



International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

Cloud Security Monitoring and Automated Incident Response Using AI

¹Batta Latesh, ²Bonela Chandramouli

^{1,2} GMRIT Deemed to be University

¹(lateshk169@gmail.com), ²(Chandrabonela3759@gmail.com)

ABSTRACT

Most organisations now run a large part of their IT infrastructure in the cloud, drawn by the scalability and cost savings it offers. That shift, however, has opened up a much wider attack surface, and attackers have been quick to take advantage of it through ransomware, phishing campaigns, insider misuse, stolen credentials, and outright data breaches. The trouble is that many security teams are still leaning on static rule sets and manual alert triage, an approach that was never built to keep pace with threats that change by the day. This project looks at how that gap can be closed by combining several complementary technologies into one working framework: Artificial Intelligence and Machine Learning for the heavy analytical lifting, SIEM for centralised log visibility, XDR for correlating signals across endpoints and networks, SOAR for turning detections into automated action, and a Zero Trust model that verifies every request rather than assuming anything inside the network is safe. On the machine learning side, the system draws on a mix of models suited to different parts of the problem - Random Forest, Isolation Forest, alongside behavioural analytics and generative AI to help connect related alerts into a coherent picture of an attack. Together, these components continuously watch cloud telemetry, flag suspicious behaviour, rank incidents by how serious they are, and trigger pre-built response playbooks without waiting on a human to act first. Because the design is cloud-native and built to keep learning from new data, it adapts reasonably well as workloads grow, while identity-aware access policies help keep false alarms and manual overhead down. Put together, the framework is meant to catch threats earlier, contain incidents faster once they are detected, and make it easier for organisations to stay on top of compliance requirements - giving security teams a practical, scalable way to defend cloud environments that keep evolving.

Keywords: Cloud Security, Artificial Intelligence, Zero Trust Architecture, Behavioural Analytics, Security Orchestration

INTRODUCTION

Cloud computing has become the foundation of modern digital services, enabling organizations to deliver applications with greater scalability, flexibility, and cost efficiency. Industries such as healthcare, banking, education, government, and e-commerce increasingly rely on cloud platforms to store sensitive data and provide uninterrupted online services. While cloud adoption offers significant benefits, it also introduces complex security challenges including unauthorized access, data breaches, misconfigured resources, insider threats, and sophisticated cyberattacks. Traditional security solutions often detect incidents only after they occur and require considerable manual effort from security teams.

To address these challenges, this project proposes Cloud Security Monitoring and Automated Incident Response using AI, an intelligent security platform designed to improve both preventive and operational cloud security. The system combines artificial intelligence, behavioral analytics, Zero Trust principles, and automated response mechanisms to identify vulnerabilities before deployment and continuously monitor cloud activities during runtime. Rather than depending on a single security technology, the proposed framework integrates multiple security functions into one unified architecture.

The project follows a three-phase security lifecycle. In Phase 1, developers upload their software project, select the application domain (such as hospital, educational, banking, financial, or general), and perform a comprehensive security assessment. The system analyzes source code, dependencies, configuration files, and infrastructure files to generate a security score and provide detailed remediation recommendations. Only projects achieving the required security threshold are approved for the next stage.

In Phase 2, the platform performs continuous cloud security monitoring by analyzing cloud events, authentication logs, resource activities, and user behavior. AI-based threat detection identifies abnormal patterns such as credential theft, brute-force attacks, privilege escalation, and suspicious access attempts while reducing false positives through behavioral validation. This enables early identification of security incidents before they cause significant damage.

Phase 3 focuses on automated incident response and digital forensics. When a validated threat is detected, the system executes predefined security playbooks, preserves forensic evidence, generates incident reports, and notifies the developer through the integrated dashboard and email alerts. An additional AI Security Assistant helps developers understand security reports, vulnerabilities, cloud security concepts, and project architecture, making the platform both a security solution and an educational support system.

Overall, this project presents a scalable, modular, and vendor-independent approach to cloud security that enhances threat detection accuracy, minimizes response time, and supports secure software development practices for modern cloud-based applications.

LITERATURE SURVEY

Deep learning methods such as CNNs, LSTMs, autoencoders and reinforcement learning are proposed to catch zero-day threats and cut false alarms in multi-tenant clouds. The authors measure none of this experimentally, and they flag privacy, adversarial manipulation and explainability as risks [1]. In a 30-day production test, an AI-enhanced SIEM, XDR and SOAR setup cut triage time from 17.4 hours to 10.7 minutes and closed 89.4% of incidents automatically. It was tested with a single vendor and without statistical testing [2]. A containerised system for Google Cloud and Azure combines a traffic classifier, a web intrusion detector and a malware analyser. It reached about 99.8% accuracy on one dataset and 91.3% recall on web anomalies, but was tested only on known attacks [3].

A zero-trust cloud-edge-terminal architecture for cloud manufacturing combines continuous identity verification, micro-isolation, blockchain-protected data and learning models. Its effectiveness is argued through a case study rather than numerical metrics [4]. A review of automation across SIEM, XDR, SOAR and threat intelligence proposes a scalable architecture and a mutual-information-based detection measure. It runs no new experiments, and legacy integration and alert overload remain concerns [5].

A CloudStack-based system uses a random forest to filter noisy alerts and link them to root causes. It reached a 0.964 noise reduction ratio and an F1-score of about 0.96, though only on one platform [6]. A narrative review argues that predictive analytics and behaviour-based detection improve detection, prioritisation and false-positive rates. It also warns about heavy computing needs, biased data, attackers using AI and compliance rules [7]. A survey of machine learning, reinforcement learning, language models and self-healing approaches finds that AI improves detection accuracy and speeds response. It lists privacy, adversarial attacks, interpretability and cost as open problems, and points to federated learning and explainable AI [8].

