© Volume 2, Issue 11, November 2026 | JATIR
e-ISSN: 2839-5786, p-ISSN: 3139-8464

AetherShield: A Unified Multi-Cloud Security Posture Management Platform

[1]Sreegokul S, [2]Shyam Karthik R, [3]Tharun Kumar A
[1,2,3]Department of Computer Science and Engineering (Internet of Things and Cyber Security including Blockchain Technology) SNS College of Engineering, Coimbatore, India

***Abstract*—Enterprises operating across Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) face fragmented visibility into their security posture, as each provider exposes configuration, identity, and compliance data through separate consoles and inconsistent terminology. This paper presents AetherShield, a full-stack Cloud Security Posture Management (CSPM) dashboard that unifies asset inventory, compliance auditing, policy enforcement, and remediation workflows across simulated multi-cloud environments into a single interface. The system computes a live, weighted security score from active findings, supports one-click and AI-assisted remediation, and includes a Chaos Simulator module that injects synthetic threat events (e.g., ransomware activity, crypto mining pod injection) to validate the responsiveness of detection-to-remediation loops. AetherShield is implemented with a React (Vite) frontend and a modular Python FastAPI backend using lightweight HMAC-based authentication, and it demonstrates how CSPM principles — continuous monitoring, automated remediation, and compliance mapping — can be realized in a compact, extensible architecture suitable for demonstrating enterprise-grade security workflows.**

***Index Terms*—Cloud Security Posture Management, Multi-Cloud Security, Compliance Automation, Security Remediation, Chaos Engineering, AI Copilot**

I. INTRODUCTION

Cloud Security Posture Management (CSPM) has emerged as a foundational discipline for organizations operating in public cloud environments, providing continuous assessment of configurations, identification of misconfigurations, and mapping of controls to regulatory frameworks such as GDPR, HIPAA, and SOC 2 [1]. As organizations increasingly distribute workloads across AWS, Azure, and GCP to gain resilience and flexibility, a new problem surfaces: each provider enforces its own identity model, resource-naming convention, and console, so security teams must manually reconcile findings from separate, disconnected tools [2].

This fragmentation has two practical consequences. First, administrators lose a single, trustworthy view of overall risk, since a role that is tightly scoped in one provider may be defined far more permissively in another purely because of naming or structural differences. Second, incident correlation across clouds becomes a manual, slow process of cross-referencing logs and alerts, which delays response and increases the likelihood that a compounding attack across providers goes unnoticed until damage has occurred.

The consequences of this fragmentation are not merely operational inconveniences; they translate directly into measurable security risk. A misconfigured storage bucket in one provider and an over-privileged service account in another may, individually, appear low-priority when viewed through separate consoles, yet together they can represent a viable attack chain that no single-provider tool is positioned to flag. Most cloud-native monitoring tools are also designed around the assumption of trust in the provider's own control plane: they tell an administrator what has changed within that provider, but are largely silent about how a change in one cloud might interact with, or be masked by, activity in another.

AetherShield addresses this gap as a demonstrative, full-stack CSPM dashboard. Rather than integrating with live cloud credentials, it simulates monitoring data across AWS, Azure, and GCP to model how a unified posture-management platform should behave: aggregating asset risk into a single score, mapping findings to compliance frameworks, enforcing policy rules with optional auto-remediation, and allowing both manual and AI-assisted remediation of active findings. A Chaos Simulator module additionally allows a user to inject realistic attack scenarios and observe how the platform's scoring and remediation loop responds in real time. This design choice — favouring a fully mocked but functionally complete data layer over partial, credential-gated integrations — allows every module of the system, from the scoring engine to the AI Copilot's remediation calls, to be inspected, tested, and demonstrated end-to-end without depending on the availability or configuration of external cloud accounts.

The motivation for this work is therefore twofold. First, it aims to demonstrate, in a compact and inspectable codebase, how the core mechanics of a modern CSPM platform — normalization of heterogeneous asset data, weighted risk scoring, policy-driven remediation, and compliance mapping — fit together as a coherent system rather than as isolated features described abstractly in vendor literature. Second, it aims to provide a reproducible testbed, through the Chaos Simulator, for observing how quickly and correctly a posture-management platform's detection-to-remediation loop responds to a realistic, multi-stage threat, which is a property that is difficult to evaluate against production tools without live incidents.

