INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-05 08:18 UTC

PyTorch Lightning 2.6.3 Backdoor Delivers ShaiWorm Infostealer Targeting Cloud and Browser Credentials

THREAT CAMPAIGN | HIGH | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0269
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	PyTorch Lightning 2.6.3 (PyPI / Lightning AI); Chrome, Firefox, Brave browsers; AWS, Azure, GCP cloud credential stores
Published	2026-05-04T13:15:27
Discovery Source	Rss

Executive Summary

A malicious version of PyTorch Lightning (v2.6.3) was published to PyPI and silently stole cloud and browser credentials on import. With approximately 11 million monthly downloads, the package is widely embedded in AI/ML development pipelines across industries. Any organization where developers installed this version may have exposed AWS, Azure, GCP credentials and browser-stored secrets, creating direct risk of cloud account takeover and data exfiltration.

Technical Analysis

PyTorch Lightning version 2.6.3 was compromised and published to PyPI as a supply chain attack (MITRE T1195.001, T1195.002). On import, the package executed a credential-stealing payload identified by Microsoft Defender as ShaiWorm (part of the Shai-Hulud themed malware family). Targeted data included cloud provider credentials (AWS, Azure, GCP via T1528, T1078.004), browser-stored credentials from Chrome, Firefox, and Brave (T1555.003), and environment files including .env secrets (T1552.001). The payload used obfuscation (T1027), Python and JavaScript execution (T1059.006, T1059.007), and exfiltrated data over C2 channels (T1041). A related package, intercom-client, has been identified as part of the same cross-ecosystem campaign. The compromise vector has not been confirmed; a full audit of adjacent releases is ongoing. A clean version has been restored on PyPI. Relevant CWEs: CWE-506 (embedded malicious code), CWE-494 (download of code without integrity check), CWE-522 (insufficiently protected credentials), CWE-312 (cleartext storage of sensitive information). No CVE has been assigned at this time. CVSS base score reported at 7.5 (High). Priority score:

0.826.

Action Checklist

1. Containment: Immediately audit all development, CI/CD, and ML pipeline environments for installed versions of PyTorch Lightning. Isolate any system where version 2.6.3 was installed. Revoke and rotate all AWS, Azure, and GCP credentials accessible from those systems, as well as any secrets stored in .env files on affected hosts.
2. Detection: Search package manager logs, container image manifests, and requirements.txt / pyproject.toml files for 'pytorch-lightning==2.6.3' or 'lightning==2.6.3'. Review cloud provider access logs (AWS CloudTrail, Azure Monitor, GCP Cloud Audit Logs) for anomalous API calls or credential use from developer or CI/CD systems. Look for unexpected outbound connections from Python processes, unusual environment variable reads, and browser profile directory access by non-browser processes. Cross-reference with Microsoft Defender detections for 'ShaiWorm'. Also audit intercom-client package versions across your dependency graph.
3. Eradication: Upgrade PyTorch Lightning to the verified clean release published after the incident (confirm version via Lightning AI's official advisory at https://lightning.ai/blog/pytorch-lightning-supply-chain-attack before upgrading). Remove version 2.6.3 from all package caches, container images, virtual environments, and artifact repositories. Purge and rebuild any Docker images or CI/CD pipeline environments that used the compromised version.
4. Recovery: After rotating credentials, validate new credentials are functioning and old credentials are fully invalidated across all cloud providers. Review cloud IAM policies for any new roles, policies, or access keys created during the exposure window. Monitor cloud billing and API usage for anomalies for at least 30 days. Confirm browser credential stores on affected developer workstations have been cleared and passwords changed for any accounts stored in Chrome, Firefox, or Brave on those systems.
5. Post-Incident: This attack exposed gaps in software supply chain integrity verification. Implement hash-pinning or signature verification for all PyPI packages in production and CI/CD pipelines. Evaluate adoption of tools such as Sigstore/Cosign or private package mirrors with integrity checks. Establish a process for monitoring security advisories from PyPI, Sonatype, and major dependency vendors. Review secrets management practices to ensure cloud credentials and .env secrets are not accessible on developer workstations in plaintext.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO, legal counsel, and cloud provider security teams immediately if CloudTrail, Azure Monitor, or GCP Audit Logs confirm any IAM key creation, role assumption, data access (S3 GetObject, Azure Blob read, GCP Storage list), or credential-based API calls originating from affected developer or CI/CD systems during the pytorch-lightning 2.6.3 exposure window, as confirmed cloud account access constitutes a reportable breach under GDPR Article 33, CCPA, and applicable state breach notification laws if PII or regulated data was accessible from the compromised cloud accounts.