AI4SOAR, built on the open-source Shuffle platform, recommends playbooks by comparing new alerts with past ones using autoencoder-compressed features and similarity scores. In an SSH brute-force demonstration it often reached near 99% similarity, but it was evaluated on only that one scenario [9]. A proposed architecture feeds threat intelligence into both SIEM and SOAR across hybrid and multi-cloud environments through a data lake, a correlation engine and playbooks. It is largely conceptual and reports no figures [10]. SOAR4BC pairs a digital twin with a deep reinforcement learning agent for critical infrastructure. On a factory simulation and an EV charging testbed it stopped 97% of intrusions at the perimeter and isolated components in about 9 seconds [11].

A hyper-automation approach uses LLM agents, guided by an investigation, validation and active-monitoring framework, to write response code and replace rigid no-code playbooks. A pilot against SSH brute-force attacks shrank a 38-step workflow into ten actions, though response-time metrics were not yet measured [12]. The ACR framework classifies attacks with an ensemble of machine learning models and selects countermeasures from a CAPEC-based ontology. It achieved a 99.13% F1-score, about 4 seconds average response time and a false-positive rate below 0.07% in a lab testbed [13].

For attacks that hop across chained services owned by different providers, a SOAR-based operational workflow is proposed, with context graphs, identity management and eBPF monitoring. It identifies five investigation obstacles, and its prototype is only partial, leaving privacy and responsibility questions unresolved [14]. A review of DevSecOps threat models, including STRIDE, PASTA and OCTAVE, proposes a layered framework for multi-cloud applications covering posture management, SIEM integration and orchestration. It has been proposed but not yet implemented or tested [15].

METHODOLOGY

1. Overview of the Approach

We built the platform as a single chain of three connected stages instead of three separate tools. A project moves forward only when it has earned the right to. Phase 1 checks whether the uploaded code is safe enough to be watched, Phase 2 watches approved projects while they run, and Phase 3 steps in only when a threat is detected with high certainty.

Two checkpoints control this movement. The first is a security-score threshold between Phase 1 and Phase 2. The second is a confidence threshold between Phase 2 and Phase 3. Every record the system produces carries one Project ID, so a scan report, a monitoring event, an incident, and an evidence file can all be traced back to the same project.

The system runs fully on a local machine. A React interface talks to a Flask backend, which stores everything in MongoDB. No live cloud account is needed, so events that would be risky or impossible to trigger on real infrastructure are simulated.

2. Phase 1: Static Security Assessment

2.1 What this phase is for

The goal of Phase 1 is to decide whether a project is secure enough to be monitored at all. Watching a badly written application at runtime without first checking its code gives a false sense of safety. For this phase we chose deterministic, rule-based analysis rather than machine learning. We had no large

labelled dataset of vulnerable source code, and rules let us point to the exact file and pattern behind every finding, which makes each result easy to explain and audit.

2.2 Handling the uploaded ZIP safely

Because we cannot trust what a user uploads, the archive is validated before anything is read. We reject empty files and files larger than the configured limit (700 MB by default). Password-protected archives are refused because their contents cannot be inspected. To guard against zip bombs, we cap the number of entries, the total uncompressed size, and the compression ratio.

To prevent Zip Slip attacks, where a file name such as `../something` tries to escape the target folder, every entry path is resolved and confirmed to stay inside the project's own directory under `uploads/<project_id>/. We deliberately avoid the standard extractall() call and extract entries one by one. Heavy folders that carry no useful security information, such as node_modules, .git, venv, and __pycache__, are skipped.`

2.3 Detecting the technology stack

Once the files are extracted, the project tree is examined to identify its languages, frameworks, and dependency manifests. This matters because the detectors use rules suited to what is actually present. A Python project is not judged by Java rules, and a project with Terraform files gets infrastructure checks that a plain web app would not.

2.4 The five detectors

Five independent detectors then walk through the relevant files, and each reports findings with a problem description, a severity, and a location.

- **Secret scanner:** looks through source files, environment files, and infrastructure definitions for hardcoded API keys, tokens, and passwords, using pattern and entropy-style checks.
- **Static code analyser:** inspects Python, JavaScript, TypeScript, Java, PHP, C#, Ruby, and Go files for risky constructs such as SQL injection, cross-site scripting, command injection, and weak cryptography.
- **Configuration checker:** reviews environment, YAML, and JSON settings for issues like debug mode left on, default credentials, and overly permissive CORS.
- **Infrastructure checker:** reads Terraform, CloudFormation, Docker, and Kubernetes files, flagging open 0.0.0.0/0 rules, public storage buckets, and privileged containers.
- **Dependency checker:** compares versions listed in files like `requirements.txt`, `package.json`, and `pom.xml` against a curated table of known-vulnerable libraries.

After all detectors finish, duplicate findings are removed so the same issue is not counted twice.

2.5 Turning findings into a score

The score is calculated in the scoring engine. Each finding carries a base penalty according to its severity: 15 for critical, 8 for high, 3.5 for medium, and 1.2 for low. If the same issue appears again and again, each repeat counts for less. The n -th occurrence of the same severity and problem is multiplied by 0.55^n , so one mistake copied across fifty files cannot wipe out the score on its own.

The project's category also matters. A banking application has its hardcoded-password, SQL injection, and API-key findings multiplied by 1.35, and healthcare projects use 1.25, because the same flaw carries more consequence in those domains.