The remainder of this paper is organized as follows. Section II surveys existing CSPM tooling and related work. Section III describes the system architecture, including the frontend, backend, and data model. Section IV details the implementation and core algorithms, including the scoring model, policy engine, authentication scheme, and Chaos Simulator. Section V presents results, evaluation, and a feature comparison against existing tools. Section VI discusses limitations, and Section VII concludes with directions for future work.

II. RELATED WORK

Commercial CSPM tooling has largely evolved within single-provider ecosystems. AWS Config continuously tracks resource configuration changes and evaluates them against defined rules, integrating tightly with AWS Security Hub and GuardDuty, but offers limited native support once resources span outside AWS [3]. Azure Security Center (now part of Microsoft Defender for Cloud) unifies visibility across Azure and on-premises assets and provides threat intelligence and continuous compliance assessment, but is similarly Azure-centric [4]. Google Cloud's Security Command Center follows the same pattern for GCP-native workloads.

Third-party multi-cloud CSPM vendors (e.g., Wiz, Prisma Cloud, Orca Security) attempt to bridge this gap by aggregating data from multiple provider APIs into a single console, and research on CSPM adoption confirms that automated misconfiguration detection meaningfully reduces the incidence of cloud security breaches compared to manual auditing [5]. The Cloud Security Alliance's guidance on critical focus areas for cloud computing emphasizes that continuous compliance monitoring and identity governance are among the highest-leverage controls an organization can automate [6], which directly informs the feature set prioritized in AetherShield's design.

However, published literature and product documentation for third-party CSPM tools rarely detail how remediation workflows are surfaced to operators, or how a unified risk score should be computed from heterogeneous, cross-provider findings — both of which are central design questions this project explores directly in a compact, inspectable implementation. Chen and Wei's study of automated misconfiguration detection in public cloud infrastructure shows that automated detection pipelines substantially outperform manual audits in both detection latency and coverage, but stops short of examining how a resulting risk score should be communicated to an operator in a way that supports fast, confident decision-making [7].

Separately, research on AI-assisted operations tooling has shown that natural-language assistants can reduce the cognitive load of triaging alerts by summarizing context and suggesting next steps. Gupta and Sharma's survey of emerging trends in CSPM identifies AI-assisted remediation guidance and automated compliance mapping as fast-growing directions in the field, but note that most current implementations treat the assistant as an advisory layer disconnected from the platform's actual remediation actions [9]. AetherShield's AI Copilot is designed to close this gap by wiring its quick actions directly into the same remediation function used by the rest of the application.

AetherShield differs from the tools and studies above by prioritizing an end-to-end, explainable pipeline over provider-API breadth: every module, from scoring to the AI copilot's remediation calls, is implemented and visible within a single codebase, making the system suitable for studying CSPM mechanics rather than only consuming them as a managed service. This is a deliberate scoping decision — the goal is not to compete with production-grade, credentialed integrations, but to provide a transparent reference implementation of the workflow those tools abstract away.

III. SYSTEM ARCHITECTURE

AetherShield follows a client-server architecture with a clear separation between presentation and domain logic. The frontend is responsible only for rendering state and dispatching user actions; all scoring, rule evaluation, and remediation logic is intended to live behind well-defined API boundaries on the backend, so that the system could later be pointed at real cloud telemetry without requiring a frontend rewrite.

A. Frontend

The frontend is built with React and Vite, and is organized into five primary views: Dashboard, Asset Inventory,

Compliance Audits, Policy Rules, and AI Copilot. Navigation is handled through a persistent left sidebar so that an operator can move between views without losing context, and a top bar exposes a global resource search, a live "Monitors Operational" status indicator, notifications, and the Chaos Simulator toggle.

A shared AppStateContext propagates state across all pages, so that a remediation performed on one screen (e.g., Asset Inventory) is immediately reflected on another (e.g., the Dashboard's security score and findings count) without a page reload. This was a deliberate architectural choice: rather than having each page independently fetch and cache its own copy of asset and finding data, a single source of truth is held at the application root and subscribed to by every page, which eliminates an entire class of state-synchronization bugs that would otherwise be likely in a dashboard with this many interconnected views.