Recovery Notes	After credential rotation, validate invalidation by attempting to use the old AWS access key via `aws sts get-caller-identity` with the revoked key — a successful call indicates rotation failed and must be retried. Monitor CloudTrail, Azure Monitor, and GCP Audit Logs continuously for 30 days post-rotation for any API calls using credentials that match the format or naming convention of keys that existed during the exposure window, as ShaiWorm may have exfiltrated credentials to an attacker infrastructure that delays use to evade detection. Browser-stored credentials on affected developer workstations should be treated as fully compromised regardless of whether browser profile access was confirmed in logs, because pytorch-lightning 2.6.3 executes on import and log visibility into filesystem reads by Python processes is typically incomplete without Sysmon or an EDR agent deployed.
Forensic Artifacts	PyPI package cache containing pytorch-lightning 2.6.3 wheel file: located at ~/.cache/pip/wheels/ (Linux/macOS) or %LOCALAPPDATA%\pip\cache\ (Windows) — the wheel file itself is the malware artifact and its SHA-256 hash should be compared against the known-malicious hash published in the Lightning AI advisory to confirm the backdoored version was present. | Cloud provider credential files read by ShaiWorm on import: ~/.aws/credentials (AWS), ~/.azure/accessTokens.json and ~/.azure/msal_token_cache.json (Azure), ~/.config/gcloud/application_default_credentials.json (GCP) — capture read-only forensic copies with timestamps intact using `cp --preserve=timestamps` before any modification. | Browser SQLite credential databases accessed by the infostealer: Chrome Login Data at %LOCALAPPDATA%\Google\Chrome\User Data\Default\Login Data (Windows) or ~/Library/Application Support/Google/Chrome/Default/Login Data (macOS); Firefox logins.json at %APPDATA%\Mozilla\Firefox\Profiles\\logins.json; Brave at %LOCALAPPDATA%\BraveSoftware\Brave-Browser\User Data\Default\Login Data — copy these files with a read-only SQLite dump (`sqlite3 'Login Data' .dump > logindata_dump.sql`) before clearing. | AWS CloudTrail events in the window from earliest pytorch-lightning 2.6.3 install to present: specifically filter for CreateAccessKey, AssumeRole, GetSecretValue, ListBuckets, and PutObject events from source IPs matching developer workstations or CI/CD runners — export as JSON using `aws cloudtrail lookup-events --lookup-attributes AttributeKey=EventName,AttributeValue=CreateAccessKey` for each relevant event type. | Python process network connection logs: on Linux, /proc//net/tcp at time of pytorch-lightning import (if captured by Sysmon or auditd); on Windows, Sysmon Event ID 3 (Network Connection Initiated) for python.exe or python3.exe processes — these capture ShaiWorm's outbound C2 exfiltration channel carrying the stolen cloud credentials and browser secrets.

Per-Action IR Details

Containment — Immediately audit all development, CI/CD, and ML pipeline environments for installed versions of PyTorch Lightning. Isolate any system where version 2.6.3 was installed. Revoke and rotate all AWS, Azure, and GCP credentials accessible from those systems, as well as any secrets stored in .env files on affected hosts.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST AC-2 (Account Management), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Run `pip show pytorch-lightning` and `pip show lightning` on each developer workstation and CI runner to confirm version 2.6.3 presence. For bulk scanning across a fleet without EDR, deploy a one-liner via SSH or Ansible: `python3 -c "import importlib.metadata; print(importlib.metadata.version('pytorch-lightning'))" 2>/dev/null`. Use AWS CLI `aws iam list-access-keys --user-name ` and `aws iam delete-access-key` to rotate IAM keys; for GCP use `gcloud iam service-accounts keys list` followed by `gcloud iam service-accounts keys delete`. Store new credentials exclusively in a secrets manager (e.g., HashiCorp Vault free tier) rather than in .env files.

Evidence: Before isolating affected systems, capture: (1) full `pip freeze` output from each affected virtual environment to establish the exact dependency graph at time of compromise; (2) a snapshot of all environment variables (`env` on Linux/macOS, `Get-ChildItem Env:` in PowerShell) to document which cloud credentials were populated in-process when pytorch-lightning 2.6.3 was imported; (3) .env files from project root directories — ShaiWorm specifically targets these for plaintext credential harvesting; (4) ~/.aws/credentials, ~/.azure/accessTokens.json, and ~/.config/gcloud/application_default_credentials.json to establish which credential files existed and were readable at time of infection; (5) running process list and open file handles at time of isolation using `lsof -p ` or `handle.exe` (Sysinternals) to capture any active exfiltration connections before network isolation.