The total penalty is then mapped to a 0 to 100 scale with a soft curve instead of simple subtraction:

$$\text{mapped} = 100 \times \left(1 - e^{-\frac{\text{raw_penalty}}{55}}\right), \quad \text{score} = \min\left(100, \max\left(0, 100 - \text{mapped}\right)\right)$$

The exponential form means a long list of findings pushes the score down steadily rather than dropping it to zero. A scan with no findings scores 100, and a floor of 12 applies when a scan is heavily flagged but not an extreme critical case. A risk band (Critical, High, Medium, or Low) is also shown for readability.

2.6 The approval decision (Gate A)

The decision engine compares the score against the approval threshold, 65 by default. A project is **Approved** if its score is at least 65 with no more than two critical findings, or at least 75 with no more than four critical findings. Anything else is marked **Pending Remediation**, and monitoring stays locked.

The gate does not demand a flawless scan. It asks whether the remaining risk is acceptable. A rejected user can fix the issues, upload the project again, and rescan.

2.7 What Phase 1 produces

The phase stores the score and status on the project, writes a security report with severity counts and recommendations, generates a PDF under reports/<project_id>/, and keeps the scan in a history used for score trends. The same Project ID is also passed into the Advanced Features layer.

3. Phase 2: Cloud Monitoring and Threat Detection

3.1 What this phase is for

Once a project is approved, Phase 2 watches it continuously. The aim is to spot abnormal behaviour such as brute-force logins, privilege abuse, and unusual data downloads, and to decide how sure the system is about each case. Here we use machine learning, because behaviour data is something we can generate in large volumes with clear labels.

3.2 Generating the event stream

A background scheduler creates a CloudTrail-style event every five seconds by default. The events cover console logins, failed logins, IAM changes, object reads and writes, large downloads, and security-group changes. Each event is normalised into a common structure with the user, IP address, country, bytes transferred, MFA status, success or failure, hour of day, and event type. The monitoring page polls the backend at a similar pace, so an analyst sees a live feed.

3.3 Feature engineering

Each event is converted into ten numbers: login hour, an encoded event type, a failed-login flag, an IAM-change flag, a data-transfer flag, MFA status, a suspicious flag, gigabytes transferred, a high-risk-country flag (Russia, China, Unknown, North Korea, Iran), and a request-failed flag.

The order of these features is identical in the training script and the live detector. If they ever differ, predictions would be wrong without any error message. Values are passed through the same StandardScaler fitted during training.

3.4 Offline model training

Training happens separately, through ai/train_model.py, and the running application never retrains. We generate 8,000 synthetic events, 15% of them malicious. To avoid a toy dataset that any model could separate perfectly, we made attack and normal behaviour overlap, including legitimate but unusual patterns such as on-call staff working at odd hours.

The data is split 80/20 with stratification and a fixed random state of 42. The scaler is fitted on the training portion only, to avoid leakage. Two models are trained:

- **Isolation Forest:** fitted on benign events only (200 trees, contamination 0.15). It learns what normal looks like and flags anything that sits far outside it, which helps with attacks it has never seen.
- **Random Forest:** a supervised classifier with 200 trees, a maximum depth of 12, a minimum of 2 samples per leaf, and balanced class weights. It outputs the probability that an event is malicious.

We evaluate with precision, recall, F1, accuracy, ROC-AUC, five-fold cross-validated F1, a confusion matrix, and feature importance. Models are saved as Joblib files (threat_detector.joblib, threat_classifier.joblib, feature_scaler.joblib) together with model_metadata.json, so the reported metrics are measured values rather than claims.

3.5 Live inference

For each incoming event, the system builds the ten-feature vector and scales it. Isolation Forest labels it an outlier or an inlier, and Random Forest gives a malicious probability. An event is treated as malicious if that probability is at least 0.5 or if Isolation Forest flags it. Confidence is the malicious probability times 100 for malicious events, or the benign probability times 100 otherwise. If only Isolation Forest is available, confidence is derived from the distance to its decision boundary.

The threat type and severity then come from the event type with a few overrides. A data transfer becomes Data Exfiltration at Critical severity, a failed and suspicious login becomes Brute Force, an IAM change without MFA becomes Privilege Escalation, and a high-risk country without MFA raises severity.

3.6 The confidence decision (Gate B)

Only events that are malicious and have confidence of at least 85% are forwarded to Phase 3. Anything below that stays on the monitoring page as an observation. The strict cutoff favours precision, so analysts are not buried under incidents caused by noisy anomalies.

3.7 Fallback and correlation

If the model files are missing or inference fails, a transparent rule score takes over: suspicious flag +40, failed login +25, IAM change +30, data transfer +35, high-risk country +20, and missing MFA +10. A total of 50 or more counts as malicious. This keeps monitoring working on a fresh setup.

Related events can also be grouped into a single story to avoid duplicate cards, and raw and normalised logs are passed on to the SIEM panel under the same Project ID.

4. Phase 3: Incident Response and Forensics

4.1 What this phase is for

Phase 3 handles a threat that has already passed the confidence gate. It records what happened, runs a repeatable response, preserves evidence, and produces a report. We use a deterministic mapping from threat type to playbook instead of another ML model, because containment actions must be explainable and predictable.

4.2 Opening the incident

When Phase 2 marks an event as a threat, the incident manager creates an incident linked to the Project ID, user, IP address, threat type, severity, and confidence. The project status changes to **Incident Detected**, and the rule engine selects a playbook.

4.3 Choosing and running a playbook

The threat type decides the playbook:

Threat	Simulated response
Credential Theft	Lock the account, end sessions, force a password reset, rotate keys
Brute Force	Apply a temporary lock, record the source IP, raise the risk level
Data Exfiltration	Suspend the download, record the transfer, block the IP
Privilege Escalation	Revoke the elevated rights, audit IAM, notify
Malware Activity	Quarantine the resource, save evidence
Public Resource Exposure	Restrict or isolate the exposed resource
Unrecognised threat	Flag the login and ask for MFA re-verification (default playbook)

Every step is clearly labelled as a **simulation**. The platform does not call AWS IAM, change a real security group, or disable a real person's account. It still demonstrates how orchestration, evidence handling, and reporting work together.

