



Heyman: AI-Powered ASPM Agent for Software Supply Chain Security

INVINCIBLE PRODUCT SECURITY

Additional information is available at <https://scribesecurity.com/>

Heyman: AI-Powered ASPM Agent for Software Supply Chain Security

Executive Summary

Heyman by Scribe Security is an AI-driven Application Security Posture Management (ASPM) agent that serves as a *virtual AppSec co-pilot* for organizations

In a market where typical ASPM tools merely aggregate scanner outputs and enforce static policies, Heyman occupies a unique position by leveraging deep, contextual data from the entire software supply chain. It integrates tightly with ScribeHub – Scribe’s continuous assurance platform – including a secure attestation lake of signed SDLC evidence and a pipeline lineage graph, to provide intelligence that traditional AppSec solutions cannot match. This enables Heyman to *detect and prioritize vulnerabilities based on your specific context* rather than generic risk scores, and to pinpoint root causes across complex CI/CD pipelines. Heyman helps organizations catch and address issues early in development (preventing costly late-stage fixes), continuously monitor software integrity, and maintain compliance. By analyzing a breadth of SDLC evidence that ScribeHub generates and collects (SBOMs, context, signatures, configurations, code provenance, build artifacts, etc.) and correlating it with known threats and up-to-date available mitigation and risk information, Heyman provides unparalleled insight into your product security posture. Just as importantly, it drives automation: the agent not only flags risks but also initiates remediation workflows – for example, tickets in Jira.

Teams interact with Heyman through natural language (via chat interfaces similar to ChatGPT) as they would with a human security engineer, dramatically reducing the need to dig through dashboards or logs. Heyman’s integration into developers’ daily

tools (e.g. Jira, VS Code*) means faster response times and better collaboration across DevSecOps. Overall, Heyman's unique combination of a rich security evidence backend and intelligent automation positions it as a cutting-edge solution that can secure an enterprise's SDLC without slowing down innovation.

Detailed Overview for AppSec and DevSecOps Teams

Core Capabilities and Features

- **Comprehensive Supply Chain Visibility:** Through accessing Scribe's *attestation lake* (Scribe's repository of security evidence) and SDLC BI infrastructure, Heyman provides visibility to the organization's SDLC platforms and assets, as well as to the software factory's processes - from code commit to deployment – including SBOMs, dependency data, build provenance records, artifact hashes/signatures, CI/CD configs, scan results, and more. This holistic visibility means no blind spots: vulnerabilities, misconfigurations, or policy violations are detected whether they reside in source code, open-source components, pipelines, or cloud infrastructure.
- **Intelligent Threat Detection & Prioritization:** Heyman analyzes the collected evidence to identify security threats and prioritize them based on real context and online intelligence. For example, it will evaluate vulnerability severity based on the published CVSS and EPSS score, but will also take into account severity evaluations by other vendors, business criticality, and the application's exposure. Heyman will also provide information about the potential exploitability of a vulnerability and the estimated complexity of handling it. This context-aware and open approach filters out noise, focuses teams on the most actionable risks first, enabling early detection of critical issues *before* they reach production, and improves the communication between the security teams and the development teams.

- Seamless Integration with Scribe's Attestation Lake: Heyman is natively integrated with ScribeHub, Scribe's secure data lake of attestations and security telemetry. It taps into signed evidence from source control, CI builds, artifact repositories, and deployment environments. All relevant security signals (e.g. SBOM entries, code integrity attestations, vulnerability scan findings, policy guardrail outputs) are available to the agent in real time. This deep integration means Heyman's analysis is always based on up-to-date, trustworthy data (cryptographically signed), yielding highly accurate assessments. The agent essentially acts as an intelligent front-end to your attestation lake – mining insights from a trove of evidence that would be impossible to manually sift through.
- Pipeline Lineage Graph & Root Cause Analysis: Scribe's platform links all artifacts and actions in a “code-to-production” lineage graph. Heyman leverages this to trace security issues to their origin. If a vulnerable library version is flagged in production, Heyman can identify the code responsible for it and the developers that are involved, and which other application releases are affected. If a CI/CD misconfiguration is found, the agent knows exactly which pipeline (and team) it relates to. In case of a compromise in the SDLC (e.g. build machine compromise), Heyman will provide the blast radius - pointing to artifacts that may have been affected, and not less important, to artifacts that do not require any further investigation. These capabilities provide root-cause and accountability data, in addition to visibility, saving AppSec engineers huge amounts of investigative time.
- Policy Enforcement and SDLC Guardrails: Beyond finding issues, Heyman works with Scribe's policy-as-code engine to enforce security *proactively*. It monitors compliance with frameworks like SSDF, SLSA, and your own internal policies. When violations occur (e.g. a deployment without a required security review, or use of an unvetted open-source component), Heyman can assist in understanding the risk, prioritization, and mitigating the violations.
- Streamlined Remediation Workflow: Heyman doesn't stop at reporting problems – it helps resolve them. The agent can create or update tickets in issue trackers