A collapsible Chaos Simulator panel, toggled from the lightning-bolt icon in the top bar, is available across every page and allows threat scenarios to be triggered contextually, without navigating away from whatever view the operator is currently using. This persistent availability was intended to mirror how, in a real incident, an operator would want to test the platform's response without losing their place in an ongoing investigation.

B. Backend

The backend is implemented in Python using FastAPI, organized into single-responsibility modules: data.py (mock multi-cloud dataset), schemas.py (Pydantic models), scoring.py (security score computation), rule_engine.py (policy evaluation), auth.py (HMAC-based token issuance and verification), assets.py, policy_rules.py, compliance.py, dashboard.py, and remediation.py (route handlers per domain), chaos.py (threat-injection endpoints), feed.py (live intelligence feed), and copilot.py (AI Copilot backend logic). main.py wires these routers together and configures CORS so the Vite development server can call the API directly during development.

This module-per-concern layout was chosen specifically so that each file maps to a single, testable responsibility: a developer extending compliance mapping only needs to read compliance.py and schemas.py, without needing to understand remediation or chaos-injection logic. All endpoints are documented and testable via FastAPI's auto-generated Swagger UI at /docs, which was used

throughout development to exercise each route independently of the frontend before wiring it into the UI.

Authentication uses a lightweight HMAC-signed token scheme rather than an external JWT library. A token is constructed from a payload (subject, issue time, expiry) and a signature computed with a server-held secret key; on each protected request, the backend recomputes the signature over the received payload and rejects the request if it does not match, or if the token has expired. This preserves the core properties needed for a stateless, verifiable session — tamper resistance and expiry — while avoiding an external dependency and keeping the authentication logic small enough to read and reason about in a single file.

C. Data Model

Cloud resources are modelled uniformly regardless of provider, each carrying a resource name, cloud provider tag (AWS, Azure, or GCP), service type, status (Healthy, Warning, Critical/Danger), and a details string describing the specific misconfiguration (e.g., public read access on a storage bucket, an open SSH port, an over-privileged IAM role). This normalization is what allows the Dashboard and Asset Inventory to present a single, filterable table across all three providers, and is the same principle that underlies how commercial multi-cloud CSPM tools reconcile provider-specific resource types into a common schema.

Findings are similarly normalized into a severity taxonomy of Critical, Warning, and Healthy, which is what allows the scoring engine (Section IV-A) to compute a single number from resources that originate from three structurally different provider APIs. Policy rules and compliance controls are modelled as separate entities that reference this same finding taxonomy, so that a single misconfiguration can simultaneously affect the live security score, appear in the relevant compliance framework's pass/fail count, and be matched against an applicable policy rule for possible auto-remediation. A typical remediation request flows from a UI trigger (a Remediate button, a Copilot quick action, or an auto-remediation from the rule engine) through the shared remediate Asset() function to an authenticated backend route, which updates the resource's status, recomputes the score, and returns both — after which AppStateContext re-renders every subscribed view consistently in the same update cycle.

IV. METHODOLOGY AND IMPLEMENTATION

A. Security Scoring Model

AetherShield computes a live security score from the set of currently active findings using a weighted-penalty formula: the average score begins at a maximum baseline and is reduced by 11 points for every active Critical finding and 4 points for every active Warning finding. This asymmetric weighting reflects the disproportionate risk contribution of critical misconfigurations (e.g., publicly exposed storage or open management ports) relative to lower-severity warnings, and was tuned so that a small number of critical findings can visibly and meaningfully move the score, rather than being diluted across a large asset inventory. The score recalculates immediately

whenever a finding is remediated or a chaos-simulated threat event is injected, giving the operator continuous, real-time feedback rather than a static, periodically refreshed metric.

This approach deliberately favors interpretability over statistical sophistication: an operator can look at the active findings list, count the Critical and Warning entries, and manually reproduce the displayed score, which is not generally possible with the proprietary, multi-factor risk-scoring models used by some commercial CSPM platforms. This transparency was considered important for a system whose purpose is partly educational — to make the mechanics of posture scoring legible rather than opaque.

