INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-11 07:40 UTC

Miasma Worm Reaches Microsoft Azure and PyPI: 73 Repositories Disabled, Hades Wave Drops 37 Malicious Python Wheels

THREAT CAMPAIGN | **CRITICAL** | CVSS 9.3

SCC Item ID	SCC-CAM-2026-0439
Type	Threat Campaign
Severity	CRITICAL
CVSS Base Score	9.3
Affected Products	Microsoft Azure/durabletask GitHub repository (versions 1.4.1-1.4.3), 19 PyPI packages including dynamo-release, spateo-release, coolbox (37 malicious wheel artifacts), AI coding agents: Claude Code, Gemini CLI, Cursor, VS Code with AI extensions
Published	2026-06-10
Discovery Source	Gemini

Executive Summary

A self-replicating supply chain worm called Miasma has compromised 73 GitHub repositories, including a Microsoft-maintained Azure SDK (durabletask versions 1.4.1-1.4.3), and poisoned 19 PyPI packages with 37 malicious Python wheel artifacts. The attack targets developer environments directly, stealing cloud credentials, API keys, and secrets when developers open affected repositories in AI coding assistants such as Claude Code, Gemini CLI, Cursor, or VS Code and interact with the code via prompt injection. Any organization using the affected Azure SDK versions or the poisoned PyPI packages may have exposed cloud infrastructure credentials, creating immediate risk of unauthorized cloud access, data exfiltration, and lateral movement across production environments.

Technical Analysis

The Miasma campaign (also tracked as Hades) operates via two distinct payload mechanisms. First, a self-replicating worm propagates through GitHub Actions workflows, compromising dependent repositories by injecting malicious workflow steps. Microsoft's Azure/durabletask SDK versions 1.4.1-1.4.3 were trojanized; these versions deliver a credential-harvesting payload that executes when a developer opens the repository in an AI coding agent, exploiting agentic context ingestion behavior (prompt injection via repository content). Second, a Bun-powered JavaScript credential stealer is embedded in 19 poisoned PyPI packages (including dynamo-release, spateo-release, and coolbox, totaling 37 malicious .whl artifacts), triggering on Python

interpreter startup at install time. Targeted credentials include cloud provider tokens (AWS, Azure, GCP), API keys, and developer secrets from the local environment. No CVE has been assigned as of this writing. Relevant CWEs: CWE-829 (Inclusion of Functionality from Untrusted Control Sphere), CWE-1357 (Reliance on Insufficiently Trustworthy Component), CWE-494 (Download of Code Without Integrity Check), CWE-522 (Insufficiently Protected Credentials), CWE-506 (Embedded Malicious Code). MITRE ATT&CK techniques: T1195.001 (Compromise Software Dependencies and Development Tools), T1195.002 (Compromise Software Supply Chain), T1554 (Compromise Client Software Binary), T1552.001 (Credentials In Files), T1059.007 (Command and Scripting Interpreter: JavaScript), T1071.001 (Application Layer Protocol: Web Protocols), T1176 (Browser Extensions, noted in technique list, likely referencing IDE extension vectors). Discovery credited to StepSecurity, Endor Labs, and SafeDep, reported approximately June 2026. The 73 affected GitHub repositories have been disabled. Patch status: As of June 2026, Microsoft has not released a patched version of durabletask. Recommend discontinuing use of durabletask or migrating to an alternative until a clean version is confirmed via official Microsoft security advisory.