(Jira, Azure Boards, GitHub issues, etc.) whenever a significant vulnerability or non-compliance is detected*. The content of these tickets is AI-generated with context, so it includes a plain-language description of the issue, its potential impact, and recommended fix steps. For example, if a secret key was exposed in a repository, Heyman's Jira ticket might note the file and commit where it occurred, the risk of not rotating the key, and an instruction to revoke and replace it. This saves security engineers time and ensures developers get clear guidance immediately. As Heyman continues to develop, it will become a part of your team as a security expert and a project manager. For example, Heyman will be able to assign tasks and track them, based on ownership concluded from the SDLC, accelerating the remediation cycle dramatically.

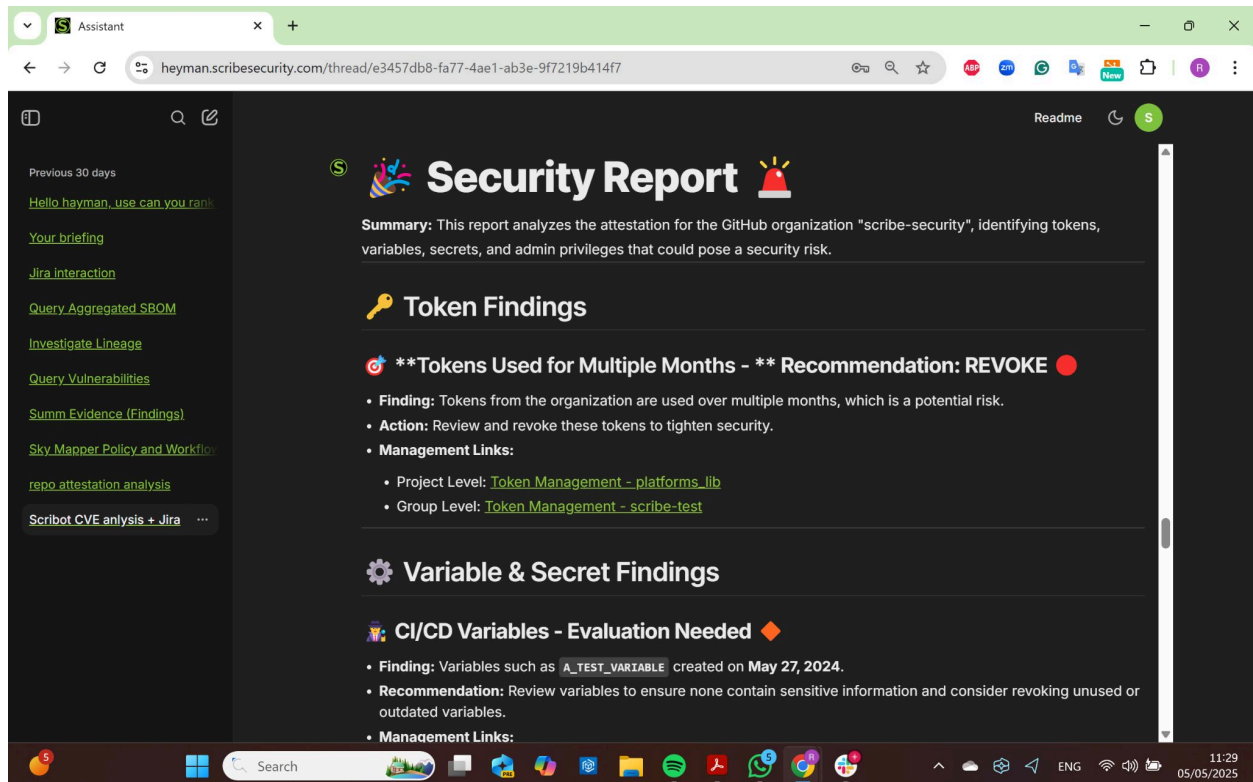
- **DevSecOps Co-Pilot (AI Assistant):** Heyman has an interactive chatbot interface. Accessible via web UI, it allows developers and security teams to *converse* with the security platform in natural language. Instead of navigating complex dashboards, a team member can ask, "Heyman, show me all high-severity vulns in the latest build of Project X" or "How was this container image built and who approved it?" and get an immediate, context-rich answer. Heyman responds like an expert colleague on demand. This co-pilot style means even non-security experts on the team can get actionable security insight just by asking, reducing the communication gap between security and development.

