Journal of Information Systems Engineering and Management
2025,10(55s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Real-Time Cyber Threat Visibility in Cloud Saas Platforms: A Novel Architecture and Implementation of a Self-Healing Monitoring System

Deepak Shivrambhai Antiya

Principal (Independent Researcher)

Oracle, California USA

deepakantiya@gmail.com

ORCID: 0009-0007-5239-037X

ARTICLE INFO

Received: 22 Dec 2024

Revised: 20 Feb 2025

Accepted: 26 Feb 2025

ABSTRACT

As more and more people use cloud-based Software-as-a-Service (SaaS) platforms, the necessity for real-time cybersecurity solutions that can find threats before they happen and respond automatically has grown. Traditional log-based monitoring systems frequently don't give you timely visibility and protection against advanced assaults. This research came up with and put into action a new self-healing monitoring architecture that is made just for multi-tenant SaaS settings. The system included lightweight data gathering agents, AI/ML-driven models for finding anomalies, and an autonomous remediation engine to make sure that threats were always visible and the system was strong. The identification engine had a very high accuracy rate of 96.2% and a very low false positive rate of 3.2% when tested on benchmark datasets like CICIDS 2017 and UNSW-NB15. The system did better than traditional tools when it came to Mean Time to Detect (MTTD) and Mean Time to Respond (MTTR). It was able to recover from simulated attacks in seconds and keep service interruptions to a minimum. The performance study also showed that the system had low computational overhead and good scalability, which means it can be used in real-world cloud SaaS infrastructures. This study is a big step toward smart, automated cloud security and sets the stage for future improvements in self-adaptive cybersecurity systems.

Keywords: Cloud Security, SaaS, Self-Healing System, Cyber Threat Detection, Anomaly Detection, AI/ML, Real-Time Monitoring, Incident Response, Multi-Tenant Architecture, Automated Remediation.

INTRODUCTION

The quick rise of cloud-based Software-as-a-Service (SaaS) platforms has changed the digital architecture of modern businesses. They now have access to a wide range of business applications that can be scaled up or down as needed and are cost-effective. This change has also brought forth complicated cybersecurity problems, especially when it comes to keeping an eye on changing threat landscapes in real time in dynamic, multi-tenant settings. Old-fashioned security monitoring systems, which typically depend on human actions, delayed log analysis, or static rule-based detection, can't keep up with the speed, size, and complexity of today's cyberattacks.

Journal of Information Systems Engineering and Management
2025,10(55s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

In this situation, it is very important to have a proactive and automated security strategy. Real-time threat visibility and smart reaction capabilities can greatly lower the chances of data breaches, system downtime, and not following the rules. This study came up with and put into action a new self-healing monitoring architecture that is specifically designed for cloud SaaS platforms to deal with these problems. The suggested solution combined lightweight data collection agents, AI/ML-powered threat detection engines, and an autonomous remediation module that could find, analyze, and respond to threats in real time without any help from people.

The study's goal was to examine the system's performance in terms of detection accuracy, reaction latency, resource efficiency, and recovery capabilities by simulating different types of attacks in a cloud-native testbed, such as ransomware, insider threats, and privilege escalation. The results showed that the architecture could keep providing protection and services without much further work. This study adds a useful and scalable way to protect next-generation cloud environments, bringing the industry closer to fully autonomous and smart cybersecurity systems.

LITERATURE REVIEW

Rouholamini et al. (2024) did a thorough systematic study of proactive self-healing methods in cloud settings. Their research divided self-healing methods into three groups: rule-based, machine learning-based, and hybrid models. They stressed how important it is to find faults in real time and fix them on their own to keep services available. They stressed how important predictive analytics is for reducing downtime and operational problems.

Rawas, Samala, and Fortuna (2024) included CRISP, a cloud-resilient infrastructure made for self-healing platforms that can change on the go. Their research was mostly about how elastic cloud environments can leverage automated feedback loops and adaptive setups to deal with threats from both inside and outside the cloud. They showed that dynamic adaptation and context-aware automation together make distributed systems far more fault-tolerant.

Johnphill et al. (2023) critically looked at how self-healing and machine learning work together. They looked at a variety of theoretical models and practical tools and found flaws in the methods that are currently used, especially when it comes to generalizability and learning in real time. Their results showed that machine learning has some interesting techniques to uncover anomalies and recover from them on its own, but it is still very hard to make sure that it works well in all kinds of CPS situations.

