pending' tab is active, showing a 'PHX-Neural 100' score, a 'COMPROMISE' status, and a 'Judge TRUE_POSITIVE' result. The dashboard also features a 'Phoenix Scorecard' with five metrics: PHX-Neural (100-H), Verdict (TRUE_POSITIVE: A-SAF-2-TP), Vulns (0 - max none), License (Unknown), and Maintenance (unknown). A 'WHY THIS PACKAGE IS COMPROMISED' section explains that the package appears to be a legitimate Azure SDK component, but all flagged heuristics are false positives due to common patterns in SDK development. A 'THREAT SUMMARY' section provides a detailed analysis of the package, noting that it is a legitimate Microsoft-published Azure SDK management client for Durable Task. The 'ANALYST' section shows a 'SAFE' rating with 100% confidence, and the 'JUDGE' section shows a 'TRUE_POSITIVE' result. The 'MITRE ATT&CK Coverage' section displays a grid of 11 categories, each with a score: Resource Dev (0), Initial Access (0), Execution (2), Persistence (12), Defense Evasion (4), Credential Access (4), Discovery (0), Collection (0), Command & Control (4), Exfiltration (0), and Impact (0).

Check malware at <https://phxintel.security/malware.html>

Affected Packages and Versions

Package	Ecosystem	Malicious Versions	Clean Version	SHA-256 (tar.gz)
durabtask	PyPI	1.4.1, 1.4.2, 1.4.3	1.4.0	1.4.1: 3de04fe2a76262743ed089efa7115f4508619838e77d60b9a1aab8b20d2cc8bf
durabtask	PyPI	1.4.2	1.4.0	85f54c089d78ebfb101454ec934c767065a342a43c9ee1beac8430cdd3b2086f
durabtask	PyPI	1.4.3	1.4.0	c0b094e46842260936d4b97ce63e4539b99a3eae48b736798c700217c52569c
rope.pyz (payload)	C2 delivery	Any	N/A	069ac1dc7f7649b76bc72a11ac700f373804bfd81dab7e561157b703999f44ce
@tanstack/* (42 packages)	npm	84 malicious versions	Latest clean	See TanStack post-mortem
VS Code extension (unconfirmed)	VS Code marketplace	Poisoned version	Verified clean	Not published

Exposure Analysis

Environment	Risk Level	Exposure Reason
Azure Functions / Durable Task workloads	Critical	durabletask imported at cold start before user code runs — payload fires automatically on any Linux host
CI/CD pipelines (GitHub Actions, Jenkins, GitLab)	Critical	Cloud credentials, publish tokens, and GitHub PATs in environment — primary theft target; worm propagation confirmed
AWS EC2 fleets with SSM agent	Critical	SSM SendCommand propagation to up to 5 instances per infected host — exponential spread potential
Kubernetes clusters	Critical	kubect exec propagation into up to 5 pods per infected workload
Developer machines (Linux)	High	Credential files, SSH keys, Docker credentials, .env files, AI tooling configs
GitHub Actions workflows publishing to PyPI/npm	High	Publishing token theft enables further package compromise — the worm’s self-replication mechanism
VS Code users (all platforms)	Medium–High	Poisoned extension vector used against GitHub employee; any developer with auto-update enabled is at risk

durabletask pulls roughly 417,000 downloads a month, according to Endor Labs. Any Linux system that upgraded past 1.4.0 before the packages were yanked should be treated as fully compromised. Azure Durable Functions runs in exactly the kinds of environments where production cloud credentials exist: workflow automation, document processing, AI agent infrastructure.

Phoenix PHX-Neural scored durabletask at 61/100 (COMPROMISE, Analysis Under Review), with 12 of 85 detection rules firing: Payload Download, Unencrypted Communication, and DGA/Dynamic C2 Resolution. The related azure-mgmt-durabletask package scored 100/100; both Analyst and Judge confirmed it as a legitimate Microsoft Azure SDK component with all signals suppressed as false positives.

