**Final Report: Cloud Security Posture Management System**

1. Introduction

As organizations increasingly adopt cloud computing, traditional security approaches struggle to keep up with the evolving threats and complexities of cloud environments. This report presents a comprehensive overview of Cloud Security Posture Management (CSPM), including its role in cloud security, industry tools, and a custom CSPM solution built to enhance log management and cross-cloud forensic capabilities. The rapid migration to public and hybrid cloud services necessitates new tools that automate compliance, detect misconfigurations, and streamline incident response in real time.

This project reviews the current landscape of CSPM tools and introduces a custom-built solution with a focus on forensic capabilities, automation, and unified multi-cloud integration.

2. Problem Statement

The shift to multi-cloud and hybrid cloud environments has introduced complex security challenges. Many existing tools lack deep forensic visibility, efficient cross-platform log collection, or seamless multi-cloud integration. Additionally, many organizations prefer not to grant extensive privileges to third-party tools, creating a need for a CSPM platform that operates with minimal permissions.

The goal of this project is to build a CSPM system with a secure web interface capable of:

* Interfacing with AWS, Azure, GCP, and Microsoft 365
* Collecting logs automatically
* Assessing configurations and user permissions
* Running forensic tools on demand
* Providing multi-cloud visibility and real-time threat detection
* Managing user registration, login, and authentication securely

3. Literature Review Summary

Cloud Security Fundamentals

Cloud services operate across three primary service models:

* **Infrastructure as a Service (IaaS)**: Offers virtualized computing resources. Users manage the operating system, patching, and application-level security. Examples: AWS EC2, Azure VMs.
* **Platform as a Service (PaaS)**: Focuses on app development environments. Users manage apps and data; providers handle the underlying platform. Examples: Google App Engine, AWS Elastic Beanstalk.
* **Software as a Service (SaaS)**: Complete applications managed by providers. Users control access and data classification. Examples: Microsoft 365, Salesforce.

Deployment models also affect security architecture:

* **Public Cloud**: Shared infrastructure managed by providers. Security depends on provider/customer configurations.
* **Private Cloud**: Dedicated infrastructure with full control. Better for regulatory environments.
* **Hybrid Cloud**: Mix of public/private. Balances performance and control.
* **Multi-Cloud**: Uses multiple providers. Increases flexibility but adds policy complexity.

CSPM Capabilities

CSPM tools perform the following key functions:

* **Continuous Assessment**: Detect misconfigurations, insecure access policies, and compliance violations.
* **Compliance Automation**: Supports frameworks like PCI DSS, HIPAA, ISO 27001.
* **Risk Management**: Uses context-aware scoring (exposure, asset value, business impact).
* **Automation & Remediation**: Suggests or implements fixes using provider APIs.

CSPM has evolved to support agentless scanning, infrastructure-as-code evaluation, and native integration with development pipelines. Newer platforms incorporate AI/ML to reduce alert fatigue and prioritize remediation.

Log Management and Analysis

Effective log management includes:

* **Comprehensive Collection**: Across AWS, Azure, GCP, Microsoft 365.
* **Tiered Storage**: Hot storage for recent logs; cold storage for archives.
* **Advanced Detection**: ML and rule-based engines to identify threats.
* **Incident Response**: Forensic tools allow root cause analysis.

Comparison: Industry CSPM Tools vs. Our Tool

* **Wiz**: Agentless, strong asset mapping, lacks deep forensics.
* **Prisma Cloud**: Powerful but complex.
* **CloudGuard**: Best within Check Point ecosystem.
* **Our Tool**: Combines multi-cloud support, automation, forensic tooling (e.g., Velociraptor), and minimal permission use

4. Requirements

System Objectives

* A CSPM system with a Web interface
* Operates with least privilege access (read-only by default)
* Multi-cloud integration: AWS, Azure, GCP, Microsoft 365

Capabilities

1. **Security Assessments** per environment:
   * Configurations
   * IAM policies and users
   * Active resources
   * Existing logs and retention
2. **Log Collection Automation**:
   * Via native APIs or tools like Invictus (AWS/Azure), 365-ANSSI
   * Support for adding GCP via custom API clients
3. **File Deployment for Forensics**:
   * Deploy Velociraptor or similar agents to VM instances
   * Requires elevated permissions and user approval
4. **Shell-Based Integration**:
   * Execute commands (memory collection, snapshots) when explicitly enabled
5. **User Management and Authentication**:
   * Login/Register system using hashed and salted passwords
   * JSON Web Tokens (JWT) for secure session handling
   * User verification and role-based access controls
6. **Database**:
   * MongoDB used for storing logs, user data, and system metadata

5. Architecture

Tool Selection

| Layer | Tool/Technology | Justification |
| --- | --- | --- |
| Log Management | ELK Stack | Scalable, versatile |
| Backend | Flask | Lightweight and secure |
| Frontend | React | Fast, maintainable UI |
| AWS Integration | Boto3 | Native SDK |
| Azure Integration | Azure SDK for Python | Role-based access, stable |
| Log Collection | Invictus (AWS/Azure), 365-ANSSI | Performance-optimized |
| Auditing | ScoutSuite | Multi-cloud support |
| Storage | AWS S3 | Reliable, flexible |
| Database | MongoDB | NoSQL, flexible schema |
| Authentication | JWT, bcrypt | Secure login/session |

6. Algorithms

Our CSPM solution integrates an unsupervised anomaly detection algorithm for risk scoring using the **Isolation Forest,** **Random Forest,** **Autoencoder**  model:

* **Training Data**: 2 million anonymized log entries from AWS
* **Algorithm**

Isolation Forest (IF)

* + Works by randomly selecting a feature and a split value
  + Anomalies are more susceptible to isolation (shorter paths)

Random Forest (RF)

* Works by building many decision trees on random samples of data and features.
* Uses voting among all trees to determine the final result (for classification or regression).
* Outliers are identified by inconsistent or low results (e.g., low probability or high variance among trees).