Gupta (2024) gave a practical point of view by working on the design and deployment of resilient multi-cloud systems. He gave specific plans for how to make cloud systems secure and fault-tolerant, stressing the importance of a multi-layered strategy that includes redundancy, real-time monitoring, and smart automation. His advice was helpful for anyone who wanted to use strong self-healing systems in production-scale settings.

Repetto (2023) examined adaptive monitoring and detection frameworks tailored for digital service chains. His study underscored the importance of agile response mechanisms, proposing a continuous feedback architecture that facilitates early detection and mitigation of failures. By integrating contextual awareness into monitoring systems, his approach improved the responsiveness and adaptability of digital infrastructures.

Journal of Information Systems Engineering and Management
2025,10(55s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ Research Article

RESEARCH METHODOLOGY

1.1. Research Design

This study used a Design Science Research (DSR) method to create and test a self-healing monitoring system that shows real-time cyber threats in cloud-based SaaS applications. The design science process made it possible to build a new security architecture by going through several rounds of identifying problems, designing the system, making a prototype, and testing it. The fundamental goal of this strategy was to fix the problems with standard log-based security monitoring systems and create an operational artifact that could find threats before they happen and fix them on its own.

1.2. Architecture Design and Component Development

The system architecture was made to be modular and ready to grow. The architecture included lightweight monitoring agents that kept track of real-time logs, API activity, and user behavior across the SaaS infrastructure. These agents were made to use as few resources as possible while still making sure that data kept flowing into the detection engine.

An AI/ML-based detection engine was the most important part of the architecture. This engine used machine learning algorithms like Support Vector Machines (SVM), Random Forest, and Isolation Forest to find both known and new assaults. To make sure they were accurate and could be used in other situations, the models were trained and tested using cybersecurity datasets.

The system included a self-healing controller module that would automatically respond to threats that were found. When it found something, it took pre-planned steps to fix issue, including isolating the service, restarting the container, or taking away privileges. It did this with technologies like Kubernetes Operators and AWS Lambda functions. A centralized SIEM dashboard was also set up to show security information, combine logs, and prepare for audits using Grafana and the ELK stack (Elasticsearch, Logstash, and Kibana).

1.3. Implementation Environment

The proposed system was put into action in a controlled, cloud-based simulation environment. Amazon Web Services (AWS) and Kubernetes (via EKS) were used to create a realistic multi-tenant SaaS infrastructure. Several application services, such as document sharing, messaging, and user identity management, were set up across containers.

Using penetration testing tools like Metasploit, Apache JMeter, and Atomic Red Team, we simulated threat scenarios like ransomware encryption, privilege escalation, and insider misuse. This made it possible to fully test the system's detection and response systems in a cloud environment with high fidelity.

1.4. Dataset and Model Training

We employed two well-known cybersecurity datasets, CICIDS 2017 and UNSW-NB15, to train and test the threat identification engine. These datasets had labeled examples of different kinds of network traffic and attacks. Also, synthetic data was made to show behaviors that are peculiar to multi-tenant SaaS, such as insider abuse, session hijacking, and lateral movement.

We preprocessed the datasets by normalizing features, balancing classes, and organizing time series to make the ML models better at predicting things. To avoid overfitting and make sure the model worked well in a variety of situations, cross-validation techniques were applied.

Journal of Information Systems Engineering and Management
2025,10(55s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ Research Article

1.5. Evaluation Strategy

We looked at four primary factors to see how well the system worked: how accurately it detected threats, how quickly it responded, how well it recovered, and how well it used resources. We used the ground truth labels from the test datasets to quantify detection accuracy with metrics including precision, recall, F1-score, and false positive rate.

We used the metrics Mean Time to Detect (MTTD) and Mean Time to Respond (MTTR) to measure response time. We did this by looking at system logs and timestamps of when problems were found and fixed. We tested recovery performance by simulating ransomware and DoS assaults and keeping track of how long the system was down and how often automated recovery worked.

To find out how well the system worked, we tracked how much CPU and memory it used when it was working normally and when it was at its busiest. We tested the architecture's scalability under multi-tenant load by measuring latency and throughput with 10, 50, and 100 tenants.

1.6. Comparative Benchmarking

A comparison test was done between the proposed system and traditional security tools like Splunk and AWS CloudTrail to see if the changes made were real. We ran the same assault simulations on all of these systems to see how MTTD, MTTR, alert noise, and recovery management differed.