Action Checklist

1. Step 1: Containment, Immediately audit your Python dependency manifests (requirements.txt, pyproject.toml, Poetry lockfiles, Conda environment files) for durabletask versions 1.4.1, 1.4.2, or 1.4.3, and for any of the 19 known poisoned PyPI packages (confirmed names include dynamo-release, spateo-release, coolbox). Remove or quarantine affected packages. Freeze installs from PyPI for these package names pending verified clean versions. Block GitHub Actions workflows that reference the 73 disabled repositories. Consistent with NIST AC-4 (Information Flow Enforcement) and CIS 2.3 (Address Unauthorized Software).
2. Step 2: Detection, Search CI/CD pipeline logs, developer workstation logs, and cloud provider access logs for anomalous outbound connections initiated at Python install time or on repository open in AI coding agents. Look for unexpected processes spawned by Bun (bun, bun.exe) or Node in Python virtual environments. Query cloud provider audit logs (AWS CloudTrail, Azure Monitor, GCP Audit Logs) for API calls from unfamiliar source IPs or at unusual times, particularly credential enumeration or token generation events. Check IDE extension logs for unexpected network calls during repository load. Consistent with NIST AU-6 (Audit Record Review, Analysis, and Reporting) and CIS 8.2 (Collect Audit Logs). D3FEND countermeasure: D3-SFA (System File Analysis), monitor Python site-packages directories and wheel installation artifacts for unexpected binaries or scripts.
3. Step 3: Eradication, Do not install or use durabletask 1.4.1-1.4.3 from PyPI. Remove all instances from developer workstations, build servers, and container images. Purge the 37 malicious wheel artifacts from any local or internal package caches (pip cache, Artifactory, Nexus, CodeArtifact). Rotate all cloud credentials, API keys, and secrets accessible from any environment where affected packages were installed or where the compromised GitHub repositories were opened in an AI coding agent. Consistent with NIST AC-2 (Account Management) and CIS 5.1 (Establish and Maintain an Inventory of Accounts). D3FEND countermeasure: D3-CRO (Credential Rotation).
4. Step 4: Recovery, After credential rotation, verify new credentials are functioning and old credentials are fully revoked across all cloud providers. Rebuild affected CI/CD pipeline containers from known-good base images. Re-run dependency installs from scratch against pinned, hash-verified package versions only. Enable or audit GitHub Actions workflow pinning (pin to full commit SHA, not mutable tags) across all organizational repositories. Validate that AI coding agent integrations do not auto-execute content from newly cloned repositories without user confirmation. Monitor cloud provider billing and access dashboards

for 30 days post-remediation for signs of persistence. Consistent with NIST AU-6 and CIS 7.1 (Establish and Maintain a Vulnerability Management Process).

5. Step 5: Post-Incident, Implement software composition analysis (SCA) tooling in CI/CD pipelines to flag new or updated dependencies before install, enforcing hash pinning and provenance verification. Establish an internal PyPI mirror or proxy with an approval gate for new packages and version updates. Review AI coding agent configurations to disable automatic code execution on repository open. Conduct a GitHub Actions workflow audit across all organizational repositories to enforce least-privilege permissions and pinned action references. Consistent with NIST CM (Configuration Management) and CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported). D3FEND countermeasures: D3-FMBV (File Magic Byte Verification) for wheel artifact integrity checks; D3-CH (Credential Hardening) to reduce secret exposure surface in developer environments.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate immediately to CISO and legal/compliance if CloudTrail, Azure Monitor, or GCP Audit Logs confirm any API calls using credentials accessible from an environment where Miasma-poisoned packages were installed, as this constitutes a confirmed credential theft and potential data breach requiring regulatory notification assessment under applicable frameworks (GDPR, state breach notification laws, SOC 2 incident reporting obligations).
Recovery Notes	After credential rotation, monitor AWS CloudTrail, Azure Monitor, and GCP Audit Logs continuously for 30 days for any activity using the rotated-out (now-invalid) credential IDs — any successful authentication against a revoked key indicates an attacker has a parallel credential set not yet identified. Validate CI/CD pipeline integrity by comparing the SHA-256 hashes of all base container images used in builds against the originating registry digests before re-enabling pipelines. Confirm AI coding agent workspace trust settings are enforced organization-wide, as Miasma's auto-execution trigger on repository open means a single misconfigured developer workstation can re-introduce the worm after environment remediation.