4.4 Evidence, timeline, notification, and report

Each playbook step is saved as a response record. An evidence collector then stores forensic documents covering the incident context, a log snapshot, and the actions taken. A timeline builder orders these events for the incident detail page.

The user receives an in-app notification (plus email if SMTP is configured), and an incident PDF is written to the reports/ folder. The incident moves from In Progress to Resolved through the rule engine, while analysts can still review, re-run, or close it manually.

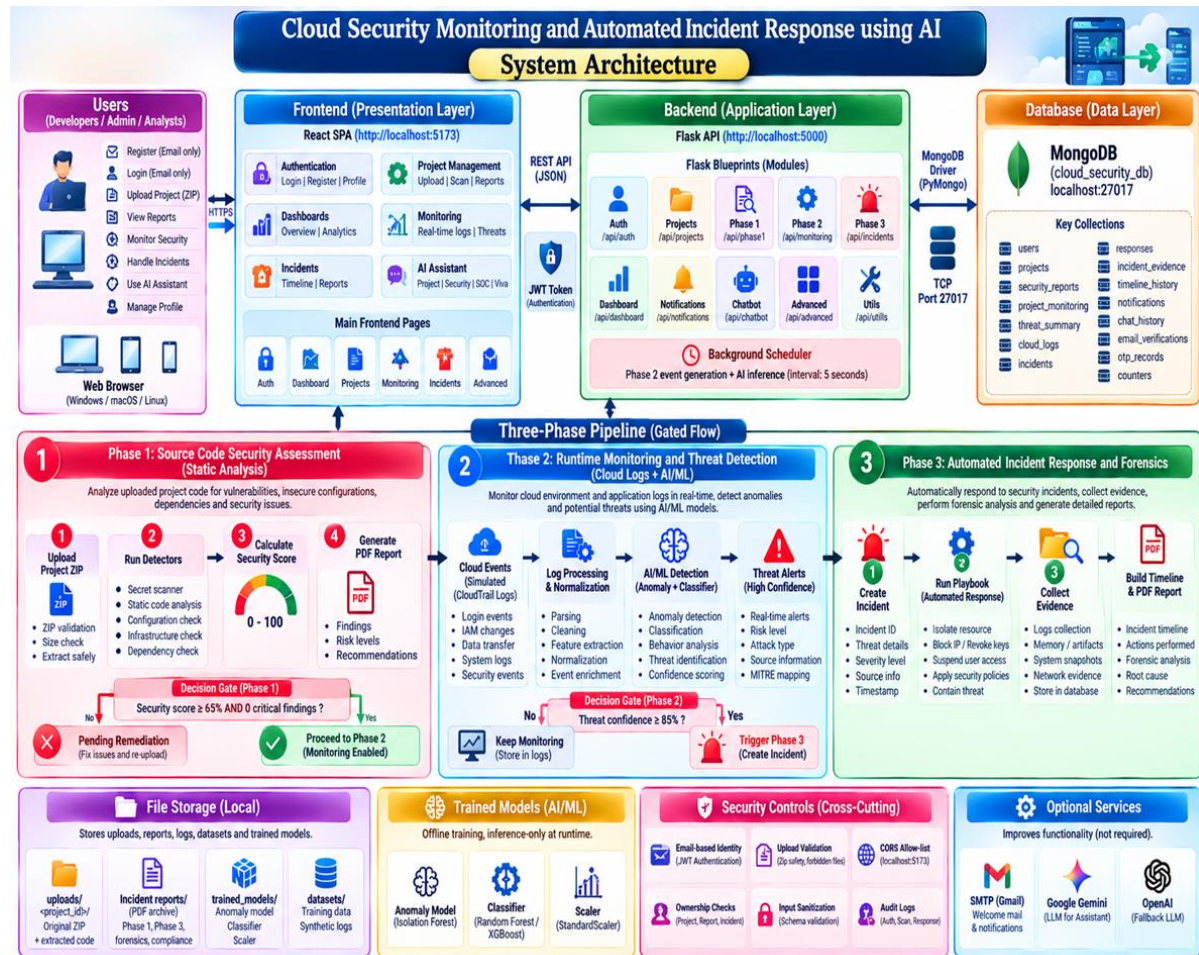
4.5 Forensic integrity

For uploaded evidence, the server computes SHA-256 and MD5 hashes along with file size and name, and opens a chain-of-custody log that records each handover (received, transferred, locker, analysis, archived). A custody text file and a forensic PDF are generated from that record. The hashes prove the evidence has not been altered. They are not used to classify threats.