Detection — Search package manager logs, container image manifests, and requirements.txt / pyproject.toml files for 'pytorch-lightning==2.6.3' or 'lightning==2.6.3'. Review cloud provider access logs (AWS CloudTrail, Azure Monitor, GCP Cloud Audit Logs) for anomalous API calls or credential use from developer or CI/CD systems. Look for unexpected outbound connections from Python processes, unusual environment variable reads, and browser profile directory access by non-browser processes. Cross-reference with Microsoft Defender detections for 'ShaiWorm'. Also audit intercom-client package versions across your dependency graph.

NIST Phase: Detection Analysis

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

Controls: NIST SI-4 (System Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-2 (Event Logging), CIS 8.2 (Collect Audit Logs), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: For package scanning: `grep -r 'pytorch-lightning==2.6.3\|lightning==2.6.3' /path/to/repos --include='*.txt' --include='*.toml' --include='*.cfg'`; scan Docker image layers with `docker history | grep pip` and `docker run --rm pip show pytorch-lightning`. For network detection without SIEM, deploy Sysmon with the SwiftOnSecurity config and filter Event ID 3 (Network Connection) for python.exe or python3 connecting to non-internal IPs on ports 443/80. Use Wireshark or `tcpdump -i any -w capture.pcap port 443 and host python_process_ip` during active pipeline execution to catch ShaiWorm's credential exfiltration C2 traffic. For CloudTrail anomaly detection without a SIEM, use the free AWS CloudTrail Lake query: `SELECT eventName, sourceIPAddress, userIdentity.arn FROM  WHERE eventTime > '' AND sourceIPAddress NOT IN ('')`. Write a Sigma rule targeting Windows Security Event ID 4663 (Object Access) for python.exe accessing `%LOCALAPPDATA%\Google\Chrome\User Data\Default\Login Data` or `%APPDATA%\Mozilla\Firefox\Profiles\*\logins.json`.

Evidence: Capture the following before this detection sweep to avoid evidence loss from log rotation: (1) AWS CloudTrail logs scoped to the window between earliest possible 2.6.3 install date and present — filter on `AssumeRole`, `GetSecretValue`, `ListBuckets`, and `CreateAccessKey` events originating from developer or CI/CD source IPs; (2) Azure Monitor sign-in logs and Azure AD audit logs for service principal token issuance anomalies during the exposure window; (3) GCP Cloud Audit Logs for `storage.objects.list`, `secretmanager.versions.access`, and `iam.serviceAccountKeys.create` from affected GCP projects; (4) Browser SQLite credential databases — on Linux: `~/.config/google-chrome/Default/Login Data`, on macOS: `~/Library/Application Support/Google/Chrome/Default/Login Data`, on Windows: `%LOCALAPPDATA%\Google\Chrome\User Data\Default\Login Data` — collect read-only copies with `sqlite3 'Login Data' .dump` before any clearing; (5) pip install log at `~/.local/share/pip/` or `%APPDATA%\pip\pip.log` to establish exact install timestamp of pytorch-lightning 2.6.3 per host.

Eradication — Upgrade PyTorch Lightning to the verified clean release published after the incident (confirm version via Lightning AI's official advisory at https://lightning.ai/blog/pytorch-lightning-supply-chain-attack before upgrading). Remove version 2.6.3 from all package caches, container images, virtual environments, and artifact repositories. Purge and rebuild any Docker images or CI/CD pipeline environments that used the compromised version.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST CM-2 (Baseline Configuration), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform

Automated Application Patch Management)

Compensating: Eradicate pytorch-lightning 2.6.3 from virtual environments with `pip uninstall pytorch-lightning lightning -y && pip cache purge`. For conda environments: `conda remove pytorch-lightning && conda clean --all`. Remove from local PyPI mirrors or Artifactory/Nexus caches by searching for the specific wheel file: `find / -name 'pytorch_lightning-2.6.3*' -o -name 'lightning-2.6.3*' 2>/dev/null` and deleting matches. Purge Docker layer cache: `docker image prune -a` after tagging clean replacement images; rebuild from a pinned Dockerfile base using the verified clean version hash. For CI/CD, update all requirements.txt and pyproject.toml files to pin the verified clean version and commit the change so pipeline history reflects the remediation. Note: the advisory URL in the original step should be validated by a human before use — this response cannot actively verify that URL resolves to current content.

