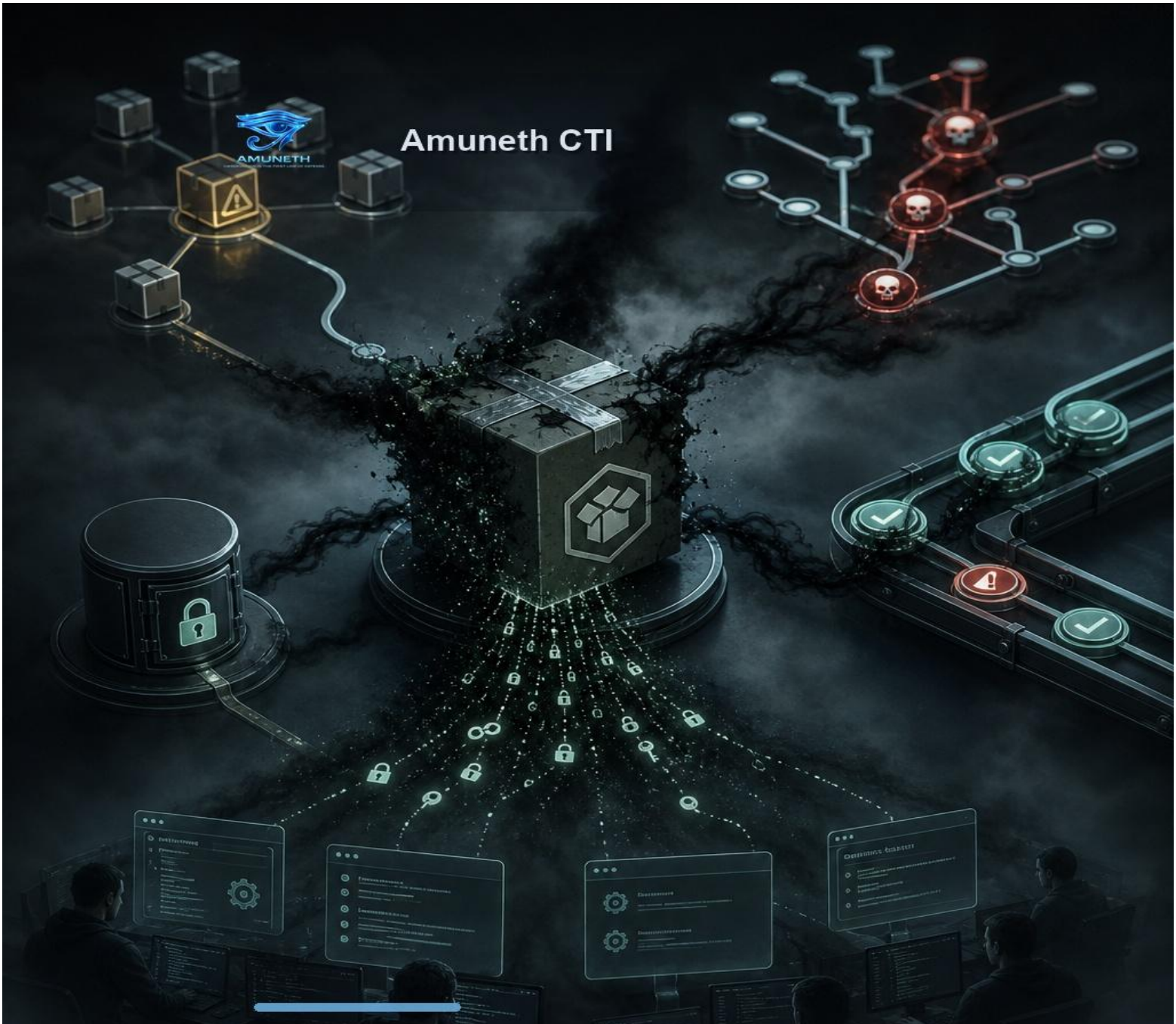

CTI-Miasma Supply Chain Worm

Erik Westhovens

June 2026

REPORT HEADER

Cyber Threat Intelligence Report

Subject: Credential theft, npm trusted-publishing abuse, CI/CD compromise, and AI coding-agent persistence