B. Policy Rules and Auto-Remediation

Each policy rule (e.g., "Enforce KMS Storage Encryption," "Block Public Storage Buckets," "Restrict Public SSH/RDP") is defined with a category, applicable provider scope, and severity, and can be independently toggled Active and Auto-Remediate, as shown in Fig. 3. When Auto-Remediate is enabled, the rule engine resolves matching findings without operator intervention; otherwise, the finding surfaces for manual or AI-assisted review. This mirrors how production CSPM tools distinguish between advisory and enforced controls, and allows an organization to phase in automation gradually — starting with detection-only visibility for a new rule category before trusting it to remediate autonomously.

Rules are also scoped by category (e.g., Data Protection, Network Security) and by cloud provider, so that a rule intended only for AWS storage buckets does not attempt to evaluate an unrelated Azure or GCP resource.

C. Remediation Workflows

Remediation can be triggered from three surfaces: inline Remediate buttons in the Asset Inventory table, a Priority Findings card on the Dashboard highlighting the highest-severity active issues, and the AI Copilot's Quick Actions panel. All three surfaces call a single shared remediateAsset() function in the frontend state layer, ensuring consistent behaviour and preventing state drift between views. Centralizing this call also simplifies future extension: adding a fourth remediation entry point (for example, a bulk-remediate action) requires only wiring a new UI trigger to the existing function rather than duplicating remediation logic.

D. AI Security Remediation Copilot

The AI Copilot page (Fig. 4) exposes a conversational interface alongside a Quick Actions & Audits panel and an Active Vulnerability Context panel. Quick actions such as "Fix GCP temp-consultant-sa Owner" or "Remediate AWS open SSH Port 22" are not merely suggestions — selecting one invokes the same remediateAsset() call used elsewhere in the application, so the assistant's actions have real, observable effects on the live security score. The Active Vulnerability Context panel keeps the currently relevant findings visible alongside the conversation, so that a remediation suggestion can be evaluated against the specific resource it concerns without the operator needing to switch to the Asset Inventory page.

E. Chaos Simulator

To validate that the detection-to-remediation loop behaves correctly under stress, AetherShield includes a Chaos Simulator that can inject two representative scenarios: an SQL Database Ransomware Threat (Azure), modeled as an external IP triggering mass data exports followed by unauthorized ALTER TABLE encryption actions, and a Cryptomining Node Injection (GCP), modelled as an unauthorized pod consuming 98% CPU while communicating with public mining pools. Triggering either scenario immediately raises Active Threats and Critical Findings counters and lowers the live security score, allowing the remediation workflow to be exercised end-to-end without live infrastructure.

Each chaos scenario is written as a short narrative description alongside a "Trigger Threat Event" button, so that the simulator doubles as a lightweight training aid: an operator unfamiliar with what a cryptomining pod injection looks like in practice can read the scenario description, trigger it, and observe its effect on the dashboard in a single, low-stakes interaction. This was intended to make the platform useful not only for testing remediation logic but also for building operator intuition about how specific attack patterns manifest in monitoring data.

F. Compliance Mapping

Active findings are additionally mapped to four compliance frameworks — CIS Benchmarks v1.4, SOC 2 Type II, HIPAA Security Rule, and GDPR Privacy Shield — with each framework's Compliance Audits card (Fig. 5) reporting the number of passing and failing controls and an overall percentage score. This mapping allows a single misconfiguration, such as a publicly readable storage bucket, to simultaneously reduce the live security score, appear as an active Critical finding in Asset Inventory, and contribute to a failed control count in one or more relevant frameworks, illustrating how a single technical fix can resolve compliance exposure across several regulatory obligations at once.

V. RESULTS AND EVALUATION

A. Dashboard and Asset Inventory

The implemented dashboard (Fig. 1) surfaces four headline metrics — Average Security Score, Active Threats, Critical Findings, and Auto-Remediations performed — alongside a live threat-intelligence feed. In the demonstrated state, the platform reports a 59% average security score with 3 active threats, 3 critical findings, and 12 completed auto-remediations, reflecting a moderately degraded posture prior to manual intervention. This baseline state was chosen deliberately for demonstration purposes so that the effect of subsequent remediation actions on the score is clearly visible.

