INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-14 05:02 UTC

# Miasma Worm Source Code Leak Amplifies Supply Chain Attack Risk Across npm, PyPI, CI/CD, and AI Coding Tools

**THREAT CAMPAIGN** | **CRITICAL** | CVSS 9.5

| | |
|---|---|
| **SCC Item ID** | SCC-CAM-2026-0455 |
| **Type** | Threat Campaign |
| **Severity** | CRITICAL |
| **CVSS Base Score** | 9.5 |
| **Affected Products** | npm, PyPI, RubyGems, GitHub, GitHub Actions, JFrog Artifactory, AWS Systems Manager, Kubernetes, Claude, Gemini, Cursor, GitHub Copilot, Kiro, Cline, Red Hat npm packages, Microsoft GitHub repositories |
| **Published** | 2026-06-10T16:27:08 |
| **Discovery Source** | Rss |

## Executive Summary

Threat actors published the source code for Miasma, a credential-stealing worm targeting software supply chains, across multiple compromised GitHub accounts. The worm harvests credentials from cloud and CI/CD environments and uniquely targets AI coding tool configurations used by developer teams, meaning malicious code can propagate through AI-assisted development workflows. Organizations with developer populations relying on npm, PyPI, GitHub Actions, or AI coding assistants face an elevated and widening threat window as lower-sophistication actors adopt the leaked code to launch derivative campaigns.

## Technical Analysis

Miasma is a supply chain worm whose source code was deliberately leaked across compromised GitHub accounts, functioning as a force-multiplier for downstream attackers. The worm operates without traditional C2 infrastructure, using GitHub itself as its command channel, which lowers the detection surface and operator skill threshold. Attack vectors include malicious packages on npm, PyPI, and RubyGems (CWE-506: Embedded Malicious Code; CWE-494: Insufficient Verification of Download Integrity), credential harvesting from CI/CD environments including GitHub Actions, AWS Systems Manager, and Kubernetes (CWE-522: Insufficiently Protected Credentials; CWE-798: Hard-coded Credentials), and poisoning of AI coding tool configurations for Cursor, GitHub Copilot, Kiro, Cline, Claude, and Gemini, enabling propagation through LLM-assisted code

generation pipelines. Relevant MITRE ATT&CK techniques include T1195.001 and T1195.002 (Supply Chain Compromise), T1059.007 (JavaScript execution), T1552.001 and T1552.004 (credential access from files and private keys), T1071.001 (web protocol C2), T1608.001 (stage capabilities: upload malware), T1528 (steal application access token), T1485 (data destruction), and T1027/T1027.002 (obfuscation). No CVE has been assigned; this is a campaign-level threat. The Shai-Hulud worm leak precedent documented a measurable surge in attack variants following source disclosure, making derivative campaign activity a near-term certainty. No vendor patch is applicable; mitigation is configuration, detection, and process-based.

## Action Checklist

1. Step 1: Containment. Immediately audit all recently added or updated dependencies in npm, PyPI, and RubyGems against known-good package manifests. Freeze dependency pinning across all CI/CD pipelines (GitHub Actions, JFrog Artifactory) to prevent auto-ingestion of new or modified packages. Revoke and rotate any secrets, tokens, or API keys exposed in CI/CD pipeline logs or environment variables, prioritizing AWS Systems Manager parameters and Kubernetes secrets. Per NIST AC-6 (Least Privilege), restrict package registry write permissions to the minimum set of accounts required.
2. Step 2: Detection. Query CI/CD and source control logs for unexpected outbound connections to GitHub from build runners, anomalous package publish events from developer accounts, and modifications to AI coding tool configuration files (e.g., .cursorrules, Copilot configuration files, Cline context files). Review AU-2 (Event Logging) coverage to confirm logging is active on package registries and CI/CD systems. Hunt for indicators consistent with T1608.001 (staged malicious packages), T1552.001 (credential file reads), and T1071.001 (HTTP/S C2 to GitHub). Flag any packages referencing Miasma or Shai-Hulud strings in metadata or code. Enable CIS 8.2 (Collect Audit Logs) across all build and deployment infrastructure if not already active.
3. Step 3: Eradication. Remove any packages identified as malicious from internal artifact repositories (JFrog Artifactory, GitHub Packages). Purge and re-provision any CI/CD runner environments where compromise is suspected. Audit and harden AI coding tool configuration files across developer workstations; remove or reset any configuration that references untrusted or unrecognized sources. Enforce CIS 2.3 (Address Unauthorized Software) for all packages in developer environments. Rotate all credentials harvested from affected systems, including AWS IAM keys, Kubernetes service account tokens, and GitHub personal access tokens.
4. Step 4: Recovery. Validate pipeline integrity by re-running builds from pinned, verified dependency versions and comparing build artifacts against pre-incident baselines. Confirm no malicious packages remain in any internal registry using hash verification (D3-FMBV: File Magic Byte Verification and artifact hash comparison). Re-enable normal CI/CD operations only after credential rotation is confirmed complete and audit logging (NIST AU-9, Protection of Audit Information) is verified active. Monitor for anomalous outbound GitHub API calls from build systems for a minimum of 30 days post-remediation.
5. Step 5: Post-Incident. Conduct a dependency provenance review: implement or strengthen software composition analysis (SCA) tooling across all pipelines to enforce CIS 7.1 (Establish and Maintain a Vulnerability Management Process) for open-source components. Establish policy for AI coding tool configuration management, treat AI tool context files as code artifacts subject to version control and review. Review separation of duties (NIST AC-5) between accounts authorized to publish packages and accounts with production access. Document lessons learned against T1195.001/T1195.002 supply chain controls and schedule a tabletop exercise simulating a derivative Miasma variant within 60 days.