Audience: SOC, Incident Response, Threat Hunting, DevSecOps, Platform Engineering, Security Leadership

Date: June 2026

Author: Erik Westhovens

Management Summary

Miasma is a fast-moving software supply-chain worm derived from the Mini Shai-Hulud ecosystem. The campaign became prominent on June 1, 2026, when maliciously modified packages were published under the trusted @redhat-cloud-services npm namespace after a compromised GitHub account was used to push unauthorized commits into Red Hat GitHub repositories.

The campaign is strategically important because it abuses trust at multiple layers at once: npm package installation, GitHub Actions OIDC trusted publishing, provenance attestations, developer workstations, cloud credentials, and CI/CD runners. The result is not just package compromise, but a repeatable mechanism for credential harvesting and downstream propagation.

Microsoft reported 32 maliciously modified packages across more than 90 versions in the Red Hat scope. The trojanized packages executed through an npm preinstall hook, downloaded the Bun runtime, and launched a secondary payload targeting GitHub, npm, AWS, Azure, GCP, HashiCorp Vault, Kubernetes, SSH keys, CLI credentials, browser data, and CI/CD secrets.

Follow-on reporting indicates that Miasma evolved quickly. By June 4, Semgrep described a version that used a malicious binding.gyp file to trigger code execution during npm install without relying on obvious package.json lifecycle scripts. StepSecurity and SafeDep also documented expansion toward AI coding-agent configuration files and GitHub repository-level persistence.

Defenders should treat Miasma as an identity-driven supply-chain incident. Package removal is only the first step. Any host, developer machine, or CI/CD runner that installed affected versions must be scoped for credential exposure, token misuse, malicious repository changes, poisoned package publishing, and AI-agent persistence artifacts.

Key Takeaways

- Miasma turns trusted npm packages and CI/CD workflows into credential-theft and propagation infrastructure.
- Valid provenance does not guarantee safe content when the upstream identity and publishing workflow are compromised.
- The threat targets developer and pipeline identities, including GitHub, npm, cloud, Vault, Kubernetes, SSH, and local credential stores.

- The campaign evolved from preinstall hooks to binding.gyp execution and AI coding-agent configuration abuse within days.
- Incident response should include dependency scoping, full credential rotation, repository review, CI/CD audit, and developer workstation checks.

Chapter 1

1. Threat Overview

Miasma represents a rapid evolution of npm worm tradecraft from package poisoning to developer-toolchain compromise.

Miasma surfaced publicly on June 1, 2026 as a supply-chain compromise affecting packages under the @redhat-cloud-services npm namespace. Red Hat confirmed that a compromised GitHub account was used to inject malicious code into packages maintained in a Red Hat GitHub organization, and that compromised versions were removed from npm following disclosure.

Microsoft's analysis described a large-scale npm supply-chain attack affecting 32 maliciously modified packages across more than 90 versions. The packages were published through a legitimate GitHub Actions OIDC trusted-publishing workflow, which allowed the malicious releases to carry authentic provenance while embedding the campaign marker Miasma: The Spreading Blight.

The campaign matters because it uses normal developer and CI/CD workflows as the delivery layer. A routine npm install can become the execution point for credential theft, cloud access discovery, GitHub token theft, npm publishing abuse, and downstream package poisoning.

Assessment

- The primary risk is credential and trust-chain compromise, not only malicious code execution.
- Miasma benefits from legitimate package namespaces, CI/CD workflows, and provenance signals.
- Affected environments should be treated as exposed even if no endpoint malware alert fired.

Chapter 2

2. Campaign Profile and Lineage

Miasma is best understood as a Mini Shai-Hulud derivative adapted for active npm and AI developer-toolchain abuse.

Snyk, Microsoft, Semgrep, and other researchers linked Miasma to the Mini Shai-Hulud family. The code and behavior align with a self-propagating credential-stealing worm model: steal secrets, identify publishing rights or repository access, modify packages or repositories, and repeat the cycle through trusted developer infrastructure.