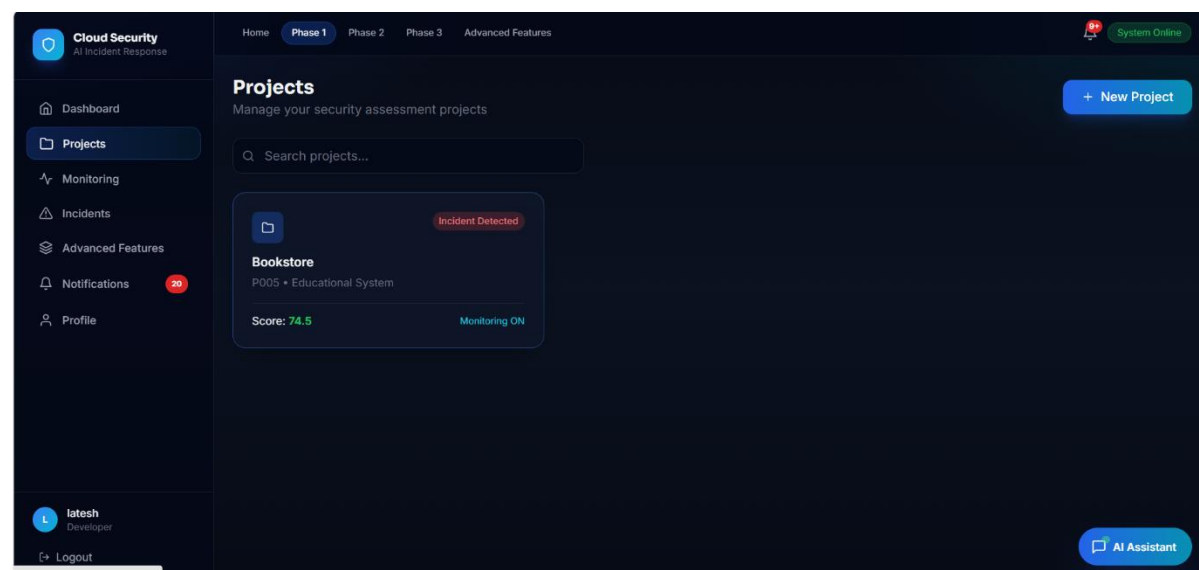
5. Limitations

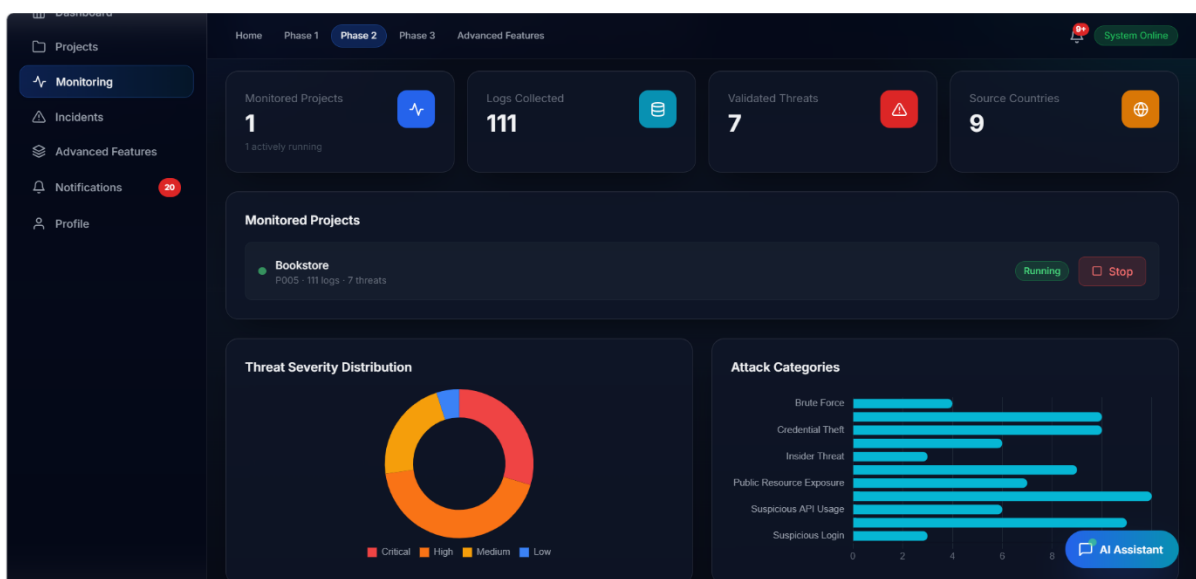
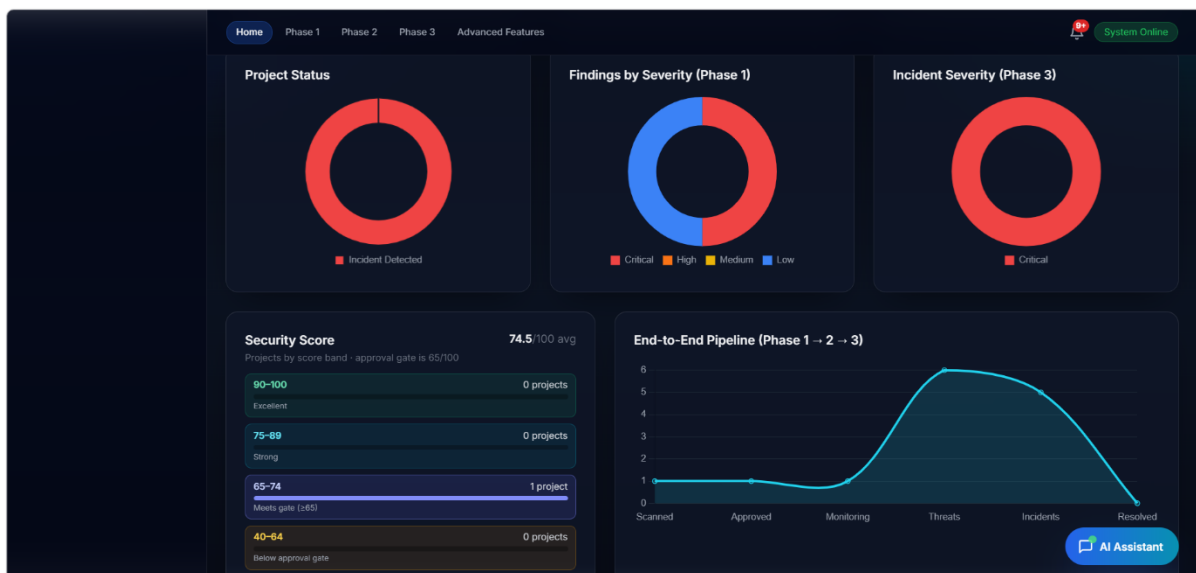
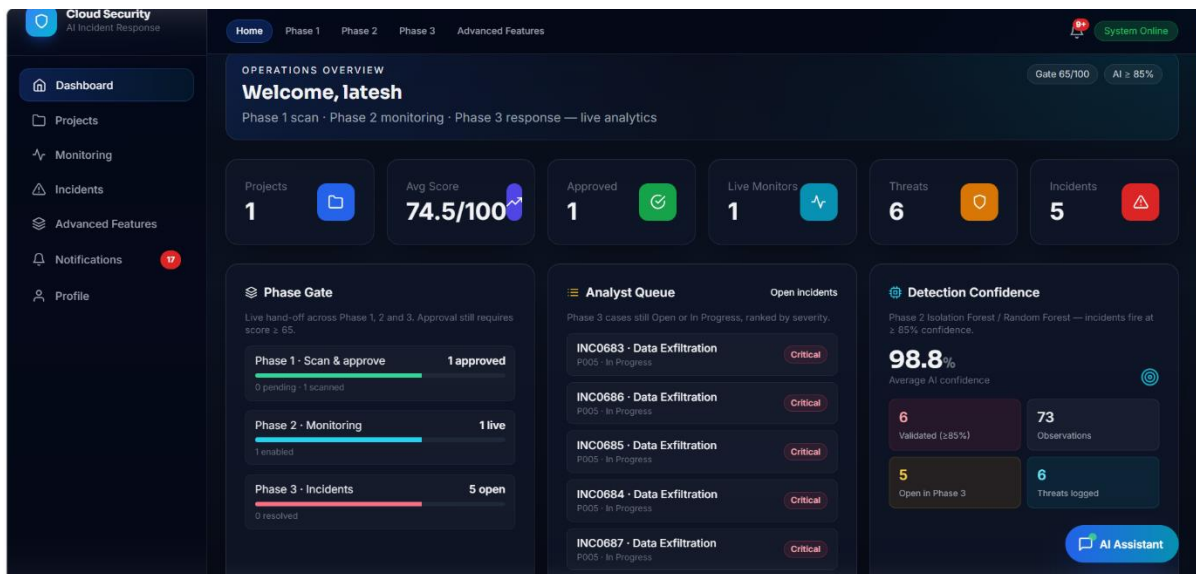
The Phase 2 events are simulated, so the models are real but the cloud feed is not. Moving to live data would mean replacing the event generator with a real collector while keeping the same ten features. The training data is synthetic, even though overlap was built in deliberately.