The Asset Inventory view (Fig. 2) lists individual resources with provider, service type, and status; for example, an AWS S3 bucket (prod-billing-invoices) flagged Danger for public read access via bucket ACL, and an AWS security group (web-server-secgroup) flagged for exposing port 22 to 0.0.0.0/0. Each is independently filterable by platform (All Platforms, AWS Only, Azure Only,

Google Cloud) and by severity (All Severities, Critical, Warning, Healthy), and each row is independently remediable, so an operator triaging a large inventory can narrow the view to, for example, only Critical AWS findings before beginning remediation.

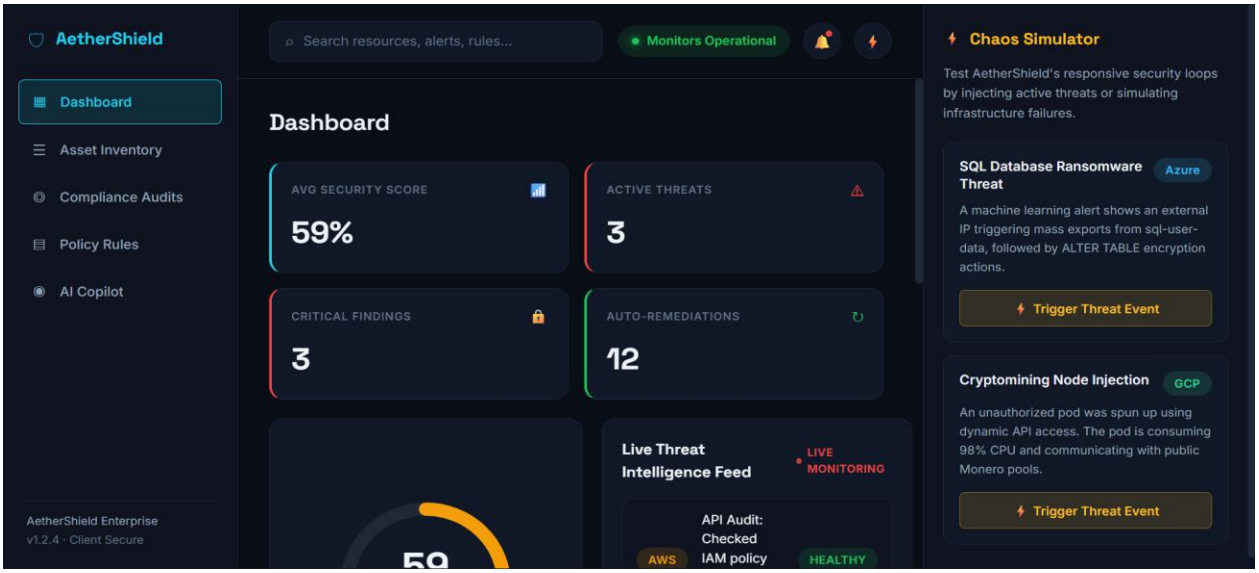


Fig. 1. AetherShield Dashboard showing live security score, active threats, and threat intelligence feed.