The Red Hat wave showed how a compromised account could become a publishing path. The attacker pushed unauthorized commits into RedHatInsights repositories, then used GitHub Actions OIDC trusted publishing to mint short-lived npm tokens and publish backdoored package versions under a legitimate namespace.

The campaign then evolved beyond the initial Red Hat scope. Reporting on June 3 and June 4 described additional npm waves, a binding.gyp-based execution method, and AI coding-agent configuration persistence. By June 5, StepSecurity reported repository-level activity in Microsoft Azure GitHub organizations, shifting part of the attack surface from package registries to code repositories opened by AI-enabled development tools.

Attribution Notes

- Miasma is a repeatable technique set, not just a single package incident.
- The tradecraft targets the trust relationship between maintainers, CI/CD, registries, repositories, and developer tools.
- Attribution should remain cautious; the defender priority is technique containment and credential scoping.

Chapter 3

3. Initial Access and Publishing Abuse

The Red Hat wave used a compromised GitHub account and trusted CI/CD publishing path to ship malicious packages.

Red Hat's bulletin states that initial investigation indicated a compromised GitHub account was used to push unauthorized commits to repositories in the RedHatInsights GitHub organization. Red Hat also stated that affected packages were frontend JavaScript libraries used by the Hybrid Cloud Console web interface, and that no Hybrid Cloud Console release was published during the compromise window.

Microsoft reported that the compromise originated from the RedHatInsights/javascript-clients CI/CD pipeline. The attacker used the legitimate GitHub Actions OIDC publishing workflow to publish trojanized packages, which meant the malicious artifacts could appear to have valid supply-chain provenance.

This is the central defensive lesson: provenance and trusted publishing reduce some risks, but they do not protect against a compromised upstream identity or malicious workflow commit. If an attacker can modify the workflow or repository that owns the publishing path, short-lived tokens and attestation can become part of the attacker-controlled release process.

Publishing Risk

- Review trusted publishing workflows, not only npm token storage.
- Monitor unauthorized commits, short-lived publishing branches, and workflow permission changes.
- Treat provenance as one signal, not proof that package contents are safe.

Chapter 4

4. Execution Chain and Payload

The first wave executed through npm preinstall hooks and launched a multi-stage JavaScript payload using Bun.

Microsoft described a heavily obfuscated dropper of approximately 4.29 MB that executed through an npm preinstall hook. The payload downloaded the Bun JavaScript runtime and launched a secondary payload tailored to credential harvesting and worm-like propagation.

The malware operated across Linux, macOS, and Windows by dynamically downloading the correct Bun runtime for the platform. Linux CI/CD runners appeared to be a primary target because they often hold high-value workflow tokens, cloud credentials, and repository permissions.

On developer systems, the payload targeted SSH keys, CLI credentials, browser data, wallet data, and local secrets. In CI/CD environments, it scraped GitHub Actions runner memory for secrets, attempted privilege escalation using passwordless sudo, and used stolen access to republish poisoned packages with forged or legitimate-looking provenance.

Execution Details

- npm install-time execution is enough to trigger compromise.
- CI/CD runners are high-value because they often combine code, cloud, and publishing authority.
- Bun download activity and large obfuscated JavaScript artifacts are useful hunt leads.

Chapter 5

5. Miasma v2 and Phantom Gyp

Miasma quickly moved from obvious lifecycle scripts to binding.gyp execution that can bypass simple package.json checks.

On June 4, 2026, Semgrep reported a Miasma v2 wave affecting 57 npm packages across more than 286 malicious versions. This version used a small malicious binding.gyp file rather than a package.json preinstall or postinstall script.

The binding.gyp technique matters because node-gyp can execute commands during native addon build preparation. Semgrep described a command-substitution action that runs node index.js while appearing as a build configuration file. This makes detection more difficult for teams that only flag package.json lifecycle scripts.