## Detection Guidance

Focus detection on four surfaces: package registries, CI/CD pipeline behavior, credential access patterns, and AI tool configuration files. On package registries (npm, PyPI, RubyGems): alert on packages published by accounts with no prior publish history, packages with names closely resembling established libraries (typosquatting), and packages that include post-install scripts making outbound network calls. In CI/CD logs (GitHub Actions, JFrog Artifactory): search for unexpected HTTP/S requests to github.com from build runner processes during execution (T1071.001), environment variable reads accessing secrets stores during non-deployment phases (T1552.001), and creation or modification of scheduled tasks or launch agents (T1053.005, T1543.001). For credential access: alert on bulk reads of files matching patterns associated with SSH private keys, cloud provider credential files (.aws/credentials, kubeconfig), and API token stores (T1552.004). For AI tool configurations: monitor file system changes to known AI coding tool configuration paths (.cursorrules, Copilot workspace settings, Cline context directories) on developer workstations, unexpected modifications are a high-confidence indicator of compromise. Apply D3-SFA (System File Analysis) and D3-LAM (Local Account Monitoring) to developer endpoints. Behavioral baseline: flag any developer account that publishes a package to a public registry outside of an established, approved release process. NIST SI-4 (System Monitoring) should be the guiding control for continuous monitoring implementation across these surfaces.

## Indicators of Compromise

| Type | Value | Context | Confidence |
|---|---|---|---|
| DOMAIN | `github.com (used as C2 channel)` | Miasma uses GitHub as its command-and-control channel rather than dedicated infrastructure; outbound calls from CI/CD runners to GitHub outside of expected clone/fetch operations are suspicious | MEDIUM |

## Framework Mappings

### MITRE-ATTACK

- **T1078.004** — Cloud Accounts
- **T1543** — Create or Modify System Process
- **T1552.004** — Private Keys
- **T1059.007** — JavaScript
- **T1059** — Command and Scripting Interpreter
- **T1195.002** — Compromise Software Supply Chain
- **T1552.001** — Credentials In Files
- **T1543.004** — Launch Daemon
- **T1071.001** — Web Protocols
- **T1021.004** — SSH

- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1485** — Data Destruction
- **T1528** — Steal Application Access Token
- **T1027** — Obfuscated Files or Information
- **T1027.002** — Software Packing
- **T1543.001** — Launch Agent
- **T1199** — Trusted Relationship
- **T1608.001** — Upload Malware
- **T1053.005** — Scheduled Task

### NIST-800-53R5

- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-7** — Software, Firmware, and Information Integrity
- **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
- **16.10** — Apply Secure Design Principles in Application Architectures
- **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

### ISO-27001-2022

- **A.8.28** — Secure coding
- **A.8.8** — Management of technical vulnerabilities
- **A.5.34** — Privacy and protection of personal information

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

**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

## MITRE ATT&CK Mapping

| Technique ID | Technique Name | Tactic |
|---|---|---|
| T1078.004 | Cloud Accounts | Defense-Evasion |
| T1543 | Create or Modify System Process | Persistence |
| T1552.004 | Private Keys | Credential-Access |
| T1059.007 | JavaScript | Execution |
| T1059 | Command and Scripting Interpreter | Execution |
| T1195.002 | Compromise Software Supply Chain | Initial-Access |
| T1552.001 | Credentials In Files | Credential-Access |
| T1543.004 | Launch Daemon | Persistence |
| T1071.001 | Web Protocols | Command-And-Control |
| T1021.004 | SSH | Lateral-Movement |
| T1195.001 | Compromise Software Dependencies and Development Tools | Initial-Access |
| T1485 | Data Destruction | Impact |
| T1528 | Steal Application Access Token | Credential-Access |
| T1027 | Obfuscated Files or Information | Defense-Evasion |
| T1027.002 | Software Packing | Defense-Evasion |
| T1543.001 | Launch Agent | Persistence |
| T1199 | Trusted Relationship | Initial-Access |
| T1608.001 | Upload Malware | Resource-Development |
| T1053.005 | Scheduled Task | Execution |

## Sources

| Source | URL | Tier |
|---|---|---|
| **Security News** | https://www.bleepingcomputer.com/news/security/the-miasma-worm-sour... | T3 |
| **The 'Miasma' worm source code briefly leaked on GitHub - Reddit** | https://www.reddit.com/r/cybersecurity/comments/1u2swcj/the_miasma_... | T3 |
| **JFrog Security Research** | https://research.jfrog.com/ | T3 |
| **Miasma: Anatomy of an Open-Source Supply-Chain Worm - Ossprey** | https://www.ossprey.com/blog/miasma-anatomy-of-an-open-source-suppl... | T2 |
| **What security headaches has AI (Copilot, Cursor, Claude etc ...** | https://github.com/orgs/community/discussions/194034 | T3 |



Generated 2026-06-14 05:02 UTC by TJS Security Command Center