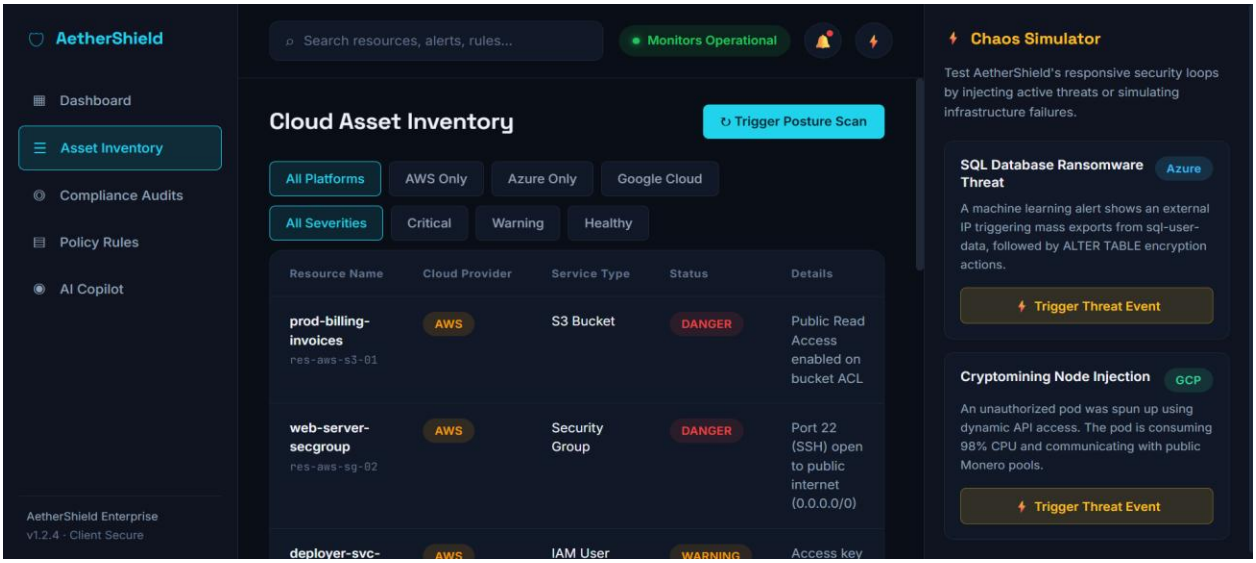


Fig. 2. Cloud Asset Inventory with cross-provider filtering and per-resource remediation.

B. Policy Rules and Compliance Audits

The Policy Rules view (Fig. 3) lists each rule with its category, provider scope, and severity badge, alongside independent Active and Auto-Remediate toggles. In the demonstrated configuration, "Block Public Storage Buckets" and "Restrict Public SSH/RDP" are both marked Critical severity and are shown with Auto-Remediate enabled, while lower-urgency rules such as "Enforce KMS Storage Encryption" are Active but left for manual review, illustrating the graduated trust model described in Section IV-B.

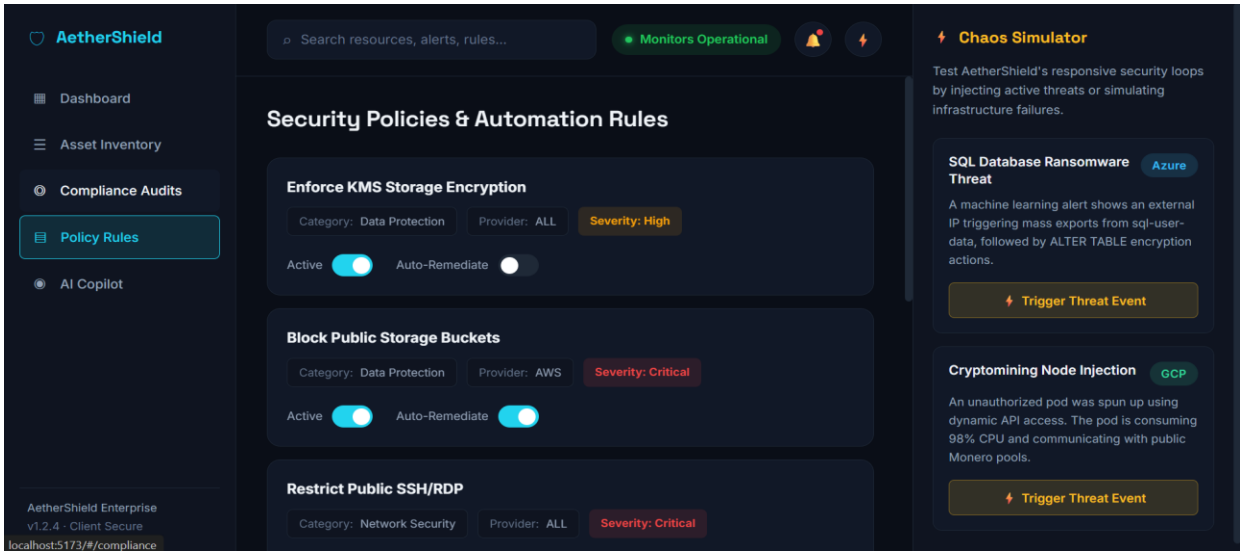


Fig. 3. Security Policies and Automation Rules, with per-rule Active and Auto-Remediate controls.