The v2 payload continued the same strategic objective: harvest cloud credentials, GitHub Actions secrets, password-manager stores, and AI assistant configuration files, then propagate through the supply chain. Chainguard and Semgrep both emphasized that dependency scanners and policy checks must inspect package contents, not only metadata.

Technique Shift

- Scan for suspicious binding.gyp files, especially command substitution patterns.
- Do not rely only on package.json lifecycle-script auditing.
- Treat unexpected root-level index.js payloads and node-gyp behavior as high-signal.

Chapter 6

6. AI Coding-Agent Persistence

Miasma also targets repository-level AI-agent configuration files as a persistence and execution surface.

SafeDep and StepSecurity reported that Miasma variants injected configuration files used by AI coding agents and IDEs. The strategic idea is simple: a repository can carry instructions or configuration that cause an AI-enabled development environment to execute attacker-controlled scripts when the repository is opened or used.

Semgrep listed persistence artifacts including .claude/setup.mjs, .claude/settings.json, .cursor/rules/setup.mdc, .gemini/settings.json, .vscode/tasks.json, .vscode/setup.mjs, and .github/setup.js. These files turn the repository itself into an execution and persistence surface beyond the npm package version that initially delivered the worm.

This changes remediation assumptions. Removing the infected package from node_modules is not enough if repository files, IDE tasks, AI-agent rules, or generated setup scripts remain in place. DevSecOps teams need policy and monitoring around agent configuration files in repositories, especially when AI tools are allowed to run commands or edit code autonomously.

AI Toolchain Risk

- Audit AI-agent and IDE configuration files in affected repositories.
- Review unexpected changes to .claude, .cursor, .gemini, .vscode, and .github paths.
- Treat AI coding-agent configuration as executable policy, not passive documentation.

Chapter 7

7. Impact and Business Risk

Miasma creates risk across development, cloud, CI/CD, and downstream customers through credential theft and propagation.

The immediate impact is exposure of secrets on any system that installed an affected package. This can include developer machines, build runners, ephemeral CI jobs, local test environments, and containerized build processes. The attacker does not need persistence on every endpoint if stolen credentials provide access to source code, cloud environments, or registries.

The business risk is broader than one poisoned dependency. Stolen GitHub tokens can alter repositories, create malicious workflows, or push additional package modifications. Stolen npm authority can publish compromised versions. Stolen cloud and Vault credentials can provide access to production infrastructure. Stolen SSH keys can enable additional lateral movement.

Red Hat stated that, based on its current findings, no customer action was required for Red Hat product use because no Hybrid Cloud Console release was published during the compromise window and affected packages are not related to several managed cloud services. Independent researchers still recommend that organizations that directly installed affected npm packages scope and rotate reachable credentials.

Risk Drivers

- Risk depends on whether affected packages were installed in developer or CI/CD environments.
- Credential exposure can outlive package removal.
- Downstream propagation risk depends on publishing rights held by compromised accounts.

Chapter 8

8. Detection and Threat Hunting

Hunting must cover dependency usage, package contents, process execution, network activity, repository changes, and credential misuse.

The first hunt question is whether any affected Miasma package versions were installed, directly or transitively, since June 1, 2026. Review package-lock.json, pnpm-lock.yaml, yarn.lock, npm cache, artifact repositories, container layers, CI/CD job logs, and developer build telemetry.

The second hunt question is whether the payload executed. Look for npm install events followed by Bun downloads, temporary Bun directories, large obfuscated JavaScript files, suspicious node or bun processes, outbound connections to GitHub APIs, unexpected repository creation or commits, and access to cloud metadata endpoints or credential stores.

The third hunt question is whether the attacker used stolen access. Review GitHub audit logs, npm publish history, new or modified GitHub Actions workflows, unusual OIDC token usage, package provenance, cloud API calls, Vault access, Kubernetes kubeconfig usage, SSH authentication, and AI-agent configuration file changes.

Hunt Leads

- Correlate package installation events with credential and repository activity.
- Scan for binding.gyp command substitution and unexpected root-level index.js payloads.
- Search repositories for AI-agent backdoor files and suspicious setup scripts.