Detection Guidance

Immediate Verification Commands

```
# Check installed version

pip show durabletask

# Scan lockfiles

grep -r 'durabletask' requirements*.txt poetry.lock Pipfile.lock uv.lock 2>/dev/null

# Check for dropped payload on disk

ls -la /tmp/managed.pyz 2>/dev/null

# Check for worm propagation markers

ls ~/.cache/.sys-update-check ~/.cache/.sys-update-check-k8s 2>/dev/null

# Check for persistence service

systemctl status pgsql-monitor.service 2>/dev/null

ls /usr/bin/pgmonitor.py ~/.local/bin/pgmonitor.py 2>/dev/null

# Scan installed package for injected code

grep -r 'git-service.com' $(pip show durabletask | grep Location | awk '{print $2}') 2>/dev/null
```

Log and Network Indicators

- **DNS / proxy logs:** Any query to check.git-service[.]com or t.m-kosche[.]com is a confirmed compromise indicator. Block *.git-service[.]com and *.m-kosche[.]com at the domain level.
- **Process creation:** python3 /tmp/managed.pyz launched as a detached process (start_new_session=True) with stdout/stderr redirected to /dev/null.
- **Network egress:** Outbound HTTPS POST to check.git-service[.]com/api/public/version; outbound connections to IPs in 83.142.209.0/24.
- **GitHub API calls:** Commit search queries for 'FIRESCALE' from non-GitHub infrastructure. Public repository creation with Russian folklore names (BABA-YAGA, KOSCHEI, FIREBIRD, etc.) followed by results.json file creation.
- **AWS CloudTrail:** ssm:SendCommand events using AWS-RunShellScript document from compromised instance. Unexpected Secrets Manager or SSM Parameter Store GetSecretValue/GetParameter calls across multiple regions.
- **Kubernetes audit logs:** kubectl exec commands originating from unexpected source pods.
- **Systemd:** New service pgsqll-monitor.service registered, described as 'PostgreSQL Monitor' — cover identity for persistence payload.

IOC Reference Table

Indicator Type	Indicator	Notes
PyPI package	durable-task==1.4.1	Malicious — do not install
PyPI package	durable-task==1.4.2	Malicious — do not install
PyPI package	durable-task==1.4.3	Malicious — do not install
File hash (SHA-256)	3de04fe2a76262743ed089efa7115f4508619838e77d60b9a1aab8b20d2cc8bf	durable-task-1.4.1.tar.gz
File hash (SHA-256)	85f54c089d78ebfb101454ec934c767065a342a43c9ee1beac8430cdd3b2086f	durable-task-1.4.2.tar.gz
File hash (SHA-256)	c0b094e46842260936d4b97ce63e4539b99a3eae48b736798c700217c52569dc	durable-task-1.4.3.tar.gz
File hash (SHA-256)	069ac1dc7f7649b76bc72a11ac700f373804bfd81dab7e561157b703999f44ce	rope.pyz second-stage payload
Domain	check.git-service[.]com	Primary C2 and payload delivery — block domain
Domain	t.m-kosche[.]com	Secondary payload delivery — block domain
Domain wildcard	*.git-service[.]com	Block full domain — subdomains used for exfil
Domain wildcard	*.m-kosche[.]com	Block full domain — subdomains may be used
URL	https://check.git-service[.]com/rope.pyz	Payload download
URL	https://check.git-service[.]com/v1/models	C2 check-in / wiper activation endpoint