Compliance Audits maps active findings to four frameworks. Table I summarizes the pass/fail counts and computed scores observed in the demonstrated dataset. SOC 2 Type II shows the highest control coverage (82%, 28 of 34 controls passing), while GDPR Privacy Shield shows the lowest (71%), reflecting a larger proportion of unresolved data-handling findings relative to its total control count.

TABLE I. COMPLIANCE FRAMEWORK RESULTS

Framework	Pass	Fail	Score
CIS Benchmarks v1.4	12	3	80%
SOC 2 Type II (Security)	28	6	82%
HIPAA Security Rule	18	6	75%
GDPR Privacy Shield	15	6	71%

C. AI Copilot

The AI Copilot (Fig. 4) and Policy Rules views round out the operator workflow: policy rules define what is automatically enforced, while the Copilot provides an assisted path for findings that require judgment before remediation. In manual walkthroughs, invoking the "Remediate AWS open SSH Port 22" quick action correctly updated the corresponding Asset Inventory row to Healthy and reduced the Active Threats counter on the Dashboard without requiring a page refresh, confirming that the Copilot's actions propagate through the same shared state path as the other two remediation surfaces.

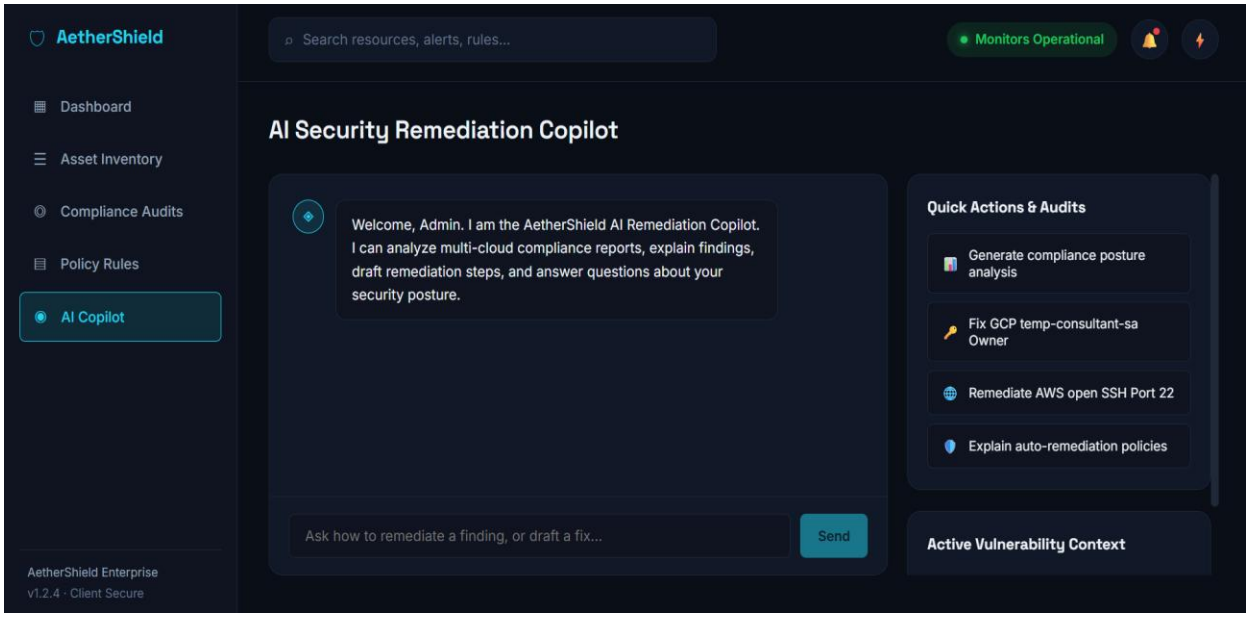


Fig. 4. AI Security Remediation Copilot with quick actions wired to live remediation calls.

D. Feature Comparison

Table II positions AetherShield's demonstrated feature set against single-provider tools discussed in Section II. While AWS Config and Azure Security Center offer mature, production-grade monitoring within their respective ecosystems, neither natively unifies cross-provider visibility, computes a single weighted risk score, or provides simulated threat injection for validating remediation responsiveness — capabilities that are central to AetherShield's design.