Evidence: Before removing the compromised package, preserve forensic copies of: (1) the malicious pytorch-lightning 2.6.3 wheel file itself from pip cache (`~/.cache/pip/wheels/` on Linux, `%LOCALAPPDATA%\pip\cache\` on Windows) — this is primary malware evidence and should be hashed with SHA-256 and submitted to your threat intel platform or VirusTotal; (2) the installed package directory (typically `site-packages/pytorch_lightning/`) — tar/zip the entire directory for later static analysis of the backdoor code before `pip uninstall` destroys it; (3) any pyproject.toml, requirements.txt, or lock files (poetry.lock, Pipfile.lock) that pinned version 2.6.3, as these establish scope and may be needed for regulatory documentation; (4) container image digests (`docker inspect | grep -i digest`) for any images built with the compromised package to support chain-of-custody documentation.

Recovery — After rotating credentials, validate new credentials are functioning and old credentials are fully invalidated across all cloud providers. Review cloud IAM policies for any new roles, policies, or access keys created during the exposure window. Monitor cloud billing and API usage for anomalies for at least 30 days. Confirm browser credential stores on affected developer workstations have been cleared and passwords changed for any accounts stored in Chrome, Firefox, or Brave on those systems.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST AC-2 (Account Management), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 5.3 (Disable Dormant Accounts), CIS 6.1 (Establish an Access Granting Process)

Compensating: For IAM drift detection without a CSPM tool: use `aws iam generate-credential-report && aws iam get-credential-report` to enumerate all access keys and their last-used dates, then flag any keys created during the exposure window; use `aws iam list-policies --scope Local` to identify custom policies created post-install; for GCP, run `gcloud projects get-iam-policy --format=json` and diff against a known-good baseline. For browser credential clearing on developer workstations, use the free BrowserPasswordDecryptor (Nirsoft) or manually delete Chrome's `Login Data` SQLite file at `%LOCALAPPDATA%\Google\Chrome\User Data\Default\Login Data` and Firefox's `logins.json` at `%APPDATA%\Mozilla\Firefox\Profiles\\logins.json`, then force password resets for all accounts identified in those stores. Set up free AWS Cost Anomaly Detection alerts (`aws ce create-anomaly-monitor`) to trigger on spend spikes indicative of cryptomining or data egress from attacker-controlled resources.

Evidence: Before marking recovery complete, document and retain: (1) a before/after diff of AWS IAM, Azure AD, and GCP IAM policies covering the full exposure window — specifically any `iam:CreateRole`, `iam:AttachRolePolicy`, or `iam:CreateAccessKey` events in CloudTrail that were not initiated by known administrators; (2) cloud billing export snapshots (AWS Cost Explorer CSV, GCP Billing export to BigQuery) for the 30-day post-compromise window as a baseline to detect delayed attacker activity; (3) confirmation screenshots or CLI output showing old access keys in INACTIVE state (`aws iam update-access-key --status Inactive`) as audit evidence of successful revocation; (4) browser profile directory hash (before and after clearing) to document that stored credentials were purged from affected developer machines.

Post-Incident — This attack exposed gaps in software supply chain integrity verification. Implement hash-pinning or signature verification for all PyPI packages in production and CI/CD pipelines. Evaluate adoption of tools such as Sigstore/Cosign or private package mirrors with integrity checks. Establish a process for monitoring security advisories from PyPI, Sonatype, and major dependency vendors. Review secrets management practices to ensure cloud credentials and .env secrets are not accessible on developer

workstations in plaintext.

NIST Phase: Post Incident

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

Controls: NIST SI-7 (Software, Firmware, and Information Integrity), NIST SA-12 (Supply Chain Protection), NIST IR-8 (Incident Response Plan), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Implement hash-pinning immediately using `pip-compile` (pip-tools) to generate a requirements.txt with exact hashes (`pip-compile --generate-hashes`), then enforce hash checking in CI with `pip install --require-hashes -r requirements.txt` — this would have blocked pytorch-lightning 2.6.3 installation if the hash differed from the pinned clean version. For advisory monitoring without a commercial feed, subscribe to OSV (osv.dev) alerts for your dependency graph using `osv-scanner --lockfile requirements.txt` (free, Google-maintained); set up a PyPI RSS feed monitor for pytorch-lightning and intercom-client specifically. Migrate cloud credentials from .env files to environment variables injected at runtime by a secrets manager — HashiCorp Vault Community Edition is free and prevents plaintext credential exposure to any process with filesystem read access, which is the exact mechanism ShaiWorm exploited. Run `trufflehog filesystem --directory=` (free) post-incident to identify any remaining plaintext secrets committed to source control.