Forensic Artifacts	Python virtual environment bin/ or Scripts/ directories on developer workstations and CI/CD build agents: look for unexpected `bun`, `bun.exe`, or `.js` files dropped by Miasma wheel post-install hooks — these are not native Python environment artifacts and confirm active dropper execution. | pip cache wheel archives (`~/.cache/pip/wheels/` on Linux/macOS, `%LOCALAPPDATA%\pip\Cache\wheels\` on Windows): the 37 malicious `.whl` files contain embedded Bun binaries and JavaScript payloads inside the zip archive that survive package removal from site-packages and serve as forensic confirmation of installation. | AWS CloudTrail logs filtered for `GetCallerIdentity`, `AssumeRole`, `CreateAccessKey`, and `ListBuckets` events correlated by source IP to developer workstation IPs within 15 minutes of `pip install` timestamps — this time correlation is the primary forensic signal linking credential theft to the Miasma install event. | AI coding agent extension host logs (VS Code: `~/.vscode/logs/exthost*/`, Cursor: `~/.cursor/logs/`, Claude Code session logs): these logs capture HTTP requests made by the extension host process at repository-open time and will contain outbound C2 calls triggered by Miasma's automatic execution payload embedded in repository content. | GitHub Actions workflow run logs for any workflow that executed `pip install` or referenced any of the 73 now-disabled Miasma-compromised repositories: the runner logs will show whether Bun was downloaded and executed within the CI/CD context, and the `ACTIONS_RUNTIME_TOKEN` or repository secrets accessible to that workflow run must be treated as compromised.

Per-Action IR Details

Step 1: Containment — Immediately audit your Python dependency manifests (requirements.txt, pyproject.toml, Poetry lockfiles, Conda environment files) for durabletask versions 1.4.1, 1.4.2, or 1.4.3, and for any of the 19 known poisoned PyPI packages (confirmed names include dynamo-release, spateo-release, coolbox). Remove or quarantine affected packages. Freeze installs from PyPI for these package names pending verified clean versions. Block GitHub Actions workflows that reference the 73 disabled repositories. Consistent with NIST AC-4 (Information Flow Enforcement) and CIS 2.3 (Address Unauthorized Software).

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy

Controls: NIST AC-4 (Information Flow Enforcement), CIS 2.3 (Address Unauthorized Software)

Compensating: Run: `pip list --format=freeze | grep -E 'durabletask|dynamo-release|spateo-release|coolbox'` across all developer workstations and build servers. For Poetry lockfiles: `grep -rE 'durabletask|dynamo-release|spateo-release|coolbox' **/poetry.lock`. Block outbound traffic to PyPI for these specific package names by adding entries to `/etc/hosts` (Linux/macOS) or Windows hosts file resolving pypi.org to 127.0.0.1 on affected hosts pending a clean version pin. Use `pip config set global.index-url` to redirect to an internal mirror or trusted proxy immediately.

Evidence: Before removing packages, snapshot the installed wheel artifacts from pip's cache directory (`~/.cache/pip/wheels/` on Linux, `%LOCALAPPDATA%\pip\Cache\wheels\` on Windows) and the site-packages directory for the affected environment. Hash every `.whl` file using `sha256sum` or `Get-FileHash -Algorithm SHA256`. Capture `pip show durabletask` output to confirm installed version. Record the full filesystem path of any `bun`, `bun.exe`, or unexpected Node.js binary dropped into the Python virtual environment's `bin/` or `Scripts/` directory — this is the Miasma dropper's primary staging artifact.

Step 2: Detection — Search CI/CD pipeline logs, developer workstation logs, and cloud provider access logs for anomalous outbound connections initiated at Python install time or on repository open in AI coding agents. Look for unexpected processes spawned by Bun (bun, bun.exe) or Node in Python virtual environments. Query cloud provider audit logs (AWS CloudTrail, Azure Monitor, GCP Audit Logs) for API calls from unfamiliar source IPs or at unusual times, particularly credential enumeration or token generation events. Check IDE extension logs for unexpected network calls during repository load. Consistent with NIST

AU-6 (Audit Record Review, Analysis, and Reporting) and CIS 8.2 (Collect Audit Logs). D3FEND countermeasure: D3-SFA (System File Analysis) — monitor Python site-packages directories and wheel installation artifacts for unexpected binaries or scripts.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), CIS 8.2 (Collect Audit Logs)

Compensating: On Linux/macOS developer workstations, run: `sudo ausearch -x bun` or `sudo ausearch -x node` filtered to Python venv paths to find Bun/Node process launches. On Windows, query Sysmon Event ID 1 (Process Creation) filtering on `ParentImage` matching `python.exe` or `pip.exe` and `Image` matching `bun.exe` or `node.exe`. For CI/CD: grep pipeline runner logs for outbound curl/wget/fetch to non-PyPI domains during `pip install` steps. For AWS CloudTrail, run: `aws cloudtrail lookup-events --lookup-attributes AttributeKey=EventName,AttributeValue=GetCallerIdentity` to find unexpected credential enumeration calls post-install. For Azure Monitor: query `AzureActivity | where OperationNameValue contains 'token' and CallerIpAddress !in (known_ip_list)`.