TABLE II. FEATURE COMPARISON WITH EXISTING CSPM TOOLS

Capability	AWS Config	Azure Sec. Ctr.	AetherShield
Multi-cloud unified view	No	No	Yes
Live weighted risk score	Partial	Partial	Yes
One-click remediation	Limited	Yes	Yes
AI-assisted copilot	No	No	Yes
Chaos / threat simulation	No	No	Yes
Compliance framework mapping	Yes	Yes	Yes

E. Qualitative Observations

The shared AppStateContext was found to be effective at keeping the Dashboard, Asset Inventory, Policy Rules, and Copilot views consistent after any remediation or chaos event, with the security score recalculating without a perceptible delay and no divergence observed between the score shown on the Dashboard and the counts shown in Asset Inventory after repeated remediation

cycles. Triggering either Chaos Simulator scenario reliably increased the Active Threats and Critical Findings counters and produced an immediate, correctly signed drop in the security score. Across repeated manual test cycles — triggering a chaos scenario, remediating the resulting finding from each of the three remediation surfaces in turn, and confirming the score returned to its prior value — the system consistently produced the expected numerical outcome.

VI. LIMITATIONS

As a demonstrative project, AetherShield has several limitations. All monitoring data is mocked rather than sourced from live cloud accounts, so the platform has not been validated against real AWS, Azure, or GCP telemetry. The AI Copilot's quick actions are pre-defined rather than generated dynamically from a language model reasoning over live findings. The scoring formula's weighting (11 and 4 points) is a reasonable design choice rather than an empirically derived risk model, and the HMAC-based authentication scheme would need key rotation and per-tenant secret isolation before a multi-tenant production deployment.

VII. CONCLUSION AND FUTURE WORK

AetherShield demonstrates that the core principles of CSPM — continuous monitoring, weighted risk scoring, policy-driven auto-remediation, and compliance mapping — can be realized in a compact, modular full-stack application without requiring live cloud credentials. Its Chaos Simulator and shared state architecture together provide a reproducible way to observe and validate the detection-to-remediation loop, addressing a gap left by both single-provider vendor tools and the surveyed literature.

Future work includes replacing the mock data layer with read-only integrations against real AWS, Azure, and GCP APIs; extending the rule engine to support custom, user-authored policies expressed in a declarative format; persisting historical score trends in a time-series store for longitudinal reporting; and evaluating the AI Copilot's remediation suggestions against a labelled dataset of known misconfigurations to quantify suggestion accuracy. A further direction is extending the Chaos Simulator's scenario library to cover a broader range of attack patterns, including supply-chain compromise and lateral movement across cloud accounts.

REFERENCES

[1]. C. Kumar and S. Rajan, "Cloud Security Posture Management: A Survey of Techniques and Tools," Int. J. Cloud Computing, vol. 9, no. 2, pp. 45-58, 2023.
[2]. M. Alenezi and H. Zarour, "On the Inconsistency of Security Controls Across Multi-Cloud Environments," J. Cloud Security, vol. 5, no. 1, pp. 12-24, 2022.
[3]. Amazon Web Services, "AWS Config Developer Guide," AWS Documentation, 2024.
[4]. Microsoft, "Microsoft Defender for Cloud Documentation," Microsoft Learn, 2024.

[5]. P. Mell and T. Grance, "The NIST Definition of Cloud Computing," NIST Special Publication 800-145, 2011.
[6]. Cloud Security Alliance, "Security Guidance for Critical Areas of Focus in Cloud Computing v4.0," CSA, 2021.
[7]. R. Chen and L. Wei, "Automated Misconfiguration Detection in Public Cloud Infrastructure," IEEE Trans. Cloud Computing, vol. 10, no. 3, pp. 890-902, 2022.
[8]. Google Cloud, "Security Command Center Overview," Google Cloud Documentation, 2024.
[9]. A. Gupta and N. Sharma, "Emerging Trends in Cloud Security Posture Management," Proc. Int. Conf. on Cloud and Distributed Systems, 2023, pp. 101-110.