Chapter 9

9. Response and Containment

Containment requires dependency removal, full credential rotation, repository cleanup, and CI/CD hardening.

If affected packages were installed, isolate the machine or runner where practical and preserve build logs, npm cache, lock files, package tarballs, process history, and network telemetry. Do not assume that deleting node_modules removes the incident.

Rotate credentials reachable from the affected environment. This includes GitHub tokens, npm tokens, cloud keys, OIDC trust relationships, Vault tokens, Kubernetes credentials, SSH keys, package registry credentials, CI/CD secrets, and any local CLI profiles present during execution.

Review repositories and pipelines for unauthorized commits, new branches, modified workflows, suspicious provenance attestations, unexpected package publishes, AI-agent configuration files, and setup scripts. Rebuild trusted dependencies from known-good sources and redeploy clean runners where possible.

Containment Checklist

- Scope installed versions before deciding response breadth.
- Rotate secrets from every environment where execution could have occurred.
- Revoke and reissue publishing authority after repository and workflow review.

Chapter 10

10. Recommendations and Outlook

Organizations should treat Miasma as a sign that developer identity and AI toolchain controls are now core supply-chain security requirements.

In the near term, enforce dependency allowlisting, lockfile review, package provenance validation, package-content scanning, npm lifecycle-script policy, node-gyp and binding.gyp inspection, and CI/CD egress monitoring. Pin GitHub Actions to commit SHAs where possible and review workflow permissions such as id-token: write and contents: write.

For identity, reduce long-lived developer and automation tokens, scope package publishing rights, require phishing-resistant MFA, monitor infostealer-exposed credentials, and alert on unusual npm publishing or GitHub repository changes. Where OIDC trusted publishing is used, protect the upstream repository and workflow path as a privileged release boundary.

For AI coding tools, inventory agent usage, restrict command execution, review repository configuration files, and prevent untrusted repository instructions from automatically executing local scripts. The likely outlook is that Miasma-style tradecraft will continue to blend package registry compromise, repository manipulation, and AI-agent persistence because each surface reinforces the others.

Recommended Actions

- Move from package metadata scanning to package-content and runtime behavior monitoring.
- Treat CI/CD workflows and AI-agent configuration as privileged execution surfaces.
- Prepare incident playbooks for install-time credential exposure and self-propagating developer-toolchain worms.

SOC Reference

Detection and Response Matrix

Use this matrix as a starting point for SOC, DevSecOps, and platform-engineering response to Miasma-style npm supply-chain compromise.

How to use this reference

- Start with dependency exposure, then determine whether the malicious payload executed.
- Treat developer machines and CI/CD runners as identity-rich environments.
- Prioritize credential rotation and publishing-right review over package removal alone.

Area	Signal	Action
Dependencies	Affected @redhat-cloud-services package versions or Miasma v2 packages in lockfiles, caches, or builds.	Identify installation scope, preserve artifacts, remove affected versions, and rebuild from trusted sources.
Execution	npm install followed by obfuscated index.js, Bun download, /tmp/b-* artifacts, or node-gyp activity.	Isolate host or runner, collect telemetry, and determine reachable credentials.
GitHub	Unauthorized commits, workflow changes, OIDC publishing activity, new branches, or suspicious repository writes.	Revoke tokens, review audit logs, remove malicious commits, and revalidate publishing workflows.
npm	Unexpected package publish, valid provenance for unexpected versions, or maintainer token misuse.	Yank malicious versions, rotate npm credentials, and restrict publishing permissions.
Cloud and Vault	AWS, Azure, GCP, Vault, Kubernetes, or SSH credential access after installation.	Rotate secrets, review cloud audit logs, and investigate lateral movement.
AI Toolchain	Unexpected .claude, .cursor, .gemini, .vscode, or .github setup files.	Remove persistence artifacts, block untrusted agent execution, and audit developer environments.

Matrix entries should be translated into environment-specific detections across SIEM, EDR, CI/CD telemetry, GitHub audit logs, cloud audit logs, package registry logs, and source-code scanning.