Evidence: Capture Sysmon Event ID 3 (Network Connection) records showing outbound connections from `bun.exe`, `node.exe`, or `python.exe` to non-PyPI, non-GitHub domains during or immediately after `pip install` of the affected packages. Collect AWS CloudTrail `AssumeRole`, `GetSessionToken`, and `ListBuckets` events occurring within 30 minutes of package installation timestamps on the same host. Collect Azure Monitor Sign-in logs and Azure Activity logs for token issuance events correlated to developer workstation IPs. Extract VS Code, Cursor, and Claude Code extension host logs from `~/.vscode/logs/`, `~/.cursor/logs/`, or equivalent, filtering for HTTP requests made during repository open events that target external IPs.

Step 3: Eradication — Do not install or use durabletask 1.4.1–1.4.3 from PyPI. Remove all instances from developer workstations, build servers, and container images. Purge the 37 malicious wheel artifacts from any local or internal package caches (pip cache, Artifactory, Nexus, CodeArtifact). Rotate all cloud credentials, API keys, and secrets accessible from any environment where affected packages were installed or where the compromised GitHub repositories were opened in an AI coding agent. Consistent with NIST AC-2 (Account Management) and CIS 5.1 (Establish and Maintain an Inventory of Accounts). D3FEND countermeasure: D3-CRO (Credential Rotation).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST AC-2 (Account Management), CIS 5.1 (Establish and Maintain an Inventory of Accounts)

Compensating: Purge pip cache with: `pip cache purge` on each affected workstation, then verify with `pip cache list | grep -E 'durabletask|dynamo-release|spateo-release|coolbox'` returns empty. For Docker/container images: `docker images --filter 'since='` to identify images built after poisoned packages became available; rebuild from scratch using `--no-cache`. For secret rotation on AWS: `aws iam create-access-key` followed by `aws iam delete-access-key --access-key-id `. For Azure: `az ad sp credential reset` for affected service principals. Enumerate all `.env` files, `~/.aws/credentials`, `~/.config/gcloud/`, and VS Code/Cursor workspace settings files on affected developer machines for hardcoded secrets that may have been exfiltrated.

Evidence: Before rotation, document the exact access key IDs, service principal client IDs, and API key prefixes (never the secrets themselves) that were accessible in the compromised environments — this establishes the scope of potential credential theft for post-incident reporting. Capture `~/.aws/credentials`, `~/.azure/`, and `~/.config/gcloud/application_default_credentials.json` metadata (not contents) to confirm which credential files existed on affected systems at time of compromise. For CI/CD: extract the list of repository secrets and environment variables accessible to any GitHub Actions workflow that referenced the 73 disabled repositories, as the Miasma worm self-replicates into CI/CD contexts.

Step 4: Recovery — After credential rotation, verify new credentials are functioning and old credentials are fully revoked across all cloud providers. Rebuild affected CI/CD pipeline containers from known-good base images. Re-run dependency installs from scratch against pinned, hash-verified package versions only. Enable

or audit GitHub Actions workflow pinning (pin to full commit SHA, not mutable tags) across all organizational repositories. Validate that AI coding agent integrations do not auto-execute content from newly cloned repositories without user confirmation. Monitor cloud provider billing and access dashboards for 30 days post-remediation for signs of persistence. Consistent with NIST AU-6 and CIS 7.1 (Establish and Maintain a Vulnerability Management Process).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Verify old AWS credentials are revoked: `aws iam list-access-keys --user-name ` — confirm `Status: Inactive` or absence of the old key ID. For Azure: `az ad sp credential list --id ` to confirm old credentials are absent. Generate `pip-compile --generate-hashes requirements.in > requirements.txt` using pip-tools to produce hash-pinned lockfiles before re-installing any dependencies. For GitHub Actions workflow pinning audit: `grep -rE 'uses: [^@]+@(?!([a-f0-9]{40}))' .github/workflows/` to find all non-SHA-pinned action references in your repositories. For AI coding agent safe mode: in VS Code settings, set `"security.workspace.trust.enabled": true` and disable auto-run extensions for untrusted workspaces.