Phase 1 rules may produce false positives or miss new vulnerability types. Playbooks demonstrate response logic without changing any real cloud state.



RESULTS





HomePhase 1Phase 2Phase 3Advanced Features

System Online

Recent AI Detections

Time	Project	Threat	Severity	Confidence	Status
22/10/2026, 11:32:28 am	P005	Privilege Escalation	High	16.8%	Observation
22/10/2026, 11:32:18 am	P005	Suspicious Login	High	79.4%	Observation
22/10/2026, 11:32:01 am	P005	Data Exfiltration	Critical	89.6%	Validated
22/10/2026, 11:31:39 am	P005	Reconnaissance	Medium	89.1%	Validated
22/10/2026, 11:31:26 am	P005	Suspicious Activity	High	28.2%	Observation
22/10/2026, 11:31:20 am	P005	Configuration Tampering	High	9.2%	Observation
22/10/2026, 11:31:15 am	P005	Privilege Escalation	High	16.8%	Observation
22/10/2026, 11:31:09 am	P005	Reconnaissance	Medium	79.9%	Observation
22/10/2026, 11:30:57 am	P005	Privilege Escalation	High	35.1%	Observation
22/10/2026, 11:30:46 am	P005	Insider Threat	Medium	3.9%	Observation
22/10/2026, 11:30:35 am	P005	Configuration Tampering	Critical	22.4%	Observation
22/10/2026, 11:30:30 am	P005	Credential Theft	High	84.7%	Observation
22/10/2026, 11:30:24 am	P005	Brute Force	High	29.7%	Observation
22/10/2026, 11:30:18 am	P005	Credential Theft	Critical	78.9%	Obs
22/10/2026, 11:30:08 am	P005	Privilege Escalation	High	6.5%	Observation

AI Assistant

HomePhase 1Phase 2Phase 3Advanced Features

System Online

22/10/2026, 11:30:30 am	P005	Credential Theft	High	84.7%	Observation
22/10/2026, 11:30:24 am	P005	Brute Force	High	29.7%	Observation
22/10/2026, 11:30:18 am	P005	Credential Theft	Critical	78.9%	Observation