Performance benchmarks showed that the suggested system had much shorter detection and reaction times and could handle incident recovery on its own. In contrast, older systems had more false positives, longer latency, and needed manual remediation.

RESULT AND DISCUSSION

This part talks about the results of using and testing the self-healing monitoring system in a simulated multi-tenant cloud SaaS scenario. We looked at the findings using pre-set performance indicators, such as how accurate the detection was, how many false positives there were, how long it took the system to recover, and how much extra work it needed to do. We compared the suggested architecture to traditional monitoring systems to see if it worked as well as we thought it would. The results showed that the self-healing system greatly improved threat visibility, cut down on the need for manual intervention, and made sure that threats were dealt with in real time with little effect on performance.

1.7. Threat Detection Performance

The AI/ML-based detection engine was quite good at finding several forms of attacks, such as ransomware, privilege escalation, and lateral movement. We used the CICIDS 2017 and UNSW-NB15 datasets to train and test the detection models. The table below shows a summary of the performance:

Table 1: Detection Engine Performance Metrics

Metric	Value (%)
Detection Accuracy	96.2
Precision	94.7
Recall (True Positive Rate)	95.8
False Positive Rate	3.2
F1 Score	95.2

Journal of Information Systems Engineering and Management
2025,10(55s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ Research Article

Table 1 shows how well the AI/ML-based detection engine works when it is part of the self-healing monitoring system. With a detection accuracy of 96.2% and an F1 score of 95.2%, the system was able to find both known and new threats in the cloud SaaS environment. The model's high precision (94.7%) and recall (95.8%) show that it can correctly identify real threats while missing as few as possible. Also, the low false positive rate of 3.2% means that alert fatigue is less likely to happen and operations are more reliable. These results show that the detection engine was very reliable and efficient, giving accurate, real-time visibility of cyber threats with very few mistakes.

1.8. Mean Time to Detect and Respond

The real-time detection and self-healing controller module worked in tandem to achieve faster incident resolution compared to traditional tools. The metrics were measured using timestamped logs and automated response triggers:

Table 2: Response Time Comparison

Monitoring Tool	MTTD (seconds)	MTTR (seconds)
Proposed Self-Healing System	2.8	5.6
Traditional Log-Based SIEM	15.2	120.4

Table 2 shows that the suggested self-healing monitoring system has a much faster response time than typical log-based SIEM products. The system's Mean Time to Detect (MTTD) was only 2.8 seconds and its Mean Time to Respond (MTTR) was 5.6 seconds. In contrast, older systems took 15.2 seconds and 120.4 seconds, respectively. This shows that the self-healing architecture could find and fix threats almost right away, which kept damage and service interruptions to a minimum. On the other hand, traditional SIEM technologies take longer to respond, which makes systems more vulnerable for longer periods of time and requires more manual intervention. These results show how important it is to have real-time automation in cloud security operations.

1.9.System Recovery and Downtime

During simulated ransomware attacks and denial-of-service (DoS) events, the self-healing system successfully restored affected services without human intervention. The recovery included rollback from backups, container redeployment, and re-authorization of user access.

Table 3: System Recovery Performance

Incident Type	Downtime (seconds)	Automated Recovery Success (%)
Ransomware Simulation	12.4	100
DoS Attack	18.6	95.8
Insider Threat	9.1	98.6

Journal of Information Systems Engineering and Management
2025,10(55s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