Indicator Type	Indicator	Notes
URL	https://check.git-service[.]com/api/public/version	Primary exfiltration endpoint
URL	https://check.git-service[.]com/audio.mp3	Wiper audio file
URL	https://t.m-kosche[.]com/rope.pyz	Fallback payload delivery
IP range	83.142.209.0/24	TeamPCP C2 subnet — confirmed across all waves
IP address	83.142.209.194	TanStack wave C2
IP address	83.142.209.11	March 2026 Checkmarx/Telnyx C2
IP address	83.142.209.203	March 2026 Checkmarx/Telnyx C2
File path	/tmp/managed.pyz	Dropped payload — confirmed compromise indicator
File path	~/.cache/.sys-update-check	Worm propagation marker (AWS SSM)
File path	~/.cache/.sys-update-check-k8s	Worm propagation marker (Kubernetes)
File path	/usr/bin/pgmonitor.py	Persistence payload (root install)
File path	~/local/bin/pgmonitor.py	Persistence payload (non-root install)
Service name	pgsql-monitor.service	Persistence service — disguised as PostgreSQL Monitor
GitHub pattern	FIRESCALE <base64>.<base64>	Dead-drop beacon in public commit messages
GitHub repo names	BABA-YAGA, KOSCHEI, FIREBIRD, PTITSA, RUSALKA, MOROZKO, LESHY	Russian folklore names — exfiltration repos created with stolen tokens

Indicator Type	Indicator	Notes
Domain registration	git-service[.]com registered 2026-05-16	Three days before attack; NameSilo, privacy-guarded

Scanner References

- Phoenix Security scanner: github.com/Security-Phoenix-demo/Shai-Hulud-Sha1-Hulud-V2-npm-compromise-scanner
- Phoenix PHX-Neural scored durabletask 0.0.0.dev73 at 61/100 — 12 of 85 detection rules triggered. Live tracking: sha1hulud-phoenix.pplx.app
- Endor Labs: Detected durabletask 1.4.1 within 2 minutes of publication (16:19 to 16:21 UTC)
- Aikido Security: Behavioral analysis of rope.pyz payload documented
- StepSecurity: SSM propagation mechanics documented
- Wiz: Root cause analysis of PyPI token compromise vector
- SafeDep: 28KB Python stealer analysis including HashiCorp Vault KV and password manager targeting
- Hunt.io: C2 infrastructure correlation across all TeamPCP waves
- SBOM scanning: Any SBOM containing durabletask without explicit version == 1.4.0 pin should be flagged

Immediate Actions

- 1. Pin durabletask to 1.4.0:** pip install durabletask==1.4.0. If using poetry: poetry add durabletask@1.4.0. If using uv: uv add durabletask==1.4.0. Do not install 1.4.1, 1.4.2, or 1.4.3.
- 2. Check for active compromise:** Run `ls -la /tmp/managed.pyz`. If the file exists, the payload ran. Isolate the host immediately before credential rotation.
- 3. Check propagation markers:** `ls ~/.cache/.sys-update-check && ls ~/.cache/.sys-update-check-k8s`. Presence confirms the worm's lateral movement logic ran.
- 4. Rotate all credentials on affected hosts:** AWS IAM keys, session tokens (check CloudTrail for unexpected API calls). Azure service principal credentials, managed identity tokens. GCP service account keys (check audit logs). Kubernetes service account tokens and kubeconfig. HashiCorp Vault tokens. 1Password and Bitwarden sessions. GitHub tokens/PATs (check for new public repos with Russian folklore names). npm, pip, PyPI tokens. SSH private keys. CI/CD pipeline secrets. Contents of any .env files accessible to the process.
- 5. Block C2 infrastructure:** Add `check.git-service[.]com`, `*.git-service[.]com`, `t.m-kosche[.]com`, `*.m-kosche[.]com` to DNS blocklists and egress proxy deny lists. Block 83.142.209.0/24 at network egress.
- 6. Check for persistence:** `systemctl status pgsql-monitor.service`. If found: `systemctl stop pgsql-monitor.service && systemctl disable pgsql-monitor.service && rm /etc/systemd/system/pgsql-monitor.service && rm /usr/bin/pgmonitor.py` (or `~/local/bin/pgmonitor.py`).
- 7. Audit VS Code extensions:** Review all installed extensions across developer and CI machines. Any extension with network access that was auto-updated recently warrants manual review of the published version's source.
- 8. Check AWS SSM propagation:** Review CloudTrail for `ssm:SendCommand` events using `AWS-RunShellScript` from any compromised instance. Investigate all target instances as potentially infected.