Live Cloud Event Stream50 most recent events

22/10/2026, 11:32:00 am

LargeFileDownload

service-account

182.213.235.57

USA

Failure

3.9 GB

22/10/2026, 11:31:55 am

CreateUser

developer

182.14.225.146

USA

Success

22/10/2026, 11:31:36 am

DatabaseQuery

admin

182.185.184.111

UK

Success

22/10/2026, 11:31:31 am

TerminateInstances

analyst

182.288.180.249

USA

Success

22/10/2026, 11:31:25 am

ConsoleLogin

admin

185.117.188.164

China

Success

22/10/2026, 11:31:20 am

TerminateInstances

service-account

182.207.239.11

Germany

Failure

22/10/2026, 11:31:14 am

AttachUserPolicy

analyst

182.79.185.33

USA

Failure

22/10/2026, 11:31:08 am

DatabaseQuery

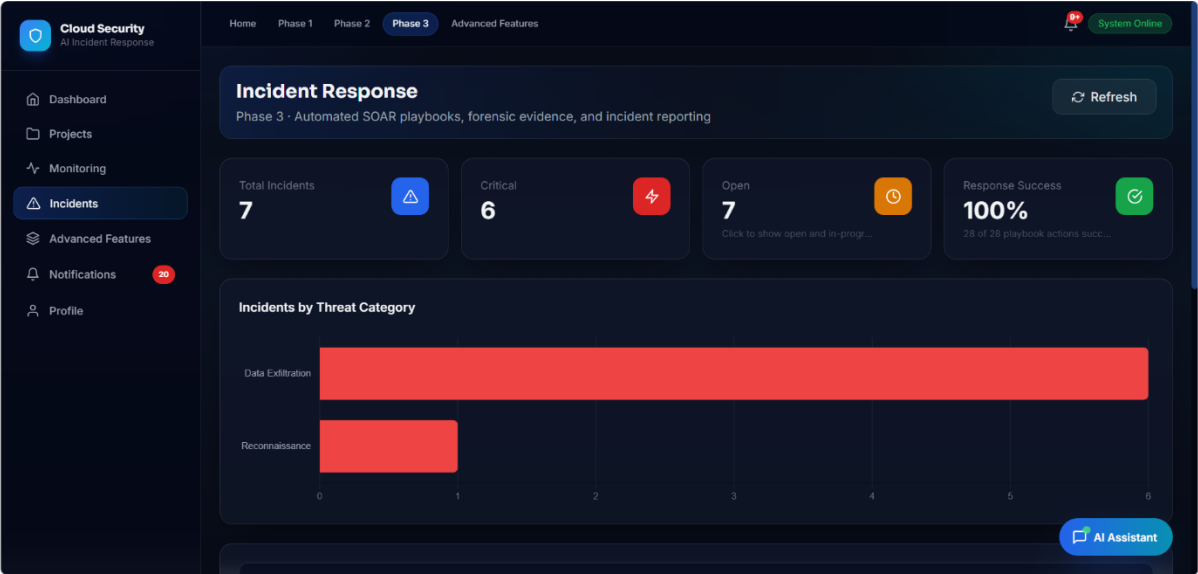
devops

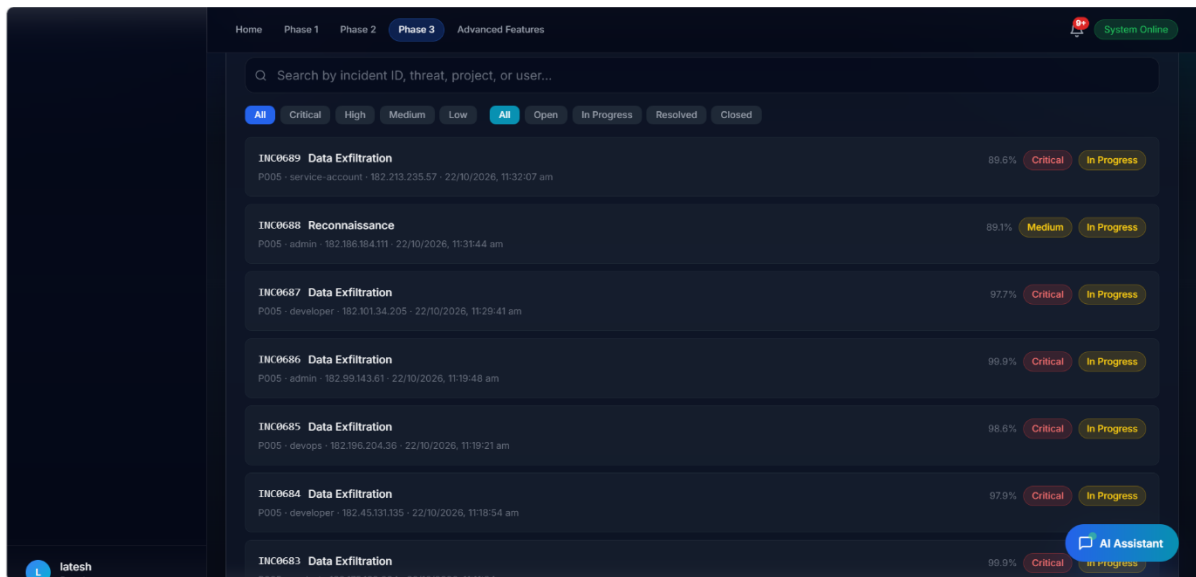
182.121.51.222

Russia

Failure

AI Assistant





CONCLUSION

The Cloud Security Monitoring and Automated Incident Response using AI project presents an integrated approach to addressing modern cloud security challenges. Traditional security approaches often depend on separate tools for vulnerability assessment, monitoring, threat detection, incident management, and response. Managing these tools independently can result in limited visibility, delayed detection, and increased manual effort.

The proposed system addresses this problem through a three-phase security architecture. Phase 1 performs pre-deployment security assessment and evaluates the security condition of an application before it proceeds to the next stage. The security-score-based approval mechanism ensures that applications meeting the defined security requirements can proceed to monitoring, while applications with significant security weaknesses can be returned for remediation.

Phase 2 extends security beyond the initial assessment by continuously analyzing security-related events and identifying suspicious activities. AI and machine-learning techniques assist in detecting abnormal behavior and potential threats from security data. This provides improved visibility into the security status of the monitored environment.