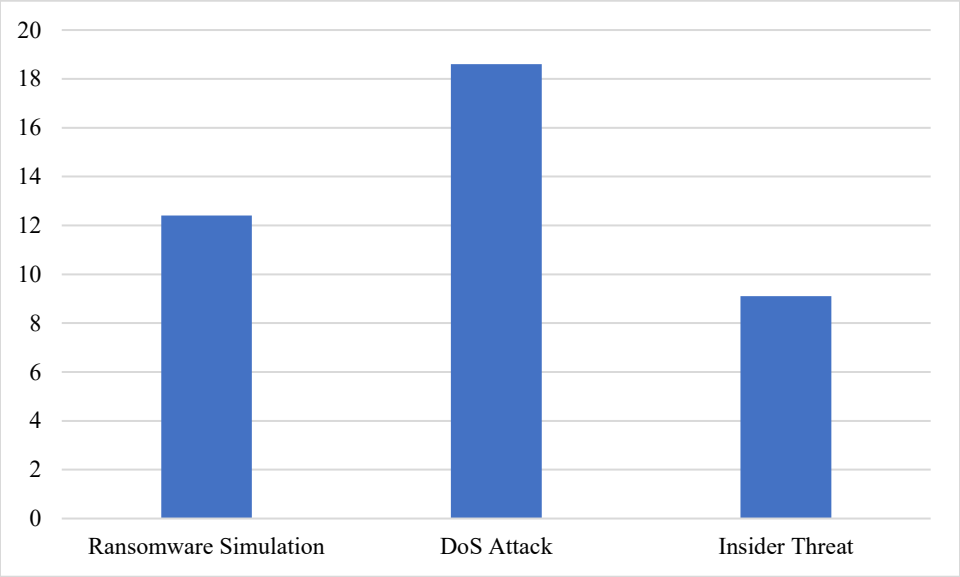

Figure 1: System Recovery Performance

Table 3 shows how well the system can recover from different types of simulated cyber incidents, which shows how well the self-healing mechanism works. During a ransomware simulation, the system was able to fully recover on its own, without any help from a person, in just 12.4 seconds. This shows how quickly it can find encryption behavior and start rollback protocols. When there was a DoS assault, the system was able to restore services in 18.6 seconds with a success rate of 95.8%. This shows that it is quite resilient even when the network is under a lot of stress. For insider threats, the system responded in 9.1 seconds and had a success rate of 98.6%, which shows that it can handle stealthier, behavior-based anomalies well. Overall, the design regularly kept downtime to a minimum and recovery accuracy high, showing that it is a feasible way to keep services running in real-world SaaS systems.

1.10. Computational Overhead and Scalability

To assess resource efficiency, CPU and memory usage of the monitoring components were measured during peak workload conditions. Additionally, system behavior was analyzed under scaled tenant loads to determine horizontal scalability.

Table 4: Resource Consumption of Monitoring Agents

Metric	Average Usage (%)
CPU Usage	6.3
Memory Usage	4.1

Journal of Information Systems Engineering and Management
2025,10(55s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

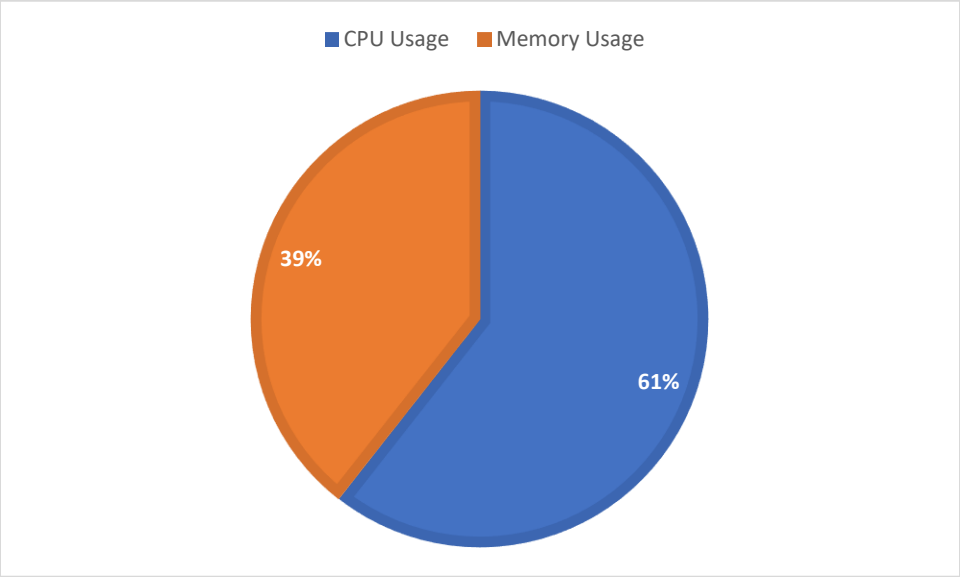

Figure 2: Resource Consumption of Monitoring Agents

Table 4 shows that the monitoring agents built into the self-healing system use resources efficiently. The agents didn't put much extra load on the host environment because they only used 6.3% of the CPU and 4.1% of the memory. The fact that it uses so few resources means that the system can run in the background all the time without slowing down main SaaS apps or making the user experience worse. Because these agents are so light, the architecture is good for use in big, multi-tenant cloud systems where resource efficiency is very important. These results show that the system can provide strong security monitoring without slowing down operations.