Deep Integration with ScribeHub's Attestation Lake

A key differentiator of Heyman is its attestation-driven intelligence. The "attestation lake" is Scribe's term for the central repository of all security evidence and metadata collected throughout the SDLC and the relationships between them, depicted in the evidence graph. This includes: Software Bill of Materials (SBOM) for every component and build, code provenance and build records (who built what, when, and how), artifact integrity proofs and signatures, AST scan results (SCA, SAST, DAST, secret scanning etc.), configuration baselines, CI/CD pipeline posture (settings, access controls), and any

policy violation logs. All evidence is cryptographically signed to prevent tampering. By having Heyman deeply integrated with this lake, the AI's knowledge spans the entire software factory, far beyond just code scanning.

Example: Heyman creates a security report directly from a GitHub organization posture attestation:

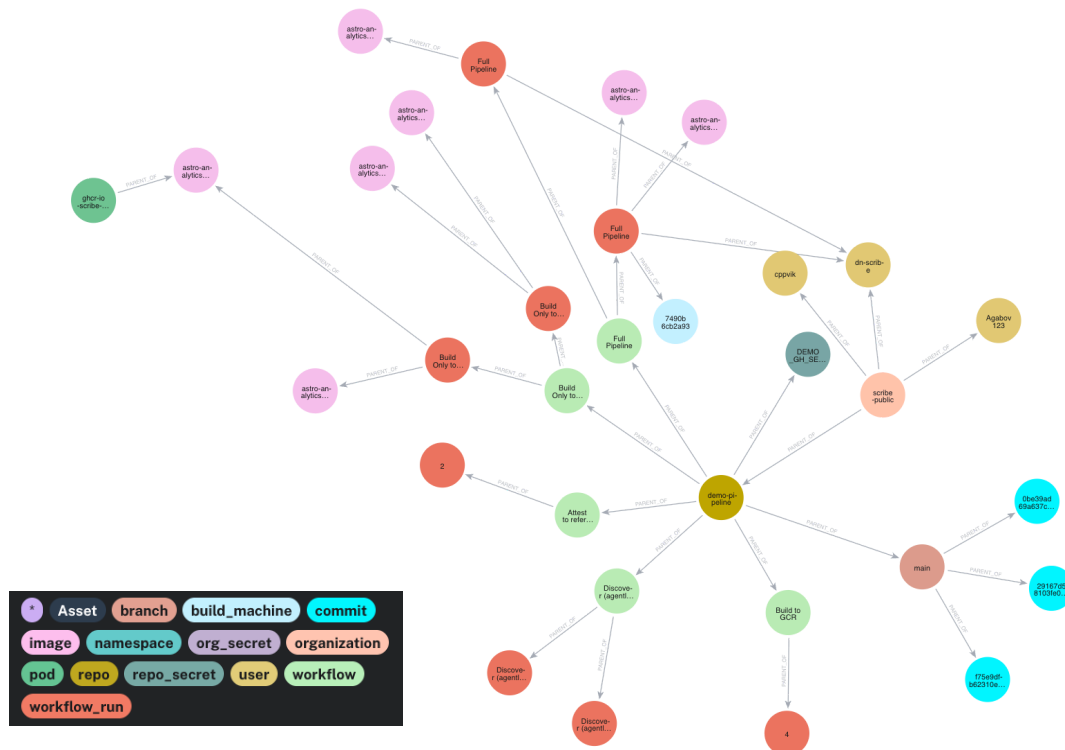


Pipeline Lineage Graph for Root Cause & Ownership Analysis

Modern software pipelines are complex, with many interconnected components and teams. When a security issue arises, understanding its origin is half the battle. Heyman addresses this via Scribe's pipeline lineage graph (also known as the evidence graph),

which maps relationships between code, builds, artifacts, deployments, and even developers. By querying this graph, Heyman can perform rapid root cause analysis.