Evidence: Before rebuilding CI/CD containers, export and preserve the full layer history of affected container images with `docker history --no-trunc` to document which build steps installed the poisoned packages. Capture GitHub Actions workflow run logs for any workflow that executed during the window when the 73 repositories were active but before disabling, specifically any `run:` steps that called `pip install` or `npm install` — these logs may contain exfiltration confirmation or Bun dropper execution evidence. Retain cloud provider billing snapshots for the 30-day monitoring baseline: unexpected compute, storage, or API Gateway charges are an indicator of Miasma persistence or unauthorized resource use by stolen credentials.

Step 5: Post-Incident — Implement software composition analysis (SCA) tooling in CI/CD pipelines to flag new or updated dependencies before install, enforcing hash pinning and provenance verification. Establish an internal PyPI mirror or proxy with an approval gate for new packages and version updates. Review AI coding agent configurations to disable automatic code execution on repository open. Conduct a GitHub Actions workflow audit across all organizational repositories to enforce least-privilege permissions and pinned action references. Consistent with NIST CM controls (no specific CM control ID is present in the provided knowledge base — no mapped control) and CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported). D3FEND countermeasure: D3-FMBV (File Magic Byte Verification) for wheel artifact integrity checks; D3-CH (Credential Hardening) to reduce secret exposure surface in developer environments.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity

Controls: CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: Deploy pip-audit (`pip-audit -r requirements.txt`) in CI/CD as a free SCA step to flag known-malicious packages against the OSV database before install. Stand up a local PyPI mirror using devpi-server (`pip install devpi-server && devpi-server --init`) and configure all developer machines with `pip config set global.index-url http://internal-devpi/simple/` to route all installs through a controlled proxy. Write a YARA rule targeting the Miasma wheel artifact pattern — specifically scanning for embedded Bun binaries (`bun`, `bun.exe`) or JavaScript payloads (`*.js`) inside `.whl` zip archives — and run it via `yara -r miasma_wheel.yar ~/.cache/pip/wheels/`. For GitHub Actions least-privilege: add `permissions: read-all` as the default at the workflow level and grant write only to specific jobs that require it.

Evidence: Document the full list of 37 malicious wheel artifact SHA-256 hashes recovered during eradication as organizational IOCs for future SCA tooling ingestion. Preserve the Miasma-modified `setup.py`, `__init__.py`, or `post-install` hook scripts extracted from affected wheels as YARA rule development artifacts. Record the specific outbound C2 domains or IPs observed in network logs during the detection phase as threat intelligence feed entries.

Capture a diff of GitHub Actions workflow `permissions:` blocks before and after the audit to evidence the least-privilege remediation for compliance records.

Detection Guidance

Focus detection on three surfaces: (1) Package installation behavior, monitor for Bun or Node processes spawned as child processes of pip or Python during package install. Alert on outbound network connections initiated within seconds of a pip install command, particularly to non-PyPI domains. Hash-verify installed wheel artifacts against PyPI's published hashes; durabletask 1.4.1-1.4.3 wheels are confirmed malicious. (2) AI coding agent context ingestion, monitor process trees for credential access (reading ~/.aws/credentials, ~/.azure, ~/.config/gcloud, or environment variables containing TOKEN, KEY, SECRET) initiated by IDE processes (code, cursor, claude, gemini-cli) immediately after repository open or file indexing. (3) Cloud credential abuse, query AWS CloudTrail for GetCallerIdentity, AssumeRole, or ListBuckets calls from developer workstation IPs or build-runner IPs outside normal business hours; query Azure Monitor for unexpected service principal activity or token issuance events. Confirmed poisoned package names: dynamo-release, spateo-release, coolbox. Complete list of 19 packages not yet published; treat any unexpected or unfamiliar Python package added to dependency manifests as suspicious pending full IOC disclosure. D3FEND countermeasures: D3-LAM (Local Account Monitoring) for developer workstation credential access events; D3-SFA (System File Analysis) for modification of Python site-packages. NIST AU-2 (Event Logging) and AU-12 (Audit Record Generation) provide the logging baseline required for this detection.