Phase 3 completes the security lifecycle by converting validated threats into incidents and applying predefined automated response actions. The integration of incident management, response playbooks, evidence preservation, notifications, and reporting helps reduce the time between threat detection and response.

The project also incorporates Zero Trust concepts, MFA, OTP-based verification, AI-assisted security explanations, SIEM/SOAR concepts, and forensic capabilities, making the platform more comprehensive than a basic monitoring system.

The major contribution of the project is therefore not a single detection algorithm, but the integration of prevention, detection, analysis, response, and explanation into one security workflow. The proposed architecture demonstrates how AI can support security teams by identifying abnormal behavior, assisting with security analysis, and accelerating incident response.

Although the current implementation has limitations related to real-cloud deployment, training data, scalability, false positives, and automated-response risks, the architecture provides a strong foundation for future development. The system can be extended with direct AWS, Azure, and Google Cloud integrations, larger real-world security datasets, advanced deep-learning models, threat-intelligence feeds, improved explainable AI, distributed log processing, and human-in-the-loop response mechanisms.

In conclusion, the proposed system demonstrates the potential of AI-driven cloud security automation to improve security visibility, reduce manual effort, accelerate incident response, and provide a unified security lifecycle from pre-deployment assessment to continuous monitoring and automated incident response. It provides a practical foundation for developing more intelligent, scalable, and adaptive cloud security platforms in the future.

REFERENCES

- [1] A. Chagovet, T. Bakardjieva, A. Ivanova, F. Sapundzhi, V. Spasova, and A. Ivanova, "AI-Driven Threat Detection and Automated Incident Response for Securing Cloud Workloads," *Applied Sciences*, vol. 16, no. 13, p. 6454, 2026, doi: 10.3390/app16136454.
- [2] L. Zhao, B. Li, and H. Yuan, "Cloud edge integrated security architecture of new cloud manufacturing system," *Journal of Systems Engineering and Electronics*, vol. 35, no. 5, pp. 1177–1189, Oct. 2024, doi: 10.23919/JSEE.2024.000112.

- [3] B. Li, M. Guo, Y. Jia, T. Zeng, and X. Liu, "Design of cloud platform alert monitoring and automatic analysis system based on random forest algorithm," *Array*, vol. 29, p. 100694, 2026, doi: 10.1016/j.array.2026.100694.
- [4] R. Guntupalli, "Enhancing Cloud Security with AI: A Deep Learning Approach to Identify and Prevent Cyberattacks in Multi-Tenant Environments," *Journal of Informatics Education and Research*, vol. 4, no. 3, pp. 3808–3813, 2024.
- [5] H. Pitkar, "Cloud Security Automation Through Symmetry: Threat Detection and Response," *Symmetry*, vol. 17, no. 6, p. 859, 2025, doi: 10.3390/sym17060859.
- [6] M. A. M. Farzaan, M. C. Ghanem, A. El-Hajjar, and D. N. Ratnayake, "AI-Powered System for an Efficient and Effective Cyber Incidents Detection and Response in Cloud Environments," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 3, pp. 623–643, 2025, doi: 10.1109/TMLCN.2025.3564912.
- [7] S. M. Shaffi, R. Vijayan, S. Vengathattil, and J. N. Sidhick, "AI-Driven Security in Cloud Computing: Enhancing Threat Detection, Automated Response, and Cyber Resilience," *Cyber Security and Applications*, 2026.
- [8] G. Sarraf and V. Pal, "Autonomous Threat Detection and Response in Cloud Security: A Comprehensive Survey of AI-Driven Strategies," *International Journal of Emerging Research in Engineering and Technology*, vol. 6, no. 4, pp. 111–121, 2025, doi: 10.63282/3050-922X.IJERET-V6I4P114.
- [9] M.-D. Nguyen, W. Mallouli, A. R. Cavalli, and E. Montes de Oca, "AI4SOAR: A Security Intelligence Tool for Automated Incident Response," in *Proceedings of the 19th International Conference on Availability, Reliability and Security (ARES '24)*, Vienna, Austria, 2024, pp. 1–8, doi: 10.1145/3664476.3670450.
- [10] C. S. Kummarapurugu, "Architectural Framework for Threat Intelligence Integration with SIEM and SOAR in Hybrid Cloud Security Environments," *International Journal of Innovative Research and Creative Technology*, vol. 10, no. 2, pp. 1–9, 2024.
- [11] P. Nguyen et al., "AI-driven digital twin-based security orchestration, automation and response for critical infrastructures," *Automated Software Engineering*, vol. 33, no. 61, 2026, doi: 10.1007/s10515-026-00612-1.
- [12] Ismail et al., "Toward Robust Security Orchestration and Automated Response in Security Operations Centers with a Hyper-Automation Approach Using Agentic Artificial Intelligence," *Information*, vol. 16, no. 5, p. 365, 2025, doi: 10.3390/info16050365.
- [13] I. Kawaminami, S. Shao, S. Hariri, and C. Tunc, "Attack Classification and Response Framework (ACR) Based on Machine Learning and CAPEC Ontology," *IEEE Access*, vol. 14, pp. 11560–11581, 2026, doi: 10.1109/ACCESS.2026.3653365.
- [14] M. Repetto, "Chaining Digital Services: Challenges to Investigate Cyber-Attacks at Run-Time," *IEEE Communications Magazine*, vol. 62, no. 5, pp. 88–94, May 2024, doi: 10.1109/MCOM.0022200942.
- [15] Nagasundari S, P. Manja, P. Mathur, and P. B. Honnavalli, "Extensive Review of Threat Models for DevSecOps," *IEEE Access*, vol. 13, pp. 45252–45271, 2025, doi: 10.1109/ACCESS.2025.3547932.