Example: a visualization of a snippet of the lineage graph:



For instance, imagine a critical vulnerability is found in a production payload running in a specific k8s pod (dark green in the graph); Heyman can traverse the lineage graph to find the users (mellow yellow) involved, the code repo where the Dockerfile introducing the vulnerability is stored. This enables either a quick fix or quick access to the relevant developers.

Similarly, for CI/CD issues, if a pipeline fails a security policy (say, missing a code review step), Heyman uses lineage data to trace that back: which repository or project had the misconfiguration and who oversees it. By analyzing past information, it can even suggest the right person to assign for remediation. Ownership analysis is critical

for DevSecOps at scale, and this is a standout benefit of Heyman – it doesn't just tell you *what* is wrong, but often *who* should fix it and *how* it got to be that way.

Example: *Heyman analyzing pod images, pointing to an orphaned image - an image that is not tracked for vulnerabilities (misconfiguration):*

```
S Your query involves retrieving image assets for the product "Astro-Analytics-Discovery" with version "3.0", specifically to find:

1. All image assets that have a pod asset as a parent.
2. All image assets that have a parent of type workflow_run.
3. List the images that have a pod parent but do not have a workflow_run parent.

Results:

1. Images with Pod Parent:
  • astro-analytics
  • nginx

2. Images with Workflow Run Parent:
  • astro-analytics

Orphan Images:
The images that have a pod parent but do not have a workflow_run parent are:
  • nginx

Thus, the final result is:
  • Orphan Images: nginx
```

AI-Driven Virtual Security Engineer

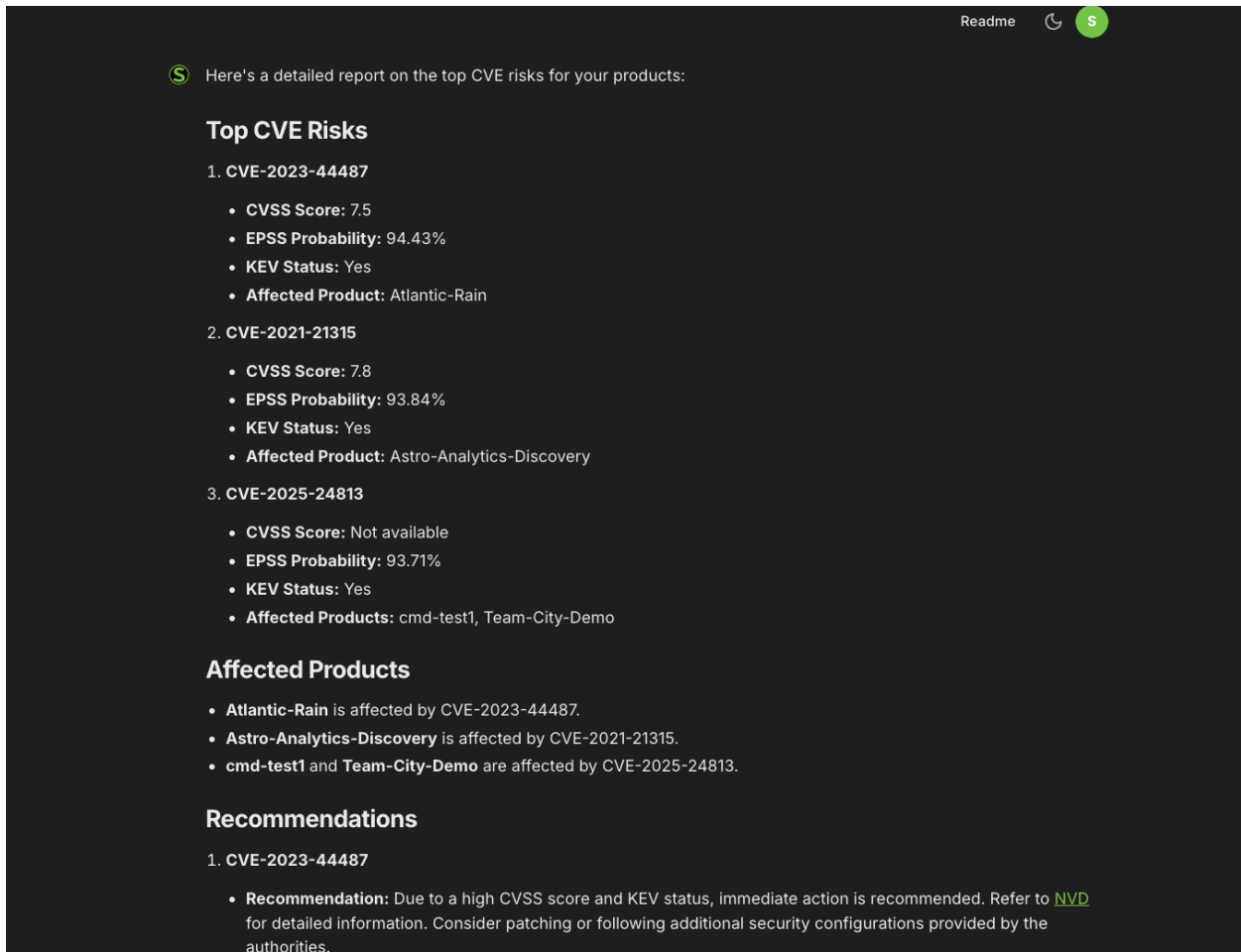
Heyman's conversational AI assistant behaves like a seasoned security engineer embedded in your DevSecOps team. It interprets questions or commands and generates helpful, context-aware responses. Some examples of how AppSec and development teams might use this virtual assistant include:

Context-Aware Explanations: A developer seeing a vulnerability alert can ask "*Heyman, how serious is this issue, and what do I need to do?*" The agent will respond with an explanation of the vulnerability's severity (e.g. "*High severity buffer overflow in library X, can lead to remote code execution*"), whether it's found in that team's specific project, and recommended actions (like patching to a safe version) drawn from both public data and the organization's context. This saves developers from having to research each

CVE – the knowledge base of Heyman (including external sources and internal data) is distilled into a concise answer

Example: *CVE Analysis by Heyman.*

First stage: point to prioritized vulnerabilities:



Readme S

S Here's a detailed report on the top CVE risks for your products:

Top CVE Risks

- CVE-2023-44487**
 - CVSS Score: 7.5
 - EPSS Probability: 94.43%
 - KEV Status: Yes
 - Affected Product: Atlantic-Rain
- CVE-2021-21315**
 - CVSS Score: 7.8
 - EPSS Probability: 93.84%
 - KEV Status: Yes
 - Affected Product: Astro-Analytics-Discovery
- CVE-2025-24813**
 - CVSS Score: Not available
 - EPSS Probability: 93.71%
 - KEV Status: Yes
 - Affected Products: cmd-test1, Team-City-Demo

Affected Products



- Atlantic-Rain is affected by CVE-2023-44487.
- Astro-Analytics-Discovery is affected by CVE-2021-21315.
- cmd-test1 and Team-City-Demo are affected by CVE-2025-24813.

Recommendations

- CVE-2023-44487**
 - Recommendation:** Due to a high CVSS score and KEV status, immediate action is recommended. Refer to [NVD](#) for detailed information. Consider patching or following additional security configurations provided by the authorities.

Second stage: deep analysis of a single vulnerability

- *Notice actionable information: Exploitability, known exploits, and upgrade risks that are crucial for effective prioritization and communication with developers.*

[Readme](#)



Exploitability:

- **How it can be exploited:** The vulnerability can be triggered via crafted input to the `patch` function via network operations, allowing an attacker to induce a double free and potentially lead to a crash or arbitrary code execution.
- **Exploitation prerequisites:** No authentication or user interaction is required, making it easier to exploit.
- **Checklist for assessment:**
 - Check if the version of GNU `patch` is 2.7.6 or earlier.
 - Evaluate the use of the `patch` utility in networked operations.
 - Assess whether the application takes untrusted input to the patching process.

Upgrade Risk:

- **Breaking changes:** Review upgrade notes for GNU `patch` to ensure compatibility with existing workflows.
- **User complaints:** Monitor discussions on platforms like StackOverflow or the GNU mailing lists about potential issues users face after upgrading.
- **Integration risks:** Confirm that dependent systems relying on previous versions don't break after the upgrade.
- **References:** Relevant discussions can often be found on [GNU bug trackers](#) or various forums.