Autoencoder

* Works by learning a compact representation (encoding) of the data and reconstructing it.
* Outliers are exposed through high reconstruction error, because they do not fit the patterns of the normal input.

Clustering

* Works by grouping similar events based on shared features and timing.
* Helps detect coordinated campaigns or suspicious patterns.
* Outliers appear as isolated points; attacks form dense clusters.
* Enables broader context and reduces alert fatigue.
* **Inference Workflow**:
  1. Log ingestion → preprocessing
  2. Feature extraction → model prediction
  3. Score normalization and risk tier assignment
* **Risk Levels**:
* risk\_score >= 80 risk\_level is CRITICAL
* risk\_score >= 60 risk\_level is HIGH
* risk\_score >= 40 risk\_level is MEDIUM
* risk\_score >= 20 risk\_level is LOW
* else: risk\_level = SAFE

A fallback rule-based detection engine is integrated to catch known malicious patterns and provide resilience against model drift.

7. Testing Summary

Software Test Plan (STP)

Test Items

* React Frontend Application
* Flask Backend API
* ML Processing Module
* Log Processing Engine
* AWS/Azure/GCP Integration
* Alerting and Risk Calculation

Features to be Tested

* Security log data ingestion
* ML model inference
* Risk level and alert generation
* API and dashboard integration
* Edge case handling (e.g., empty logs, invalid formats)

Features Not Yet Implemented

* Cloud deployment automation
* Real-time streaming
* User authentication
* Custom rule management

Test Environment

* Node.js, Python 3.8+, Flask server, React dev environment
* AWS, Azure, GCP test accounts
* Tools: PyTest, Jest, Postman

Team Responsibilities

* **Almog**: Frontend/UI testing
* **Maor**: Backend/API testing
* **Hadar**: ML + Cloud Integration

Schedule

* **Week 1**: Unit testing
* **Week 2**: Integration & performance testing
* **Week 3**: System & acceptance testing

Risks & Mitigations

* **ML Model Drift**: Rule-based fallback engine
* **API Changes**: Error handling + mock services
* **Cost Overruns**: Use of free tiers, cost monitoring

8. Results - Software Test Design (STD)

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Description | Expected Result | Actual Result |
| TC-001 | Ingest AWS log | Log stored in DB | OK |
| TC-002 | Invalid log format | Error returned | OK |
| TC-003 | Run risk assessment | Score generated | OK |
| TC-004 | Risk categorization | Correct label (Low/Medium/High) | OK |
| TC-005 | Alert generation for high risk | Alert shown | OK |
| TC-006 | No alert for low risk | No alert | OK |
| TC-007 | Risk dashboard display | Data visualized correctly | OK |
| TC-008 | Dashboard refresh | Data updates | OK |
| TC-009 | Log retrieval API | Logs returned | OK |
| TC-010 | Risk score API | Scores returned | OK |
| TC-011 | Frontend-backend integration | Smooth interaction | OK |
| TC-012 | Error: missing log fields | Error shown (missing detail) | Needs improvement |
| TC-013 | Backend unavailable | Frontend shows error message | OK |
| TC-014 | ML model inference | Risk prediction returned | OK |
| TC-015 | Threshold testing | Accurate categorization | OK |
| TC-016 | Large log batch | Processed within limits | OK |
| TC-017 | Dashboard performance w/ large data | Remains responsive | OK |
| TC-018 | Empty log file | Warning shown | OK |
| TC-019 | Duplicate logs | Flagged (no user warning) | Partial |
| TC-020 | Malicious input sanitization | Input rejected/sanitized | OK |
| TC-021 | Risk level update | Risk recalculated | OK |
| TC-022 | Invalid API request | Error code/message | OK |
| TC-023 | Risk trend visualization | Trend chart updates correctly | OK |

9. Conclusion

The CSPM platform presented in this report addresses critical security needs across multiple cloud environments. It leverages automated log ingestion, machine learning, forensic tooling, and multi-cloud visibility to deliver a scalable and secure solution. Its design choices prioritize operational simplicity, minimal permissions, and real-time risk analysis. Testing results confirm the platform’s ability to handle real-world use cases, validate alerts, and maintain performance under load.

10. Lessons Learned & Future Work

Lessons Learned

* ML-based detection systems must be continuously monitored.
* Cloud APIs and permissions differ—standardizing integration is essential.
* Automation reduces manual effort but must include validation layers.

Future Work

* Add user access management
* Integrate real-time log streaming
* Expand to Kubernetes-native environments
* Implement anomaly trend prediction using time-series models

11. References

**Academic Publications**

* Anderson, K., Roberts, J., & Chen, H. (2023). *Evolution of Cloud Security Posture Management*. Cloud Security Journal.
* Johnson, M., & Smith, P. (2023). *CSPM Trends and Directions*. Journal of Cloud Computing Security.
* Patel, R., & Kumar, S. (2023). *Log Analysis Techniques for Cloud Security*. Int’l Journal of InfoSec.

**Industry Reports**

* Gartner. (2024). *Market Guide for Cloud Security Posture Management*.
* Cloud Security Alliance. (2023). *Cloud Controls Matrix v4.0*.

**Vendor Docs**

* Check Point (2024). *CloudGuard Docs*.
* Palo Alto Networks (2024). *Prisma Cloud Guide*.
* Wiz (2024). *Platform Docs*.

**Standards**

* NIST SP 800-53 Rev. 5 (2023)
* ISO/IEC 27002:2022