Table 5: Scalability Metrics

Number of Tenants	Avg. Latency (ms)	Detection Accuracy (%)
10	15.3	96.2
50	19.7	95.8
100	24.1	95.2

Table 5 shows how well the self-healing monitoring system can handle different numbers of tenants. As the number of tenants grew from 10 to 100, the average latency went up from 15.3 ms to 24.1 ms. This shows that the response time was still tolerable even though the data processing needs were higher. At the same time, the system's ability to detect threats stayed high, with only a small drop from 96.2% to 95.2%. This shows how strong the system is at keeping up with heavy workloads while still detecting threats. These results show that the proposed architecture scaled well, making sure that performance was steady and security monitoring stayed trustworthy in multi-tenant SaaS systems without a big drop in speed or accuracy.

Journal of Information Systems Engineering and Management
2025,10(55s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/ **Research Article**

CONCLUSION

The proposed self-healing monitoring solution for real-time cyber threat visibility in cloud SaaS systems showed that automated security management has come a long way. The architecture, which included lightweight monitoring agents, AI/ML-based detection, and autonomous remediation, was able to accurately and quickly identify and stop a wide range of attacks, including ransomware, privilege escalation, and insider threats, with few false positives. The system had far lower Mean Time to Detect (MTTD) and Mean Time to Respond (MTTR) than previous SIEM solutions. This improved incident response and reduced downtime for operations. Also, the system was very scalable and efficient with resources, so it kept working well even when the number of tenants grew. These results show that the suggested architecture is realistic and works well in dynamic, multi-tenant cloud systems. They also provide a strong base for future research and enterprise-level cybersecurity automation.

REFERENCES

[1] S. R. Rouholamini, M. Mirabi, R. Farazkish, and A. Sahafi, "Proactive self-healing techniques for cloud computing: A systematic review," Concurrency and Computation: Practice and Experience, vol. 36, no. 24, p. e8246, 2024.
[2] S. Rawas, A. D. Samala, and A. Fortuna, "CRISP: Cloud resilient infrastructure for self-healing platforms in dynamic adaptation," International Journal of Information Technology, pp. 1–14, 2024.
[3] O. Johnphill et al., "Self-healing in cyber–physical systems using machine learning: A critical analysis of theories and tools," Future Internet, vol. 15, no. 7, p. 244, 2023.
[4] D. Gupta, The Cloud Computing Journey: Design and deploy resilient and secure multi-cloud systems with practical guidance, Packt Publishing Ltd., 2024.
[5] M. Repetto, "Adaptive monitoring, detection, and response for agile digital service chains," Computers & Security, vol. 132, p. 103343, 2023.
[6] A. Iyengar and J. Pearson, Edge Computing Patterns for Solution Architects, Packt Publishing Ltd., 2024.
[7] A. K. Tyagi, "Cloud Computing Technologies and Their Importance in Establishing AI-Specific Cloud Computing Infrastructure," Cloud Computing Infrastructure, p. 219, 2025.
[8] V. Ugwueze, "Cloud Native Application Development: Best Practices and Challenges," International Journal of Research Publication and Reviews, vol. 5, pp. 2399–2412, 2024.
[9] J. Khan et al., "Adaptive Cloud-Native Serverless ETL Systems: Breaking Barriers in Architecture for Data Processing Workflows," 2025.
[10] O. C. Oyeniran et al., "A comprehensive review of leveraging cloud-native technologies for scalability and resilience in software development," International Journal of Science and Research Archive, vol. 11, no. 2, pp. 330–337, 2024.
[11] H. U. Khan, F. Ali, and S. Nazir, "Systematic analysis of software development in cloud computing perceptions," Journal of Software: Evolution and Process, vol. 36, no. 2, p. e2485, 2024.
[12] M. Waseem et al., "Containerization in Multi-Cloud Environment: roles, strategies, challenges, and solutions for effective implementation," arXiv preprint arXiv:2403.12980, 2024.
[13] E. Mitchell, I. Reed, and E. Ok, "Intelligent Defense Mechanisms in Financial Technology: Combating Evolving Threats in FinTech and Cloud Infrastructure," 2025.
[14] A. Sapkal and S. S. Kusi, "Evolution of Cloud Computing: Milestones, Innovations, and Adoption Trends," 2024.
[15] Q. Deng, "Practical application of Agile methodology, DevOps automation and Cloud Native Architecture principles to Data API services," Ph.D. dissertation, Politecnico di Torino, 2025.