Alternative Mitigations:

- **WAF/IDS rules:** Consider using Web Application Firewalls or Intrusion Detection Systems with rules to detect attempts to exploit this vulnerability.
- **Permission/config changes:** Limit the network exposure of systems running vulnerable versions.
- **Vendor suggestions:** Regularly review advice from GNU patches [advisories](#) for workload protection tips.

Detection / IoCs:

- **Indicators of Compromise:** Log entries indicating unusual `patch` command executions from untrusted sources.
- **Monitoring:** Set alerts for unexpected behavior from the `patch` command processes. Specific log entries to watch for include:
 - Segmentation faults related to the `patch` process.
- **Detection rules:** Consider creating Snort/Suricata signatures for known exploit patterns.

Known Exploits:

- No public exploits have been documented, and there are no known working Proofs of Concept (PoC) available on platforms like GitHub or ExploitDB.

Third stage: Guided Remediation and Best Practices: If a policy violation is flagged (for example, k8s manifest missing resource limitations), a team member can first get a clear understanding of it:

Evidence Summary:

The evidence type is a "Finding" statement which details the various security checks performed on the pod "ghcr-io-scribe-public-astro-analytics-11-pod" and the corresponding failures observed.

Top Findings:

Title	Severity	Description	References
Linux hardening	Low	Failed control for Linux hardening requirements.	Link
Non-root containers	Medium	The pod is not configured to run as a non-root user.	Link
Ingress and Egress blocked	Medium	Network policies allowing ingress and egress are not set.	Link
Immutable container filesystem	Low	Pod does not have a read-only root filesystem configuration.	Link
Allow privilege escalation	Medium	Pod settings allow privilege escalation.	Link
Ensure memory limits are set	High	Memory limits are not configured for the pod.	Link
Ensure CPU limits are set	High	CPU limits are not configured for the pod.	Link

Recommendations:

- **Overall Handling:** Conduct a thorough review of the findings, focusing on the medium and high severity issues, as they pose a significant risk to security and compliance.
- **Specific Recommendations:**
 - **For Critical Findings:** Immediately address the findings related to CPU and memory limits, as these are critical for application performance and resource control. Implement the recommended configurations to ensure these limits are set properly.
 - **For Medium Severity Findings:** Review and modify the pod's configuration to ensure it runs as a non-root container and restricts privilege escalation.
 - **For Low Severity Findings:** While these findings should be addressed, they can be prioritized after the medium and high severity issues are resolved. Implement recommended configurations for Linux hardening and a read-only root filesystem.

Then he can ask, "How do I fix that?" Heyman might respond with exact guidance:

Readme 5

S Based on your findings list, we can modify the Kubernetes YAML manifest to address the identified issues. Here's a revised version of the manifest with improvements based on the findings:

Revised Kubernetes YAML Manifest with Fixes

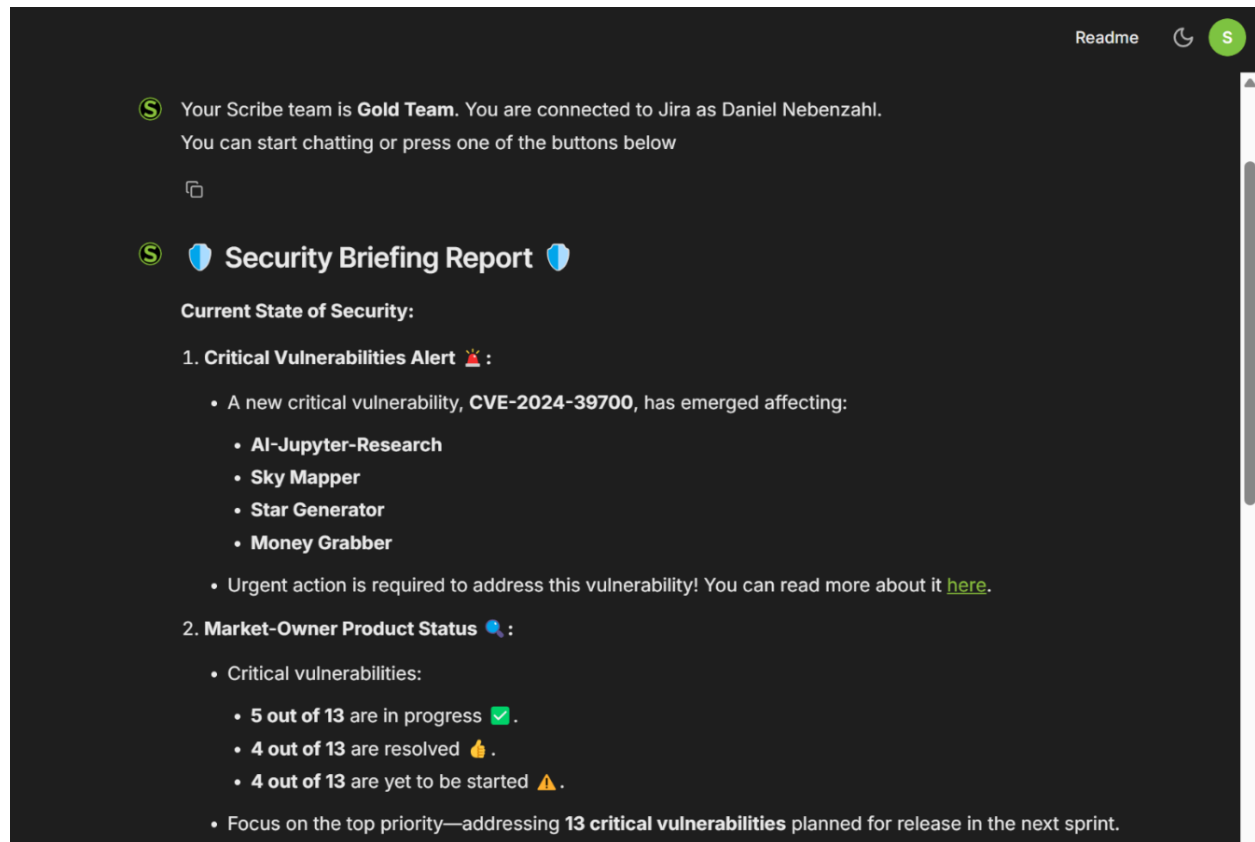
```

yml
apiVersion: v1
kind: Namespace
metadata:
  name: astro-analytics-discovery
  labels:
    kubernetes.io/metadata.name: astro-analytics-discovery
---
apiVersion: v1
kind: Pod
metadata:
  name: ghcr-io-scribe-public-astro-analytics-11-pod
  namespace: astro-analytics-discovery
  labels:
    run: ghcr-io-scribe-public-astro-analytics-11-pod
spec:
  containers:
    - name: ghcr-io-scribe-public-astro-analytics-11-pod
      image: ghcr.io/scribe-public/astro-analytics:11
      ports:
        - containerPort: 8080 # Replace with actual port if necessary
      volumeMounts:
        - name: kube-api-access-qpc2r
          mountPath: /var/run/secrets/kubernetes.io/serviceaccount
          readOnly: true
      securityContext:
        runAsUser: 1000 # Run as a non-root user
        runAsGroup: 1000
        allowPrivilegeEscalation: false # Prevent privilege escalation
        readOnlyRootFilesystem: true # Set root filesystem to read-only
      resources:
        requests:
          memory: "256Mi" # Minimum amount of memory the container will require
          cpu: "500m" # Minimum amount of CPU the container will require
        limits:
          memory: "512Mi" # Maximum amount of memory the container can use
          cpu: "1" # Maximum amount of CPU the container can use


```

Prioritization and Risk Communication: The AI agent helps translate raw security data into insights that management and engineers care about. It might summarize the day's findings: *"Daily Security Brief: 5 new vulnerabilities (1 critical in Service X, 4 low), 2 policy violations resolved, 1 pending. Top recommendation: address the critical vuln in Service X (patch available)."* By prioritizing responses and focusing attention on the highest risks, Heyman ensures the team isn't overwhelmed. It effectively triages issues across the organization continuously.

Example: A “Good Morning Security Briefing”:



Integration and Automation Benefits

Heyman’s integration into the SDLC toolchain means it fits naturally into existing workflows. For example, as soon as a CI/CD pipeline finishes, Scribe’s attestation engine records all relevant data (build outputs, signatures, test results). Heyman can automatically evaluate that against policies – if everything looks good, perhaps it posts a “ All clear” message; if not, it might open an issue or alert the team with the specifics of what failed. Over time, organizations can even measure improvement (since Heyman can report metrics like “average time to remediate critical vulns” or “policy adherence rate”), showing the ROI of an AI-assisted security approach.