Indicators of Compromise

Type	Value	Context	Confidence
HASH	`not confirmed in available sources`	Malicious durabletask wheel artifacts (versions 1.4.1–1.4.3) — hash values not published in T3 sources reviewed; verify against PyPI audit data from Endor Labs or SafeDep advisories	LOW
DOMAIN	`not confirmed in available sources`	C2 or exfiltration endpoints used by Bun-powered JavaScript credential stealer — not disclosed in reviewed sources; monitor for anomalous outbound connections from Python/Bun processes	LOW
URL	`https://pypi.org/project/durabletask/`	Official PyPI listing for durabletask — verify installed version is not 1.4.1, 1.4.2, or 1.4.3; treat those versions as malicious	HIGH
HASH	`not confirmed in available sources`	dynamo-release, spateo-release, coolbox wheel artifacts — specific hashes not published in reviewed sources; cross-reference with SafeDep advisory for full IOC list	LOW

Framework Mappings

MITRE-ATTACK

- **T1176** — Software Extensions
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1071.001** — Web Protocols
- **T1059.007** — JavaScript
- **T1554** — Compromise Host Software Binary
- **T1195.002** — Compromise Software Supply Chain
- **T1552.001** — Credentials In Files

NIST-800-53R5

- **CM-7** — Least Functionality
- **SA-9** — External System Services
- **SR-3** — Supply Chain Controls and Processes
- **SI-7** — Software, Firmware, and Information Integrity
- **CM-3** — Configuration Change Control
- **IA-5** — Authenticator Management
- **SR-2** — Supply Chain Risk Management Plan

OWASP-TOP10-2021

- **A08:2021** — Software and Data Integrity Failures
- **A04:2021** — Insecure Design
- **A07:2021** — Identification and Authentication Failures

CIS-V8

- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **5.2** — Use Unique Passwords
- **6.3** — Require MFA for Externally-Exposed Applications
- **15.1** — Establish and Maintain an Inventory of Service Providers

HIPAA-SECURITY

- **164.308(a)(5)(ii)(D)** — Password Management
- **164.312(d)** — Person or Entity Authentication

SOC2-TSC

- **CC6.1** — Logical access security software, infrastructure, and architectures
- **CC9.2** — Manages risks associated with vendors and business partners

NIST-CSF-2

- **GV.SC-01** — Cybersecurity supply chain risk management program

ISO-27001-2022

- **A.5.21** — Managing information security in the ICT supply chain
- **A.5.23** — Information security for use of cloud services

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1176	Software Extensions	Persistence
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1071.001	Web Protocols	Command-And-Control
T1059.007	JavaScript	Execution
T1554	Compromise Host Software Binary	Persistence
T1195.002	Compromise Software Supply Chain	Initial-Access
T1552.001	Credentials In Files	Credential-Access

Sources

Source	URL	Tier
Miasma Worm Hits Microsoft Again: Azure Functions Action and 72 ...	https://www.stepsecurity.io/blog/miasma-worm-hits-microsoft-again-a...	T3
Trojanized Microsoft SDK: durabletask 1.4.1 through ... - Endor Labs	https://www.endorlabs.com/learn/trojanized-microsoft-sdk-durabletas...	T3
Malicious durabletask on PyPI: Multi-Cloud Credential Stealer with ...	https://safedep.io/malicious-durabletask-pypi-supply-chain-attack	T3
Supply Chain Attack Hits Microsoft GitHub Repos, AI Coding Tools	https://redmondmag.com/articles/2026/06/08/supply-chain-attack-hits...	T3
73 Microsoft GitHub Repositories Compromised via AI Coding Tools	https://www.rescana.com/post/miasma-worm-supply-chain-attack-73-mic...	T3

DISCLAIMER

This intelligence report is produced by Tech Jacks Solutions Security Command Center (SCC) for informational purposes only. It does not constitute professional security advice, legal counsel, or an incident response engagement. The information herein is derived from publicly available sources and AI-assisted analysis; while every effort is made to ensure accuracy, Tech Jacks Solutions makes no warranties regarding completeness or timeliness. Organizations should conduct their own validation and consult qualified security professionals before taking action based on this report. Tech Jacks Solutions is not liable for any damages resulting from the use of this information.

Generated 2026-06-11 07:40 UTC by TJS Security Command Center