Temporary Mitigations (If Immediate Patching Is Delayed)

- Restrict outbound egress from CI runners and cloud workloads to known-good domains. The dropper requires DNS resolution of `check.git-service[.]com` to execute the payload.
- Implement pip `--require-hashes` with a pinned requirements file. A hash mismatch on 1.4.1–1.4.3 would have failed the build before any code ran.
- Disable auto-update on VS Code extensions in developer environments pending review.
- Monitor for the FIRESCALE pattern in GitHub commit search API calls originating from your infrastructure.
- Restrict SSM agent permissions using AWS IAM policy conditions to limit lateral movement blast radius.

Longer-Term Hardening

- Enforce minimum release age controls on package managers (OpenAI deployed this as a post-Axios control — the two affected employee devices had not yet received the updated configuration).
- Require SBOM generation and signing for all builds. Alert on transitive dependency version changes between builds.

- Scope CI pipeline permissions to least privilege. Credential harvesting only pays off when those credentials exist on the build machine.
- Review all package `__init__.py` files for network calls. SDKs do not make outbound connections at import time. Any `urllib`, `requests`, or `subprocess` call in a package's `__init__.py` warrants investigation.
- Implement GitHub Actions permissions pinning (pin actions to commit SHA, not tag) to prevent workflow token exfiltration via tampered action versions.

Phoenix Security Platform Recommendations

Traditional CVE-based scanners have no record of any package in this campaign. No CVEs exist for durabletask 1.4.1–1.4.3, the VS Code extension, or the TanStack wave. PHX-Neural flagged durabletask on behavioral signals alone, without a CVE to match against.

- **Supply chain compromise detection:** Phoenix PHX-Neural behavioral analysis flagged durabletask as COMPROMISE-tagged with 12 behavioral signals triggered. This detection does not depend on CVE assignment — it operates on package behavior, network call patterns, and code heuristics. Teams using Phoenix's package intelligence feed received a COMPROMISE signal on durabletask within the publication window.
- **FIRESCALE campaign tracking:** Phoenix maintains active tracking of the Mini Shai-Hulud / FIRESCALE campaign lineage at `sha1hulud-phoenix.pplx.app` with live IOC feeds and affected package lists. Package intelligence at `phxintel.security/package.html` provides per-package assessment against known campaign IOCs.
- **Reachability analysis:** For teams where durabletask is a transitive dependency, Phoenix reachability analysis identifies which workloads actually import the package at runtime versus which have it in the dependency graph but never execute it. This narrows remediation prioritization to environments where the payload would have fired.
- **Attack surface management:** Phoenix ASM identifies internet-facing services and CI pipelines running affected package versions. For the GitHub breach vector, identifying which developer machines and CI runners had the compromised VS Code extension installed — and which repositories those machines had access to — defines the blast radius for credential rotation.
- **Remediation campaigns:** Create a TeamPCP Wave Four remediation campaign in Phoenix, assign owners by repository and team, track the pin-to-1.4.0 fix across all services, and verify remediation via SBOM re-scan. The credential rotation scope (AWS, Azure, GCP, GitHub, Vault) can be tracked per-service against the reachability-confirmed exposure list.

Phoenix maps compromise signals to actual workload exposure: which pipelines ran affected packages, which developers had the extension, which repositories were in scope for credential rotation. The campaign is sprawling. The remediation backlog shouldn't be.