From a technology standpoint, Heyman is built on robust AI and analytics. But importantly, it is *not* a black box solely relying on LLM predictions – it's grounded in your actual security telemetry. This hybrid of AI + real data is what makes it trustworthy and enterprise-grade. Heyman's insights and suggestions are backed by the evidence collected by ScribeHub's collectors, so that users can drill into them in the ScribeHub interface to see more details (e.g., a recommendation to revoke a token comes with a link to the evidence of that token's age and usage). This transparency helps build trust in the recommendations and facilitates learning – developers not only get a fix, they can understand the why behind it.

Conclusion

Heyman represents a significant advancement in application security management. It uniquely fuses a comprehensive software supply chain security platform with the convenience and intelligence of an AI assistant. For AppSec and DevSecOps leaders, it means unprecedented insight and control over their SDLC risk posture, and the ability to respond to threats faster and smarter. For developers and engineers, it means security information and guidance is available on demand, in digestible form, and even taken care of automatically in many cases. With Heyman, organizations can achieve a stronger security posture without sacrificing speed or agility, effectively baking security into development and operations with the help of AI-driven automation.

Appendix - Main Capabilities

Heyman - Your AI Conversational Co-Pilot for Secure Software Supply Chains

Instant Context. Less Noise. Total Control.

DevSecOps teams drown in alert overload, manual ticketing burdens, and hidden supply-chain risks. Scribe Heyman delivers security intelligence and automated fixes via natural-language chat.

Pain Points Heyman Solves:

- **Alert Fatigue:** AppSec tools generate too many alerts
 - **Supply-Chain Blind Spots:** Unmonitored pipelines and misconfigurations can lead to risks in production
 - **Manual Remediation:** Engineers spend hours crafting, prioritizing, and validating tickets.
-

Heyman's AI-Powered Edge

1. Conversational Interface

- Ask in plain English: "Show me critical vulns in the production build."
- Ambiguous queries trigger clarifying prompts; all interactions are logged for audit.

2. Graph-Driven Visibility

- Live map of repos, pipelines, registries, admission controllers, and K8s clusters via ScribeHub's cryptographically-signed Attestation Lake—updated via API scans and optional CI/CD plugins.
- Trace any artifact from commit to production in seconds.

3. Evidence & Provenance Intelligence

- Ingests SBOMs, third-party SBOMs, scanner outputs, logs, and build metadata (environment variables, build IDs) for repos and container workloads.
- Detects version drift and missing SBOMs post-deployment; verifies cryptographic signatures and flags any tampering.

4. Policy-As-Code Enforcement

- Out-of-the-box support for Open Policy Agent (Rego) and custom YAML rules.
- Records a pass/fail record for every security check; auto-gates or triggers remediation on policy violations.

5. Real-Time Threat Prioritization

- Correlates public CVE feeds and exploit data with your environment context.
- Surfaces only high-confidence, actionable risks.

6. Automated Remediation Workflows

- Drafts and assigns Jira, GitHub, or Azure tickets with validated, code-specific fix guidance.
- Includes hyperlinks to the underlying attestation records from tickets and chat answers for full transparency.

7. Seamless Pipeline Integration

- Prebuilt connectors for GitHub, GitLab, Jenkins, Azure Pipelines, and more.
- Cuts configuration and deployment time of security integrations.

8. Automated Briefings & Reports

- Schedule daily, weekly, or custom security briefings via chat or email.
- Generate trend reports on vulnerabilities, policy compliance, and remediation metrics.

Practitioner Insights & Deployment

- **Scalable Graph Sync:** Incremental diffs and parallel API calls handle thousands of services in minutes.
- **BI & Adoption Metrics:** Leverages attestation and telemetry data to gauge adoption of security controls and DevSecOps frameworks across teams, projects, and production applications—identifying gaps and tracking continuous improvement.
- **IDE Integration:** VS Code plugin and CLI tools embed Heyman into developers' workflows for in-editor chat, insights, and alerts.
- **Drill-Down Evidence Links:** Every ticket and chat answer includes hyperlinks to the underlying attestation records, enabling security teams to verify and learn from the source data.
- **Audit & Compliance:** Full audit trail for all AI interactions, policy decisions, and remediation actions.
- **Compliance Alignment:** Meets SSDF, SLSA, NIST 800-218, and FedRAMP standards with cryptographically-signed evidence.