Reference Material

Sources

Microsoft Security Blog - Preinstall to persistence: Inside the Red Hat npm Miasma credential-stealing campaign

https://www.microsoft.com/en-us/security/blog/2026/06/02/preinstall-persistence-inside-red-hat-npm-miasma-credential-stealing-campaign/

Published June 2, 2026. Source for Red Hat npm scope, CI/CD OIDC trusted publishing abuse, preinstall execution, Bun runtime use, credential targets, and propagation behavior.

Red Hat Customer Portal - RHSB-2026-006 Supply chain compromise of @redhat-cloud-services npm packages

https://access.redhat.com/security/vulnerabilities/RHSB-2026-006

Created June 1, 2026 and updated June 3, 2026. Source for Red Hat's official executive and technical summary, product-impact statements, and ongoing investigation status.

Snyk - Miasma supply chain attack: malicious code found in @redhat-cloud-services npm packages

https://snyk.io/blog/miasma-supply-chain-attack-malicious-code-redhat-cloud-services-npm-packages/

Published June 1, 2026. Source for package namespace impact, preinstall behavior, credential-harvesting risk, and Mini Shai-Hulud lineage.

Semgrep - Miasma v2: Self-Spreading npm Worm Now Uses Malicious binding.gyp file and Compromises 57 Packages

https://semgrep.dev/blog/2026/miasma-v2-self-spreading-npm-worm-now-uses-malicious-bindinggyp-file-and-compromises-57-packages/

Published June 4, 2026. Source for the binding.gyp technique, affected scale, v2 execution path, and AI assistant persistence artifacts.

StepSecurity - Miasma Worm Hits Microsoft Again: Azure Functions Action and 72 Other Repositories Disabled

https://www.stepsecurity.io/blog/miasma-worm-hits-microsoft-again-azure-functions-action-and-72-other-repositories-disabled-after-supply-chain-attack-targeting-ai-coding-agents

Published June 5, 2026. Source for repository-level activity targeting AI coding-agent workflows and the shift from package registry poisoning to repository-based execution.

SafeDep - Miasma Worm Targets AI Coding Agents via GitHub Repos

https://safedep.io/miasma-worm-ai-coding-agent-config-injection/

Published June 2026. Source for AI coding-agent configuration injection and repository-level prompt/execution risk.

Wiz - Miasma: Supply Chain Attack Targeting RedHat npm Packages

https://www.wiz.io/blog/miasma-supply-chain-attack-targeting-redhat-npm-packages

Published June 1, 2026. Source for initial identification, affected namespace, weekly-download estimate, and investigation updates.

Partner Profile

About Amuneth

Amuneth provides intelligence-led security operations designed to help organizations detect, investigate, and contain modern threats across identity, endpoint, network, cloud, and software supply-chain environments. Our reporting bridges executive decision-making and operational action by combining threat context, analyst judgment, and practical detection guidance.

For CTI consumers, this final page gives reusable context on the operating model behind the report and clarifies how Amuneth supports ongoing monitoring, escalation, and proactive hunting beyond the specific topic covered in the report.

Security Operations Center

- Continuous monitoring across endpoint, identity, cloud, CI/CD, repository, and network telemetry.
- Triage and escalation workflows that prioritize verified risk over raw alert volume.
- Operational coordination with customer stakeholders during incidents, containment, and follow-up investigation.

Threat Hunting Capabilities

- Hypothesis-driven hunting for identity abuse, credential theft, stealthy persistence, lateral movement, and data exfiltration patterns.
- Targeted hunts based on current CTI findings, campaign tradecraft, and environment-specific risk signals.
- Cross-source correlation between endpoint, Microsoft 365, Entra ID, GitHub, CI/CD, cloud, firewall, proxy, and package-management telemetry.

CTI and Reporting Approach

Amuneth reports are built to support both management and frontline defenders. Each report explains why the development matters, how the threat behaves, where detection opportunities exist, and which defensive actions should be prioritized next.

This standardized format keeps reporting visually consistent while making room for actor-specific findings, malware analysis, campaign tradecraft, and strategic risk interpretation.