Evidence: For the post-incident review, assemble: (1) a full dependency graph snapshot (pip freeze or poetry.lock) from each affected environment at time of compromise — this establishes whether intercom-client or other transitive dependencies of pytorch-lightning 2.6.3 also contained malicious code requiring separate investigation; (2) a timeline reconstruction from pip install logs, CloudTrail timestamps, and CI/CD build logs correlating the first install of 2.6.3 to the first anomalous cloud API call, establishing the attacker's dwell time and the credential-to-action gap; (3) a lessons-learned document specifically addressing why pytorch-lightning 2.6.3 passed through existing dependency review controls, to serve as the basis for supply chain process improvement under NIST IR-8 (Incident Response Plan) update requirements.

Detection Guidance

Primary detection signals: (1) Presence of 'pytorch-lightning==2.6.3' or 'lightning==2.6.3' in any pip freeze output, requirements file, container layer, or artifact. (2) Microsoft Defender detections tagged 'ShaiWorm' on developer or CI/CD hosts. (3) Outbound network connections from Python interpreter processes to unexpected external IPs or domains, review EDR telemetry and network flow logs for Python process network activity during the window the package was available (approximate exposure window: confirm with Lightning AI advisory). (4) Anomalous AWS CloudTrail events: ListBuckets, DescribeInstances, GetSecretValue, or AssumeRole calls from developer machines or CI systems. (5) Azure Monitor / GCP Cloud Audit Logs: credential use outside normal service account patterns. (6) File system access to browser profile directories (e.g., Chrome User Data, Firefox profiles) by non-browser processes. (7) Reads of .env files or environment variables by Python processes outside expected application context. Also audit intercom-client package for the same indicators, it has been identified as part of the same campaign. IOCs: verify Kodem Security, Sonatype, and Semgrep sources for published network indicators, domains, and file hashes. If not publicly available, monitor vendor intelligence feeds for updated IOC releases.

Indicators of Compromise

Type	Value	Context	Confidence
HASH	`Not publicly confirmed in available sources`	Package hash for pytorch-lightning 2.6.3 malicious release — confirm via Sonatype or PyPI audit logs; do not rely on unverified values	LOW
DOMAIN	`Not publicly confirmed in available sources`	C2 exfiltration infrastructure associated with ShaiWorm — monitor vendor feeds from Sonatype, Semgrep, and Kodem Security for confirmed network IOCs	LOW

Framework Mappings

MITRE-ATTACK

- **T1059.007** — JavaScript
- **T1528** — Steal Application Access Token
- **T1059** — Command and Scripting Interpreter
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1555.003** — Credentials from Web Browsers
- **T1078.004** — Cloud Accounts
- **T1027** — Obfuscated Files or Information
- **T1059.006** — Python
- **T1041** — Exfiltration Over C2 Channel
- **T1195.002** — Compromise Software Supply Chain
- **T1552.001** — Credentials In Files

NIST-800-53R5

- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-7** — Software, Firmware, and Information Integrity
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SA-9** — External System Services
- **SR-3** — Supply Chain Controls and Processes
- **IA-5** — Authenticator Management
- **CM-3** — Configuration Change Control
- **SR-2** — Supply Chain Risk Management Plan

OWASP-TOP10-2021

- **A04:2021** — Insecure Design
- **A07:2021** — Identification and Authentication Failures

- **A08:2021** — Software and Data Integrity Failures

CIS-V8

- **5.2** — Use Unique Passwords
- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **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
T1059.007	JavaScript	Execution
T1528	Steal Application Access Token	Credential-Access
T1059	Command and Scripting Interpreter	Execution
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1555.003	Credentials from Web Browsers	Credential-Access
T1078.004	Cloud Accounts	Defense-Evasion
T1027	Obfuscated Files or Information	Defense-Evasion
T1059.006	Python	Execution
T1041	Exfiltration Over C2 Channel	Exfiltration
T1195.002	Compromise Software Supply Chain	Initial-Access

Technique ID	Technique Name	Tactic
T1552.001	Credentials In Files	Credential-Access

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/backdoored-pytorch-l...	T3
How the PyTorch Lightning Community Discovered a Supply Chain ...	https://lightning.ai/blog/pytorch-lightning-supply-chain-attack	T3
Malicious PyTorch Lightning Packages Found on PyPI - Sonatype	https://www.sonatype.com/blog/malicious-pytorch-lightning-packages-...	T3
Mini Shai-Hulud Strikes PyTorch Lightning and intercom-client	https://www.kodemsecurity.com/resources/mini-shai-hulud-strikes-pyt...	T3
Shai-Hulud Themed Malware Found in the PyTorch Lightning AI ...	https://semgrep.dev/blog/2026/malicious-dependency-in-pytorch-light...	T3

Generated 2026-05-05 08:18 UTC by TJS Security Command Center