MITRE ATT&CK Mapping

Technique ID	Name	Description
T1195.001	Supply Chain Compromise: Compromise Software Dependencies	Malicious versions of durabletask, TanStack packages published via stolen PyPI/npm tokens
T1059.006	Command and Scripting Interpreter: Python	<code>rope.pyz</code> executes Python modules for credential theft, propagation, and persistence
T1078	Valid Accounts	Stolen GitHub tokens, cloud credentials used for lateral movement and exfiltration
T1552	Unsecured Credentials	Credential files (<code>~/.aws</code> , <code>~/.kube</code> , <code>~/.vault-token</code> , <code>.env</code> , Docker socket, etc.) harvested
T1021.007	Remote Services: Cloud Services (SSM)	AWS SSM SendCommand used to propagate worm to up to 5 EC2 instances per infected host
T1609	Container Administration Command (kubectl exec)	Kubernetes pod lateral movement via <code>kubectl exec</code>
T1041	Exfiltration Over C2 Channel	Encrypted credential bundles POSTed to <code>check.git-service[.]com</code>

Technique ID	Name	Description
T1567.001	Exfiltration to Code Repository	GitHub token abuse to create exfiltration repositories with Russian folklore names
T1071.001	Application Layer Protocol: Web Protocols	HTTPS used for C2, payload delivery, and exfiltration
T1485	Data Destruction	rm -rf /* on Israeli/Iranian-fingerprinted hosts via roulette.py
T1547.001	Boot or Logon Autostart: Systemd Service	pgsql-monitor.service persistence registered as 'PostgreSQL Monitor'
T1102.001	Web Service: Dead Drop Resolver (GitHub)	FIRESCALE mechanism uses GitHub commit search API for C2 rotation with RSA-signed URLs

External References

1. GitHub statement on internal repository compromise (May 20, 2026) — @github on X
2. Wiz: durabletask compromise analysis — 'The attacker compromised a GitHub account via a previous attack, dumped GitHub secrets...' (May 19, 2026)
3. Endor Labs: durabletask malicious package analysis — detection within 2 minutes of 1.4.1 publication (May 19, 2026)
4. Aikido Security: rope.pyz payload behavioral analysis — SSM propagation, kubectl exec, roulette.py wiper
5. StepSecurity: SSM SendCommand propagation mechanics and FIRESCALE documentation
6. SafeDep: 28KB Python stealer module analysis including HashiCorp Vault and password manager targeting
7. Hunt.io: C2 infrastructure correlation — 83.142.209.0/24 subnet across all TeamPCP waves (May 16, 2026)
8. The Hacker News: 'GitHub Investigating TeamPCP Claimed Breach of ~4,000 Internal Repositories' (May 20, 2026)
9. The Hacker News: 'TanStack Supply Chain Attack Hits Two OpenAI Employee Devices, Forces macOS Updates' (May 16, 2026)
10. OpenAI: 'Our response to the TanStack npm supply chain attack' — openai.com/index/our-response-to-the-tanstack-npm-supply-chain-attack (May 15, 2026)
11. BleepingComputer: 'OpenAI confirms security breach in TanStack supply chain attack' (May 14, 2026)
12. Socket Security: TanStack compromise attribution to Mini Shai-Hulud operation
13. Phoenix Security scanner (Shai-Hulud): github.com/Security-Phoenix-demo/Shai-Hulud-Sha1-Hulud-V2-npm-compromise-scanner
14. Phoenix Security live tracking: sha1hulud-phoenix.pplx.app — Mini Shai-Hulud campaign IOC feed
15. Phoenix Package Intelligence: phxintel.security/package.html
16. Phoenix Security editorial archive: phoenix.security/?s=sha1
17. @xploitsturtle2 (box turtle / shai-huturtle, linked to TeamPCP) — 'Github knew for hours, they delayed telling you and they won't be honest in the future.' X post, 3:40 AM May 20, 2026. x.com/xploitsturtle2/status/2056927898771067006

Source: <https://phoenix.security/teampcp-github-breach-durabletask-pypi-supply-chain-wave-four-2026/>